

Scheduling und Lineare Programmierung

Nach J. K. Lenstra, D. B. Shmoys und É.

Tardos

Janick Martinez Esturo jmartine@techfak.uni-bielefeld.de

xx.08.2007 Sommerakademie Görlitz Arbeitsgruppe 5

Gliederung

Inhaltsverzeichnis

1 Lineare Programmierung	1
1.1 LP und IP Problembeschreibung	1
2 Scheduling	2
2.1 Scheduling LP Parametric Pruning	2
2.2 Extreme-Point-Lösungen	3
2.3 Algorithmus	6

1 Lineare Programmierung

1.1 Problembeschreibung linearer und integer Programmierung

Lineare und Integer Programmierung

Definition 1 (Lineares und Integer Programm). • Ein *lineares Programm LP* ist ein Optimierungsproblem der Form: Gegeben $\vec{b} \in \mathbb{R}^m$, $\vec{c} \in \mathbb{R}^n$ und $\mathbf{A} \in \mathbb{R}^{m \times n}$, finde $\vec{x} \in \mathbb{R}^n$ damit

$$\min_{\vec{x} \in \mathbb{R}^n} \left\{ \vec{c}^T \vec{x} : \mathbf{A} \vec{x} \leq \vec{b}, \vec{x} \leq 0 \right\}$$

• Ein *integer Programm IP* ist ein Optimierungsproblem der Form: Gegeben $\vec{b} \in \mathbb{R}^m$, $\vec{c} \in \mathbb{R}^n$ und $\mathbf{A} \in \mathbb{R}^{m \times n}$, finde $\vec{x} \in \mathbb{R}^n$ damit

$$\min_{\vec{x} \in \mathbb{R}^n} \left\{ \vec{c}^T \vec{x} : \mathbf{A} \vec{x} \leq \vec{b}, \vec{x} \leq 0, \forall j \in J x_j \in \mathbb{Z}^+ \right\}$$

- Das IP heißt *pur*, falls $J = \{1, 2, \dots, n\}$, sonst *gemischt*
- Falls $x_j \in \{0, 1\}$, ist das IP *binär*

Relaxation eines Integer Programmierung

Definition 2 (Relaxation eines Integer Programmierung). • Die *Relaxation eines IPs (IPR)*

$$\min_{\vec{x} \in \mathbb{R}^n} \left\{ \vec{c}^T \vec{x} : \mathbf{A} \vec{x} \leq \vec{b}, \vec{x} \leq 0, \forall j \in J x_j \in \mathbb{Z}^+ \right\}$$

ist das Optimierungsproblem

$$\min_{\vec{x} \in \mathbb{R}^n} \left\{ \vec{c}^T \vec{x} : \mathbf{A} \vec{x} \leq \vec{b}, \vec{x} \leq 0 \right\}$$

- Beschränkung auf ganzzahlige Variablen x_j fallengelassen
- Erlaubt u.U. effiziente Bestimmung einer „guten“ Lösung. Ist für ursprüngliches IP nicht effizient möglich
- Integralitätslücke (*integrality gap*) eines Problems Π ist gegeben durch $\sup_{I \in \Pi} \frac{\text{OPT}_{IP}(I)}{\text{OPT}_{IPR}(I)}$

2 Scheduling

2.1 Scheduling durch LP Parametric Pruning

Scheduling auf unverbundenen parallelen Maschinen

Definition 3 (Schedule). • Gegeben sind Menge J von n Aufgaben und Menge M von m Maschinen

- Ein *Schedule* ist eine *Zuteilung* jeder Aufgabe auf einen (oder mehrere) Zeitintervalle von einer (oder mehrerer) Maschinen

Problem 4 (Scheduling auf unverbundenen parallelen Maschinen). • Für alle Aufgaben $j \in J$ und Maschinen $i \in M$ ist $p_{ij} \in \mathbb{Z}^+$ die benötigte Zeit, um j auf i zu bearbeiten

- Ziel: Plane die Aufgaben auf den Maschinen so, dass die gesamte Bearbeitungszeit (Makespan) minimiert wird, d.h. minimiere die maximale Bearbeitungszeit jeder Maschine

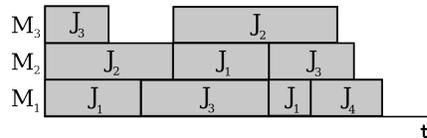
Variationen des Problems

- Hier Maschinen „unverbunden“, da *keine Relation* der Bearbeitungszeiten einer Aufgabe zwischen ihnen angenommen wird
- Für *identische* Maschinen, d.h. Aufgabe j hat auf jeder Maschine Bearbeitungszeit p_j , kann PTAS angegeben werden
- PTAS ebenfalls für *uniforme* Maschinen i angebar, die Geschwindigkeit s_i besitzen und Aufgabe j in Zeit p_j/s_i bearbeiten

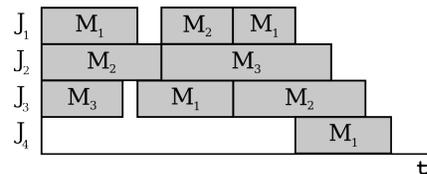
Scheduling Beispiele

Beispiele 5 (Scheduling). • *Gantt charts* zur Visualisierung eines Schedules

- Maschinenorientiert:



- Aufgabenorientiert:



Scheduling und lineare Programme

Definition 6 (Scheduling Integer Programm (SIP)). • x_{ij} sind nm Indikatorvariablen, ob Aufgabe j auf Maschine i läuft, t ist der *Makespan*

$$\begin{aligned} &\text{minimiere } t \\ &\text{unter } \sum_{i \in M} x_{ij} = 1 \quad j \in J \\ &\quad \sum_{j \in J} x_{ij} p_{ij} \leq t \quad i \in M \\ &\quad x_{ij} \in \{0, 1\} \quad i \in M, j \in J \end{aligned}$$

IP Scheduling Beispiel

Beispiel 7 (IP Scheduling mit nur einem Job). • Gegeben sei nur eine Aufgabe mit Bearbeitungsdauer m auf jeder Maschine

- SIP \Rightarrow $Makespan = m$
- SIP Relaxation $\Rightarrow x_i = \frac{1}{m} \Rightarrow Makespan = 1$
- Beispiel hat Integralitätslücke von $\frac{m}{1} = m$, das Problem hat unbeschränkte Integralitätslücke

Analyse IP Relaxation

- IP setzt $x_{ij} = 0$ wenn $p_{ij} > t$
- IPR findet günstigere Lösung, da gebrochene Werte für x_{ij} erlaubt
- Mögliche Lösung: *zusätzliche* Bedingung

$$\forall i \in M, j \in J : \text{wenn } p_{ij} > t \text{ dann } x_{ij} = 0$$

- Aber: keine lineare Bedingung \Rightarrow Verwende *Parametric Pruning*

Scheduling durch LP Parametric Pruning

Definition 8 (Parametric Pruning Scheduling (PPS)). • *Idee*: Lasse Aufgaben-Maschinenpaare (i, j) mit $p_{ij} > T$ aus

- Parameter $T \in \mathbb{Z}^+$ ist *Schätzung* einer unteren Schranke des optimalen Makespans
- Definiere Menge guter Paare $S_T = \{(i, j) : p_{ij} \leq T\}$
- $LP(T)$ ist Familie von LPs, die zulässige gebrochene Lösungen \vec{x} mit $Makespan \leq T$ suchen:

$$\begin{array}{l} \text{existiert } \vec{x}, \\ \text{so dass} \end{array} \quad \begin{array}{l} \sum_{i:(i,j) \in S_T} x_{ij} = 1, \quad j \in J \\ \sum_{j:(i,j) \in S_T} x_{ij} p_{ij} \leq T, \quad i \in M \\ x_{ij} \geq 0, \quad (i, j) \in S_T? \end{array}$$

2.2 Extreme-Point-Lösungen

Verfahrensansatz

- Finde durch *binäre Suche* den kleinsten Wert von T , so dass $LP(T)$ eine zulässige Lösung hat. Dieser Wert sei T^*
- T^* ist eine *untere* Schranke für OPT
- Algorithmus wird eine Extreme-Point-Lösung zu $LP(T^*)$ runden, so dass $Makespan \leq 2T^*$
- Nützliche Eigenschaften von Extreme-Point-Lösungen werden dabei ausgenutzt

Definition 9 (Extreme-Point-Lösung (EPL)). • Eine Extreme-Point-Lösung ist eine zulässige Lösung auf einem *Eckpunkt* des alle zulässigen Lösungen beschreibenden Polyhedrons

- Eine EPL kann durch *keine* konvexe Kombination zweier zulässiger Lösungen ausgedrückt werden

Lemma 10 (Beschränkung der Variablen einer EPL). *Jede EPL zu $LP(T)$ hat höchstens $n + m$ Variablen ungleich Null*

Beweis. • Sei $r = |S_T|$ Anzahl Variablen durch die $LP(T)$ definiert ist

- Satz: Eine zulässige Lösung zu $LP(T)$ ist eine EPL, falls r (lin. unab.) Bedingungen von $LP(T)$ auf Gleichheit gesetzt werden
- Von diesen r Bedingungen müssen mindestens $r - (n + m)$ aus der dritten Gruppe ($x_{ij} \geq 0$) sein. Setze diese Null
- \Rightarrow Jede EPL zu $LP(T)$ hat höchstens $n + m$ Variablen ungleich Null

□

Definition 11 (Integrale und partielle Aufgaben). Eine Aufgabe j heißt *integral gesetzt*, falls sie in einer Lösung \vec{x} nur einer Maschine zugeordnet wird. Sonst heißt sie *partiell gesetzt*

Folgerung 12 (Beschränkung der Aufgaben einer EPL). • *Jede EPL zu $LP(T)$ hat mindestens $n - m$ Aufgaben integral gesetzt*

- *Jede EPL zu $LP(T)$ hat höchstens m Aufgaben partiell gesetzt*

Beweis. • Sei \vec{x} eine EPL zu $LP(T)$ und α und β die Anzahl der Aufgaben, die in \vec{x} integral bzw. partiell gesetzt sind

- Jede partiell gesetzte Aufgabe wird auf mindestens zwei Maschinen ausgeführt und hat daher mindestens zwei Komponenten in \vec{x} ungleich Null
- $\Rightarrow \alpha + \beta = n$ und $\alpha + 2\beta \leq n + m$
- $\Rightarrow \beta \leq m$ und $\alpha \geq n - m$

□

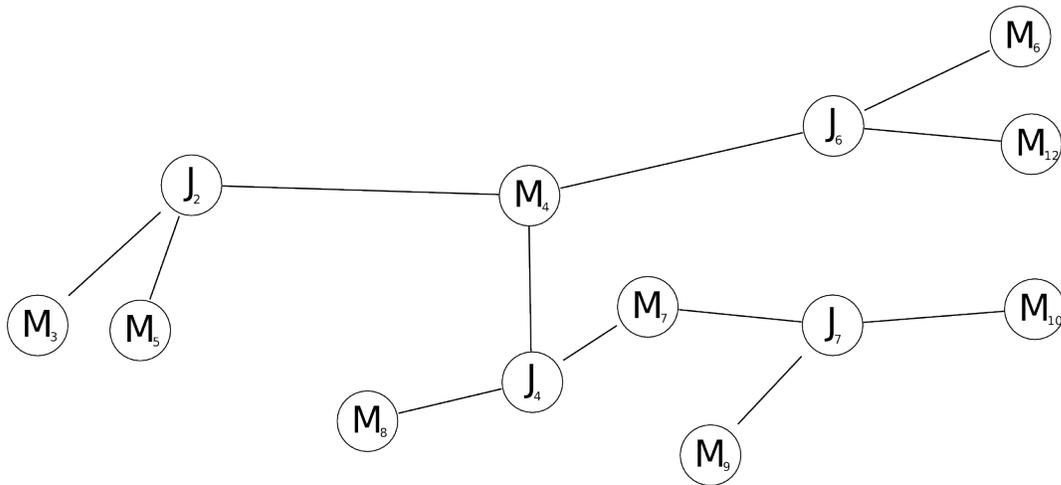
Graph einer Extreme-Point-Lösungen

Definition 13 (Bipartiter Graph G einer EPL). Gegeben einer EPL \vec{x} zu $LP(T)$, sei $G = (J, M, E)$ der *bipartite Graph* mit den Knoten $J \cup M$, so dass $(i, j) \in E$ wenn $x_{ij} \neq 0$

Definition 14 (Untergraph H von G). • $F \subset J$ sei die Menge der Aufgaben, die *partiell* in \vec{x} gesetzt sind

- H sei der *Untergraph* von G über der Knotenmenge $F \cup M$
- (i, j) ist Kante in H , falls $0 < x_{ij} < 1$
- Ein Matching in H heißt *perfektes Matching*, falls jede Aufgabe $j \in F$ abgedeckt wird
- Algorithmus nutzt aus, dass H ein perfektes Matching besitzt

Beispiel Scheduling EPL Graphen



Integrale

Aufgaben entfernt \Rightarrow Untergraph H

Pseudo-Bäume und Pseudo-Wälder

Definition 15 (Pseudo-Baum). • Ein verbundener Graph V heißt *Pseudo-Baum*, wenn er maximal $|V|$ Kanten enthält

- Da V verbunden ist, hat er mindestens $|V| - 1$ Kanten \Rightarrow er ist entweder ein Baum oder ein Baum plus einer Kante
- Im Zweiten Fall hat er einen eindeutigen Zyklus

Definition 16 (Pseudo-Wald). Ein Graph V heißt *Pseudo-Wald*, wenn all seine verbundenen Komponenten Pseudo-Bäume sind

EPL Graphen

Lemma 17 (Struktur von G). *Der zu einer EPL gehörende Graph G ist ein Pseudo-Wald*

Beweis. • Idee: Zeige, dass Anzahl an Kanten jeder verbundenen Komponente G_i von G nach oben durch Anzahl an Knoten beschränkt ist \Rightarrow Jede verbundene Komponente ist ein Pseudo-Baum

- Betrachte G_c . Beschränke $LP(T)$ und \vec{x} auf die in G_c enthaltenen Maschinen und Aufgaben zu $LP_c(T)$ und \vec{x}_c
- \vec{x}_c sei der Rest von \vec{x} . Beobachte, dass \vec{x}_c eine EPL zu $LP_c(T)$ sein muss
- B.d.W.: Wenn \vec{x}_c keine EPL zu $LP_c(T)$ ist, dann ist \vec{x}_c konvexe Kombination zweier zulässiger Lösungen zu $LP_c(T)$
- Jede der beiden Lösungen sind zusammen mit \vec{x}_c eine zulässige Lösung zu $LP(T)$
- D.h.: \vec{x} ist konvexe Kombination zweier zulässiger Lösungen zu $LP(T)$ \Rightarrow \nexists Widerspruch
- Aus Anwendung des Lemmas „Beschränkung der Variablen einer EPL“ folgt, dass G_c ein Pseudo-Baum ist

□

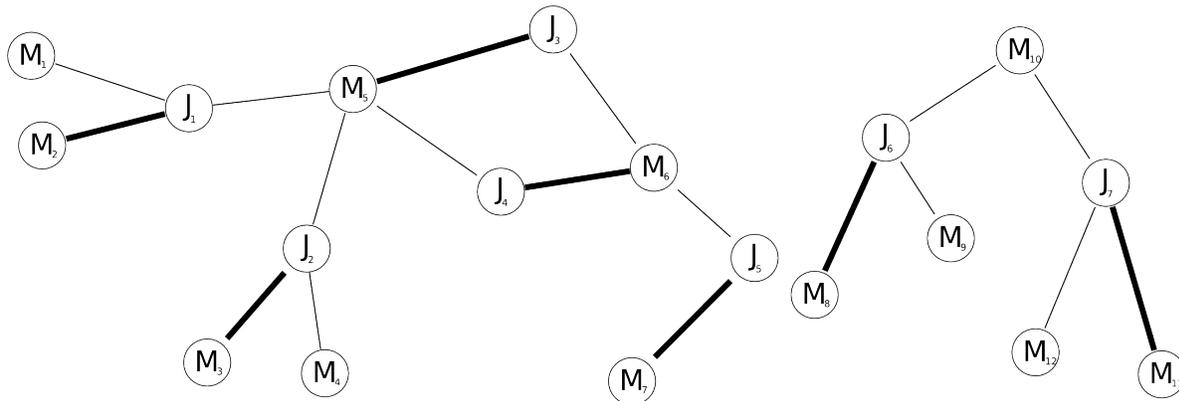
Lemma 18 (Matching von H). *Der Graph H zu einer EPL besitzt ein perfektes Matching*

Beweis. Perfektes Matching Prozedur

1. Jede integral in \vec{x} gesetzte Aufgabe hat nur eine einfallende Kante. Entferne beide \Rightarrow Untergraph H . Da gleiche Anzahl von Kanten und Knoten entfernt wurde ist H ebenfalls ein Pseudo-Wald
2. Jede Aufgabe hat mindestens Grad 2 \Rightarrow Alle Blätter müssen Maschinen sein. *Matche* jeweils Blatt mit der zugehöriger Aufgabe und entferne beide (In jedem Schritt müssen Blätter Maschinen sein)
3. Übrig bleiben gleichmäßige Zyklen. Matching von gegenüberliegenden Kanten jedes Zyklus gibt ein perfektes Matching

□

Perfektes Matching Prozedur



Perfektes Matching von H

2.3 Der Algorithmus

Scheduling Approximation Algorithmus

Algorithmus Initialisierung

Suche Bereich der Werte T durch Berechnung des *Makespans* α der *Greedy Suche* \Rightarrow Suchintervall: $[\frac{\alpha}{m}, \alpha]$

Algorithmus: Scheduling auf unverbundenen parallelen Maschinen

1. Suche per *Binäre-Suche* den kleinsten Wert $T^* \in \mathbb{Z}^+$, für den $LP(T)$ eine zulässige Lösung besitzt
2. Finde eine *EPL* \vec{x} zu $LP(T^*)$
3. Weise Maschinen alle *integralen Aufgaben* \vec{x} zufolge zu
4. Erzeuge Graph H und finde *perfektes Matching* M in H (durch Perfektes Matching Prozedur)
5. Weise Maschinen die *partiellen Aufgaben* Matching M zufolge zu

Approximationsgarantie des Algorithmus

Satz 19 (Approximationsgarantie des Algorithmus). *Der Algorithmus erreicht eine Approximationsgarantie von Faktor 2 für das Problem Scheduling auf unverbundenen parallelen Maschinen*

Beweis. • $T^* \leq \text{OPT}$, da $LP(\text{OPT})$ eine zulässige Lösung besitzt

- Die EPL \vec{x} zu $LP(T^*)$ hat einen partiellen *Makespan* $\leq T^*$
- Daher hat die Beschränkung auf integrale Aufgaben in \vec{x} einen (integralen) *Makespan* $\leq T^*$
- Jede Kante (i, j) von H genügt $p_{ij} \leq T^*$. Das perfekte Matching von H teilt höchstens *eine zusätzliche Aufgabe* pro Maschine zu
- \Rightarrow Gesamte *Makespan* $\leq 2T^* \leq 2 \text{OPT}$
- Der Algorithmus läuft in *polynomieller* Zeit

□

Abschliessendes Beispiel

Beispiel 20. • Definiere Familie von Problemen, deren Instanz m aus $m^2 - m + 1$ Aufgaben, die auf m Maschinen gesetzt werden sollen, besteht

- Erste Aufgabe hat Bearbeitungszeit m auf allen Maschinen, alle anderen haben Einheits-Bearbeitungszeit
- Optimaler Schedule setzt erste Aufgabe auf eine Maschine und verteilt jeweils m der restliche Aufgaben auf übrige $m - 1$ Maschinen
- \Rightarrow $Makespan = m$. Ausserdem hat $LP(T)$ keine zulässige Lösung für $T < m$
- Angenommen es wird die EPL zu $LP(m)$ gewählt, die $\frac{1}{m}$ der ersten Aufgabe und $m - 1$ andere Aufgaben jeder der m Maschinen zuweist
- \Rightarrow Der Algorithmus rundet eine Lösung mit $Makespan = 2m - 1$

Zusammenfassung

- Faktor 2 Algorithmus zum Scheduling auf unverbundenen parallelen Maschinen angegeben
- Verfahren basiert auf LP-Rounding mit LP-Parametric-Pruning
- Spezielle Eigenschaften einer EPL des relaxierten Problems werden zur Approximation ausgenutzt

Literatur

[J. K. Lenstra and D. B. Shmoys and É. Tardos, 1990] J. K. Lenstra and D. B. Shmoys and É. Tardos *Approximation algorithms for scheduling unrelated parallel machines* Math. Program. 46:259-271, 1990

[Vijay V. Vazirani, 2001] Vijay V. Vazirani *Approximation Algorithms* Springer-Verlag, Berlin, 2001