Kapitel 6 Miscellaneous

JOHANNES GUTENBERG UNIVERSITÄT MAINZ

Lars Porth

Parametrisierte Algorithmen Seminar

16. 11. 2015

Inhaltsverzeichnis

1 6.1 Dynamische Programmierung über Teilmengen
- 6.1.1 Set Cover
- 6.1.2 Steiner Baum

2 6.2 Ganzzahlige Lineare Programmierung
- 6.2.1 Das Beispiel der Unausgewogenheit

6.1 Dynamische Programmierung über Teilmengen

6.1.1 Set Cover

Set Cover

Definition: Überdeckung

Sei $\mathcal{F}$ eine Familie von Mengen in einem Universum $\mathcal{U}$. Für eine Unterfamilie $\mathcal{F}' \subseteq \mathcal{F}$ und einer Teilmenge $\mathcal{U}' \subseteq \mathcal{U}$ sagen wir, dass $\mathcal{F}'$ $\mathcal{U}'$ überdeckt, wenn jedes Element von $\mathcal{U}'$ zu mindestens einer Menge von $\mathcal{F}'$ gehört ($\mathcal{U}' \subseteq \bigcup \mathcal{F}'$).

Set Cover

Im Set Cover Problem ist eine Familie von Mengen $\mathcal{F}$ gegeben in einem Universum $\mathcal{U}$ und eine positive ganze Zahl k. Die Aufgabe ist zu überprüfen ob eine Unterfamilie, mit maximal k Elementen, $\mathcal{F}' \subseteq \mathcal{F}$ existiert, so dass $\mathcal{F}'$ $\mathcal{U}$ überdeckt.

Theorem 6.1

Theorem 6.1

Gegeben eine Instanz des SET COVER Problems $(\mathcal{U}, \mathcal{F}, k)$, kann die minimal mögliche Größe einer Unterfamilie $\mathcal{F}' \subseteq \mathcal{F}$, die $\mathcal{U}$ überdeckt, in Zeit $2^{|\mathcal{U}|}(|\mathcal{U}| + |\mathcal{F}|)^{\mathcal{O}(1)}$ gefunden werden.

Theorem 6.1

Beweis

- Sei $\mathcal{F} = \{F_1, F_2, ..., F_{|\mathcal{F}|}\}$
- Wir definieren die dynamische Programmierungs Tabelle wie folgt:
- Für jede Untermenge $X \subseteq \mathcal{U}$ und jede ganze Zahl $0 \leq j \leq |\mathcal{F}|$, definieren wir $T[X, j]$ als die minimale Größe einer Untermenge $\mathcal{F}' \subseteq \{F_1, F_2, ..., F_j\}$, die X überdeckt.
- Falls keine solche Teilmenge $\mathcal{F}'$ existiert setzten wir $T[X, j] = +\infty$

Theorem 6.1

Beweis

- In unserem dynamischen Algorithmus Programm berechnen wir alle $2^{|\mathcal{U}|}(\mathcal{F}+1)$ Werte $T[X,j]$
- Basis Fall: $T[\emptyset,0]=0$, $T[X,0]=+\infty$, für $X\neq\emptyset$
- Rekusiver Fall: Für $X\subseteq\mathcal{U}$ und $0<j\leq|\mathcal{F}|$ zeigen wir, dass $T[X,j]=min(T[X,j-1],1+T[X\backslash F_j,j-1])$

Theorem 6.1
Beweis

- Wir wollen zeigen, dass
  $T[X,j] = min(T[X,j-1], 1 + T[X \backslash F_j, j-1])$
- Dafür zeigen wir, dass in beiden Richtungen Ungleichheit gilt

Theorem 6.1

Beweis

- Wir wollen zeigen, dass $T[X,j] = min(T[X,j-1], 1 + T[X \backslash F_j, j-1])$
- Dafür zeigen wir, dass in beiden Richtungen Ungleichheit gilt
- Für $\geq$: Sei $\mathcal{F}' \subseteq \{F_1, F_2, ..., F_j\}$ eine Familie mit minimaler Größe, die X überdeckt, wir unterscheiden zwei Fälle:

Theorem 6.1

Beweis

- Wir wollen zeigen, dass
  $T[X, j] = min(T[X, j-1], 1 + T[X \backslash F_j, j-1])$
- Dafür zeigen wir, dass in beiden Richtungen Ungleichheit gilt
- Für $\geq$: Sei $\mathcal{F}' \subseteq \{F_1, F_2, ..., F_j\}$ eine Familie mit minimaler Größe, die X überdeckt, wir unterscheiden zwei Fälle:
  - $F_j \notin \mathcal{F}'$, dann ist $\mathcal{F}'$ auch ein zulässiger Kandidat für $T[X, j-1]$

Theorem 6.1
Beweis

- Wir wollen zeigen, dass
  $T[X,j] = min(T[X,j-1], 1 + T[X \backslash F_j, j-1])$
- Dafür zeigen wir, dass in beiden Richtungen Ungleichheit gilt
- Für $\geq$: Sei $\mathcal{F}' \subseteq \{F_1, F_2, ..., F_j\}$ eine Familie mit minimaler Größe, die X überdeckt, wir unterscheiden zwei Fälle:
  - $F_j \notin \mathcal{F}'$, dann ist $\mathcal{F}'$ auch ein zulässiger Kandidat für $T[X, j-1]$
  - $F_j \in \mathcal{F}'$, dann ist $\mathcal{F}' \backslash F_j$ ein zulässiger Kandidat für $T[X \backslash F_j, j-1]$

Theorem 6.1

Beweis

- Wir wollen zeigen, dass $T[X,j] = min(T[X,j-1], 1 + T[X \backslash F_j, j-1])$
- Dafür zeigen wir, dass in beiden Richtungen Ungleichheit gilt
- Für $\geq$: Sei $\mathcal{F}' \subseteq \{F_1, F_2, ..., F_j\}$ eine Familie mit minimaler Größe, die X überdeckt, wir unterscheiden zwei Fälle:
  - $F_j \notin \mathcal{F}'$, dann ist $\mathcal{F}'$ auch ein zulässiger Kandidat für $T[X, j-1]$
  - $F_j \in \mathcal{F}'$, dann ist $\mathcal{F}' \backslash F_j$ ein zulässiger Kandidat für $T[X \backslash F_j, j-1]$
- Für $\leq$:

Theorem 6.1

Beweis

- Wir wollen zeigen, dass $T[X,j] = min(T[X,j-1], 1 + T[X \backslash F_j, j-1])$
- Dafür zeigen wir, dass in beiden Richtungen Ungleichheit gilt
- Für $\geq$: Sei $\mathcal{F}' \subseteq \{F_1, F_2, ..., F_j\}$ eine Familie mit minimaler Größe, die X überdeckt, wir unterscheiden zwei Fälle:
  - $F_j \notin \mathcal{F}'$, dann ist $\mathcal{F}'$ auch ein zulässiger Kandidat für $T[X,j-1]$
  - $F_j \in \mathcal{F}'$, dann ist $\mathcal{F}' \backslash F_j$ ein zulässiger Kandidat für $T[X \backslash F_j, j-1]$
- Für $\leq$:
  - jeder zulässige Kandidat $\mathcal{F}'$ für $T[X,j-1]$ ist auch ein zulässiger Kandidat für $T[X,j]$

Theorem 6.1

Beweis

- Wir wollen zeigen, dass $T[X,j] = min(T[X,j-1], 1 + T[X \backslash F_j, j-1])$
- Dafür zeigen wir, dass in beiden Richtungen Ungleichheit gilt
- Für $\geq$: Sei $\mathcal{F}' \subseteq \{F_1, F_2, ..., F_j\}$ eine Familie mit minimaler Größe, die X überdeckt, wir unterscheiden zwei Fälle:
  - $F_j \notin \mathcal{F}'$, dann ist $\mathcal{F}'$ auch ein zulässiger Kandidat für $T[X,j-1]$
  - $F_j \in \mathcal{F}'$, dann ist $\mathcal{F}' \backslash F_j$ ein zulässiger Kandidat für $T[X \backslash F_j, j-1]$
- Für $\leq$:
  - jeder zulässige Kandidat $\mathcal{F}'$ für $T[X,j-1]$ ist auch ein zulässiger Kandidat für $T[X,j]$
  - für jeden zulässigen Kandidaten $\mathcal{F}'$ für $T[X \backslash F_j, j-1]$ gilt, dass $\mathcal{F}' \cup F_j$ ein zulässiger Kandidat für $T[X,j]$ ist

Theorem 6.1
Beweis

- Mit diesem dynamischen Programm können wir für alle $X \subseteq U$ und $0 \leq j \leq |\mathcal{F}|$ den Wert $T[X, j]$ in versprochener Zeit $2^{|\mathcal{U}|}(|\mathcal{U}| + |\mathcal{F}|)^{\mathcal{O}(1)}$ berechnen.
- Der Wert den wir suchen ist $T[\mathcal{U}, |\mathcal{F}|]$

6.1.2 Steiner Baum

Steiner Baum

Steiner Baum

Sei G ein ungerichteter Graph mit n Knoten und $K \subseteq V(G)$ eine Menge von Endpunkten (terminals) aus G. Ein Steiner Baum für K in G ist ein zusammenhängender Teilgraph H von G, der K enthält ($K \subseteq H$)

Steiner Baum Problem

Im (gewichteten) Steiner Baum Problem bekommt man einen ungerichteten Graphen G, eine Gewichtungsfunktion $w : E(G) \to \mathbb{R}_{>0}$ und eine Teilmenge von Endpunkten $K \subseteq V(G)$ gegeben. Das Ziel ist es einen Steiner Baum H für K in G zu finden, in dem $w(H) = \sum_{e \in V(H)} w(e)$ minimal ist.

Steiner Baum Problem

Dynamischer Algorithmus

- Das Ziel ist es einen dynamischen Algorithmus zu entwickeln, der in Zeit $3^{|K|}n^{\mathcal{O}(1)}$ ($n = |V(G)|$) das Steiner Baum Problem löst.

Notation: $dist(v, u)$

Für ein Knotenpaar $u, v \in V(G)$, notieren wir die Kosten des kürzesten Pfad von v nach u als $dist(v, u)$.

- Erinnerung $dist(v, u)$ kann durch Algorithmen, wie der kürzeste Pfad Algorithmus von Dijkstra, in Polynomialzeit bestimmt werden.

Steiner Baum Problem

Vorverarbeitung

- Wir nehmen an, dass $|K| > 1$, weil sonst die Instanz des Problems trivial wäre.
- Ohne Einschränkungen nehmen weiterhin an, dass G verbunden ist.
- Als letztes setzten wir voraus, dass jeder Endpunkt aus K in G genau Grad 1 hat und sein einziger Nachbar kein Knoten aus K ist.
  - Um diese Bedingung zu erfüllen erzeugen wir für jeden Knoten $t \in K$ einen neuen Knoten t' und eine Kannte tt'

JG|U

Steiner Baum Problem

Dynamischer Algorithmus

- Wir definieren nun die dynamische Tabelle: Für jede nicht leere Teilmenge $D \subset K$ und jeden Knoten $v \in V(G) \setminus K$ sei $T[D, v]$ das minimale Gewicht von einem Steiner Baum für $D \cup \{v\}$ in G.
- Als Basisfall betrachten wir die Teilmengen $D \subset K$ für die gilt $|D| = 1$. Dann gilt:
  - Sei $D = \{t\}$, dann ist einem Steiner Baum mit minimalem Gewicht von $D \cup \{v\} = \{t, v\}$.
  - Somit ist der Wert für $T[\{t\}, v] = \mathit{dist}(t, v)$.

Lemma 6.2

Lemma 6.2

Für jedes $D \subseteq K$ von einer Größe von mindestens 2 und jedes $v \in V(G) \setminus K$ gilt folgendes:

$$T[D, v] = \min_{\substack{u \in V(G) \setminus K \\ \emptyset \neq D' \subsetneq D}} \{T[D', u] + T[D \setminus D', u] + dist(u, v)\}$$

Lemma 6.2

Beweisidee

- Man zeigt wieder die Ungleichheit in beide Richtungen.
- Für $\leq$:
  - Fixiere $u \in V(G)$ und $\emptyset \neq D' \subsetneq D$
  - Sei H_1 der Baum, der den Wert $T[D', u]$ bestimmt, H_2 der Baum, der den Wert $T[D \backslash D', u]$ bestimmt, und P der kürzeste Pfad zwischen v und u.
  - $w(H_1 \cup H_2 \cup P) \leq w(H_1) + w(H_2) + w(P) = T[D', u] + T[D \backslash D', u] + dist(u, v))$

Lemma 6.2
Beweisidee

- Für $\geq$:
  - Sei H ein Steiner Baum für $D \cup \{v\}$ in G mit minimalem Gewicht, also $w(H) = T[D, v]$.

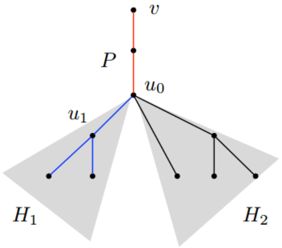

Fig. 6.1: Decomposition of H

  - $T[D, v] = w(H) = T[D', u_0] + T[D \setminus D', u_0] + dist(v, u_0) \geq \min_{\substack{u \in V(G) \setminus K \\ \emptyset \neq D' \subsetneq D}} \{T[D', u] + T[D \setminus D', u] + dist(u, v)\}$

Theorem 6.3

Theorem 6.3

Das Steiner Baum Problem kann in Zeit $3^{|K|} n^{\mathcal{O}(1)}$ gelöst werden.

Theorem 6.3

Beweis

- Sei (G, w, K) eine Instanz vom Steiner Baum Problem nach den Vorverarbeitungsschritten.
- Um im allgemeinen Fall $T[D, v]$ berechnen zu können benötigen wir die Werte $T[D', u]$ und $T[D \backslash D', u]$
- Auf die Art wie in Lemma 6.2 beschrieben kann man für feste Werte D und v $T[D, v]$ in Zeit, $2^{|D|} n^{\mathcal{O}(1)}$ berechnen.

Theorem 6.3

Beweis

- Als gesamt Laufzeit des Algorithmus ergibt sich:

$$\sum_{v \in V(G) \setminus K} \sum_{D \subseteq K} 2^{|D|} n^{\mathcal{O}(1)} \leq n \sum_{j=2}^{|K|} \binom{|K|}{j} 2^j n^{\mathcal{O}(1)} = 3^{|K|} n^{\mathcal{O}(1)}$$

- Wenn die Vorverarbeitungsschritte durchgeführt worden sind, enthält jeder Steiner Baum von K in $V(G)$ mindestens einen Steiner Punkt (Punkt aus $V(G) \setminus K$) und daher entspricht $\min_{v \in V(G) \setminus K} T[K, v]$ dem Wert des minimalen Steiner Baum für K in G.

6.2 Ganzzahlige Lineare Programmierung

Ganzzahlige Lineare Programmierungs Machbarkeit

Definition: Ganzzahlige Lineare Programmierungs Machbarkeit

Beim Ganzzahligen Linearen Programmierungs Machbarkeits Problem (Integer Linear Programming Feasibility Problem) bekommt man ein p Variablen $x_1, x_2, \dots, x_p$ und eine Menge an m Ungleichungen in der Form:

$$a_{1,1}x_1 + a_{1,2}x_2 + \dots + a_{1,p}x_p \leq b_1$$
$$\vdots$$
$$a_{m,1}x_1 + a_{m,2}x_2 + \dots + a_{m,p}x_p \leq b_m$$

Wobei $a_{i,j}$, b_j und x_i ganze Zahlen sein müssen.
Das Ziel ist es herauszufinden ob man eine Belegung für die x_i finden kann, so dass alle Ungleichungen erfüllt sind.

Theorem 6.4

Theorem 6.4

Eine Ganzzahlige Lineare Programmierungs Machbarkeits Instanz der Größe L mit p Variablen kann mit $\mathcal{O}(p^{2.5p+o(p)} \cdot L)$ arithmetischen Operationen und mit Platz polynomiell in L gelöst werden.

Ganzahlige Lineare Programmierung

Definition: Ganzzahliges Lineares Programmierungs Problem

Beim Ganzzahligen Linearen Programmierungs Problem, bekommt man eine Instanz des Ganzzahlige Lineare Programmierungs Machbarkeits gegeben (z.B als Matrix $A \in \mathbb{Z}^{m \times p}$ und einem Vektor $b \in \mathbb{Z}^p$) und zusätzlich noch einen Vektor $c \in \mathbb{Z}^p$.
Das Ziel ist es einen Vektor $x \in \mathbb{Z}^p$ zu finden, der alle alle Ungleichungen in $Ax \leq b$ erfüllt und die objective function $c \cdot x$ minimiert.

Theorem 6.5

Theorem 6.5

Eine Ganzzahlige Lineare Programmierungs Instanz (Linear Integer Programming Instance) der Größe L mit p Variablen kann mit $\mathcal{O}(p^{2.5p+o(p)} \cdot (L + \log M_x)(\log(M_x M_c)))$ arithmetischen Operationen und mit Platz polynomiell in $L + \log M_x$ gelöst werden.

Theorem 6.5

Beweis

- Wir stellen fest, dass der Betrag des Werts der objective function maximal pM_xM_c ist, solange der maximale Betrag der Variablen M_x ist.
- Wir wenden nun Binäre Suche an um den minimalen Wert der objective function zu finden.
- Dafür fügen wir zur Instanz des Machbarkeit Problem die Ungleichung $cx \leq t$ hinzu, wobei t ein fester Wert ist für den gilt $-pM_xM_c \leq t \leq pM_xM_c$. Und wenden darauf einen Algorithmus, der Theorem 6.4 erfüllt an.

Theorem 6.5

Beweis

- Die Instanz hat die Größe $\mathcal{O}(L + p\log(pM_x M_c)) = \mathcal{O}(p(L + \log M_x))$ und daher Läuft der Algorithmus von Theorem 6.4 in Zeit $\mathcal{O}(p^{2.5p+o(p)} \cdot (L + \log M_x))$.
- Durch anwenden der binären Suche mit t finden wir so in versprochener Zeit $\mathcal{O}(p^{2.5p+o(p)} \cdot (L + \log M_x)(\log(M_x M_c)))$ eine optimale Lösung t_0 des Linearen Programm.

6.2.1 Das Beispiel der Unausgewogenheit

Das Beispiel der Unausgewogenheit

Definition: Ordung

Sei G ein ungerichteter Graph mit n Knoten. Eine Ordnung von $V(G)$ ist eine bijektive Funktion $\pi : V(G) \rightarrow \{1, 2, ..., n\}$

Definition: Imbalance

Für $v \in V(G)$ definieren wir $L_\pi(v) = \{u \in N(v) : \pi(u) < \pi(v)\}$ und $R_\pi(v) = \{u \in N(v) : \pi(u) > \pi(v)\}$ Wir definieren die Imbalance an Knoten v als $\iota_\pi(v) = ||L_\pi(v)| - |R_\pi(v)||$ und die Imbalance von der Ordnung π als $\iota(\pi) = \sum_{v \in V(G)} \iota_\pi(v)$.

- Beim Imbalance Problem wollen wir nun eine Ordnung π finden, so dass $\iota(\pi)$ minimal ist.

Parametrisierung

- Wir werden das Imbalance Problem durch die Größe einer Knoten Überdeckung des Graphen parametrisieren.
- Wir nehmen an wir bekommen einen Graphen G zusammen mit seiner Knotenüberdeckung X der Größe k
  - In diesem Fall wäre es nicht unbedingt notwendig die Knoten Überdeckung mit übergeben zu bekommen, da 2-Approximations Algorithmen oder FPT Algorithmen für das Knoten Überdeckungs Problem bekannt sind.

Parametrisierung

- Im Folgenden werden wir zeigen, wie man Theorem 6.5 einsetzen kann um einen FPT Algorithmus für das Imbalance Problem parametrisiert durch die Größe der gegebenen Knotenüberdeckung von G.
- Um die minimale Ordnung ι_π zu finden werden wir für alle möglichen Ordnungen $\pi_X : X \to \{1, 2, ..., k\}$ der gegebenen Knotenüberdeckung, werden wir die beste Ordnung π finden, die mit π_X übereinstimmt.
    - Wir sagen, dass π_X mit π übereinstimmt, wenn gilt $\pi_X(u) < \pi_X(v)$, genau dann wenn $\pi(u) < \pi(v)$.

Parametrisierung

- Daher können wir annehmen, dass eine optimale Ordnung π existiert, so dass für $X = \{u_1, u_2, ..., u_k\}$ gilt: $\pi(u_1) < \pi(u_2) < ... < \pi(u_k)$.
- Wir definieren $X_i = \{u_1, u_2, ..., u_i\}$ für alle $0 \leq i \leq k$
- Weil X eine Knoten Überdeckung von G ist, sind die Knoten $I = V(G) \backslash X$ unabhängig voneinander und wir können jeden Knoten aus I einem Typ zuweisen.

Definition: Typ

Der Typ eines Knotens $v \in I$ ist die Menge $N(v) \subseteq X$. Für einen Typ $S \subseteq X$ ist die Menge I(S) die Menge aller Knoten in I von Typ S.

JG|U

Parametrisierung

- Jeder Knoten aus I ist entweder zwischen zwei Knoten aus X, links vom Knoten u_1 oder rechts vom Knoten u_k
- Wir sagen, dass ein Knoten $v \in I$ an Position 0 ist, wenn $\pi(v) < \pi(u_1)$ und dass ein Knoten an Position i ist, wenn i die größte Zahl ist, so dass $\pi(u_i) < \pi(v)$
- Wir notieren L_i für die Menge aller Knoten aus an Position i.

Parametrisierung

- Die Aufgabe eine optimale Permutation zu finden lässt sich nun in zwei Teile aufteilen:
  - Zerteilen der Menge I in $L_0, ..., L_k$.
  - Eine optimale innere Ordnung an allen Positionen zu finden.
- Das Ziel ist das Zerteilen von I in Mengen $L_0, ..., L_k$ als Ganzahliges Lineares Problem zu formulieren.

Formulierung des Linearen Probelems

- Für die Anzahl eines Typen S an der Position i führen wir die Variable x_S^i ein.
- Wir definieren für alle $u_i \in X$ eine Variable y_i als untere Schranke für die Imbalance von u_i.
- Außerdem definieren wir für alle u_i aus X $e_i = |N(u_i) \cap X_{i-1}| - |N(u_i) \cap (X \backslash X_{i-1})|$
- Dies führt uns zu einer Bedingung für alle u_i:

$$y_i \geq \left| e_i + \sum_{\substack{S \subseteq X \\ u_i \in S}} \left(\sum_{j=0}^{i-1} x_S^j - \sum_{j=i}^{k} x_S^j \right) \right|$$

- Als letztes definieren wir $z_S^i = ||S \cap X_i| - |S \cap (X \backslash X_i)||$ als Konstante für die Imbalance eines Knoten von Typ S, wenn er an Position i platziert wird.

JG|U

Formulierung des Linearen Problems

$$
\begin{array}{lll}
\min & \sum_{i=1}^{k} y_i + \sum_{i=0}^{k} \sum_{S \subseteq X} z_S^i x_S^i & \\
\text{s.t.} & \sum_{i=0}^{k} x_S^i = |I(S)| & \forall S \subseteq X \\
& y_i \geq e_i + \sum_{\substack{S \subseteq X \\ u_i \in S}} \left(\sum_{j=0}^{i-1} x_S^j - \sum_{j=i}^{k} x_S^j\right) & \forall 1 \leq i \leq k \\
& y_i \geq -e_i - \sum_{\substack{S \subseteq X \\ u_i \in S}} \left(\sum_{j=0}^{i-1} x_S^j - \sum_{j=i}^{k} x_S^j\right) & \forall 1 \leq i \leq k \\
& x_S^i \geq 0 & \forall 0 \leq i \leq k, S \subseteq X
\end{array}
$$

JG|U

Formulierung des Linearen Problems

- Durch anwenden des Theorems 6.5 kann man dieses Lineare Problem parametrisiert lösen.
- Daher folgt: Das Imbalance Problem, parametrisiert durch die Größe einer Knoten Überdeckung des Graphen ist FPT.