

Makespan-Scheduling auf allgemeinen Maschinen

Es gibt verschiedene Varianten des Makespan-Scheduling-Problems.

- Scheduling auf identischen Maschinen: Job $i \in [n]$ hat Laufzeit p_i auf jeder Maschine.
- Scheduling auf Maschinen mit Geschwindigkeiten s_1, \dots, s_m : Job $i \in [n]$ hat Laufzeit $\frac{p_i}{s_j}$ auf Maschine $j \in [m]$.
- Scheduling auf allgemeinen Maschinen: Die Eingabe besteht aus einer Matrix $(p_{ij})_{i \in [n], j \in [m]}$. Dabei bezeichnet p_{ij} die Laufzeit von Job $i \in [n]$ auf Maschine $j \in [m]$.

Die ersten beiden Probleme haben ein PAS. Das Approximationsschema für das erste Problem haben wir vorgestellt. Das Schema für das zweite Problem ist ähnlich.

Für das dritte Problem, Makespan-Scheduling auf allgemeinen Maschinen, ist kein PAS bekannt. Der beste bekannte Algorithmus hat den Approximationsfaktor 2. Diesen Algorithmus von Lenstra, Shmoys und Tardos (1990) werden wir uns im Folgenden näher anschauen.

ILP-Formulierung des allgemeinen Schedulingproblems

Wir verwenden Indikatorvariablen

$$x_{ij} \in \{0, 1\}, i \in [n], j \in [m]$$

sowie eine Variable t , die dem Makespan entspricht.

Die Zielfunktion lautet

minimiere t

Die Nebenbedingungen sind

$$\forall i \in [n] : \sum_{j \in [m]} x_{ij} \geq 1$$

$$\forall j \in [m] : \sum_{i \in [n]} x_{ij} p_{ij} \leq t$$

$$\forall i \in [n], j \in [m] \quad x_{ij} \in \{0, 1\}$$

Wir können dieses ILP relaxieren, indem wir die Ganzzahligkeit aufgeben. Das so erhaltene LP kann in Polynomialzeit gelöst werden. Anschließend könnte man versuchen durch LP-Rundung eine ganzzahlige Lösung zu erhalten, die nah an der Lösung des LPs ist. Das folgende Beispiel zeigt jedoch, dass diese Idee so nicht aufgehen kann.

Beispiel: Wir nehmen an, es gibt nur einen Job und dieser Job hat Laufzeit 1 auf jeder Maschine. Dann hat die optimale ILP-Lösung den Wert 1. Die optimale Lösung des durch Relaxierung entstandenen LPs hingegen hat den Wert $\frac{1}{m}$. Damit ist der Faktor zwischen ILP- und LP-Optimum, das sogenannte Integrality-Gap, gleich m .

Ein derartig großes Integrality-Gap bedeutet, dass das Ergebnis des randomisierten Rundens sehr weit vom optimalen Wert des LPs entfernt sein muss. In diesem Fall liefert eine LP-Rundung kein brauchbares Ergebnis.

Um ein kleines Integrality-Gap zu erhalten, entwickeln wir eine alternative ILP-Formulierung.

Wir nehmen an, ein Orakel verrät uns den optimalen Makespan T . Das Orakel können wir, wie schon im Fall der identischen Maschinen gesehen, durch eine Binärsuche in Polynomialzeit simulieren. Die alternative ILP-Formulierung mit vorgegebenem Makespan T bezeichnen wir mit $\text{ILP}(T)$.

Wenn der Makespan bekannt ist, haben wir die folgende Zusatzinformation, die wir in unser ILP einfließen lassen: Ein Job i kann nur dann auf einer Maschine j platziert werden, falls gilt $p_{ij} < T$, denn sonst würde die Laufzeit dieses einzelnen Jobs bereits den vorgegebenen Makespan überschreiten.

Alternative ILP-Formulierung – ILP(T)

Definiere $S_T := \{(i, j) \in [n] \times [m] \mid p_{ij} \leq T\}$.

ILP(T) hat die Variablen x_{ij} nur für Paare $(i, j) \in S_T$.

Eine Zuteilung von Job i auf Maschine j für $(i, j) \notin S_T$ ist damit, wie erwünscht, nicht möglich.

Der Lösungsraum von ILP(T) wird beschrieben durch die Nebenbedingungen:

$$\begin{aligned} \forall i \in [n] : \quad & \sum_{j:(i,j) \in S_T} x_{ij} \geq 1 \\ \forall j \in [m] : \quad & \sum_{i:(i,j) \in S_T} x_{ij} p_{ij} \leq T \\ \forall (i, j) \in S_T \quad & x_{ij} \geq 0 \end{aligned}$$

Wir spezifizieren keine Zielfunktion. Es ist ausreichend eine beliebige zulässige Lösung zu berechnen, weil jede zulässige Lösung den Makespan (höchstens) T hat.

Relaxierung und Berechnung einer LP-Lösung

- Wir lassen die Ganzzahligkeitsbedingung fallen und erhalten aus ILP(T) das lineare Programm LP(T).
- Mit der Ellipsoidmethode berechnen wir eine zulässige Lösung für LP(T) in Polynomialzeit.

Diskussion verschiedener LP-Rundungsverfahren

Die nicht-ganzzahlige Lösung von $LP(T)$ kann durch randomisiertes Runden in eine ganzzahlige Lösung transformiert werden. Dabei geht man analog zum randomisierten Runden für das Routingproblem vor. Man zeigt, dass mit hoher Wahrscheinlichkeit auf jeder Maschine die zugewiesene Last nur um den Faktor $O(\log m)$ vom Wert der LP-Lösung abweicht. Daraus ergibt sich ein logarithmischer Approximationsfaktor. Eine etwas genauere Rechnung zeigt, dass randomisiertes Runden den Approximationsfaktor $\Theta\left(\frac{\log m}{\log \log m}\right)$ liefert. Unser Ziel in diesem Kapitel ist es, einen wesentlich besseren Faktor zu erreichen.

Wir werden im Folgenden ein deterministisches LP-Rundungsverfahren mit Approximationsfaktor 2 beschreiben. Unsere Analyse beruht auf interessanten, kombinatorischen Eigenschaften des Lösungspolyhedrons des linearen Programms $LP(T)$. Dazu müssen wir unser Wissen über lineare Programme ein wenig auffrischen und ergänzen.

Exkursion: Eigenschaften von Basislösungen

Wie die Simplexmethode berechnet auch die Ellipsoidmethode eine sogenannte „Basislösung“, die einem Knoten des Lösungspolyhedrons entspricht. Wir gehen von einem zulässigen und beschränkten LP mit endlicher Lösung aus. Bezeichne D die Dimension des LPs, also die Anzahl der Variablen, und $C \geq D$ die Anzahl der Nebenbedingungen (Constraints). Wir erinnern uns:

Eine Basislösung des LPs entspricht einem Knoten des Lösungspolyhedrons, also dem Schnittpunkt von D linear unabhängigen Nebenbedingungen bzw. der zugehörigen Hyperebenen im \mathbb{R}^D .

Somit ist eine Basislösung also in mindestens D der zu den Nebenbedingungen gehörenden Hyperebenen enthalten. Es folgt, dass in einer Basislösung mindestens D der C Nebenbedingungen des LPs exakt (also mit Gleichheit) erfüllt sind. Entsprechend sind höchstens $C - D$ der Nebenbedingungen nicht exakt erfüllt (d.h. diese Bedingungen sind übererfüllt).

Basislösungen werden häufig auch als Extrempunktlösungen bezeichnet. Dies hat seine Ursache in der geometrischen Interpretation des Lösungsraums als Polyhedron.

Ein Punkt x eines Polyhedrons \mathcal{P} ist genau dann ein Extrempunkt oder auch Eckpunkt von \mathcal{P} , wenn es *keine* Punkte $x_1, x_2 \in \mathcal{P} \setminus \{x\}$ mit der Eigenschaft gibt, dass x auf der Verbindungslinie zwischen x_1 und x_2 liegt.

Wir erinnern uns, die Verbindungslinie zwischen zwei Punkten x_1 und x_2 besteht aus allen Linearkombinationen $\lambda x_1 + (1 - \lambda)x_2$, $\lambda \in [0, 1]$.

Aus der Konvexität des Polyhedrons \mathcal{P} folgt, dass Extrempunkte nichts anderes sind als die Knoten von \mathcal{P} . Somit sind Basislösungen also Extrempunkte, und wir erhalten die folgende alternative Charakterisierung von Basislösungen.

Eine zulässige Lösung eines LPs ist genau dann eine Basis- oder Extrempunktlösung, wenn sie sich *nicht* als Linearkombination aus anderen zulässigen Lösungen bilden lässt.

Lemma 1 In einer Basislösung für $LP(T)$ haben alle bis auf höchstens $n + m$ der Variablen den Wert 0 haben.

Beweis:

- $LP(T)$ hat $D \leq nm$ Variablen und $C = D + n + m$ Nebenbedingungen.
- In der berechneten Basislösung sind höchstens $C - D = n + m$ viele Nebenbedingungen nicht exakt bzw. nicht mit Gleichheit erfüllt.
- Somit sind auch höchstens $n + m$ der Nichtnegativitätsbedingungen (also der Bedingungen $x_{ij} \geq 0$) nicht mit Gleichheit erfüllt.
- Es folgt, alle bis auf höchstens $n + m$ Variablen haben den Wert 0.

□

Der Allokationsgraph G

Zu einer Basislösung x von $LP(T)$ definieren den Allokationsgraphen $G = ([n] \cup [m], E)$. G ist ein bipartiter Graph, dessen Knoten den Jobs und Maschinen entsprechen. Job $i \in [n]$ ist mit Maschine $j \in [m]$ genau dann durch eine Kante verbunden, wenn $x_{ij} > 0$.

Ein Pseudobaum mit Knotenmenge V ist ein zusammenhängender Graph mit höchstens $|V|$ Kanten. Jeder Spannbaum enthält $|V| - 1$ Kanten. Ein Pseudobaum ist also entweder ein Spannbaum oder ein Spannbaum mit Zusatzkante, d.h. ein Graph, der höchstens *einen* Kreis enthält.

Lemma 2 Falls G zusammenhängend ist, so ist G ein Pseudobaum.

Beweis: Gemäß Annahme ist G ein zusammenhängender Graph mit $n+m$ Knoten. Aus Lemma 1 folgt, G hat höchstens $n+m$ Kanten. Damit ist G ein Pseudobaum. \square

Ein Graph ist ein Pseudowald, wenn jede Zusammenhangskomponente jeweils einem Pseudobaum entspricht.

Lemma 3 Der Allokationsgraph G ist ein Pseudowald.

Beweis: Betrachte eine beliebige Zusammenhangskomponente $G' = (V', E')$. Die Knotenmenge V' besteht aus Teilmengen der Jobs und Maschinen, d.h. $V' = J' \cup M'$ mit $J' \subseteq [n]$ und $M' \subseteq [m]$. Wir müssen zeigen, dass G' ein Pseudobaum ist.

Das Tupel (J', M') definiert ein eingeschränktes Schedulingproblem, bei dem die Jobs aus J' auf die Maschinen in M' verteilt werden sollen. Sei $LP'(T)$ die Relaxierung zum Schedulingproblem (J', M') mit vorgegebenem Makespan T . Wenn wir die Basislösung x für $LP(T)$ auf die Variablen x_{ij} mit $i \in J'$ und $j \in M'$ einschränken (d.h. alle anderen Variablen streichen), dann erhalten wir eine zulässige Lösung x' für $LP'(T)$.

Behauptung: x' ist eine Basislösung für $LP'(T)$.

Widerspruchsbeweis: Wir nehmen an x' ist keine Basislösung. Dann ist x' kein Extrempunkt des Lösungspolyhedrons von $LP'(T)$. Also gibt es zwei andere Lösungen x'_1 und x'_2 für $LP'(T)$, so dass gilt

$$x' = \lambda x'_1 + (1 - \lambda)x'_2$$

für ein geeignetes $\lambda \in [0, 1]$. Wir ergänzen x'_1 und x'_2 um die fehlenden Variablen aus x und erhalten die Lösungen $x_1 \neq x$ und $x_2 \neq x$ für $LP(T)$. Auf diese Art gilt

$$x = \lambda x_1 + (1 - \lambda)x_2 .$$

Dies ist aber ein Widerspruch zur Eigenschaft, dass x eine Extrempunktlösung für $LP(T)$ darstellt. Also ist x' auch eine Basislösung.

Damit ist G' also ein zusammenhängender Allokationsgraph zur Basislösung x' von $LP'(T)$. Jetzt folgt aus Lemma 2, dass G' ein Pseudobaum ist. Da diese Eigenschaft für jede Zusammenhangskomponente von G gilt, ist G somit ein Pseudowald.

□

LP-Rundung mit Hilfe des Allokationsgraphen

- **Ungeteilte Jobs:** Falls die Basislösung x für einen Job $i \in [n]$ eine ganzzahlige Zuteilung berechnet hat, d.h. es gibt eine Maschine $j \in [n]$ mit $x_{ij} = 1$, so wird diese Zuteilung direkt übernommen. Wir entfernen die entsprechenden Jobknoten und die inzidente Kanten aus dem Graphen G und erhalten dadurch einen Graphen, den wir H nennen.
- **Geteilte Jobs:** Wir berechnen ein einseitig perfektes Matching M für H , d.h. eine Teilmenge der Kanten, so dass jeder Jobknoten zu genau einer Kante in M inzident ist und jeder Maschinenknoten zu höchstens einer Kante. M ordnet jedem geteilten Job i also genau eine Maschine j zu, und wir setzen $x_{ij} = 1$ und $x_{ij'} = 0$ für $j' \neq j$. Beachte, jede Maschine erhält bei diesem Rundungsschritt höchstens einen zusätzlichen Job.

Wir müssen noch die Existenz des einseitig perfekten Matchings M auf H nachweisen und zeigen, wie dieses Matching effizient berechnet werden kann. Wir beobachten, dass der Graph H ein bipartiter Pseudowald ist, in dem alle Blätter Maschinenknoten sind, weil alle Jobknoten mit Grad 1 durch Streichung der ungeteilten Jobs entfernt wurden. Diese Eigenschaft werden wir ausnutzen.

Lemma 4 Der Allokationsgraph H hat ein einseitig perfektes Matching, und dieses Matching kann in Polynomialzeit berechnet werden.

Beweis: Wir beschreiben einen einfachen Algorithmus, der das Matching berechnet. Wir nutzen aus, dass alle Blätter im Pseudowald H Maschinenknoten sind. Zunächst entfernen wir alle isolierten Maschinenknoten, d.h. alle Zusammenhangskomponenten, die nur aus einem Maschinenknoten bestehen. Den folgenden Schritt wenden wir solange an, bis kein Blatt mehr verfügbar ist.

Wähle ein beliebiges Blatt $j \in [m]$, und füge die inzidente Kante $\{j, i\}$ ($i \in [n]$) zu M hinzu. Dann entferne die Knoten j und i mit allen inzidenten Kanten aus H und lösche die dadurch möglicherweise neu entstandenen isolierten Maschinenknoten.

Da der Graph H ein bipartiter Pseudowald ist, verbleiben nach dem iterierten Entfernen aller Blätter nur ein paar Kreise gerader Länge. Von diesen Kreisen nehmen wir jede zweite Kante zum Matching M hinzu. Auf diese Art wird jeder Jobknoten durch genau eine Kante aus M abgedeckt, und wir haben ein einseitig perfektes Matching konstruiert.

□

Theorem 5 Das Makespan-Scheduling-Problem für allgemeine Maschinen hat einen Polynomialzeitalgorithmus mit Approximationsfaktor 2.

Beweis: Wir müssen uns nur noch um den Nachweis des Approximationsfaktors kümmern.

- Auf jeder Maschine verursachen die durch die Basislösung ungeteilt zugewiesenen Jobs eine Last von höchstens T .
- Beim Runden der geteilten Jobs wird jeder Maschine maximal ein Job zugeordnet, weil wir ein Matching verwenden. Eine Matchingkante $\{i, j\}$ existiert nur dann, wenn in der Basislösung $x_{ij} > 0$ gilt. Notwendige Bedingung dafür ist aber, dass $(i, j) \in S_T$ ist. Aus $(i, j) \in S_T$ folgt aber $p_{ij} \leq T$. Deshalb erzeugt die Zuteilung entlang der Matchingkanten einen Lastzuwachs von höchstens T je Maschine.

Somit ist der berechnete Makespan höchstens zweimal so groß wie der optimale Makespan T . □