Lösungen zu den Übungsaufgaben im Buch *Automaten und Sprachen: Theoretische Informatik für die Praxis* von Andreas Müller, ISBN 978-3-662-70145-4 (Softcover), ISBN 978-3-662-70146-1 (eBook), https://link.springer.com/book/10.1007/978-3-662-70146-1. Website zum Buch: https://autospr.ch

Kapitel 13: NP-Vollständigkeit

13.1. Ein Verkehrsnetz soll regelmäßig durch Mitarbeiter kontrolliert werden, die ihre Basis an einzelnen Knotenpunkten des Netzes haben. Kann man auf effiziente Art und Weise herausfinden, an welchen Knotenpunkten man Kontrolleure stationieren muss, damit jede Strecke in einem Knoten mit Kontrolleur endet?

Lösung. Dieses Problem is äquivalent zu *VERTEX-COVER*, und damit NP-vollständig, also nach aktuellem Wissen für große Probleme nicht effizient lösbar. Die Reduktionsabbildung bildet die Objekte wie folgt ab

$$\begin{aligned}\text{Knoten} &\mapsto \text{Knoten}\\ \text{Strecke} &\mapsto \text{Kante}\\ \text{Anzahl Kontrolleure} &\mapsto k\\ \text{Knoten mit Kontrolleur} &\mapsto \text{Knoten aus dem Vertex-Cover}\end{aligned}$$

○

13.2. Für eine Gruppenarbeit sollen k Gruppen gebildet werden. Um die Zeit für das gegenseitige Kennenlernen möglichst kurz zu halten, sollen sich die Leute einer Gruppe bereits gegenseitig kennen. Alle Leute sollen beschäftigt sein. Können Sie einen effizienten Algorithmus formulieren, mit dem eine solche Gruppeneinteilung auch bei einer großen Teilnehmerzahl gefunden werden kann?

Lösung. Das Problem ist äquivalent zu *CLIQUE-COVER*, also NP-vollständig. Nach aktuellem Wissen gibt es also keine effizienten Algorithmus, der das Problem lösen würde. Die Äquivalenz wird durch folgende Abbildung vermittelt

$$\begin{aligned}\text{Teilnehmer} &\mapsto \text{Knoten}\\ \text{kenne sich} &\mapsto \text{Kante}\\ \text{Anzahl Gruppen} &\mapsto k\\ \text{Gruppe} &\mapsto \text{Clique}\end{aligned}$$

○

13.3. Ein aufstrebendes Film-Festival ist derart gewachsen, dass der Vorführsaal nicht mehr reicht. Daher müssen jetzt zwei gleich große Säle verwendet werden, und trotzdem ist das Festival wieder ausverkauft, und zwar in einem Maße, dass überhaupt nur Stars und Prominente samt ihrer Entourage eingelassen werden können. Für einzelne Besucher gibt es keine Plätze.

Doch die Stars stören sich daran, dass sie möglicherweise nicht ihre ganze Entourage im gleichen Saal haben können. Daher muss kurzfristig eine Aufteilung der Festival-Gäste gefunden werden, so dass die beiden Säle so gefüllt werden können, dass jede Entourage vollständig in einem der Säle Platz nimmt.

Der Festival-Direktor ist jedoch sehr überrascht, dass die Bestimmung einer solchen Aufteilung so lange dauert. Warum sind Sie nicht überrascht?

Lösung. Zu jedem Star $i \in I$ gibt es eine Entourage mit c_i Mitgliedern. Diese Menge muss jetzt in zwei Teilmengen A (Stars samt Entourage, die in Saal A Platz nehmen) und B (Stars samt Entourage, die in Saal B Platz nehmen) aufgeteilt werden, so dass $I = A \cup B$. Die Aufteilung muss so sein, dass in beiden Sälen gleich viele Leute Platz nehmen, also

$$\sum_{i \in A} c_i = \sum_{i \in B} c_i.$$

Dies ist das Problem *PARTITION*. Das gestellte Problem ist also äquivalent zum NP-vollständigen Problem *PARTITION*, und ist daher ebenfalls NP-vollständig. Man kann daher nach aktuellem Wissen nicht erwarten, dass es dafür einen effizienten Algorithmus gibt. ○

Diskussion. Man kann auch versuchen, das Problem Festival-Problem mit *SUBSET-SUM* vergleichen. Zu jedem Star $i \in I$ gehört die Zahl s_i der Mitglieder in der Entourage des Stars. Sei $S = \{s_i \mid i \in I\}$ die Menge all dieser Mitgliederzahlen. Sei t die Platzzahl eines der Vorführsäle. Man muss jetzt eine Teilmenge von $I' \subset I$ auswählen, so dass

$$\sum_{i \in I'} s_i = t.$$

Dies sieht auf den ersten Blick aus wie des *SUBSET-SUM*-Problem, beim genaueren Hinschauen erkennt man jedoch den Unterschied: In der Liste der s_i können einzelne Zahlen mehrfach vorkommen. Es könnte sogar sein, dass die Menge S überhaupt nur eine Zahl enthält, zum Beispiel wenn dass Filmfestival allen Stars die gleiche Zahl von Freikarten für die Entourage abgegeben hat. Dann ist das erzeugte *SUBSET-SUM*-Problem gar nicht mehr lösbar, während das ursprüngliche Problem genau dann lösbar ist, wenn t durch die immer gleiche Größe der Entourage teilbar ist.

Eine allgemeinere Formulierung des Rucksack-Problems hingegen wäre durchaus ein mögliches Vergleichsproblem, denn oben haben wir ja eine Reduktion des Festival-Problems auf das Problem konstruiert, für eine Familie (nicht Menge!) $(s_i)_{i \in I}$ von Zahlen eine Teilfamilie (nicht Teilmenge!) $(s_i)_{i \in I'}$ zu finden, so dass die Summe der Zahlen einen bestimmten Wert $t = \sum_{i \in I'} s_i$ erreicht.

13.4. In einem Entwicklungsland sollen die aus dem Ausland erhaltenen Unterstützungsmittel dazu verwendet werden, endlich alle Ortschaften mit mindestens 100 Einwohnern ans Stromnetz anzuschließen. Der Bau von Leitungen zwischen einzelnen Ortschaften ist je nach Gelände unterschiedlich teuer, zum Teil auch schlicht unmöglich. Es wird entschieden, dass man in einer ersten Phase auf Redundanz des neu zu erstellenden Netzes verzichten will. Der Minister möchte endlich wissen, ob das vorhandene Geld für das Projekt ausreicht und ist sehr ungehalten darüber, dass die Verwaltung so lange braucht, diese Frage zu beantworten. Kann man dies erklären?

Lösung. Dieses *STROMNETZ* genannte Problem ist äquivalent zu *STEINER-TREE* wie folgt. Die Ortschaften des Landes entsprechen der Menge V aller Vertices des Graphen. Die untereinander zu verbindenden Ortschaften bilden eine Teilmenge $R \subset V$. Eine Kante im Graphen G entspricht zwei verbindbaren Ortschaften, jeder solchen Kante sind die Kosten für den Bau einer Verbindungsleitung zugeordnet. Es ist nun eine Menge von Verbindungsleitungen zu wählen, die einen Baum (keine Redundanz) bilden und so, dass die Summe der Gewichte das Budget nicht übersteigt.

STEINER-TREE ↔ *STROMNETZ*

Knoten ↔ Ortschaften

$$\begin{aligned}\text{Knoten in } R &\leftrightarrow \text{zu erschließende Ortschaften}\\ \text{Gewicht } w \text{ einer Kante} &\leftrightarrow \text{Baukosten einer Verbindungsleitung}\\ \text{maximales Gewicht } k &\leftrightarrow \text{Budget}\end{aligned}$$

Da das Problem *STEINER-TREE* bekanntermaßen NP-vollständig ist und daher keinen effizienten Lösungsalgorithmus besitzt, ist nicht überraschend, dass das äquivalente Problem *STROMNETZ* von der Verwaltung nicht effizient gelöst werden konnte. ○

Diskussion. *HAMPATH* ist nicht geeignet, denn Stromnetze werden ja nicht als eine einzige Leitung gebaut, die nacheinander durch alle Ortschaften gezogen wird. Ein Stromnetz ist nicht die Tour de Suisse.

MAX-CUT ist ebenfalls nicht geeignet. *MAX-CUT* sucht die maximalen Investitionen, die man in den Sand setzen kann, indem man eine Menge von Verbindungen durchschneidet. Das ist so ziemlich das Gegenteil von dem, was man tatsächlich machen will.

Die Tatsache, dass man ein gewisses Budget k ausgeben kann, könnte einen dazu verleiten, eine Reduktion auf *SUBSET-SUM* zu versuchen. Aber in *SUBSET-SUM* kann man nicht abbilden, dass die gebauten Leitungen ein zusammenhängendes Netz bilden müssen. *SUBSET-SUM* wäre die richige Wahl, wenn es darum ginge herauszufinden, ob man genau das gesamte Budget ausgeben kann, unabhängig davon, ob damit die Ortschaften tatsächlich mit Strom versorgt werden.

13.5. Für eine medizinische Studie ist eine große Zahl von Probanden rekrutiert worden. Sie sind bereits auf Allergien getestet worden, man weiß also von jedem Probanden, auf welche Allergene (Pollen, Katzenhaare, Hausstaub, Lactose,...) er allergisch reagiert. Die Untersuchung soll sich auf eine Teilmenge von $k = 17$ oder noch mehr ausgewählten Allergenen beschränken, die so beschaffen ist, dass kein Proband auf mehr als eines der ausgewählten Allergene reagiert. Es stellt sich als schwierig heraus, eine solche Teilmenge zu finden. Warum?

Lösung. Dies ist das Problem *SET-PACKING*, wenn man folgende Identifikation vornimmt:

$$\begin{aligned}\text{Allergene} &\leftrightarrow I\\ \text{auf Allergen } i \text{ allergische Probanden} &\leftrightarrow S_i\\ \text{ausgewählte Allergene} &\leftrightarrow J\\ \text{Ausschlussbedingung zwischen Allergenen } i \text{ und } j &\leftrightarrow S_i \cap S_j = \emptyset\end{aligned}$$

Es wird verlangt, k Allergene auszuwählen, also eine Teilmenge $J \subset I$ mit $|J| = k$ zu finden. ○

Diskussion. Man könnte auch eine Reduktion auch auf *HITTING-SET* versuchen, etwa mit der folgenden Identifikation:

$$\begin{aligned}\text{Allergene} &\leftrightarrow S\\ \text{Probanden} &\leftrightarrow I\\ \text{Allergien eines Probanden } i &\leftrightarrow S_i, i \in I\\ \text{Auswahl von Allergenen} &\leftrightarrow H \subset S\\ \text{Proband hat nur eine der Allergien} &\leftrightarrow |S_i \cap H| = 1 \forall i \in I\end{aligned}$$

Leider funktioniert das nicht, weil im Aufgabenproblem die Anzahl der Allergene vorgegeben ist, das entspräche hier der Vorgabe der Anzahl der Elemente von H. Es fehlt also ein Element des Aufgabenproblems in *HITTING-SET*.

Manchmal ist es schwierig *SET-COVERING*, *EXACT-COVER* und *SET-PACKING* auseinander zu halten. Man beachte:

- In *SET-COVERING* und in *SET-PACKING* kommt eine Zahl k vor, nicht aber in *EXACT-COVER*.
- In *SET-COVERING* dürfen sich die Mengen schneiden, müssen aber auch alles abdecken. In *SET-PACKING* dürfen sich die Mengen nicht schneiden, müssen aber auch nicht alles abdecken.
- In *EXACT-COVER* dürfen sich die Mengen nicht schneiden, müssen alles abdecken, aber es kommt nicht auf ihre Anzahl an.

Oder tabellarisch:

	SET-COVERING	*SET-PACKING*	*EXACT-COVER*
Anzahl Mengen	k	k	
Vereinigung	ganze Menge		ganze Menge
paarweise Schnittmengen		$\emptyset$	$\emptyset$

13.6. Die Prüfungsvorbereitungszeit ist intensiv, manchmal reicht die Zeit nicht, alle Prüfungen vorzubereiten, bis die Prüfungssession beginnt. Es ist daher unumgänglich, während der Prüfungssession weiter zu lernen. Nehmen wir an, dass für die Vorbereitung der p Fächer $1, \ldots, p$ die Vorbereitungszeiten $t_1, \ldots, t_p$ notwendig sind. Ebenfalls bekannt ist, dass die Prüfungen zu den Zeiten $d_1, \ldots, d_p$ stattfinden, dann muss die Vorbereitung abgeschlossen sein, weil die Prüfung sonst nicht zu bestehen ist. Es stellt sich daher die Frage, in welcher Reihenfolge die Vorbereitungen durchgeführt werden sollen, damit höchstens k Prüfungen nicht bestanden werden. Ein Student, der *Automaten und Sprachen* nicht studiert hat, wendet viel Zeit dafür auf herauszufinden, ob es überhaupt eine Reihenfolge dieser Art gibt, versteht aber nicht, warum dies so aufwendig ist. Können Sie dies erklären?

Lösung. Das vorliegende Problem ist das Problem *SEQUENCING*, welches bekanntermaßen NP-vollständig ist.

$$\begin{aligned} \text{Job } i &\leftrightarrow \text{Vorbereitung für Fach } i \\ \text{Ausführungszeit } t_i &\leftrightarrow \text{Vorbereitungszeit } t_i \\ \text{Deadline } d_i &\leftrightarrow \text{Prüfungszeitpunkt } d_i \end{aligned}$$

Die Strafe für eine nicht bestandene Prüfung ist $s_1 = \cdots = s_p = 1$, die Reihenfolge muss so gewählt werden, dass die Gesamtstrafe $\leq k$ ist. ○