ETH zürich
Departement Informatik

Algorithmik für schwere Probleme

Prof. Dr. Dennis Komm
Dr. Hans-Joachim Böckenhauer
Dr. Richard Kralovič
https://courses.algodi.inf.ethz.ch/schwereprob-2025

Lösungsvorschläge – Blatt 1

Zürich, 25. Februar 2025

Lösung zu Aufgabe 1

(a) In der hier betrachteten veränderten Version des Rucksackproblems dürfen Objekte mehrfach verwendet werden. Die Summe der Gewichte kann jedoch b nicht überschreiten. Daher kann ein Objekt (w_i, c_i) höchstens $\lfloor b/w_i \rfloor$-mal verwendet werden.

Wir passen nun den Algorithmus DPKP an das veränderte Problem an. Dazu genügt es, eine Vorberechnung einzufügen. Sei $I = (w_1, w_2, \ldots, w_n, c_1, c_2, \ldots, c_n, b)$ die gegebene Eingabe. Dann lassen wir DPKP auf der Eingabe

$$(w_{1,1}, w_{1,2}, \ldots, w_{1,\lfloor b/w_1 \rfloor}, w_{2,1}, \ldots, w_{n,\lfloor b/w_n \rfloor}, c_{1,1}, c_{1,2}, \ldots, c_{1,\lfloor b/w_1 \rfloor}, c_{2,1}, \ldots, c_{n,\lfloor b/w_n \rfloor}, b)$$

laufen, wobei $w_{i,j} = w_i$ und $c_{i,j} = c_i$ für alle i und j gilt.

Wir können zeigen, dass der Algorithmus pseudopolynomiell bleibt. Dazu müssen wir nach Definition 3.2.1.1 sicherstellen, dass die Laufzeit durch ein Polynom in der Länge der Eingabe und Max-Int(I), dem maximalen Wert innerhalb der ursprünglichen Eingabe, beschränkt ist. Offensichtlich ändert sich durch die Vorverarbeitung der Wert von Max-Int(I) nicht. Die Länge der Eingabe vergrössert sich höchstens um einen Faktor $b \leq$ Max-Int(I). Die sich ergebende Laufzeit ist also immer noch beschränkt durch ein Polynom in beiden Parametern.

(b) Beachten Sie, dass jede optimale Lösung *alle* Objekte mit Gewicht $w_i \leq 0$ enthält. Wir können also all diese Objekte direkt in die Lösung übernehmen. Seien o.B.d.A. $w_1, w_2, \ldots, w_j$ die einzigen nicht positiven Gewichte der Eingabe.

Wir können die Eingabe zu folgender Eingabe umwandeln:

$$I = \left(w_{j+1}, w_{j+2}, \ldots, w_n, c_{j+1}, c_{j+2}, \ldots, c_n, b + \sum_{i=1}^{j} |w_i| \right).$$

Im Wesentlichen wird also die Grösse des Rucksacks um $\sum_{i=1}^{j} |w_i|$ vergrössert.

Auch hier können wir zeigen, dass der Algorithmus pseudopolynomiell bleibt. Dazu erweitern wir die Definition von Max-Int auf Eingaben mit negativen Zahlen wie folgt:

$$\text{Max-Int}(I) = \max\{|\text{Nummer}(x_i)| \mid i = 1, 2, \ldots, n\}.$$

Daraus folgt sofort, dass kein Gewicht der Eingabe kleiner als $-\text{Max-Int}(I)$ sein kann. Insgesamt kann die Grösse des Rucksacks also höchstens

$$\sum_{i=1}^{j}|w_i| + b \leq (n+1) \cdot \text{Max-Int}(I)$$

sein. Der Algorithmus bleibt daher pseudopolynomiell.

(c) Objekte mit negativem Profit und positivem Gewicht kann der Algorithmus ignorieren. Der Umgang mit Objekten mit negativem Profit und negativem Gewicht ist etwas komplizierter. Wir können zum Beispiel wie in Teilaufabe (b) am Anfang alle solchen Objekte in den Rucksack packen. Damit erhöht sich wieder die Kapazität des Rucksacks, der Gesamtprofit nimmt aber ab.

Im Gegensatz zur Situation in (b) müssen wir uns aber die Möglichkeit offenlassen, diese Objekte wieder zu entfernen. Wenn wir ein Objekt mit Profit $c_i < 0$ und Gewicht $w_i < 0$ aus dem Rucksack entfernen, ist das äquivalent dazu, ein Objekt mit Gewicht $|w_i|$ und Profit $|c_i|$ einzupacken.

Wenn wir also o.B.d.A. annehmen, dass die Objekte mit negativem Profit und negativem Gewicht genau die Objekte $1, 2, \ldots, j$ sind, können die Eingabe umwandeln zu

$$I = \left(|w_1|, \ldots, |w_j|, w_{j+1}, \ldots, w_n, |c_1|, \ldots, |c_j|, c_{j+1}, c_{j+2}, \ldots, c_n, b + \sum_{i=1}^{j}|w_i|\right).$$

Dann können wir den Algorithmus aus (b) verwenden. Am Ende müssen wir von der Lösung noch $\sum_{i=1}^{j}|c_i|$ abziehen.

Analog zu (b) bleibt auch dieser Algorithmus pseudopolynomiell.

(d) Falls es ein Objekt gibt, das kein positives Gewicht hat, dann wird der Profit beliebig gross, denn jedes Objekt hat nach Definition des Rucksackproblems einen positiven Profit. Es gibt also kein sinnvolles Ergebnis. Man kann solche Eingaben separat betrachten und den Rest wie in (a) behandeln.