Ablaufplanung für unabhängige parallele Maschinen

und

Parametric Pruning

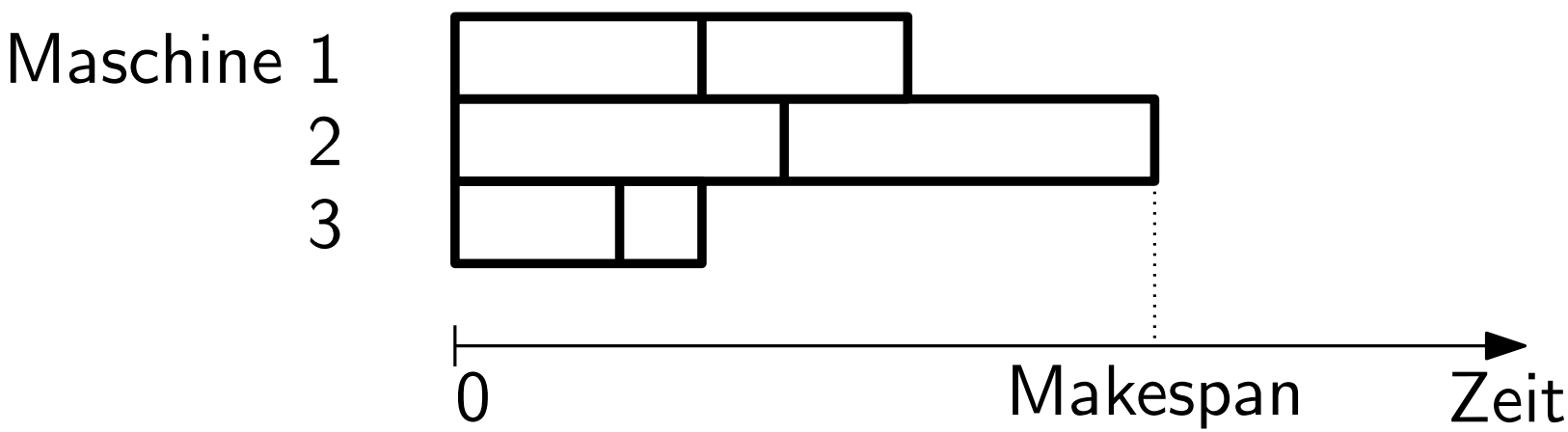
[Kap. 21 von: Vazirani, *Approximation Algorithms*, 2. Aufl., Springer '03]

Alexander Wolff
TU Eindhoven

What's the problem?

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).



IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

IP:

IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

IP: min t unter den Nebenbedingungen...

IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

IP: min t unter den Nebenbedingungen...

– jeder Auftrag wird erledigt:

IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

IP: min t unter den Nebenbedingungen...

– jeder Auftrag wird erledigt:

$$\sum_{i \in M} x_{ij} = 1 \quad \text{für alle } j \in J$$

IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

IP: min t unter den Nebenbedingungen...

– jeder Auftrag wird erledigt:
$\sum_{i \in M} x_{ij} = 1$ für alle $j \in J$

– keine Maschine arbeitet länger als t:

IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

IP: min t unter den Nebenbedingungen...

– jeder Auftrag wird erledigt:
$\sum_{i \in M} x_{ij} = 1$ für alle $j \in J$

– keine Maschine arbeitet länger als t:
$\sum_{j \in J} p_{ij} x_{ij} \leq t$ für alle $i \in M$

IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

IP: min t unter den Nebenbedingungen...

– jeder Auftrag wird erledigt:
$\sum_{i \in M} x_{ij} = 1$ für alle $j \in J$

– keine Maschine arbeitet länger als t:
$\sum_{j \in J} p_{ij} x_{ij} \leq t$ für alle $i \in M$

– Aufträge werden nicht geteilt:

IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

IP: min t unter den Nebenbedingungen...

- jeder Auftrag wird erledigt:
  $\sum_{i \in M} x_{ij} = 1$ für alle $j \in J$
- keine Maschine arbeitet länger als t:
  $\sum_{j \in J} p_{ij} x_{ij} \leq t$ für alle $i \in M$
- Aufträge werden nicht geteilt:
  $x_{ij} \in \{0, 1\}$

IP-Formulierung

Gegeben Menge J von Aufträgen (*jobs*)
Menge M Maschinen
Auftragszeiten $p_{ij} \in \mathbb{N}$ für alle $i \in M$ und $j \in J$,

finde einen kürzestmöglichen Ablaufplan (min. *Makespan*).

~~IP:~~
LP: min t unter den Nebenbedingungen...

– jeder Auftrag wird erledigt:

$$\sum_{i \in M} x_{ij} = 1 \quad \text{für alle } j \in J$$

– keine Maschine arbeitet länger als t:

$$\sum_{j \in J} p_{ij} x_{ij} \leq t \quad \text{für alle } i \in M$$

– Aufträge werden ~~nicht~~ geteilt:

~~$x_{ij} \in \{0, 1\}$~~ $x_{ij} \geq 0$

Integrality Gap

IP:

$$\min t$$

$$\sum_{i \in M} x_{ij} \geq 1 \quad \text{für alle } j \in J$$

$$\sum_{j \in J} p_{ij} x_{ij} \leq t \quad \text{für alle } i \in M$$

$$x_{ij} \in \{0, 1\}$$

Integrality Gap

IP:

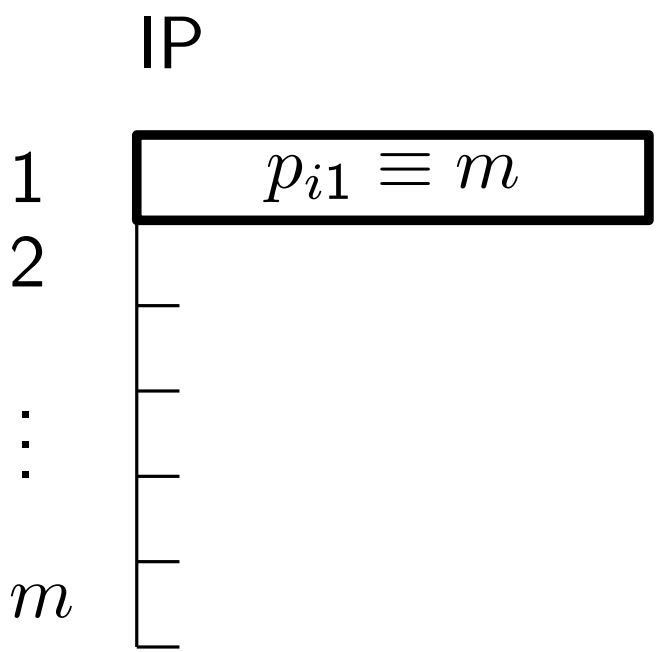
$$\min t$$

$$\sum_{i \in M} x_{ij} \geq 1 \quad \text{für alle } j \in J$$

$$\sum_{j \in J} p_{ij} x_{ij} \leq t \quad \text{für alle } i \in M$$

$$x_{ij} \in \{0, 1\}$$

Problematisches Beispiel:



Integrality Gap

IP:

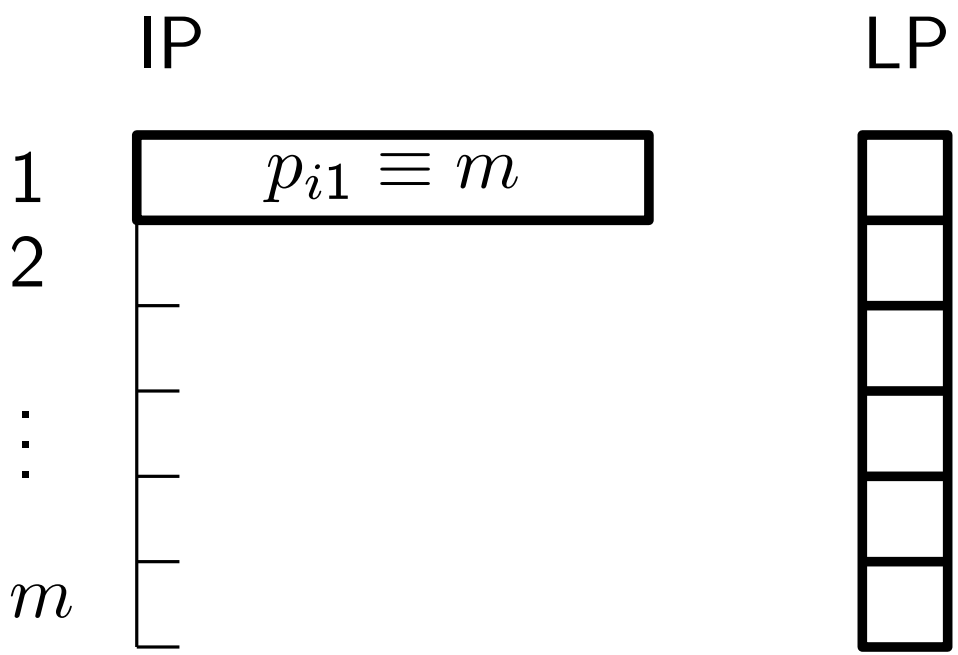
$$\min t$$

$$\sum_{i \in M} x_{ij} \geq 1 \quad \text{für alle } j \in J$$

$$\sum_{j \in J} p_{ij} x_{ij} \leq t \quad \text{für alle } i \in M$$

$$x_{ij} \in \{0, 1\}$$

Problematisches Beispiel:



Integrality Gap

IP:

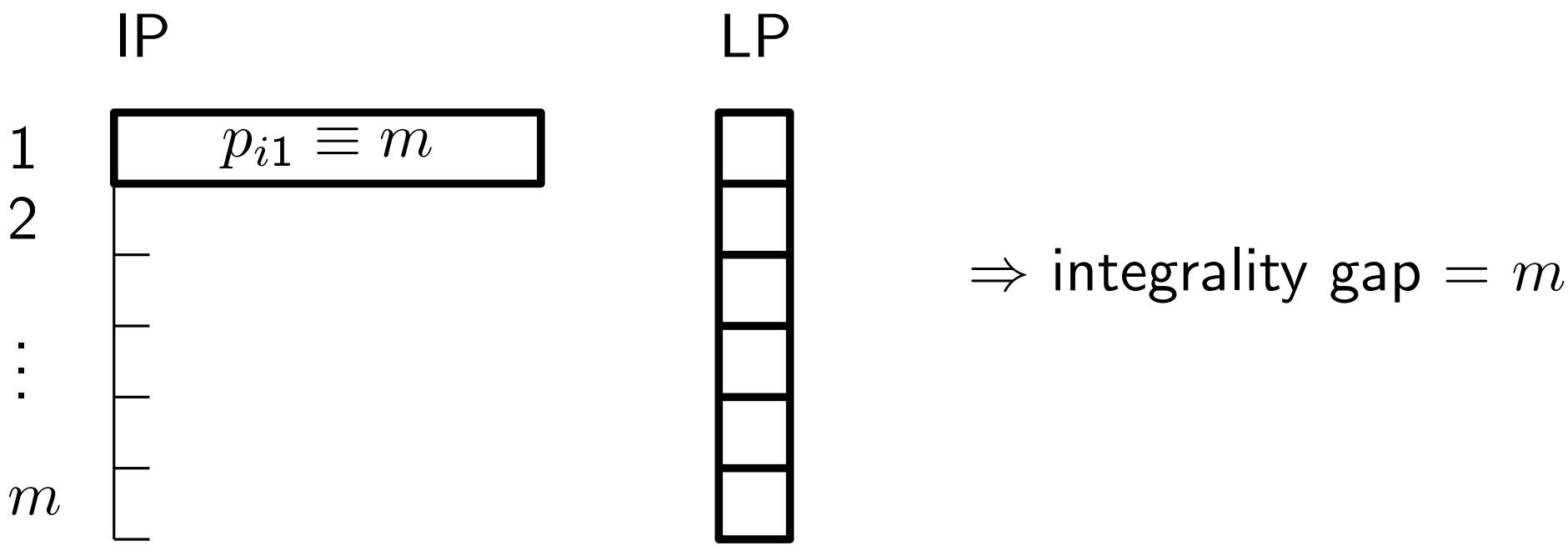
$$\min t$$

$$\sum_{i \in M} x_{ij} \geq 1 \quad \text{für alle } j \in J$$

$$\sum_{j \in J} p_{ij} x_{ij} \leq t \quad \text{für alle } i \in M$$

$$x_{ij} \in \{0, 1\}$$

Problematisches Beispiel:



$\Rightarrow$ integrality gap $= m$

Intuition

LP kann auch große Jobs gleichmäßig verteilen.

Intuition

LP kann auch große Jobs gleichmäßig verteilen.

$\Rightarrow$ Lösung hat viel kleineren Makespan

Intuition

LP kann auch große Jobs gleichmäßig verteilen.

$\Rightarrow$ Lösung hat viel kleineren Makespan

IP setzt automatisch $x_{ij} = 0$, falls $p_{ij} > t$.

Intuition

LP kann auch große Jobs gleichmäßig verteilen.

$\Rightarrow$ Lösung hat viel kleineren Makespan

IP setzt automatisch $x_{ij} = 0$, falls $p_{ij} > t$.

Bräuchten für das LP also zusätzliche Beschränkungen der Art:

für alle $i \in M, j \in J$: falls $p_{ij} > t$, dann $x_{ij} = 0$,

Intuition

LP kann auch große Jobs gleichmäßig verteilen.

$\Rightarrow$ Lösung hat viel kleineren Makespan

IP setzt automatisch $x_{ij} = 0$, falls $p_{ij} > t$.

Bräuchten für das LP also zusätzliche Beschränkungen der Art:

für alle $i \in M, j \in J$: falls $p_{ij} > t$, dann $x_{ij} = 0$,

aber das sind natürlich keine linearen Beschränkungen. :(

Lösung: parametric pruning!

Lösung: parametric pruning!

Wir führen einen Parameter T ein.

Lösung: parametric pruning!

Wir führen einen Parameter T ein.

T ist unsere Schätzung für den wahren Makespan.

Lösung: parametric pruning!

Wir führen einen Parameter T ein.

T ist unsere Schätzung für den wahren Makespan.

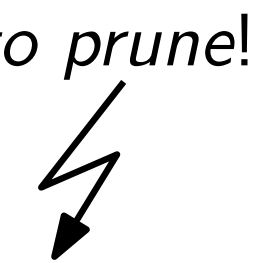
Mithilfe von T können wir alle (i, j) mit $p_{ij} > T$ wegschneiden.

Lösung: parametric pruning!

Wir führen einen Parameter T ein.

T ist unsere Schätzung für den wahren Makespan.

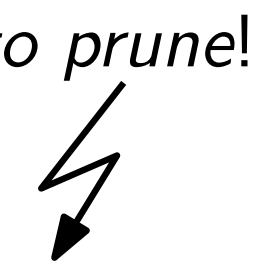
Mithilfe von T können wir alle (i, j) mit $p_{ij} > T$ wegschneiden.



Lösung: parametric pruning!

Wir führen einen Parameter T ein.

T ist unsere Schätzung für den wahren Makespan.



Mithilfe von T können wir alle (i, j) mit $p_{ij} > T$ wegschneiden.

Sei $S_T = \{(i, j) \in M \times J \mid p_{ij} \leq T\}$.

Lösung: parametric pruning!

Wir führen einen Parameter T ein.

T ist unsere Schätzung für den wahren Makespan.

to prune!

Mithilfe von T können wir alle (i, j) mit $p_{ij} > T$ wegschneiden.

Sei $S_T = \{(i, j) \in M \times J \mid p_{ij} \leq T\}$.

LP(T):

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1 \quad \text{für alle } j \in J$$

$$\sum_{j:(i,j)\in S_T} p_{ij}x_{ij} \leq T \quad \text{für alle } i \in M$$

$$x_{ij} \geq 0 \quad \text{für alle } (i, j) \in S_T$$

Lösung: parametric pruning!

Wir führen einen Parameter T ein.

T ist unsere Schätzung für den wahren Makespan.

to prune!

Mithilfe von T können wir alle (i, j) mit $p_{ij} > T$ wegschneiden.

Sei $S_T = \{(i, j) \in M \times J \mid p_{ij} \leq T\}$.

LP(T):

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1 \quad \text{für alle } j \in J$$

$$\sum_{j:(i,j)\in S_T} p_{ij}x_{ij} \leq T \quad \text{für alle } i \in M$$

$$x_{ij} \geq 0 \quad \text{für alle } (i, j) \in S_T$$

} Familie von LPs

Lösung: parametric pruning!

Wir führen einen Parameter T ein.

T ist unsere Schätzung für den wahren Makespan.

to prune!

Mithilfe von T können wir alle (i, j) mit $p_{ij} > T$ wegschneiden.

Sei $S_T = \{(i, j) \in M \times J \mid p_{ij} \leq T\}$.

keine Zielfunktion!

LP(T):

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1 \quad \text{für alle } j \in J$$

$$\sum_{j:(i,j)\in S_T} p_{ij}x_{ij} \leq T \quad \text{für alle } i \in M$$

$$x_{ij} \geq 0 \quad \text{für alle } (i, j) \in S_T$$

Familie von LPs

Plan

Binäre Suche in geeignetem Intervall ergibt kleinsten Parameter T^*, so dass LP(T^*) $\neq \emptyset$.

Plan

Binäre Suche in geeignetem Intervall ergibt kleinsten Parameter T^*, so dass LP(T^*) $\neq \emptyset$.

Dann gilt: $T^* \leq$ OPT

Plan

Binäre Suche in geeignetem Intervall ergibt kleinsten Parameter T^*, so dass LP(T^*) $\neq \emptyset$.

Dann gilt: $T^* \leq$ OPT (Warum?)

Plan

Binäre Suche in geeignetem Intervall ergibt kleinsten Parameter T^*, so dass LP(T^*) $\neq \emptyset$.

Dann gilt: $T^* \leq$ OPT (Warum?)

Wir betrachten einen Extrempunkt x von LP(T^*), also eine Ecke des Lösungspolytops.

Plan

Binäre Suche in geeignetem Intervall ergibt kleinsten Parameter T^*, so dass LP(T^*) $\neq \emptyset$.

Dann gilt: $T^* \leq$ OPT (Warum?)

Wir betrachten einen Extrempunkt x von LP(T^*), also eine Ecke des Lösungspolytops.

Ziel: x so runden (ganzzahlig machen), dass der Makespan dabei höchstens verdoppelt wird.

Plan

Binäre Suche in geeignetem Intervall ergibt kleinsten Parameter T^*, so dass LP(T^*) $\neq \emptyset$.

Dann gilt: $T^* \leq$ OPT (Warum?)

Wir betrachten einen Extrempunkt x von LP(T^*), also eine Ecke des Lösungspolytops.

Ziel: x so runden (ganzzahlig machen), dass der Makespan dabei höchstens verdoppelt wird.

$\Rightarrow$ 2-Approximation

Lemmas

Lemma 0. Jeder Extrempunkt von LP(T) hat höchstens $n + m$ Nicht-Null-Variable.

Lemmas

Lemma 0. Jeder Extrempunkt von LP(T) hat höchstens $n + m$ Nicht-Null-Variable.

Ein *Pseudobaum* ist ein Baum oder ein Baum + 1 Kante.

Ein *Pseudowald* ist ein Graph, dessen Komp. Pseudobäume sind.

Lemmas

Lemma 0. Jeder Extrempunkt von LP(T) hat höchstens $n + m$ Nicht-Null-Variable.

Ein *Pseudobaum* ist ein Baum oder ein Baum + 1 Kante.

Ein *Pseudowald* ist ein Graph, dessen Komp. Pseudobäume sind.

Sei $T \in \mathbb{N}$ und x ein Extrempunkt von LP(T).

Sei $G_x = (J \cup M, E_x)$ mit $(i, j) \in E_x$ gdw. $x_{ij} > 0$.

Lemmas

Lemma 0. Jeder Extrempunkt von LP(T) hat höchstens $n + m$ Nicht-Null-Variable.

Ein *Pseudobaum* ist ein Baum oder ein Baum + 1 Kante.

Ein *Pseudowald* ist ein Graph, dessen Komp. Pseudobäume sind.

Sei $T \in \mathbb{N}$ und x ein Extrempunkt von LP(T).

Sei $G_x = (J \cup M, E_x)$ mit $(i, j) \in E_x$ gdw. $x_{ij} > 0$.

Lemma 1. G_x ist ein Pseudowald.

Lemmas

Lemma 0. Jeder Extrempunkt von LP(T) hat höchstens $n + m$ Nicht-Null-Variable.

Ein *Pseudobaum* ist ein Baum oder ein Baum + 1 Kante.

Ein *Pseudowald* ist ein Graph, dessen Komp. Pseudobäume sind.

Sei $T \in \mathbb{N}$ und x ein Extrempunkt von LP(T).

Sei $G_x = (J \cup M, E_x)$ mit $(i, j) \in E_x$ gdw. $x_{ij} > 0$.

Lemma 1. G_x ist ein Pseudowald.

Sei $H_x \leq G_x$ induziert durch fraktionell gesetzte Aufträge.

Lemmas

Lemma 0. Jeder Extrempunkt von LP(T) hat höchstens $n + m$ Nicht-Null-Variable.

Ein *Pseudobaum* ist ein Baum oder ein Baum + 1 Kante.

Ein *Pseudowald* ist ein Graph, dessen Komp. Pseudobäume sind.

Sei $T \in \mathbb{N}$ und x ein Extrempunkt von LP(T).

Sei $G_x = (J \cup M, E_x)$ mit $(i, j) \in E_x$ gdw. $x_{ij} > 0$.

Lemma 1. G_x ist ein Pseudowald.

Sei $H_x \leq G_x$ induziert durch fraktionell gesetzte Aufträge.

Lemma 2. H_x hat ein perfektes Auftragsmatching.

Algorithmus

- $\alpha :=$ Makespan von Greedy-Ablaufplan

Algorithmus

- $\alpha :=$ Makespan von Greedy-Ablaufplan

- Binäre Suche in $[\alpha/m, \alpha]$:
  Finde kleinstes $T^* \in \mathbb{N}$, so dass LP$(T^*) \neq \emptyset$.

Algorithmus

- $\alpha :=$ Makespan von Greedy-Ablaufplan
- Binäre Suche in $[\alpha/m, \alpha]$:
  Finde kleinstes $T^* \in \mathbb{N}$, so dass LP(T^*) $\neq \emptyset$.
- Finde Extrempunkt x von LP(T^*).

Algorithmus

- $\alpha :=$ Makespan von Greedy-Ablaufplan
- Binäre Suche in $[\alpha/m, \alpha]$:
  Finde kleinstes $T^* \in \mathbb{N}$, so dass LP$(T^*) \neq \emptyset$.
- Finde Extrempunkt x von LP(T^*).
- Ein Aufrag $j \in J$ is *ganzzahlig gesetzt*, wenn es eine Maschine $i = i(j)$ mit $x_{ij} = 1$ gibt.
  Für jeden solchen Auftrag j: weise j der Maschine $i(j)$ zu.

Algorithmus

- $\alpha :=$ Makespan von Greedy-Ablaufplan
- Binäre Suche in $[\alpha/m, \alpha]$:
  Finde kleinstes $T^* \in \mathbb{N}$, so dass LP$(T^*) \neq \emptyset$.
- Finde Extrempunkt x von LP(T^*).
- Ein Aufrag $j \in J$ is *ganzzahlig gesetzt*, wenn es eine Maschine $i = i(j)$ mit $x_{ij} = 1$ gibt.
  Für jeden solchen Auftrag j: weise j der Maschine $i(j)$ zu.
- Konstruiere H_x und perfektes Auftragsmatching M in H.

Algorithmus

- $\alpha :=$ Makespan von Greedy-Ablaufplan
- Binäre Suche in $[\alpha/m, \alpha]$:
  Finde kleinstes $T^* \in \mathbb{N}$, so dass LP$(T^*) \neq \emptyset$.
- Finde Extrempunkt x von LP(T^*).
- Ein Aufrag $j \in J$ is *ganzzahlig gesetzt*, wenn es eine Maschine $i = i(j)$ mit $x_{ij} = 1$ gibt.
  Für jeden solchen Auftrag j: weise j der Maschine $i(j)$ zu.
- Konstruiere H_x und perfektes Auftragsmatching M in H.
- Weise *fraktionell gesetzte* Aufträge gemäß M an Maschinen zu.

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.
  (2. Beschränkung des LPs!)

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.
  (2. Beschränkung des LPs!)
- Einschränkung von x auf ganzzahlig gesetzte Aufträge hat dann natürlich auch Makespan $\leq T^*$.

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.
  (2. Beschränkung des LPs!)
- Einschränkung von x auf ganzzahlig gesetzte Aufträge hat dann natürlich auch Makespan $\leq T^*$.
- Für jede Kante (i, j) von H_x gilt $p_{ij} \leq T^*$.

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.
  (2. Beschränkung des LPs!)
- Einschränkung von x auf ganzzahlig gesetzte Aufträge hat dann natürlich auch Makespan $\leq T^*$.
- Für jede Kante (i, j) von H_x gilt $p_{ij} \leq T^*$. (Def. S_{T^*}!)

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.
  (2. Beschränkung des LPs!)
- Einschränkung von x auf ganzzahlig gesetzte Aufträge hat dann natürlich auch Makespan $\leq T^*$.
- Für jede Kante (i, j) von H_x gilt $p_{ij} \leq T^*$. (Def. S_{T^*}!)
- Das Matching M weist jeder Maschine ≤ 1 Auftrag zu.

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.
  (2. Beschränkung des LPs!)
- Einschränkung von x auf ganzzahlig gesetzte Aufträge hat dann natürlich auch Makespan $\leq T^*$.
- Für jede Kante (i, j) von H_x gilt $p_{ij} \leq T^*$. (Def. S_{T^*}!)
- Das Matching M weist jeder Maschine ≤ 1 Auftrag zu.

$\Rightarrow$ Makespan(Lösung) $\leq 2T^*$

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.
  (2. Beschränkung des LPs!)
- Einschränkung von x auf ganzzahlig gesetzte Aufträge hat dann natürlich auch Makespan $\leq T^*$.
- Für jede Kante (i, j) von H_x gilt $p_{ij} \leq T^*$. (Def. S_{T^*}!)
- Das Matching M weist jeder Maschine ≤ 1 Auftrag zu.

$\Rightarrow$ Makespan(Lösung) $\leq 2T^* \leq 2\,\mathrm{OPT}$

Analyse

Satz. Der Algorithmus ist eine 2-Approximation.

Beweis.

- Ein Extrempunkt x von LP(T^*) hat Makespan $\leq T^*$.
  (2. Beschränkung des LPs!)
- Einschränkung von x auf ganzzahlig gesetzte Aufträge hat dann natürlich auch Makespan $\leq T^*$.
- Für jede Kante (i, j) von H_x gilt $p_{ij} \leq T^*$. (Def. S_{T^*}!)
- Das Matching M weist jeder Maschine ≤ 1 Auftrag zu.

$\Rightarrow$ Makespan(Lösung) $\leq 2T^* \leq 2\,\mathrm{OPT}$

($T^* \leq$ OPT, da LP(OPT) $\neq \emptyset$.)

□