Beispiele von NP-vollständigen Problemen

Ergänzung zum Kapitel 13

Dies ist ein Ergänzungsartikel zum Buch *Automaten und Sprachen: Theoretische Informatik für die Praxis*, von Andreas Müller. Erschienen im Verlag Springer-Vieweg, ISBN 978-3-662-70145-4 (Softcover) und ISBN 978-3-662-70146-1 (eBook).

Die Rechte an den Bilder gehören den in der Bildunterschrift angegebenen Bildquellen, wenn keine Quelle angegeben ist dem Autor. Die Teile des Autors werden unter der Lizenz CC BY-SA 4.0 zur Verfügung gestellt, Details: https://creativecommons.org/licenses/by-sa/4.0/.

Website zum Buch: https://autospr.ch

Springer-Link: https://link.springer.com/book/10.1007/978-3-662-70146-1

Beispiele von NP-vollständigen Problemen

Andreas Müller

Zusammenfassung

Die NP-vollständigen Probleme aus dem Katalog von Richard Karp sind sehr abstrakt formuliert. Für den Anfänger ist es schwer, sich darunter etwas vorzustellen. Es wird einfacher, wenn man sich zu den einzelnen Problemen weniger abstrakte Anwendungsbeispiele ausdenkt, die man sich leichter merken kann.

1 Das Problem

Die NP-vollständigen Probleme aus dem Katalog von Richard Karp sind sehr abstrakt formuliert. Für den Anfänger ist es schwer, sich darunter etwas vorzustellen. Es wird einfacher, wenn man sich zu den einzelnen Problemen weniger abstrakte Anwendungsbeispiele ausdenkt, die man sich leichter merken kann.

In den nachfolgenden Abschnitten werden zu einigen der in Kapitel 13 vorgetellten NP-vollständigen Problemen des Katalogs von Karp "anschaulichere" Beispiele oder Illustrationen gegeben. Die Probleme sind gleich gruppiert wie im Buchkapitel. Es wird jeweils die abstrakte Formulierung des Problems aus dem Buch vorangestellt.

2 SUBSET-SUM (13.3.3)

Problem 13.11 (*SUBSET-SUM*)**.** *Gegeben ist eine Liste*

$$S = [s_i \mid 1 \le i \le n]$$

von natürlichen Zahlen $s_i \in \mathbb{N}$ *und eine natürliche Zahl* $t \in \mathbb{N}$*. Sie heißt auch der* Rucksack*. Ist es möglich, aus* S *eine Teilliste* $T \subset S$ *auszuwählen derart, dass die Elemente von* T *die Summe*

$$t = \sum_{s \in T} s$$

haben?

Beispiel 13.11. Kurz vor Jahresende in einer großen Software-Firma: Wie jedes Jahr stellt der Gruppenleiter Software-Entwicklung fest, dass noch nicht sein ganzes Budget verbraucht ist. Er entscheidet, dass diesmal der gesamte Restbetrag bis auf den letzten Rappen ausgegeben werden soll. Daher sammelt der Gruppenleiter von seinem Team Vorschläge, was mit dem verbleibenden Geld gemacht werden könnte, und gibt die Liste seiner Sekretärin. Sie soll daraus so einige Dinge auswählen, dass genau das Restbudget verbraucht

wird. Einige Tage später wundert er sich, dass die Sekretärin total überfordert ist. Ist er am richtigen Ort?

Nein, der Gruppenleiter hätte erkennen müssen, dass er der Sekretärin ein NP-vollständiges Problem gestellt hat, wofür es nach aktuellem Wissen keinen Algorithmus mit polynomieller Laufzeit gibt. Die Vorschläge des Teams bilden eine Menge $\{b_1, b_2, \ldots, b_n\}$ von Beträgen b_i. Daraus soll eine Teilmenge $I = \{i_1, \ldots, i_m\}$ gebildet werden, so dass der Restbetrag r dadurch aufgebraucht wurde:

$$\sum_{k=1}^{m} b_{i_k} = r.$$

Dies ist das Problem *SUBSET-SUM*. ○

3 Probleme zu Graph-Überdeckungen (13.4.3)

Problem 13.26 (*CLIQUE-COVER* und *EXACT-CLIQUE-COVER*)**.** *Gegeben ein Graph G und eine Zahl k, gibt es k Cliquen im Graphen G, die alle Knoten des Graphen enthalten? Können die Cliquen zudem disjunkt gewählt werden? Als Sprachprobleme:*

$$\textit{CLIQUE-COVER} = \left\{ \langle G, k \rangle \;\middle|\; \begin{array}{l} \textit{Es gibt } k \textit{ Cliquen} \\ G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k), \textit{ die zusammen} \\ \textit{alle Knoten von } G \textit{ abdecken, also } V = \bigcup_{i=1}^{k} V_i. \end{array} \right\}$$

bzw.

$$\textit{EXACT-CLIQUE-COVER} = \left\{ \langle G, k \rangle \;\middle|\; \begin{array}{l} \textit{Es gibt } k \textit{ disjunkte Cliquen} \\ G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k),\ V_i \cap V_j = \emptyset \\ \textit{für } i \neq j, \textit{ die zusammen alle Knoten von } G \\ \textit{abdecken, also } V = \bigcup_{i=1}^{k} V_i. \end{array} \right\}.$$

Wir schreiben auch k-CLIQUE-COVER, wenn die Abhängigkeit von k deutlich gemacht werden soll.

Beispiel 13.26. 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?

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

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

○

3.1 Abgeschwächtes CLIQUE-COVER

Es gibt auch eine abgeschwächte Form des *CLIQUE-COVER*-Problems, wo nicht verlangt wird, dass die Cliquen disjunkt sind. Wir könnten dieses Problem *WEAK-CLIQUE-COVER* nennen. Ein Beispiel für dieses abgeschwächte Problem könnte das folgende sein.

Beispiel 13.27. In einem Online-Spiel treten die Spieler jeweils paarweise gegeneinander an. Eine zuverläßige Rangliste der Spieler kann nur innerhalb einer Gruppe erstellt werden, in der jeder Spieler gegen jeden anderen gespielt hat. Trotzdem soll jetzt versucht werden, ein Gesamt-Ranking zu erstellen. Grundlage dafür ist die Auswahl einer Menge von Gruppen, in der jeder gegen jeden gespielt hat. Die Auswahl soll möglichst klein sein, denn man hofft so, die Schnittmengen, in denen systembedingt am ehesten Ranking-Widersprüche auftreten könnten, möglichst klein zu halten. Allerdings stellt es sich als schwierig heraus, eine solche Auswahl zu treffen, warum?

Die Aufgabe verlangt, in dem Graphen bestehend aus den Spielern als Knoten und Kanten, die angeben, ob die zwei Spieler schon gegeneinander gespielt hat, eine Überdeckung mit einer möglichst kleinen Anzahl Cliquen zu finden. Dies ist das NP-vollständige Problem *WEAK-CLIQUE-COVER*. Nach aktuellem Wissen gibt es dafür keinen Algorithmus mit polynomieller Laufzeit. ○

Problem 13.28 (*VERTEX-COVER*)**.** *Gegeben ein Graph G und eine Zahl k, gibt es eine Menge W von k Knoten derart, dass jede Kante von G einen Endpunkt in W hat? Als Sprachproblem:*

$$\textit{VERTEX-COVER} = \left\{ \langle G = (V, E), k \rangle \;\middle|\; \begin{array}{l} \textit{Es gibt eine } k\textit{-elementige Teilmenge } W \subset V \\ \textit{mit } W \cap e \neq \emptyset \textit{ für alle } e \in E. \end{array} \right\}.$$

Wir schreiben auch k-VERTEX-COVER, wenn die Abhängigkeit von k deutlich gemacht werden soll.

Beispiel 13.28. Diktator Revoc Xetrev unterdrückt sein Volk gnadenlos. Schon länger lässt er von der staatlichen Telefongesellschaft erheben, welche Telefonanschlüsse genau miteinander telefonieren. Damit lassen sich hervorragend Leute als Regimefeinde entlarven. Nachdem allerdings eine ganze Reihe von hohen Militärs als Landesverräter verurteilt und erschossen worden waren, stellte sich heraus, dass eine Gruppe von inzwischen ebenfalls verurteilten Regimekritikern sich genau dies zu Nutze machten: sie riefen die Militärs gegen deren Willen an. Da es keine Aufzeichnung über den Inhalt der Gespräche gab, konnten die Militärs ihre Unschuld nicht beweisen, mit fatalen Folgen für ihre Karriere.

Diese Affäre hatte das Militär einige der fähigsten Köpfe beraubt, so dass eine zuverläßigere Methode gefunden werden musste, um Regimefeinde zu erkennen. Wenn sich damit gegenüber dem Ausland auch noch ein Anschein von Rechtssystem etablieren ließ, würde das Diktator Xetrev den Zugang zu westlichen Waffenlieferanten etwas erleichtern.

Aus technischen Gründen ist eine lückenlose Überwachung aller Anschlüsse nicht möglich. Die Software kann mit maximal k Anschlüssen umgehen, die abgehört werden können. Diktator Xetrev befiehlt daher, dass innert einer Woche eine Menge von k Anschlüssen definiert werden müsse, so dass jedes Telefongespräch abgehört werden kann.

Diese Ankündigung verursacht erst einmal eine neue Fluchtwelle von Mathematikern und Informatikern. Nach einer Woche wird der erste Experte wegen Befehlsverweigerung erschossen. Im Prozess, der nur 15 Minuten gedauert hatte, hatte er feige behauptet, es sei nicht möglich, in der kurzen Zeit eine entsprechende Menge zu finden. Ja es sei nicht einmal möglich, zu entscheiden, ob es eine solche Menge überhaupt gäbe. Der Staatsanwalt legte ihm das als Sabotage der Pläne des grossen Diktators Xetrev aus und machte kurzen Prozess mit ihm.

Als aber nach einer weiteren Woche erneut ein Saboteur angeklagt wurde, der genau die gleiche Verteidigungstrategie verwendete, obwohl doch mittlerweile bekannt war, dass sie nicht zum Erfolg führen konnte, begann sich Diktator Xetrev zu fragen, ob darin vielleicht ein Körnchen Wahrheit stecken könnte. Könnte es sein, dass es tatsächlich zu schwierig ist, eine solche Anschlussauswahl zu treffen?

Das Problem, welches die Experten lösen sollten, ist das Problem *VERTEX-COVER*. Die Telefonanschlüsse bilden die Knoten eines Graphen, die Kanten verbinden diejenigen Anschlüsse, die tatsächlich miteinander telefonieren. Gesucht ist eine Menge von k Knoten so, dass jedes mögliche Gespräch abgehört werden kann, also jede Kante in einem ausgewählten Knoten endet. ○

4 Probleme über Familien von Mengen (13.4.4)

Problem 13.31 (*SET-COVERING*)**.** *Gegeben die Familie $(S_i)_{i\in I}$ und eine natürliche Zahl k, gibt es eine Teilfamilie $J \subset I$ von k Mengen, $|J| = k$, die die gleiche Vereinigung*

$$\bigcup_{j\in J} S_j = U = \bigcup_{i\in I} S_i$$

haben? Als Sprachproblem:

$$\textit{SET-COVERING} = \left\{ \langle (S_i)_{i\in I}, k\rangle \;\middle|\; \begin{array}{l} \textit{Es gibt eine } k\textit{-elementige Teilfami-} \\ \textit{lie } J \subset I,\ |J| = k, \textit{ mit } \bigcup_{j\in J} S_j = U. \end{array} \right\}.$$

Beispiel 13.31. Der populistische Politiker Tesco Vering versprach während des Wahlkampfes, das Dickicht von Steuererleichterungen, welches für jeden Bürger das Ausfüllen der Steuererklärung zur Qual machte, zu vereinfachen. Nach seinem Amtsantritt als Ministerpräsident stellte er jedoch fest, dass jeder seiner Wähler von irgendeiner der Vergünstigungen profitierte. Um seine Wiederwahl nicht zu gefährden, beauftragte er daher eine Kommission eine Liste von Vergünstigungen aufzustellen, so dass jeder seiner Wähler immer noch von einer Vergünstigung profitiert. Niemand soll im Wahlkampf sagen können: "Vering hat mir alle Vergünstigungen weggenommen!" Kurz vor Beginn des nächsten Wahlkampfs war die Kommission jedoch immer noch nicht fertig, so dass sich Vering als neues Wahlkampfthema auf den Kampf gegen die allgemeine Unfähigkeit der Verwaltung verlegte, um davon abzulenken, dass er das letzte Wahlversprechen nicht eingelöst hatte. Trotzdem wurde er nicht wiedergewählt. Eine detaillierte Analyse der Wahlresultate ergab, dass Vering vor allem die Stimmen der Mathematiker und Informatiker gefehlt hatten. Sie warfen ihm Unverständnis für die Komplexität seiner Aufgabe vor und hielten ihn deshalb für nicht tauglich für ein politisches Amt.

Tatsächlich ist das Programm von Tesco Vering das Problem *SET-COVERING*. Die Steuervergünstigungen sind numeriert von 1 bis n. Sei S_i die Menge der Wähler von Vering, die von Vergünstigung i profitieren. Der Auftrag der Kommission war, eine Teilmenge $I = \{i_1, i_2, \ldots, i_k\}$ zu finden, so dass

$$\bigcup_{i=1}^{n} S_i = \bigcup_{i \in I} S_i$$

ist. Dieses Problem ist NP-vollständig, nach aktuellem Wissen gibt es also keinen polynomiellen Algorithmus um zu entscheiden, ob es überhaupt eine Lösung des Problems gibt. ○

Problem 13.34 (*EXACT-COVER*)**.** *Gegeben die Familie $(S_i)_{i \in I}$, gibt es eine Teilfamilie $J \subset I$ von disjunkten Mengen $S_k \cap S_l = \emptyset \; \forall k, l \in J, k \neq l$ mit der gleichen Vereinigung*

$$\bigcup_{j \in J} S_j = U = \bigcup_{i \in I} S_i?$$

Als Sprachproblem:

$$\textit{EXACT-COVER} = \left\{ (S_i)_{i \in I} \;\middle|\; \begin{array}{l} \textit{Es gibt eine Teilfamilie } J \subset I \textit{ paarweise} \\ \textit{disjunkter Mengen mit der gleichen Vereinigung.} \end{array} \right\}.$$

Beispiel 13.34. In einem Straflager müssen Sträflinge in diversen Bauprojekten der Umgebung arbeiten. Viele der Sträflinge sind jedoch eher unbegabt und daher kaum fähig, diese Arbeiten zur Zufriedenheit der Lagerleitung auszuführen. Jeder Sträfling ist für mindestens eines der Projekte nicht geeignet. Der sadistische Lagerkommandant ersinnt sich dazu noch einen teuflischen Plan, wie er möglichst viele Sträflinge wegen Sabotage bestrafen kann. Er gibt seinen Schergen folgende Regeln für die Zuteilung der Sträflinge zu den Projekten:

1. An jedem Tag soll nur an einem Teil der Projekte gearbeitet werden.
2. Alle für das Projekt ungeeigneten Sträflinge arbeiten an diesem Tag an dem Projekt.
3. Trotzdem muss eindeutig bestimmt sein, an welchem Projekt ein Sträfling arbeiten soll.
4. Jeder Sträfling muss eingesetzt sein.

Im Lagerapport am nächsten Tag rücken die Aufseher damit heraus, dass sie keine Zuteilung gefunden hätten, die diesen Regeln entspricht, und dass deshalb an diesem Tag nichts gearbeitet worden sei. Der Lagerkommandant schäumt vor Wut und droht damit die Aufseher eigenhändig zu erschießen, wenn sie sich weiterhin seinen Befehlen widersetzen würden.

Der Lagerkommandant hat seinen Aufsehern ein Problem gestellt, welches äquivalent zu *EXACT-COVER* ist. Die Menge aller Sträflinge ist U. Die Projekte seien numeriert von

1 bis n. Die Menge $S_i \subset U$ besteht aus den Sträflingen, die für das Projekt i ungeeignet sind. Weil jeder Sträfling für mindestens ein Projekt ungeeignet ist, ist

$$U = \bigcup_{i=1}^{n} S_i.$$

Gesucht wird jetzt eine Teilmenge von Projekten $I = \{i_1, \ldots, i_m\} \subset \{1, \ldots, n\}$, so dass jeder Sträfling auf einem Projekt arbeitet:

$$\bigcup_{j=1}^{m} S_{i_j} = U$$

aber kein Sträfling mehr als einem Projekt zugeteilt wird:

$$S_{i_j} \cap S_{i_k} = \emptyset \qquad \forall j \neq k.$$

Dieses Problem ist NP-vollständig, nach aktuellem Wissen gibt es also keinen polynomiellen Algorithmus, mit dem entschieden werden könnte, ob es überhaupt eine Lösung hat. ○

Problem 13.36 (*HITTING-SET*)**.** *Gibt es eine Menge $W \subset U$ derart, dass W mit jeder Menge S_i nur ein Element gemeinsam hat? Als Sprachproblem*

$$\textit{HITTING-SET} = \left\{ \langle (S_i)_{i \in I} \rangle \;\middle|\; \begin{array}{l} \textit{Es gibt eine Teilmenge } W \subset U = \bigcup_{i \in I} S_i \\ \textit{derart, dass } |W \cap S_i| = 1 \textit{ für alle } i \in I. \end{array} \right\}.$$

Beispiel 13.36. Die Lobby-Organisation Bribe-A-Scientist will eine Menge von Wissenschaftlern bestechen. Sie sucht Fachleute, die zu einer Anzahl wichtiger Themen im Sinne von BAS Stellung nehmen können. Bei ihrer Suche stellt BAS fest, dass einige Wissenschafter zu mehreren Themen Stellung nehmen könnten. BAS will verhindern, dass der Eindruck entsteht, gewisse Leute ließen sich für jede Aussage kaufen. Die ausgewählten Wissenschafter sollen im jeweiligen Fachgebiet die einzigen sein, die in Frage kommen. Ist so eine Auswahl effizient auffindbar?

Sei S_i Menge der Wissenschafter, die zum Thema i Stellung im Sinne von BAS beziehen können. Gesucht wird eine Auswahl H von Wissenschaftern, so dass jeder Wissenschafter genau ein Fachgebiet hat, für das er zur Stimme von BAS werden soll, also $|H \cap S_i| = 1$. Dies ist das NP-vollständige Problem *HITTING-SET*, welches nach heutigem Wissen nicht in polynomieller Zeit gelöst werden kann. ○

5 FEEDBACK-NODE-SET und FEEDBACK-ARC-SET (13.4.5)

Problem 13.39 (*FEEDBACK-NODE-SET*)**.** *Gibt es zu einem gerichteten Graphen G und einer Zahl k eine Menge $W \subset V$ von k Knoten derart, dass jeder geschlossene Zyklus durch einen dieser Knoten geht?*

Beispiel 13.39. In einer großen Fabrik bringen Transportroboter die Teile zu den Verarbeitungsstationen. Die Roboter fahren dabei entlang vorgegebener Bahnen, die sie mit Hilfe von Bodenmarkierungen einhalten. Die Roboter sind immer für die gleichen Prozesse im Einsatz und fahren daher mehrmals am Tag immer die gleichen Runden ab.

Die Wartungsvorschriften verlangen, dass die Roboter täglich einmal einem Sicherheitscheck unterzogen werden, der nur wenige Minuten dauert. Wegen des weitläufigen Fabrikgeländes lohnt es sich nicht, alle Roboter jeden Tag an eine zentrale Kontrollstelle zu fahren, wo sie überprüft werden können. Sie würden viel zu lange ausfallen. Daher wurde entschieden, an einigen Knoten des Netzes der Roboter-Fahrwege Prüfstellen einzurichten.

Sachbearbeiter Heiri Muster soll herausfinden, ob das Budget reicht, um genügend Prüfstellen einzurichten, so dass jeder Roboter täglich geprüft werden kann. Drei Monate später wird er wegen Burnout freigestellt.

Das Budget reicht für die Ausrüstung von k Prüfstellen. Die Roboter bewegen sich auf dem gerichteten Graphen der Bodenmarkierungen. Sie fahren die Zyklen des Graphen ab. Gesucht wird jetzt eine Menge von k Netzknoten für die Prüfstellen, so dass jeder Zyklus des Graphen durch eine dieser Prüfstellen verläuft. Dies ist das NP-vollständige Problem *FEEDBACK-NODE-SET*. Nach aktuellem Wissen gibt es keinen Algorithmus mit polynomieller Laufzeit, der entscheiden könnte, ob das Problem überhaupt eine Lösung hat. ○

Problem 13.39 (*FEEDBACK-ARC-SET*)**.** *Gibt es zu einem gerichteten Graphen G und einer Zahl k eine Menge $F \subset E$ von k Kanten derart, dass jeder geschlossene Zyklus eine dieser Kanten enthält?*

Beispiel 13.39. Kurz nachdem die Pläne für das Prüfstellennetz aus dem vorangegangenen Beispiel von der Geschäftsleitung bewilligt worden waren, änderte der Hersteller der Roboter bei einem Softwareupdate die Wartungsvorschriften. Die Prüfung erfolgt jetzt nicht mehr im Stehen, sondern während der Fahrt. Durch diese dynamische Prüfung werden Probleme erkennbar, die bei der statischen Prüfung nicht gefunden werden konnten. Heiri Musters Nachfolger wird beauftragt, die Pläne entsprechend anzupassen. Drei Monate später ist auch er wegen Burnout freigestellt.

Das Problem wurde durch die Umstellung zu einer Instanz des Problems *FEEDBACK-ARC-SET* modifiziert. Es müssen k Kanten gefunden werden, so dass jeder Zyklus eine diese Kanten durchläuft. Auch dieses Problem ist NP-vollständig, und dürfte daher für ein kompliziertes Streckennetz nicht nur einen Sachbearbeiter überfordern. ○

6 Probleme mit einer Aufteilung in zwei Mengen (13.4.7)

Problem 13.43 (*PARTITION*)**.** *Gibt es eine Unterteilung einer Liste von ganzen Zahlen in zwei disjunkte Listen mit gleicher Summe? Als Sprachproblem:*

$$PARTITION = \left\{ S = [s_1, \ldots, s_n] \;\middle|\; \begin{array}{l} \textit{Es gibt eine Teilliste } T = [s_{i_1}, \ldots, s_{i_k}] \subset S\textit{, deren} \\ \textit{Elemente die gleiche Summe haben wie die} \\ \textit{Elemente der Liste } \overline{T} = S \setminus T. \end{array} \right\}.$$

Beispiel 13.43. Ein reicher König ist gestorben, und hinterlässt seinen beiden Söhnen, eineiigen Zwillingen, ein grosses Vermögen aus Schlössern und Burgen. Die Berater des Königs werden zusammengerufen um darüber zu entscheiden, wie die Güter gerecht an die beiden Söhne verteilt werden können. Warum sind die Berater auch nach einem Jahr noch nicht zur Einsicht gelangt, ob eine gerechte Teilung überhaupt möglich ist?

Bezeichnen wir mit c_i den Wert der Immobilie i, dann besteht die Aufgabe der Berater darin eine Aufteilung der Menge A aller zur Teilung stehenden Immobilien in zwei Mengen I und $A \setminus I$ zu vollziehen, die Immobilien jeder Menge jeweils einem der Söhne zugeteilt werden sollen. Natürlich sollen die beiden Mengen gleichen Gesamtwert haben. Der Gesamtwert ist jeweils die Summe der einzelnen Werte, gesucht ist also $I \subset A$ so dass

$$\sum_{i \in I} c_i = \sum_{i \in A \setminus I} c_i$$

Dies ist das Problem *PARTITION*, es ist NP-vollständig. Daher gibt es nach aktuellem Wissen keinen polynomiellen Algorithmus, der es entscheiden könnte. ○

Problem 13.45 (*MAX-CUT*)**.** *Gegeben ein Graph $G = (V, E)$ mit einer Gewichtsfunktion $\varphi\colon E \to \mathbb{Z}$ der Kanten und einer ganzen Zahl $w \in \mathbb{Z}$, gibt es eine Aufteilung der Knotenmenge $V = V_1 \cup V_2$ in zwei disjunkte Teilmengen, so dass das Gewicht der Kanten, die V_1 und V_2 verbinden, mindestens w ist:*

$$\varphi(V_1, V_2) = \sum_{v_1 \in V_1, v_2 \in V_2, e = \{v_1, v_2\} \in E} \varphi(e) \geq w?$$

Beispiel 13.45. Nord- und Süd-Computanien haben sich nach langem Krieg wieder vereinigt. Ein Förderungs-Programm versucht die wirtschaftliche Zusammenarbeit der ehemals verfeindeten Staaten dadurch zu fördern, dass Transporte, die ehemalige Demarkationslinie überschreiten, großzügig subventioniert werden. So großzügig, dass die Auftraggeber, wenn sie denn unterbezahlte Fahrer aus den Nachbarländern dafür anheuern, sogar mehr an Subventionen erhalten können als die eigentliche Fahrt kostet. Eine Firma möchte sich daran eine goldene Nase verdienen, und verlangt, dass die Produktionsstandorte so verteilt werden, dass möglichst viele grenzquerende Transporte nötig sind. Das muss schnell geschehen, bevor das Förderprogramm eingestellt wird. Hat dieser Plan Chancen?

Die Produktionsstationen bilden einen Graphen, dessen Kanten anzeigen, ob zwischen den beiden Stationen ein Transport notwendig ist. Jeder Kante ist der mögliche Gewinn zugeordnet, der winkt, wenn der Transport grenzquerend durchgeführt werden kann. Gesucht wird jetzt eine Aufteilung der Produktionsstationen auf die beiden Landesteile Nord- und Süd-Computanien, so dass der Gewinn aus Subventionen die Ziele W des Managements übersteigen. Dies ist das Problem *MAX-CUT*, welches NP-vollständig ist und daher nach gegenwärtigem Wissen nicht von einem Algorithmus in polynomieller Zeit gelöst werden kann. ○

7 3D-MATCHING (13.4.8)

Problem 13.47 (*3D-MATCHING*)**.** *Gegeben sind drei endliche Mengen X, Y und Z mit gleich vielen $n = |X| = |Y| = |Z|$ Elementen und $T \subset X \times Y \times Z$. Gibt es eine n-elementige*

Teilmengen $M \subset T$ derart, dass keine zwei Tripel von M in einer Koordinate übereinstimmen?

Beispiel 13.47. In der Provinz Gnichtam De herrscht grosse Wohnungsnot. Eine Analyse durch das Innenministerium ergab, dass die Ursache die vielen Singles sind, die jeweils alleine eine gemäss Plan der staatlichen Wohnungsbaubehörde für Familien vorgesehene Wohnung belegen. Würde man die Singles paarweise den Wohnungen zuteilen, würden die Wohnungen reichen.

Daher beschliesst der Familienminister ein Programm zur Zwangsverheiratung der Singles und beauftragt seine Behörde mit der Ausarbeitung einer Zuordnung, wer mit wem verheiratet werden soll und wo das frisch verheiratete Paar wohnen soll. Natürlich ist nicht jede Kombination aus einem Mann, einer Frau und einer Wohnung akzeptabel, immerhin müssen Arbeitswege vernünftig bleiben, die Bürger sollen ja als Arbeiter weiterhin produktiv bleiben um die hohen Steuern bezahlen zu können. Das Ministerium beginnt also eine Liste von möglichen Zwangsfamilien zu erstellen.

Nach einem Jahr wird der Familienminister ungeduldig, denn obwohl die genannte Liste nach wenigen Wochen fertig war, konnte daraus immer noch keine geeignete Familienplanung abgeleitet werden, die Fachleute wissen noch nicht einmal, ob es überhaupt möglich ist, eine Planung zu finden, welche alle geforderten Rahmenbedingungen erfüllt.

Das Problem, welches das Familienministerium der Provinz Gnichtam De lösen soll, ist *3D-MATCHING*. Es gibt n unverheiratete Frauen und Männer, und n mögliche Wohnungen. Die ursprünglich erstellte Liste ist eine Menge von Tripeln (x, y, z), wobei die Zahlen x, y und z aus der Menge $T = [n] = \{1, 2, 3, \ldots, n\}$ stammen. Aus dieser Teilmenge $U \subset T \times T \times T$ soll jetzt eine Teilmenge $W \subset U$ von genau n Tripeln ausgewählt werden, so dass jeder Mann, jede Frau und jede Wohnung in genau einem Tripel vorkommt. Dieses Problem ist NP-vollständig, nach heutigem Wissen gibt es keinen polynomiellen Algorithmus, mit dem entschieden werden könnte, ob es überhaupt so eine Teilmenge W gibt. ○

8 Kombinatorische Optimierungsprobleme (13.4.9)

Problem 13.48 (*STEINER-TREE*)**.** *Gibt es zu einem Graphen $G = (V, E)$ mit einer Gewichtsfunktion $w\colon E \to \mathbb{Z}$, einer Teilmenge $R \subset V$ der Knoten und einer Zahl k einen Teilbaum $T = (V_1, E_1) \subset G$ des Graphen, der alle Knoten von R enthält und dessen Kantengewicht*

$$\sum_{e \in E_1} w(e) \leq k$$

ist?

Beispiel 13.48. Eine Bank hat bis anhin ihre Filialen immer vom Hauptsitz aus mit Geld versorgt. Die dafür nötigen Hochsicherheitstransporte sind jedoch teuer, daher soll ein neues Transportmodell gefunden werden. Neu sollen auch Transporte zwischen Filialen möglich sein. Am Hauptsitz werden dazu die Beträge für mehrere Filialen verladen, die Filialen müssen die Lieferungen auseinander nehmen und auf neue Transporte verladen.

Das Management will wissen, ob sich mit dieser Methode die Kosten unter den Betrag k senken lassen.

Für jedes Paar von Filialen sind die Kosten eines Transportes bekannt. Der billigste Logistikplan wird einen Baum verwenden. Wäre das gewählte Verfahren nicht ein Baum, könnte man durch Umverteilung eines Teils des Geldes nämlich einen Transport weglassen, wodurch die Kosten geringer würden. Es liegt also eine Instanz des Problems *STEINER-TREE* vor, welches NP-vollständig ist und daher nach allem, was wir wissen, nicht von einem Algorithmus mit polynomieller Laufzeit gelöst werden kann. ○

Problem 13.49 (*SEQUENCING*)**.** *Die Komponenten der drei Vektoren*

$$\begin{aligned} \textit{Laufzeiten:} \quad & T = (T_1, \ldots, T_p) \in \mathbb{Z}^p \\ \textit{Deadlines:} \quad & D = (D_1, \ldots, D_p) \in \mathbb{Z}^p \\ \textit{Strafen:} \quad & P = (P_1, \ldots, P_p) \in \mathbb{Z}^p \end{aligned}$$

beschreiben je einen Job. Job i hat Laufzeit T_i, sollte zur Zeit D_i abgeschlossen sein und kostet die Strafe P_i, wenn er nicht rechtzeitig fertig ist. Die Jobs sollen nacheinander laufen. Sobald ein Job fertig ist, wird der nächste gestartet. Ist es möglich, die Reihenfolge der Ausführung der Jobs so zu planen, dass die entstehenden Strafen $\leq k$ sind?

Beispiel 13.49. Wenn Züge sich verspäten oder ausfallen hat dies oft weitreichende Auswirkungen auf viele weitere Verbindungen. Aus diesem Grund muss die SBB zum Beispiel dem Zürcher Verkehrsverbund Strafe für verspätete ZVV-Züge bezahlen. Aber auch dort, wo keine direkte Strafzahlung fällig wird, entsteht mindestens ein Image-Schaden. Ursache der Abhängigkeiten ist oft die Tatsache, dass ein Streckenabschnitt zu einer bestimmten Zeit nur von einem Zug durchfahren werden kann. Die Züge sind verschieden schnell, die benötigte Durchfahrtszeit variiert.

Die Kosten sollen jetzt dadurch gesenkt werden, dass den Disponenten ein Computerprogramm zur Verfügung gestellt wird, mit dem sie nach einem Zwischenfall auf einer Strecke die optimale Reihenfolge der Züge festlegen können, mit der der geringste Schaden entsteht. Hat dieses Projekt Aussicht auf Erfolg?

Nur beschränkt. Die Durchfahrzeit des Zugs mit Nummer i durch die Strecke ist t_i. Wenn der Zug später als zur Zeit d_i ankommt, ist die Strafe s_i fällig. Werden die Züge in der durch die Permutation π permutierten Reihenfolge $\pi(1), \pi(2), \ldots, \pi(n)$ abgefertigt, beträgt die für Zug Nummer j anfallende Strafe:

$$\vartheta(t_{\pi(1)} + t_{\pi(2)} + \cdots + t_{\pi(j)}) s_{\pi(j)}$$

Die Gesamtstrafe ist daher

$$\sum_{j=1}^{n} \vartheta(t_{\pi(1)} + t_{\pi(2)} + \cdots + t_{\pi(j)}) s_{\pi(j)}.$$

Das Problem ist somit äquivalent zu *SEQUENCING*, einem NP-vollständigen Problem, für das es nach aktuellem Wissen keinen polynomiellen Algorithmus steht. Das Projekt wäre besser beraten, statt nach der optimalen Abfolge nach einer angenähert optimalen Abfolge zu suchen. ○