ETH zürich
Departement Informatik

Approximations- und Online-Algorithmen

Dr. Hans-Joachim Böckenhauer
Dr. Dennis Komm

courses.ite.inf.ethz.ch/approx_online_alg_22

Lösungsvorschläge – Blatt 3

Zürich, 17. März 2022

Lösung zu Aufgabe 4

(a) Bei dem in der Aufgabe beschriebenen Problem handelt es sich um das Makespan Scheduling Problem. Wir entwerfen einen Greedy-Algorithmus Greedy-Makespan für dieses Problem analog zum Greedy-Algorithmus für das Rucksack-Problem in Abschnitt 4.3.4 des Buchs *Algorithmics for Hard Problems.*

> Algorithmus Greedy-Makespan
>
> **Eingabe:** Positive ganze Zahlen $w_1, \dots, w_n, k$.
>
> 1. Sortiere $w_1, \dots, w_n$ absteigend (o.B.d.A. sei danach $w_1 \geq \dots \geq w_n$).
> 2. $(I_1, \dots, I_k) \coloneqq (\emptyset, \dots, \emptyset)$.
> 3. **for** $i \coloneqq 1$ **to** n **do**
> $m \coloneqq \operatorname{argmin}_{j \in \{1,\dots,k\}} \left\{ \sum_{w \in I_j} w \right\}$.
> $I_m \coloneqq I_m \cup \{w_i\}$.
>
> **Ausgabe:** $(I_1, \dots, I_k)$.

Die Idee des Algorithmus ist es, den jeweils nächsten Prozess dem am wenigsten belasteten Prozessor zuzuweisen. Dabei werden die längsten Prozesse zuerst behandelt. Die Laufzeit wird je nach Grösse von k von der for-Schleife oder vom Sortieren im ersten Schritt dominiert und ist bei entsprechender Wahl der Datenstruktur in $O(nk + n \log n)$, also polynomiell.

Wir zeigen nun, dass die Approximationsgüte dieses Algorithmus höchstens 4/3 ist. Obwohl in der Aufgabenstellung nur eine Analyse für den Fall $k = 3$ gefordert wurde, geben wir hier eine allgemeine Analyse an, unabhängig von der Anzahl Prozessoren.

Falls $n \leq k$, berechnet Greedy-Makespan immer eine optimale Lösung. Im Folgenden gehen wir also davon aus, dass $n > k$. Wir unterscheiden zwei Fälle.

(1) Falls $w_n > \text{cost}(Opt)/3$ ist, dann können in einer optimalen Lösung auf jedem Prozessor höchstens zwei Prozesse laufen. Würden mindestens drei auf einem

Prozessor laufen, hätte dieser eine Last von mehr als $3 \cdot \text{cost}(\text{Opt})/3 = \text{cost}(\text{Opt})$, was zu einem Widerspruch führt. Also wissen wir, dass $n \leq 2k$ gilt. Unser Algorithmus Greedy-Makespan verteilt zunächst die k grössten Prozesse auf die k Prozessoren und verteilt dann die verbleibenden $n - k \leq k$ Prozesse so, dass jeweils der grösste verbleibende Prozess dem am wenigsten belasteten Prozessor zugewiesen wird. Jede Lösung, die zwei der k grössten Prozesse auf einem Prozessor laufen lässt, ist somit mindestens so teuer wie die berechnete Lösung. Da $n > k$ ist, muss jede Lösung, insbesondere auch die optimale, zwei der grössten $k + 1$ Prozesse auf einem Prozessor laufen lassen, also berechnet Greedy-Makespan in diesem Fall eine optimale Lösung.

(2) Falls $w_n \leq \text{cost}(\text{Opt})/3$ ist, folgt die Behauptung wie folgt: w_n wird einem Prozessor zugewiesen, der schon die Last P hat, also ist die Last aller anderen Prozessoren auch mindestens P. Damit ist P eine untere Schranke für $\text{cost}(\text{Opt})$. Die Länge der berechneten Lösung ist $P + w_n \leq \text{cost}(\text{Opt}) + w_n \leq 4/3 \cdot \text{cost}(\text{Opt})$.

Greedy-Makespan berechnet also in jedem Fall eine Lösung mit einem Wert von höchstens $4/3 \cdot \text{cost}(\text{Opt})$ und ist somit ein 4/3-Approximationsalgorithmus.

(b) Wir verwenden nun die Ideen aus (a), um ein PTAS für drei Prozessoren zu konstruieren.

> Algorithmus Makespan-PTAS
>
> **Eingabe:** Positive ganze Zahlen $w_1, \ldots, w_n$ und ein $\varepsilon \in \mathbb{R}^+$.
>
> 1. $t := \lceil 3/\varepsilon \rceil$.
> 2. Sortiere $w_1, \ldots, w_n$ absteigend (o.B.d.A. sei danach $w_1 \geq \cdots \geq w_n$).
> 3. **for all** Verteilungen von $w_1, \ldots, w_t$ auf I_1, I_2, I_3 **do**
>   Verfahre wie in Greedy-Makespan, um die übrigen Bearbeitungszeiten auf die Prozessoren zu verteilen.
>
> **Ausgabe:** Die kürzeste Lösung, die in der Schleife berechnet wurde.

Ist $t \geq n$, so berechnet der obige Algorithmus Makespan-PTAS offensichtlich eine optimale Lösung, also gehen wir im Folgenden davon aus, dass $t < n$. Mit einer ähnlichen Argumentation wie in (a) können wir nun schliessen, dass die Länge der berechneten Lösung höchstens w_n grösser ist als eine optimale Lösung. Dazu unterscheiden wir wieder die gleichen zwei Fälle:

(1) Falls $w_n > \text{cost}(\text{Opt})/3$ ist, überlegen wir uns wieder, dass in der optimalen Lösung auf jedem Prozessor höchstens zwei Prozesse laufen können, es aber auch mindestens einen Prozessor geben muss, der zwei Prozesse enthält. Falls $t > k$, berechnet Makespan-PTAS somit eine optimale Lösung, denn t Prozesse werden optimal verteilt, und für die restlichen Prozesse ist es optimal, sie mit dem Greedy-Verfahren zu verteilen. Falls nun $t \leq k$ ist, überlegt man sich, dass in der for-Schleife von Makespan-PTAS jede Verteilung von $w_1, \ldots, w_t$ berücksichtigt wird, insbesondere auch diejenigen, bei denen diese t Prozesse auf t Prozessoren verteilt werden. Danach folgt mit der gleichen Argumentation

wie in Fall (1) aus Aufgabenteil (a), dass Makespan-PTAS eine optimale Lösung berechnet.

(2) Falls $w_n \leq \text{cost}(\text{Opt})/3$ ist, kann man mit derselben Argumentation wie in (a) zeigen, dass die Länge der berechneten Lösung höchstens $\text{cost}(\text{Opt}) + w_n$ ist.

Sei nun $S := \sum_{i=1}^{n} w_i$. Die Länge einer optimalen Lösung ist mindestens $S/3$. Nehmen wir nun an, es wäre $w_n > S/t$. Mit unserer obigen Voraussetzung $n > t$ wäre dann $S = \sum_{i=1}^{n} w_i > S/t \cdot n > S$, was zu einem Widerspruch führt. Also können wir schliessen, dass $w_n \leq S/t$ gilt. Dann ist die Approximationsgüte beschränkt durch

$$\begin{aligned}\frac{\text{cost}(\textsc{Makespan-PTAS})}{\text{cost}(\text{Opt})} &\leq \frac{\text{cost}(\text{Opt}) + w_n}{\text{cost}(\text{Opt})} \leq 1 + \frac{w_n}{\text{cost}(\text{Opt})}\\ &\leq 1 + \frac{S/t}{S/3} = 1 + 3/t \leq 1 + \varepsilon.\end{aligned}$$

Wir müssen nun noch zeigen, dass die Laufzeit polynomiell in der Eingabelänge ist. Dazu bemerken wir, dass es $3^t = 3^{\lceil 3/\varepsilon \rceil}$ Möglichkeiten gibt, die t Prozesse auf drei Prozessoren zu verteilen. Da t nur von ε abhängt, hängt also die Anzahl 3^t der Schleifendurchläufe auch nur von ε ab. Im Übrigen muss der Algorithmus sortieren und Greedy-Makespan ausführen. Die sich ergebende Laufzeit ist also in $O(3^t\, n \log n)$.

(c) Die Laufzeit von Makespan-PTAS ist von der Anzahl Prozessoren abhängig. Falls die Beschränkung $k = 3$ wegfällt, jedoch eine konstante Anzahl c von Prozessoren vorgegeben ist, verschlechtert sich zwar die Laufzeit, jedoch ist die entsprechend angepasste Version von Makespan-PTAS immer noch ein PTAS, weil die Anzahl $c^{c/\varepsilon}$ der Schleifendurchläufe nicht von der Länge der Eingabe abhängt.

Wenn jedoch die Anzahl Prozessoren Teil der Eingabe ist, ist die Anzahl der Schleifendurchläufe gegebenenfalls superpolynomiell. Damit ist der Algorithmus kein PTAS mehr.

Lösung zu Aufgabe 5

Die Idee unseres Greedy-Algorithmus App-MSKP ist, den Rucksack mit vielen Exemplaren der kleinsten Sorte zu füllen, ausser wenn nur ein Objekt in den Rucksack passt.

Eingabe: Positive ganze Zahlen $w_1, \ldots, w_n, b$, mit $w_1 \leq b, \ldots, w_n \leq b$.

1. Sortiere alle w_i absteigend (oBdA sei danach $b \geq w_1 \geq \cdots \geq w_n$).
2. **if** $w_n > b/2$
   $a_1 := 1$, $a_2 := \cdots := a_n := 0$.
   else
   $a_1 := \cdots := a_{n-1} := 0$ und $a_n := \lfloor b/w_n \rfloor$.

Ausgabe: $(a_1, \ldots, a_n)$.

Nun zeigen wir, dass App-MSKP ein polynomieller 1.5-Approximationsalgorithmus ist.

Die Laufzeit ist durch das Sortieren im ersten Schritt beschränkt, da im zweiten Schritt nur linear viele Wertzuweisungen stattfinden. Die Laufzeit ist also in $O(n \log n)$. Wenn man statt zu sortieren nur das grösste und das kleinste Objekt sucht (die anderen werden nicht benötigt), geht das sogar in Linearzeit.

Die Approximationsgüte kann folgendermassen bestimmt werden: Falls $w_n > b/2$, sind alle Objekte grösser als $b/2$, also kann nur eines einmal eingepackt werden, und es ist optimal, das grösste einmal zu nehmen. Das tut der Algorithmus in diesem Fall.

Falls $b/2 \geq w_n > b/3$, so ist $a_n := \lfloor b/w_n \rfloor = 2$ und somit $a_n w_n > 2b/3$. Der Wert der optimalen Lösung *Opt* ist nach oben beschränkt durch die Kapazität b des Rucksacks, also ist die Güte der berechneten Lösung in diesem Fall

$$\frac{Opt}{cost(a_1, \dots, a_n)} \leq \frac{b}{a_n w_n} < \frac{b}{2b/3} = \frac{3}{2}.$$

Falls schliesslich $w_n \leq b/3$, so bleibt im Rucksack weniger als w_n Platz frei, denn sonst hätten wir das Objekt der Grösse w_n noch ein weiteres Mal eingepackt. Also ergibt sich auch in diesem Fall eine Approximationsgüte von

$$\frac{Opt}{cost(a_1, \dots, a_n)} < \frac{b}{b - w_n} \leq \frac{b}{b - b/3} \leq \frac{b}{2b/3} = \frac{3}{2}.$$

Somit ist App-MSKP ein 3/2-Approximationsalgorithmus.