Lösungsstrategien für NP-schwere Probleme der Kombinatorischen Optimierung

— Übungsblatt 7 —

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

Abgabe am: 6. Juni 2016

Aufgabe 7.1

Algorithm 1 ApproxWightedHittingSet

1: **procedure** ApproxWightedHittingSet(A, B)
2: R := relation from *element* $\rightarrow$ *quality index*
3: **for all** $a_i \in A$ **do**
4: n := amountOfAppearances(a_i)
5: $quality := weight(a_i)/n$
6: addToRelation(R, $a_i \rightarrow quality$)
7: **end for**
8: **for all** $a_i \in R$ **with** $R(a_k) \leq R(a_{k+1})$ **and** $B \neq \emptyset$ **do**
9: $H \leftarrow H \cup \{a_i\}$
10: $B \leftarrow B \setminus \{$allSetsHitBy($a_i$)$\}$
11: **end for**
12: **end procedure**

Beschreibung

R ist eine Relation, die jedes $a_i \in A$ auf einen Qualitätsindex abbildet. Dieser Index ist einfach das Gewicht $\frac{weight(a_i)}{n}$, wobei $weight(a_i)$ das Gewicht von a_i ist und n die Anzahl der Vorkommen angibt.

Die zweite `for`-Schleife (Zeile 8-11) geht alle a_i durch, wobei bei dem a_i mit den geringsten Gewicht begonnen wird, sodass $R(a_k) \leq R(a_{k+1})$ gilt. Zudem bricht die Schleife ab, sobald kein $B_j \in B$ mehr existiert (A muss dabei nicht leer sein), sprich sobald $B = \emptyset$ gilt.

In der Schleife wird nun jedes a_i in H aufgenommen und jedes getroffene B_j aus B entfernt, sodass man keine Elemente doppelt aufgenommen werden, die evtl. gar keine neuen B_j treffen.

Laufzeitanalyse

Die beiden Schleifen durchlaufen maximal $|A|$ viele Elemente.
Der Schleifenrumpf der ersten Schleife enthält die Funktion `amountOfAppearances`, der die Anzahl der Vorkommen des übergebenen a_i bestimmt. Eine Brude-Force Implementation würde jedes $B_i \in B$ durchgehen in prüfen ob das a_i darin vorkommt. Es gibt potentiell $|B| \cdot |A|$ viele Elemente, die man untersuchen muss, daher ist die Laufzeit in $\mathcal{O}(|B| \cdot |A|)$.

Die Berechnung der Qualität und das einfügen in die Relation ist nicht zwangsläufig von der Eingabegröße Abhängig, daher kann man das mit einer Laufzeit in $\mathcal{O}(1)$ implementieren.

Die zweite Schleife fügt zunächst ein a_i der Menge H hinzu. Implementiert man H als verkettete Liste geht das in $\mathcal{O}(1)$.
Als nächstes wird die `allSetsHitBy` Funktion benutzt, welcher alle Mengen sucht in denen a_i vorkommt. Die Funktionsweise ist identisch mit der von `amountOfAppearances`, nur wird eine Menge von Mengen zurückgegeben, statt einer Zahl. Somit ist die Laufzeit von `allSetsHitBy` in $\mathcal{O}(|B| \cdot |A|)$.

Insgesamt hat man also die Laufzeit von $\mathcal{O}(|A| \cdot (|B| \cdot |A|) + |A| \cdot (|B| \cdot |A|)) = \mathcal{O}\left(2 \cdot |B| \cdot |A|^2\right) = \mathcal{O}\left(|B| \cdot |A|^2\right)$, was polynomiell in der Eingabe ist.

Beweis: `ApproxWightedHittingSet` **ist ein b-Approximationsalgorithmus**

Aufgabe 7.2