Lösungsstrategien für NP-schwere Probleme der Kombinatorischen Optimierung

— Übungsblatt 2 —

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

Abgabe am: 25. April 2016

Aufgabe 2.1

Aufgabe 2.1.0.a

Das RESOURCE RESERVATION PROBLEM ist NP-Völlständig, da es das Problem SET PACKING leicht anders aufgeschrieben ist und wir schon gezeigt haben, dass SET PACKING NP-Vollständig ist. Die Familie S_i die SET PACKING übergeben bekommt ist die Menge P, wobei die Elemente von P die benötigten Ressourcen, die der jeweilige Prozess benötigt, in einer Menge sind. Das k aus dem RESOURCE RESERVATION PROBLEM entspricht dem von SET PACKING. Wenn nun k disjunkte Mengen vorhanden sind, sind auch k verschiedene Prozesse aktiv, da durch das disjunkt sein sichergestellt ist, dass die Ressourcen die ein Prozess benötigt kein anderer der gefunden Prozesse benötigt.

Aufgabe 2.1.1 Spezialfall $k = 2$

Beim Spezialfall für $k = 2$ kann man tatsächlich einen Algorithmus angeben, dessen Laufzeit polynomiell in der Eingabe ist.

Algorithm 1 reserve

```
1:  procedure RESERVE(P, R, k)
2:      for all p∈P do
3:          R_rest = weiseResourcenZu(p, R) // gibt alle noch nicht zugewiesenen Ressourcen zurück
4:          P = P − p
5:          for all q∈P do
6:              b = istVerteilungMöglich(q, R_rest)
7:              if b == true then return true
8:              end if
9:          end for
10:     end forreturn false
11: end procedure
```

Die Laufzeit beträgt dabei $\mathcal{O}(|P| \cdot n + |P|^2 \cdot n)$, wobei $n = |R|$ ist.

Man Geht in der äußeren Schleife alle $p \in P$ durch , also $|P|$ mal. Dort verteilt man zunächst maximal n viele Ressourcen, ergo n viele Schritte. Nun wird p aus P entfernt, was mit einer schlauen Implementierung (z.B. als Liste) in konstanter Zeit machbar ist.

Danach beginnt die innere Schleife, welche alle $q \in P$ durch geht und somit $|P| - 1$ mal durchläuft. Dort wird dann geprüft ob eine weitere Verteilung der restlichen Ressourcen auf q möglich ist. Auch da benötigt man maximal n viele Schritte.
Man erhält also eine Laufzeit von $\mathcal{O}(|P| \cdot (n + |P| \cdot n))$, was mit $\mathcal{O}(|P| \cdot n + |P|^2 \cdot n)$ äquivalent ist (hier erkennt man aber schön das Polynom).

Aufgabe 2.1.1.a

Für diesen Spezialfall können wir den Algorithmus von Edmonds und Karp benutzen. Wir wissen das jeder Prozess genau eine Ressource „Person “und genau eine Ressource „Ausrüstungsgegenstand “benötigt. Damit wir mit dem Algorithmus eine Lösung bekommen erstellen wir uns einen bipartiten Graph indem wir jede „Person “der Menge X hinzufügen und jeden „Ausrüstungsgegenstand “der Menge Y hinzufügen. Nun erstellen wir für jedes Element aus X und Y einen Knoten. Die Kanten erstellen wir uns mit den Prozessen, diese haben jeweils ein x-Element und ein y-Element und genau diese verbinden wir nun mit einer Kante und da ein Prozess keine 2 x- oder y-Elemente haben kann erzeugen wir so einen bipartiten Graphen. Mit dem Algorithmus von Edmonds und Karp können wir jetzt in Polynomialzeit ein maximales Matching bestimmen, dabei sind die Matchingkanten die Prozesse, die gleichzeitig aktiv sein können, da bei einem Matching keine 2 unterschiedlichen Kanten an den gleichen Knoten anstoßen, demnach keine 2 Prozesse die gleiche „Person “oder den gleichen „Ausrüstungsgegenstand “benutzen. Ist die Anzahl der Matchingkanten nun $\geq$ k, haben wir eine Ja-Instanz, andernfalls eine Nein-Instanz.

Aufgabe 2.1.1.b

–

Aufgabe 2.2

Aufgabe 2.2.1 $3-$SAT auf SET SPLITTING reduzieren

Aufgabe 2.2.1.a Zeigen, dass SET SPLITTING$\in NP$ gilt

Mit einem Verfifikationsalgorithmus, welcher die Eingabe $\langle S_1, S_2, C\rangle$ bekommt, kann man in polynomieller Laufzeit prüfen, ob eine Aufteilung in Klassen (also ein set splitting), korrekt ist.

Dazu geht man zunächst jedes $c \in C$ durch und nimmt ein Element e_1 aus c. Für dieses Element prüft man nun ob es ein $e_2 \in c$ gibt, bei dem gilt $class(e_1) \neq class(e_2)$ (also ob e_1 in einer anderen Klasse ist, als e_2). Gibt es ein solches e_2, ist die Aufteilung der Teilmenge c schon mal gültig und verifiziert. Zu prüfen sind noch die restlichen Teilmengen.

Die Laufzeit ist polynomiell, da man $|C|$ viele Teilmengen durch geht und pro Teilmenge maximal $|S|$ Elemente. Es gibt maximal $|S|$ viele Teilmengen (jede Teilmenge mit je einem Element), damit wäre die Laufzeit in $\mathcal{O}(|S|^2)$.

Aufgabe 2.2.1.b Reduktion angeben

Gegeben sei eine Aussagenlogische Formel A mit den Klauseln $K_1, K_2, \ldots, K_k$ und den Literalen $x_1, x_2, \ldots, x_n$, wobei jede Klausel maximal drei Literale enthält. Um eine Eingabe für das SET SPLITTING Problem zu bekommen, wandelt man A wie folgt um:

Jedes Literal wird mit seinem Komplement in eine Menge geschlossen, also $\{x_1, \overline{x_i}\}$, analog wird jede Klausel in eine Menge geschlossen. Dazu wird ein neues Element F erzeugt und jeder bisherigen Teilmenge hinzugefügt, sodass F in jeder Teilmenge vorhanden ist. F wird als `false` interpretiert und ist immer in der Partition enthalten, in der alle Literale, welche zu 0 ausgewertet werden enthalten sind.

Die Reduktion geschieht in polynomieller Zeit, da man zunächst alle Literale (n viele) in Mengen vereinigt, dann zu jeder Klausel eine Menge erzeugt (es gibt k viele Klauseln), dann F erzeugt (geht in konstanter Zeit) und danach noch jeder Klauselmenge das F hinzufügt (wieder k mal).

Insgesamt ergibt das eine Laufzeit von $\mathcal{O}(n + 2 \cdot k)$, was ein Polynom ist.

Damit ist die Reduktion fertig und man muss nur noch das Orakel von SET SPLITTING befragen.

Aufgabe 2.2.1.c Korrektheit für "⇒"

Behauptung: A besitzt eine erfüllende Belegung, dann existiert ein SET SPLITTING in obiger Konstruktion.

Seien $x_1^*, x_2^*, \ldots, x_n^*$ die Belegungen für die Literale aus A, welche A erfüllen. Man teilt nun die Elemente der zugehörigen Mengen aus der Konstruktion so auf, dass in S_1 alle die jenigen Literale enthalten sind, die zu 1 und in S_2 die jenigen, die zu 0 ausgewertet werden. Durch das neue Element F gelingt es nun eine Partitionierung zu finden, da man zwei Fälle unterscheiden kann:
Entweder besitzt eine betrachtete Klausel mindestens ein zu einer 1 ausgewertetes und mindestens ein zu einer 0 ausgewertetes Literal(sodass in sowohl in S_1, als auch in S_2 ein Literal enthalten ist), oder es besitzt nur zu 1 ausgewertete Literale (nur zu 0 ausgewertete treten nicht auf, da A erfüllbar ist und wir eine passende Belegung haben). Für den Fall von ausschließlich zu 1 ausgewerteten Literalen gibt es das F, welches als zusätzliches Literal dafür sorgt, dass auch diese rein "positive" Klausel eine gültige Partitionierung erzeugt.

In allen Fällen ist eine gültige Partitionierung möglich.

Aufgabe 2.2.1.d Korrektheit für "⇐"

Behauptung: Es gibt eine gültige Partitionierung, damit ist A erfüllbar.

Sei S_1, S_2 eine gültige Partitionierung von S mit der Teilmengenmenge C, sowie der dahinter liegenden Formel A, welche wiederum die Literale $x_1, x_2, \ldots, x_n$ und die Klauselmenge K besitzt. F ist in der Partition enthalten, in der alle Literale enthalten sind, die in A zu 0 ausgewertet werden sollen. Es kann in dieser Partition nach Definition von SET SPLITTING keine Teilmenge $c \in C$ liegen. In der anderen Partition sind somit alle Literale enthalten, die zu 1 ausgewertet werden sollen. Wertet man nun jedes Literal aus A zu 1 aus, welches in dieser Partition vorkommt, ist A erfüllt.
Dies ist durch die Konstruktion der Partitionierung gegeben, die besagt, dass Komplemente von Literalen stets in der jeweils anderen Partition enthalten sind. Dadurch wird garantiert, dass es keine Wiedersprüche in der Auswertung geben kann.
In dieser Partition taucht jedes Literal auf und jedes Literal wird zu 1 ausgewertet, somit ist A mit dieser Belegung erfüllt.

Aufgabe 2.2.2

Aufgabe 2.2.3 SUBSET-SUM auf NUMBER PARTITION INTO EQUAL PARTS reduzieren

Aufgabe 2.2.3.a Zeigen, dass NUMBER PARTITION INTO EQUAL PARTS $\in NP$ gilt

Wir geben einen polynomiellen Verifikationsalgorithmus für ein Zertifikat $\langle S, S_1, S_2 \rangle$ an.

Zuerst prüfen wir dazu, ob S_1 und S_2 wirklich eine Partitionierung von S darstellen. Dazu prüfen wir für jedes Element aus S_1 und S_2, ob dieses jeweils in S enthalten ist und markieren jedes gefundene Element in S. Dies braucht maximal - nämlich bei gültigen Eingaben - $|S|^2$ viele Operationen, da dann für jedes Element aus S eine Suche auf S durchgeführt werden muss (naive Suche ist in $O(|S|)$). Schlägt eine Suche fehl terminiert der Algorithmus und falsifiziert das Zertifikat. Nachdem alle Suchen ausgeführt wurden, muss für jedes Element aus S noch geprüft werden, ob es markiert wurde, was in $|S|$ Schritten möglich ist. Ist ein Element nicht markiert, kann der Algorithmus auch hier mit dem Ergebnis `flase` terminieren. Ansonsten können wir nun von einer korrekten Partitionierung ausgehen.

Danach müssen wir nur noch die Summe der Elemente von S_1 und S_2 bestimmen, was bei $|S|$ Elementen auch $|S|$ Operationen braucht, da wir jedes Element ein mal betrachten müssen. Am Ende braucht es noch einen Vergleich beider Summen. Sind die Summen gleich, gibt der Algorithmus `true`

aus, ansonsten `false`.

Der Algorithmus verifiziert damit das Zertifikat und liegt in $O(|S^2|)$, ist also polynomiell. Damit liegt NUMBER PARTITION INTO EQUAL PARTS folglich auch in NP.

Aufgabe 2.2.3.b Reduktion angeben

Wir gehen von einer Eingabe $\langle S, W\rangle$ mit $S = w_1 \dots w_n$ für NUMBER PARTITION INTO EQUAL PARTS aus, wobei wir nur mit positiven ganzen Zahlen arbeiten.

Wir konstruieren nun eine Eingabe $\langle S'\rangle$ für SUBSET-SUM.

Dazu bilden wir zuerst $X = \sum_{w \in S} w$.

Jetzt berechnen wir noch die Werte $a = 2X - W$ und $b = X + W$.

Die Menge S' sei nun gegeben durch $S \cup \{a, b\}$ und damit ist die Reduktion abgeschlossen.

Da $\sum_{w \in S'} w = 4X$, fragen wir jetzt also, ob wir zwei Teilmengen von S' finden können, sodass deren Summe jeweils 2X beträgt.

Die Reduktion erfolgt dabei in Polynomialzeit, weil wir lediglich die Summe von n Elementen bilden müssen und dann noch 2 weitere Zahlen berechnen und eine Vereinigung von n + 2 Elementen bilden müssen. Die Reduktion liegt damit in $O(n)$.

Aufgabe 2.2.3.c Korrektheit für "⇒"

Behauptung: Gibt es eine gültige Partitionierung für NUMBER PARTITION INTO EQUAL PARTS, existiert auch eine entsprechende Teilmenge für SUBSET-SUM.

In einer gültigen Partitionierung von S' können a und b nicht in derselben Teilmenge sein, weil schon $a + b = 3X$ und beide Teilmengen müssen sich jeweils genau zu 2X addieren.

Da $a = 2X - W$ müssen in der Teilmenge von a die anderen Elemente auch genau die Summe W bilden. Lässt man daher a aus dieser Teilmenge heraus, hat man eine Teilmenge von S vorliegen, die sich genau zu W addiert, also eine gültige Einagbe für SUBSET-SUM.

Aufgabe 2.2.3.d Korrektheit für "⇐"

Behauptung: Gibt es eine gültige Teilmenge für SUBSET-SUM, existiert auch eine entsprechende Partitionierung für NUMBER PARTITION INTO EQUAL PARTS.

Da alle Elemente aus S auch in S' sind, findet sich die gültige Teilmenge von S mit Summe W auch in S' wieder. Fügen wir jetzt das a zu genau dieser Teilmenge hinzu, ergibt sich als Summe dieser Menge genau 2X.

Da die Summe aller Elemente aus S' genau 4X ist, addieren sich die übrigen Elemente aus S' also ebenfalls genau zu 2X, sodass es eine gültige Partitionierung für NUMBER PARTITION INTO EQUAL PARTS geben muss.