Lösungsstrategien für NP-schwere Probleme der Kombinatorischen Optimierung

— Übungsblatt 1 —

Walter Stieben (4stieben@inf) Tim Reipschläger (4reipsch@inf) Louis Kobras (4kobras@inf) Hauke Stieler (4stieler@inf)

Abgabe am: 18. April 2016

Aufgabe 1.1

Aufgabe 1.1.1 4D-MATCHING

Eingabe: Disjunkte Mengen W, X, Y und Z mit $|W| = |X| = |Y| = |Z| = n$ sowie eine Menge $Q \subseteq W \times X \times Y \times Z$ von Quadrupeln.

Frage: Gibt es eine Menge von n Quadrupeln in Q, so dass jedes Element aus $W \cup X \cup Y \cup Z$ in genau einem dieser Quadrupel vorkommt?

Um zu zeigen das 4D-MATCHING *NP*-Vollständig ist müssen wir zuerst zeigen, dass 4D-MATCHING in *NP* liegt, indem wir einen Verifizierungsalgorithmus für 4D-MATCHING finden.

Verifizierungsalgorithmus

Der Algorithmus bekommt als Zertifikat die Quadrupelmenge Q übergeben, welche die n Quadrupel enthält, die zusammen genau jedes Element der Mengen einmal enthalten. Es wird zuerst geprüft ob $|Q| \geq n$ ist und anschließend wird jedes Element der Mengen W, X, Y und Z überprüft, ob es in genau einem der Quadrupel aus Q enthalten ist. Ist das Element in keinem Quadrupel oder in mehr als nur einem Quadrupel handelt es sich um eine Nein-Instanz. Falls alle Elemente in den Quadrupeln aus Q genau ein mal vorkommen, handelt es sich um eine Ja-Instanz.
Das Überprüfen ist in Polynomialzeit möglich, da für jedes der $4n$ Elemente nur jeweils n Quadrupel durchgegangen werden müssen, demnach $4n^2$ Überprüfungen durchgeführt werden.

Damit haben wir einen Verifizierungsalgorithmus gefunden und gezeigt das 4D-MATCHING in *NP* liegt. Nun müssen wir noch ein Problem in Polynomialzeit auf 4D-MATCHING reduzieren, dass bereits als *NP*-Vollständig bekannt ist. Wir nehmen hier 3D-MATCHING.

Reduktion

Um 3D-MATCHING auf 4D-MATCHING zu reduzieren müssen wir die Eingabe von 3D-MATCHING in Polynomialzeit umformen sodass wir sie in 4D-MATCHING einspeisen können und jede Ja-Instanz von 3D-MATCHING eine Ja-Instanz von 4D-MATCHING ist. Gleiches gilt für Nein-Instanzen.
Dazu nehmen wir die Eingabe vom 3D-MATCHING und fügen den Mengen X, Y und Z eine Menge W hinzu, für die gilt, dass sie zum einen disjunkt zu den Mengen X, Y und Z ist und $|W| = n$. Wie die Elemente genau aussehen ist nicht wichtig, solange die Elemente nicht dafür sorgen, dass die eben genannten Kriterien von W verletzt werden.

Wir haben somit nun eine Menge $W = \{w_1, ..., w_n : w_i \notin (X \cup Y \cup Z)\}$ mit $i \in \{1, 2, 3, ..., n\}$.

Nun müssen wir noch die gegebene Tripelmenge T so umformen, dass wir eine Quadrupelmenge T' bekommen, die wir für die Eingabemenge Q von 4D-MATCHING benutzen können.
Dafür hängen wir an jedes Tripel jeweils ein Mal alle Elemente der Menge W an, sodass jedes vorhandene Tripel n unterschiedliche Quadrupelformen hat, die sich nur durch das Element der Menge W unterscheiden. Diese Quadrupel sind nun unsere Menge T' die wir in 4D-MATCHING als Eingabe für die Menge Q nehmen.
Damit haben wir nun die Eingabe von 3D-MATCHING in eine Eingabe für 4D-MATCHING umgewandelt und das in Polynomialzeit.

Begründung

Führen wir die Umformungen die im Reduktionsteil beschrieben wurden durch, haben wir aus einer 3D-MATCHING-Instanz eine 4D-MATCHING-Instanz gemacht. Der Grund warum eine Ja-Instanz nach dem Umformen immer noch eine Ja-Instanz ist und auch Nein-Instanzen weiterhin Nein-Instanzen sind ist der, dass durch das vorher beschriebene Bauen der Quadrupel die Matchingdimension nur um 1 erhöht wird.

Angenommen wir haben die Mengen X, Y und Z und angenommen $|T| = n$. Wir wissen, dass sofern nicht alle X Elemtene in den Tupeln vorkommen, dass es sich um eine Nein-Instanz handelt. Gleiches Gilt für die Mengen Y und Z.
Haben wir nun den Fall, dass eine Nein-Instanz vorliegt, dann wird durch das hinzufügen der W-Elemente nichts an der Tatsache verändert, dass die Tupel eine Menge nicht ganz abdecken, wie bei dem eben erwähnten Sachverhalt die Menge X.
Liegt eine Ja-Instanz vor, dann bleibt sie auch weiterhin eine Ja-Instanz, da ja für ein Tripel jeweils alle Kombinationen mit den Elementen aus der neuen Menge W vorkommen. Demnach können wir uns beim Ersten Tripel aussuchen welchen W-wert wir nehmen und für die restlichen Tripel bleiben noch die restlichen W-Werte. Somit können wir den n Tripeln die n unterschiedlichen W-Werte zuweisen und habe anschließend noch eine Ja-Instanz.
Angenommen wir haben die Mengen X, Y und Z und angenommen $|T| < n$. Wir Wissen, dass keine Ja-Instanz existieren kann, da min 1 Element aus jeder Menge nicht abgebildet werden kann. Das ganze bleibt wie vorher schon beschrieben auch nach dem Umformen erhalten.

Angenommen wir haben die Mengen X, Y und Z und angenommen $|T| > n$. Liegt eine Ja-Instanz vor, so können n Elemente aus T den Tripeln hinzugefügt werden. Allerdings sind dann noch Elemente aus T übrig, die nicht abgedeckt werden. Somit liegt nach Reduktion keine Ja-Instanz vor. Bei einer Nein-Instanz verändert das Hinzufügen von Elementen einer neuen Menge nichts daran, dass die alten Mengen nicht vollständig abgedeckt sind. Wäre es durch Hinzufügen eines $(n+1)$-ten Elementes aus T möglich, ein komplettes Matching so erreichen, so hätte es dieses bereits vorher geben müssen. Also liegt auch hier eine Nein-Instanz vor.

Aufgabe 1.1.2

Zuallererst wollen wir nachweisen, dass 2-CLIQUE nicht in *NPC* liegt, indem wir dafür einen polynomiellen Algorithmus angeben, der das Problem löst. Dieser Algorithmus muss nur überprüfen, ob die Kantenmenge des Graphen leer ist. Wenn sie leer ist, gibt es keine 2-CLIQUE, andernfalls gibt es mindestens eine Kante, also eine Menge von 2 Knoten, die untereinander "alle" verbunden sind und somit eine 2-CLIQUE. Dies ist in einem Schritt, also polynomiell möglich.

Für alle k-CLIQUE n mit $k \geq 3$ weisen wir im Folgenden nach, dass sie in *NPC* liegen.

Um k-CLIQUE$\in$ *NP* zu zeigen, geben wir zuerst einen Algorithmus an, der ein Zertifikat für k-CLIQUE in polynomieller Zeit verifiziert.
Dieser Algorithmus nimmt das Zertifikat, also eine Menge von Knoten, die eine k-CLIQUE sein sollen, entgegen und zählt die zunächst. Sind es wirklich k Knoten, prüft der Algorithmus nun zu jedem angegebenen Knoten, ob von diesem zu allen anderen angegebenen Knoten auch Kanten bestehen. Wenn dies der Fall ist, akzeptiert der Algorithmus, ansonsten lehnt er ab. Das zählen der Knoten sind

k Schritte und das Prüfen der Kanten $k^2 - k$ Schritte (k Schritte entfallen, weil die k Knoten keine Kanten zu sich selbst besitzen müssen), also liegt der Algorithmus insgesamt in $\mathcal{O}(k^2) \in P$. Damit gilt dann auch k-CLIQUE$\in$ *NP*.

Um k-CLIQUE$\in$ *NPC* zu zeigen, müssen wir von einem als *NP*-Vollständig nachgewiesenen Problem aus darauf reduzieren. Als dieses Ausgangsproblem wählen wir INDEPENDENT SET.

Die Reduktion damit ist sehr kurz: Sei $\langle G, k\rangle$ mit $G = (V, E)$ eine Eingabeinstanz von INDEPENDENT SET. Eine Instanz für $\langle G', k'\rangle$ k-CLIQUE erhalten wir, indem wir $k' = k$ setzen (1 Schritt) und G' konstruieren, indem wir alle Kanten in G invertieren (V^2 Schritte). Diese Reduktion liegt in $O(V^2)$ und ist damit polynomiell.

Die Korrektheit folgt entsprechend einfach:

- Angenommen, es gibt einen vollständigen Graphen der Größe k in k-CLIQUE. Dann gibt es k Knoten die paarweise verbunden sind. Diese Kanten müssen alle bei der Reduktion entstanden und vorher nicht existent gewesen sein. D.h. die Knoten waren alle paarweise nicht verbunden und somit eine unabhängige Menge der Größe k.
- Angenommen, es gibt eint unabhängige Menge der Größe k in INDEPENDENT SET. Dann gibt es k Knoten die paarweise nicht verbunden sind. Nach der Reduktion werden diese dann paarweise verbunden sein und eine k-CLIQUE bilden.

□

Aufgabe 1.2

Aufgabe 1.2.1 Zeigen, dass HITTING SET$\in$ *NPC* gilt

Aufgabe 1.2.1.a Zeigen, dass HITTING SET$\in$ *NP* gilt

Um zu zeigen, dass HITTING SET$\in$ *NP* gilt muss ein Verifikationsalgorithmus ein Zertifikat in polynomieller Zeit entscheiden können.

Die Bedingung, dass $|H| \leq k$ gilt, ist trivial und geht natürlich in mit einer Laufzeit von $\mathcal{O}(|H|)$.

Um zu überprüfen ob H wirklich ein HITTING SET ist, geht man für jedes $h \in H$ durch jedes B_i mit $0 \leq i \leq m$ und prüft ob h in einem B_i vorkommt. Wenn dem so ist, ist H ein HITTING SET, ansonsten nicht.
Die Laufzeit ist dabei polynomiell in der Anzahl der Teilmengen B_i. Für alle $|H|$-Elemente aus H geht man jedes Element aus jedem B_i durch, was maximal n viele sind. Dadurch hat man die Laufzeit $n \cdot m \cot |H|$.

Aufgabe 1.2.1.b Reduktion von $3-$SA auf HITTING SET angeben

Gegeben sei eine Eingabe für SAT, also eine Aussagenlogische Formel A mit n vielen Literalen $x_0, \ldots, x_n$ in k vielen Klauseln zu je bis zu drei Literalen.

Um eine Eingabe von HITTING SET zu erzeugen, benötigt man eine Menge an Zahlen M, Teilmengen $B_0, \ldots, B_i$ und eine Schranke $k \in \mathbb{N}$.
Zu jedem Literal aus A wird nun eine Zahl zugeordnet, welche dann in M enthalten ist (dadurch ist $|M| = n$). Hier wird immer der Index des Literals herangezogen, also für x_i ist es i, für $\neg x_i$ ist es $-i$. Wichtig ist dabei, dass keine Zahlen mehrfach vorkommen, auch nicht bei x_i und $\neg x_i$.
Die Schranke k ist die Anzahl an Literalen, also $k = n$. Diese Konstruktion ist jedoch noch nicht vollständig, denn wir benötigen noch ein Gadget um eine Korrekte Abbildung von Ja- und Nein-Instanzen zu gewährleisten.

Als Gadget fügt man n weitere Mengen zu den bisherigen Teilmengen $B_0, \ldots, B_i$ hinzu. Nennen wir sie $B_{i+1}, \ldots, B_{i+n}$. In diesen Mengen ist jeweils die Zahl eines Literals, sowie die seines Komplements enthalten, also z.B. enthält B_{i+2} die Zahlen der Literale x_2 und $\neg x_2$, also ist $B_{i+2} = \{2, -2\}$.

Nun bilden die Mengen M, $B_0, \ldots, B_{i+n}$ und k die Eingabe für das HITTING SET Orakel.

Aufgabe 1.2.1.c Korrektheit für "⇒"

Behauptung: A besitzt eine erfüllende Belegung, dann gibt es für $B_0, \ldots, B_{i+n}$ ein HITTING SET.

Seien $x_0^*, \ldots, x_i^*$ die Belegungen für die jeweiligen Literale $x_0, \ldots, x_i$ aus A, welche A erfüllen. In den Dazugehörigen Mengen $B_0, \ldots, B_i$ wählt man nun aus jeder Teilmenge das Literal aus, welches für die erfüllende Belegung benutzt wurde. Dabei wird aus jeder Teilmenge ein Element ausgewählt, da auch bei jeder Klausel aus A ein Literal benutzt wird im A zu erfüllen.

Aus dem Gadget $B_{i+1}, \ldots, B_{i+n}$ wählt man das Element aus, welches man bereits aus $B_0, \ldots, B_i$ gewählt hat, wodurch die Anzahl der ausgewählten Elemente nicht über die Schranke k hinaus wachsen kann.

Die ausgewählten Elemente bilden nun ein HITTING SET, da aus jeder Teilmenge mindestens ein Element ausgewählt wurde. Somit ist die Behauptung gezeigt.

Aufgabe 1.2.1.d Korrektheit für "⇐"

Behauptung: Gibt es für $B_0, \ldots, B_{i+n}$ ein HITTING SET, dann besitzt A eine erfüllende Belegung.

Seien Menge M eine Menge an Zahlen und $B_0, \ldots, B_i$ maximal dreielementige Teilmengen mit verschiedenen Elementen aus M und $B_{i+1}, \ldots, B_{i+n}$ Teilmengen mit je zwei zueinander Komplementären Elementen. Die Mengen $B_{i+1}, \ldots, B_{i+n}$ sind dabei paarweise disjunkt. Außerdem ist eine Menge H mit $H \subseteq M$ gegeben, welche auch als HITTING SET bezeichnet wird (es existiert für H die bekannte Eigenschaft, dass $H \cap B_i = \emptyset$ für alle $i \in \{1, \ldots, m\}$ gilt).

Für jede Menge L aus $B_{i+1}, \ldots, B_{i+n}$ nimmt man nun das positive Element e heraus. Nun wählt man dazu ein positives (also nicht negiertes) Literal x und wählt passend dazu für das jeweilige Komplement von e das negierte Literal, also $\neg x$. Es wird also e das x und dem Komplement von e das $\neg x$ zugeordnet.

Bildet man nun jede Menge L aus $B_0, \ldots, B_i$ auf eine Disjunktion von den gerade gewählten Literalen ab und verbindet man diese mit Konjunktionen (bildet also eine KNF), hat man eine gültige Aussagenlogische Formel A.

A ist erfüllbar, da H jeweils ein Element oder dessen Komplement enthält. Literale können sich somit nicht widersprechen, was bedeutet, dass es keine Situation gibt in der man sowohl x, als auch $\neg x$ zu `wahr` auswerten muss um A zu erfüllen. Dadurch ist A wiederum erfüllbar.

Aufgabe 1.2.2

Wir haben zum Reduzieren als alternatives Ausgangsproblem VERTEX COVER gewählt.

Sei $\langle G, k\rangle$ mit $G = (V, E)$ eine Eingabeinstanz von VERTEX COVER. Eine Instanz $\langle A, B, k'\rangle$ für HITTING SET erhalten wir, indem wir $k' = k$ setzen (1 Schritt) $A = V$ setzen (1 Schritt) und B konstruieren, indem wir für alle Kanten $(u, v) \in E$ eine Menge $\{u, v\}$ zu B hinzufügen ($|E|$ Schritte).

Diese Reduktion liegt in $\mathcal{O}(()|E|^2)$ und ist damit polynomiell.

Zur Korrektheit:

- Angenommen, es gibt ein Hitting Set der Größe k in HITTING SET. Dann wird in jeder Menge $B_1 \ldots B_n$ mindestens ein Element durch H getroffen.
  Dies ist durch die Reduktion gleichbedeutend damit, dass für VERTEX COVER mindestens ein Endknoten einer jeden Kante in der Menge S ist, darum ist $S = H$ eine Knotenüberdeckung.
- Angenommen, es gibt eine Knotenüberdeckung der Größe k in VERTEX COVER. Dann muss mindestens ein Endknoten von jeder Kante in S sein, weshalb sich mit $H = S$ ein Hitting Set der Größe k ergibt.