# Lösungsstrategien für NP-schwere Probleme der Kombinatorischen Optimierung

— Übungsblatt 3 —

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

Abgabe am: 2. Mai 2016

## Aufgabe 3.1

### Aufgabe 3.1.1

Da $c-$ HITTING SET ein NP-vollständiges Problem ist, muss es hierfür einen deterministischen und polynomiellen Verifikationsalgoritmus geben.

Mit diesem Wissen wollen wir nun einen Algorithmus angeben, der seine eigenen Zertifikate erstellt und diese in Polynomialzeit verifiziert. Den Algorithmus zum Verifizieren entwickeln wir analog zu demjenige, den wir für Hausaufgabe 1.2 aus Aufgabenblatt 1 angefertigt hatten.

Die Bedingung, dass $|H| \leq k$ gilt, ist trivial lässt sich in einer Laufzeit von $\mathcal{O}(|H|)$ prüfen .

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 $c$ viele sind. Dadurch hat man mit $|H| \leq k \leq n$ die Laufzeit $c \cdot m \cdot n$.

Da es diesen Algorithmus gibt, können wir außerdem sagen, dass $c-$ HITTING SET in NP liegt. Wir wollen jetzt alle möglichen Zertifikate für diesen Algorithmus erstellen und wenn eines verifiziert wird, akzeptiert der Algorithmus die Eingabe, ansonsten nicht. Ein einzelnes Zertifikat kann erstellt werden, indem k Elemente aus n ausgewählt werden. Die Funktion f(k) soll dann die Anzahl der Möglichkeiten angeben, diese Elemente auszuwählen. Weiter sind wir nicht gekommen; uns fehlte eine konkrete Idee, die Anzahl der Zertifikate nur mit k und c abzuschätzen, ohne n und m zu benutzen.

### Aufgabe 3.1.2

Folgende Dinge können wir mit Sicherheit sagen:

- Da das Problem NP-Vollständig ist und p(n,m) polynomiell ist, muss f(k) nicht exponentiell sein, weil sonst der ganze Algorithmus polynomiell wäre und dann P = NP gelten würde, was sehr unwahrscheinlich ist.
- Da k der einzige Parameter für die Funktion f ist, muss er in dieser als Exponent vorkommen, damit der Algorithmus exponentiell in der Eingabe ist.

- Wir können in polynomieller Laufzeit ein Zertifikat erstellen und verifizieren. die Auswahl der k Elemente aus A für ein Zertifikat ist durch n beschränkt und der Algorithmus zur Verifikation durch $c \cdot m \cdot n$.

f(k) haben wir zu $f_c(k) = 2^{c \cdot k}$ geschätzt und p(n,m) zu $p(n, m) = c \cdot m \cdot n + n$ ermittelt. Damit ergibt sich dann auch die Laufzeit von $\mathcal{O}(f_c(k) \cdot p(n, m))$

## Aufgabe 3.1.3

Wir wollen zeigen, dass $2-$ HITTING SET mit dem kleinsten erlaubten c NP-vollständig ist und dann begründen, warum $c+1-$ HITTING SET ebenfalls NP-vollständig sein muss, um insgesamt zu zeigen, warum $c-$ HITTING SET NP-vollständig ist. Dies ging leider nicht besonders kurz. Für den Beweis, dass $c-$ HITTING SET in NP liegt, siehe Aufgabenteil a.

Hier zur Wiederholung unsere Lösung für Aufgabe 1.2.b.) von Aufgabenblatt 1 :

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 $O(|E|)$ und ist damit polynomiell.

Zur Korrektheit:

- Angenommen, es gibt ein Hitting Set der Größe $k$ in HITTING SET. Dann wird in jeder Menge $B_1 \dots B_m$ 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.

Wir stellen fest, dass durch die Art der Reduktion alle Mengen $B_1 \dots B_m$ immer genau 2 Elemente haben und identifizieren diese Reduktion somit insbesondere auch als Reduktion auf $2-$ HITTING SET.

### a) Reduktion angeben

Eine Reduktion von $c-$ HITTING SET auf $c+1-$ HITTING SET ist trivial, weil nämlich die Eingabe für $c-$ HITTING SET prinzipiell direkt als Eingabe für $c+1-$ HITTING SET übernommen werden kann. Damit hier nicht das Problem auftritt, dass für $c-$HITTING SET ungültige Eingaben mit einem $|B_i| > c$ als gültige Eingaben in $c+1-$ HITTING SET übernommen werden, müssen wir alle Mengen $B_1 \dots B_m$ vorher durchgehen und die Elemente zählen, was für gültige Eingaben jeweils nur c Elemente sein können, weshalb diese Überprüfung und damit auch die ganze Reduktion in $O(m \cdot c)$ liegt. Schlägt diese Überprüfung fehl, wird direkt `false` ausgegeben, ansonsten wird die Eingabe wie oben beschrieben übernommen.

### b) Korrektheit für "⇒"

Behauptung: Gibt es ein gültiges Hitting Set in $c+1-$ HITTING SET, so ist dieses ebenfalls ein gültiges Hitting Set in $c-$ HITTING SET.

Da wir durch die Reduktion wissen, dass wir keine Mengen $|B_i| > c$ in der Eingabe für $c+1-$ HITTING SET hatten, sind also alle Faktoren bis auf das c gleich geblieben und die lockerere Schranke für c wurde nicht genutzt, demnach muss das gefundene Hitting set auch ein Hitting Set in $c-$ HITTING SET sein.

**c) Korrektheit für "⇐"**

Behauptung: Gibt es ein gültiges Hitting Set in $c-$ HITTING SET, so ist dieses ebenfalls ein gültiges Hitting Set in $c+1-$ HITTING SET.

Ein Hitting Set in $c-$ HITTING SET ist für größere c immer auch ein Hitting Set, da jedes $B_i$ aus $B_1 \dots B_m$ die Eigenschaft $|B_i| \leq c$ und somit automatisch auch $|B_i| \leq c+1$ erfüllt und das Hitting Set, sowie die Menge A nicht selbst von c abhängig sind.

# Aufgabe 3.2

**d)**

Wenn $\forall v \in D : grad(v) = n > k$ gelten soll, dann gibt es zu viele Kanten ($(n)^2$ viele), als dass man diese durch $k$ viele Knoten überdecken könnte. Mit $k$ Knoten mit $grad(v) = n$ kann man maximal $(n) \cdot k$ Kanten überdecken.

Bildet $D$ eine $|D|$-Clique in $G$, so ist $k = |D| - 2$ und bildet eine Ausnahme, da man mit $k$ vielen Knoten zwar rechnerisch genug Kanten abdecken könnte, jedoch zwei Knoten in der $|D|$-Clique übrig bleiben, die gemeinsam wieder eine Kante hätten. Diese Kante würde nicht abgedeckt werden.

Wenn also $|D| > k$ gilt, gibt es keine Knotenüberdeckung in $G$.

**e)**

**Hinrichtung**
Angenommen $G$ hat eine Knotenüberdeckung mit $k$ Knoten.
In $G$ und $G'$ werden nun alle Kanten getroffen, wobei es Kanten von $G$ zu seinem Untergraphen $G'$ gibt. Nimmt man nun aus $G$ eben die $|D|$ vielen Knoten heraus durch die man $G'$ erhält, so gibt es keine Kanten mehr, die von $G$ nach $G'$ führen. Kanten, die nur innerhalb von $G'$ existieren bleiben jedoch bestehen.
Somit hat $G$ eine $k'$ Knotenüberdeckung, wenn es eine $k$ Knotenüberdeckung in $G$ gibt.

**Rückrichtung**
Angenommen $G'$ hat eine Knotenüberdeckung mit $k'$ Knoten.
Fügt man die in $D$ enthaltenen Knoten zu $G'$ hinzu, so hat $G$ genau $k$ viele Knoten, da $k' = k - |D| \Leftrightarrow k = k' + |D|$ gilt. Alle hinzugefügten Knoten nimmt man in die Knotenüberdeckung von $G$ mit auf, somit sind alle zu diesen "neuen" Knoten inzidenten Kanten auch in der Knotenüberdeckung enthalten und es gibt keine Kanten, die nicht getroffen werden.
Somit hat $G$ eine Knotenüberdeckung mit $k$ Knoten, wenn $G'$ eine mit $k'$ vielen Knoten hat.

**f)**

Im folgenden wird angenommen, dass $G'$ genau $k' \cdot (k+1)$ viele Knoten und eine $k'$ Knotenüberdeckung besitzt. Hat $G'$ weniger Knoten, so kann man welche Hinzufügen, ohne, dass sich etwas ändert.

Behauptung:
Die Knoten der $k'$ Knotenüberdeckung können keine weiteren Kanten aufnehmen.

Beweis:
Jeder der $k'$ vielen Knoten in der Überdeckung hat genau $k$ viele zu ihm inzidente Kanten in $G'$. Dadurch gibt es neben diesen Knoten noch $k' \cdot k$ weitere Knoten, die mit den $k'$ vielen Knoten der Überdeckung die insgesamt $k \cdot k' + k' = k' \cdot (k+1)$ viele Knoten in $G'$ ergeben.

Angenommen einer der $k'$ vielen Knoten hat mehr als $k$ Kanten, dann wäre er in $D$ und nicht mehr in $G'$, wodurch $G'$ keine $k' \cdot (k+1)$ Knoten mehr hätte. Angenommen einer der $k'$ vielen Knoten hat weniger als $k$ viele Kanten, dann müsste ein anderer Knoten $v$ aus der Überdeckung $k+1$ viele Kanten besitzen, damit $G'$ weiterhin $k' \cdot (k+1)$ Knoten und eine $k'$ Knotenüberdeckung besitzt. Dadurch wäre aber $v$ in $D$ und nicht mehr in $G'$, wodurch $G'$ keine $k' \cdot (k+1)$ vielen Knoten mehr hätte.

Es hat also jeder Knoten der $k'$ Knotenüberdeckung in $G'$ genau $k$ viele zu ihm inzidente Kanten.

Behauptung:
Enthält $G'$ mindestens $k' \cdot (k+1) + 1$ Knoten, so gibt es keine Überdeckung in $G$ und $G'$.

Beweis:
Angenommen $G'$ enthält $k' \cdot (k+1) + n$ Knoten, dann folgt aus obigen, dass die zusätzlichen Knoten $v_1, \ldots, v_n$ jeweils nicht mit einem Knoten $u$ aus der Knotenüberdeckung in $G'$ verbunden sein dürfen (da $u$ sonst $k+1$ inzidente Kanten besäße und nicht in $G'$ wäre, s.o.).
Daher muss er mit einem Knoten $w$ verbunden sein, der nicht in der Knotenüberdeckung aber noch in $G'$ ist. Folglich gibt es eine Kante $(w, v_i)$ mit $1 \leq i \leq n$, welche weder durch die Knotenüberdeckung in $G'$, noch durch die in $G$ getroffen wird.

In $G'$ kann $(w, v_i)$ nicht getroffen werden, da sich dann die Anzahl der Knoten in der Überdeckung ändern würde, die jedoch durch $k' = k - |D|$ fest gegeben ist.
In $G$ kann $(w, v_i)$ aber auch nicht getroffen werden, da jeder zusätzliche Knoten $v_i$ nicht mit einem Knoten der Überdeckung in $G$, sondern nur mit "unbeteiligten" Knoten verbunden ist. Nimmt man also jeden Knoten $v_i$ mit in die Überdeckung aus $G$ mit auf, erhöht sich dadurch die Größe der Überdeckung für jeden zusätzlichen Knoten, womit die minimale Überdeckung in $G$ die Größe $k+n$ hätte.

Dadurch ist es nicht möglich, dass $G'$ mehr als $k' \cdot (k+1)$ Knoten enthält und $G$ eine Überdeckung mit der Größe $\leq k$ besitzt.

□

**g)**

Man kann beim Einlesen von $E(G)$ für jeden Knoten dessen Grad speichern indem man für jede Kante den Grad der beiden Knoten jeweils inkrementiert. Dies ist mit indizierten Knoten in $\mathcal{O}(1)$ machbar, ansonsten in $\mathcal{O}(|V(G)|)$.
Mit dem gespeicherten Grad für jeden Knoten kann man durch einmaliges iterieren über die Knotenmenge $V(G)$ die Menge $D$ erzeugen. Das entspricht einer Laufzeit von $\mathcal{O}(|V(G)|)$.
Einlesen, speichern und $D$ erzeugen ist also in $\mathcal{O}(|V(G)|)$ möglich.

Auch das löschen von $D$ aus $G$ geht schnell, denn man kann in $\mathcal{O}(|V(G)|)$ die Knoten und in $\mathcal{O}(|E(G)|)$ die Kanten löschen. Speichert man alle Mengenkardinalitäten zwischen (also von $G, V(G), D$), was kaum Zeit kostet bei jeder Änderung die Kardinalität zu aktualisieren, so kann man in konstanter Zeit prüfen, ob $|V(G')| > k' \cdot (k+1)$ gilt.