Technische Universität München
Zentrum Mathematik

Diskrete Optimierung: Fallstudien aus der Praxis

Barbara Wilhelm | Michael Ritter

Station 1: Komplexitätsklassen

Diskutieren Sie folgende Fragen in der Gruppe und tragen Sie Ihre Antworten auf das Arbeitsblatt ein.

1. In welchem Verhältnis stehen die Komplexitätsklassen $\mathcal{P}$ und $\mathcal{NP}$? Sind sie disjunkt oder haben sie nichtleeren Schnitt, gibt es eine Teilmengenrelation (welche?) oder sind die Klassen sogar identisch? Beweisen Sie Ihre Behauptung!

Es gilt $\mathcal{P} \subseteq \mathcal{NP}$. Um das zu zeigen, sei $\Pi \in \mathcal{P}$ und $\mathcal{A}$ ein Algorithmus, der jede Instanz von Π in polynomieller Zeit löst. Um zu zeigen, dass $\Pi \in \mathcal{NP}$ gilt, benötigen wir

a) eine Definition, wie ein Zertifikat für Π aussehen soll (das muss polynomiell codierbar sein) und

b) einen polynomiellen Algorithmus $\mathcal{A}'$, der als Input eine Instanz von Π und ein Zertifikat akzeptiert, den Output „Ja“ oder „Nein“ produziert und folgende Eigenschaft hat:
   - Für jede Nein-Instanz I von Π und jedes Zertifikat Z ist $\mathcal{A}'(I, Z) = Nein$.
   - Für jede Ja-Instanz I von Π gibt es ein Zertifikat Z, so dass $\mathcal{A}'(I, Z) = Ja$.

Ein Zertifikat für eine Instanz von Π ist typischerweise ein „leicht überprüfbarer“ Beweis, dass es sich um eine Ja-Instanz handelt (für Nein-Instanzen gibt es so einen Beweis natürlich nicht). Achtung: Für Nein-Instanzen muss es kein Zertifikat geben, das beweisen würde, dass es sich um eine Nein-Instanz handelt! (Das ist i. A. auch gar nicht so einfach. Denken Sie z. B. an das Hamilton-Kreis-Problem: Die Existenz eines Hamilton-Kreises in einem gegebenen Graphen kann man leicht beweisen, indem man den Hamilton-Kreis einfach einzeichnet. Aber der Beweis, dass ein Graph keinen Hamilton-Kreis enthält, ist meistens sehr viel schwerer zu führen.)

Bei dieser Aufgabe kann man einen ganz einfachen Weg wählen: Zu jeder beliebigen Instanz von Π nehmen wir als Zertifikat immer die leere Menge und als Algorithmus $\mathcal{A}' = \mathcal{A}$. Das tut genau das Gewünschte: Für jede Nein-Instanz liefert $\mathcal{A}'$ jetzt „Nein“, für jede Ja-Instanz liefert $\mathcal{A}'$ die Antwort „Ja“ (sogar für jedes Zertifikat – das wäre nicht nötig, schadet aber auch nicht). Und offenbar ist das Zertifikat $\emptyset$ polynomiell codierbar und der Algorithmus $\mathcal{A}'$ ist wegen $\Pi \in \mathcal{P}$ auch polynomiell. Damit ist $\Pi \in \mathcal{NP}$.

Die Frage, ob $\mathcal{P}$ und $\mathcal{NP}$ identisch sind, hat (erwartungsgemäß) leider niemand beantwortet. Es gab lediglich ausweichende Antworten wie „Wenn wir das wüssten, würden wir jetzt unsere Million verprassen.“

Bitte wenden!

2. Betrachten Sie die Entscheidungs- bzw. Evaluationsversion des Knapsack-Problems:

   Input: $n \in \mathbb{N}$, Gewichte $w \in \mathbb{N}^n$, Werte $v \in \mathbb{N}^n$, eine Kapazität $K \in \mathbb{N}$ und eine natürliche Zahl C.

   Frage (Entscheidungsproblem): Gibt es eine zulässige Lösung des Knapsack-Problems mit einem Gesamtwert von mindestens C? Eine zulässige Lösung des Knapsack-Problems ist eine Teilmenge $I \subseteq \{1, \ldots, n\}$ mit der Eigenschaft $\sum_{i \in I} w_i \leq K$, der Wert einer solchen Lösung ist $\sum_{i \in I} v_i$.

   Aufgabe (Evaluationsproblem): Bestimme das Maximum des Gesamtwerts aller zulässigen Lösungen oder stelle fest, dass keine zulässige Lösung existiert!

   Wenn Sie die Evaluationsversion lösen könnten, wäre natürlich auch die Entscheidungsversion gelöst. Aber angenommen, Sie kennen ein Orakel, das die Entscheidungsversion löst – könnten Sie damit auch das Evaluationsproblem in polynomieller Zeit (besser „in Orakel-polynomieller Zeit“) lösen? Sie dürfen das Orakel mehrmals (polynomiell oft) aufrufen und dabei ggf. beliebige Eingabedaten (polynomieller Größe) verwenden.

Bei der Frage waren die meisten Gruppen zumindest auf der richtigen Spur. Eine häufig genannte Idee:

a) Starte mit $C := V := \sum_{i=1}^{n} v_i$.

b) Rufe das Entscheidungsproblem mit dem aktuellen C-Wert auf.

c) Falls die Antwort „ja“ lautet, ist das Maximum gefunden (nämlich C).

d) Falls die Antwort „nein“ lautet, verringere C um 1 und gehe zu 2b.

Das funktioniert, hat aber einen Haken: Im schlimmsten Fall wird der Algorithmus V-mal aufgerufen. Da man zur Codierung von V aber $\log_2(V)$ Bits (Größenordnung) benötigt, wären das $2^{\log_2(V)}$ Durchläufe. Damit ist die Laufzeit leider nicht polynomiell in der Inputlänge.

Man kann dem aber relativ einfach abhelfen: Anstatt C bei jedem Durchlauf um 1 zu verringern, führt man eine binäre Suche durch. Das sieht dann so aus:

a) Starte mit $C := V := \sum_{i=1}^{n} v_i$, $B^+ := V$, $B^- := 0$.

b) Rufe das Entscheidungsproblem mit $C = V$ auf. Bei Antwort „ja“ ist das Maximum gefunden (nämlich V). Abbruch.

c) Setze $C := \lfloor 1/2 (B^+ + B^-) \rfloor$.

d) Rufe das Entscheidungsproblem mit dem aktuellen C-Wert auf.

e) Falls die Antwort „ja“ lautet, setze $B^- := C$, sonst $B^+ := C$.

f) Falls $B^+ - B^- \leq 1$, ist das Maximum gefunden (nämlich B^-). Sonst gehe zu 2c.

Mit diesem Ansatz erreichen wir die Lösung mit nur $\log_2(V)$ Durchläufen, damit haben wir einen polynomiellen Algorithmus gefunden.

Einige haben sich auch noch Gedanken gemacht, wie man neben dem Zielfunktionswert auch eine optimale Lösung bestimmen kann. Ein Ansatz dafür: Man bestimmt zunächst den Zielfunktionswert. Dann nimmt man den Gegenstand 1 aus dem Problem heraus (oder setzt einfach $w_1 := K + 1$, so dass der Gegenstand auf keinen Fall genommen wird) und bestimmt erneut den maximalen Zielfunktionswert. Verringert sich der Zielfunktionswert, dann muss

der Gegenstand 1 in jeder optimalen Lösung enthalten sein und wir nehmen ihn wieder ins Problem auf. Bleibt der Wert gleich, dann ist 1 fürs Optimum nicht relevant und wir lassen den Gegenstand weiter außen vor. Genauso verfährt man der Reihe nach mit den Gegenständen 2 bis n, am Ende bleibt eine Optimallösung übrig. Dazu benötigt man $n + 1$ Aufrufe des Evaluations-Orakels, das wiederum polynomiell oft das Entscheidungsproblem aufruft. Das heißt, wenn wir das Entscheidungsproblem polynomiell lösen könnten (was wir bisher nicht können!), dann wären auch Evaluations- und Optimierungsversion polynomiell lösbar. Das rechtfertigt (zumindest teilweise) die Beschränkung auf Entscheidungsprobleme im Rahmen der Komplexitätstheorie.