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:     while B ≠ ∅ do
3:         a := null                               // the element with the best quality
4:         q_min := ∞
5:         for all a_i ∈ A do
6:             n := amountOfAppearances(a_i)
7:             quality := weight(a_i)/n
8:             if quality < q_min then
9:                 q_min ← quality
10:                a ← a_i
11:            end if
12:        end for
13:        H ← H ∪ {a}
14:        B ← B \ allSetsHitBy(a)
15:        A ← A \ a
16:    end while
17: end procedure
```

Beschreibung

Zu Beginn jedes Schleifendurchgangs (Zeile 3-15) wird das Element aus A bestimmt, welches die beste Qualität hat. Die Qualität beschreibt wie viel Gewicht pro getroffenem $B_j \in B$ aufgenommen wird. Je weniger Gewicht aufgenommen wird, desto besser das Resultat. Dadurch hat ein kleinerer Wert eine höhere (bessere) Qualität.

Bestimmt wird das Element a mit der besten Qualität in Zeile 5-12 in der alle verbleibenden $a_i \in A$ durchgegangen werden. Die Funktion `amountOfAppearances` berechnet dabei die Anzahl der getroffenen Mengen $B_j \in B$ durch das übergebene Element a_i.

Am Ende (Zeile 13-15) wird das gefundene Optimum für diesen Durchlauf (a) in das Hitting Set H aufgenommen. Zudem wird aus B jede Menge entfernt, die durch a getroffen wurde. Auch wird a aus A entfernt, sodass nun ein neues Optimum berechnet werden kann.

Terminierung

Der angegebene Algorithmus terminiert, da in jedem $B_i \in B$ nur Elemente aus A vorkommen. Da im worst-case jedes A einmal die beste Qualität hat, wird jedes B_i getroffen. Elemente mit gleicher Qualität werden nacheinander behandelt, sodass es auch dort keine Terminierungsprobleme geben kann.

Laufzeitanalyse

Auch wenn die Bedingung der `while`-Schleife $B \neq \emptyset$ lautet, so wird diese maximal $|A|$ mal ausgeführt, da bei jedem Durchlauf A um ein Element verkleinert wird. Ist $A = \emptyset$, so ist auch $B = \emptyset$ (s. Abschnitt Terminierung oben).

Die innere `for`-Schleife wird ebenfalls $|A|$ mal ausgeführt. In ihr wird `amountOfAppearances` aufgerufen, was die Anzahl der "Hits" ausgibt. Dabei wird in einer intuitiven Implementation jede Menge in B in jedes Element in dieser Menge durchgegangen. Jede Menge in B kann dabei maximal $|A|$ viele Elemente beherbergen, wodurch sich eine Laufzeit ergibt, die in $\mathcal{O}(|B| \cdot |A|)$ ist. Alle anderen Schritte haben eine konstante Laufzeit.

Am Ende (Zeile 13-15) wird a in H aufgenommen, was in konstanter Zeit machbar ist und a aus A gelöscht, was in linearer Zeit machbar ist (sofern man bei beiden von einer verketteten Liste ausgeht). Die Funktion `allSetsHitBy`, welche Aufgerufen wird funktioniert in einer intuitiven Implementation genauso wie `amountOfAppearances`, nur wird hier ein anderes Ergebnis zurückgegeben. Die Laufzeit von `allSetsHitBy` ist somit ebenfalls in $\mathcal{O}(|B| \cdot |A|)$.

Insgesamt ergibt sich also eine in der Eingabe polynomielle Laufzeit in

$$\mathcal{O}(|A| \cdot (|A| \cdot (|B| \cdot |A|) + (|B| \cdot |A|))) = \mathcal{O}\left(|A|^3 \cdot |B| + |A|^2 \cdot |B|\right).$$

Beweis: `ApproxWightedHittingSet` ist ein b-Approximationsalgorithmus

Aufgabe 7.2