ADS: Algorithmen und Datenstrukturen 2

Teil X

Peter F. Stadler & Konstantin Klemm

Bioinformatics Group, Dept. of Computer Science & Interdisciplinary Center for Bioinformatics, **University of Leipzig**

09. Juni 2010

Greedy-Algorithmus I

Greedy-Algorithmen sind mit dem dynamischen Programmieren verwandt, jedoch einfacher.

Die Grundsituation ist dieselbe:
Es geht um ein Optimierungsproblem; es soll sukzessiv eine optimale Lösung — in Bezug auf eine gegebene Bewertungsfunktion — konstruiert werden. Während man beim dynamischen Programmieren solche optimale Lösung für alle kleineren Teilprobleme konstruiert und mit Hilfe dieser in einer Tabelle eingetragenen Daten die nächst größere optimale Lösung konstruiert, verzichtet man bei Greedy auf die Buchführung mittels einer Tabelle. Stattdessen wird der nächste Erweiterungsschritt zu einer (hoffentlich) optimalen Lösung lediglich auf Grund der lokal verfügbaren Informationen getätigt.

Greedy-Algorithmus II

Annahme: Es gibt eine Gewichtsfunktion w, die die Güte einer Lösung (auch einer Teillösung) misst. Es soll eine Lösung mit maximalen w-Wert konstruiert werden.

1. Starte mit der leeren Lösung.
2. Erweitere die bisher konstruierte Teillösung wie folgt: Wenn zur Erweiterung dieser Teillösung k Erweiterungsmöglichkeiten zur Verfügung stehen, die auf die vergrößerten Teillösungen $L_1, \ldots, L_k$ führen, wähle diejenige Teillösung L_i mit $w(L_i)$ maximal.

Der Name Greedy = *gefräßig* erklärt sich dadurch, dass ein Greedy-Algorithmus nach der Methode vorgeht: *Nimm immer das größte Stück*. Dieses Greedy-Prinzip kann in vielen Fällen funktionieren und tatsächlich auf eine optimale Lösung führen — ohne den Aufwand, der bei dynamischem Programmieren betrieben werden muss.

Unabhängigkeitssystem

- Sei E eine endliche Menge und $\mathcal{M}$ eine Menge von Teilmengen von E, also $\mathcal{M} \subseteq \mathcal{P}(X)$. Dann heißt $(E, \mathcal{M})$ *Mengensystem*.
- Ein Mengensystem $(E, \mathcal{M})$ heißt *Unabhängigkeitssystem*, wenn $\emptyset \in \mathcal{M}$ und zu jeder Menge in $\mathcal{M}$ auch alle Teilmengen enthalten sind, also gilt:

  $$\text{Aus } A \in \mathcal{M} \text{ und } B \subseteq A \text{ folgt } B \in \mathcal{M}.$$

- Ein Unabhängigkeitssystem ist somit *abgeschlossen* unter Bildung von Teilmengen. Man kann eine in $\mathcal{M}$ gefundene Menge "kleiner" machen und bleibt in $\mathcal{M}$.

Gewichtsfunktion und Optimierungsproblem

- Sei $(E, \mathcal{M})$ ein Mengensystem und $w : E \to \mathbb{R}^+$.
- w heißt *Gewichtsfunktion*
- Optimierungsproblem: Finde $L \in \mathcal{M}$, das die Gewichtssumme maximiert. Für alle $A \in \mathcal{M}$ soll also gelten:

$$\sum_{x \in L} w(x) \geq \sum_{x \in A} w(x)$$

Kanonischer Greedy-Algorithmus

Der einem Unabhängigkeitssystem $(E, \mathcal{M})$ und Gewichtsfunktion $w : E \to \mathbb{R}^+$ zugeordnete kanonische Greedy-Algorithmus arbeitet wie folgt:

Ordne alle Elemente in E nach absteigendem Gewicht:
$w(e_1) \geq w(e_2) \geq \ldots w(e_{|E|})$;
$L \leftarrow \emptyset$;
for $k \leftarrow 1$ **to** $|E|$ **do**
  if $L \cup \{e_k\} \in \mathcal{M}$ **then**
    $T \leftarrow T \cup \{e_k\}$
  end
end
Ausgabe der Lösung T ;

Dieser Algorithmus liefert allerdings nicht immer die optimale Lösung.

Beispiel

- $E = \{e_1, e_2, e_3\}$
- $\mathcal{M} = \{\, \emptyset, \{e_1\}, \{e_2\}, \{e_3\}, \{e_2, e_3\} \,\}$
- $w(e_1) = 3,\ w(e_2) = w(e_3) = 2$
- Dazu liefert der kanonische Greedy-Algorithmus die Lösung $L = \{e_1\}$ mit Gewichtssume $w(e_1) = 3$
- Die optimale Lösung ist $L^* = \{e_2, e_3\}$ mit Gewichtssumme $w(e_2) + w(e_3) = 4$.

Kriterium, wann Greedy das Opimierungsproblem löst?

Matroid

- Ein Unabhängigkeitssystem $(E, \mathcal{M})$ heißt *Matroid*, wenn für alle $A, B \in \mathcal{M}$ die folgende *Austauscheigenschaft* gilt:
  Ist $|A| > |B|$, dann gibt es $x \in A \setminus B$, so dass $B \cup \{x\} \in \mathcal{M}$.
- Sei $(E, \mathcal{M})$ ein Matroid und $B \in \mathcal{M}$ ein maximales Element, also für alle $x \in E \setminus B$ gilt: $B \cup \{x\} \notin \mathcal{M}$. Wir bezeichnen B als *Basis* des Matroids $(E, \mathcal{M})$.
- Beobachtung: Alle Basen eines Matroids enthalten dieselbe Anzahl Elemente. Anders gesagt, sind B und C Basen von $(E, \mathcal{M})$, dann gilt $|B| = |C|$

Exkurs: Bezug zur Linearen Algebra

Matroideigenschaft ist eine abstrakte Charakterisierung von linearer Unabhängigkeit und Basen.

- E = Vektorraum
- $\mathcal{M}$ = Menge aller linear unabhängigen Teilmengen von E
- Austauscheigenschaft: vgl. Basisergänzungssatz aus der linearen Algebra.
- Basis = linear unabhängige Menge, bei der jede Obermenge linear abhängig ist.

Matroid garantiert Optimalität

Sei $(E, \mathcal{M})$ ein Mengensystem.

Satz (Greedy-Optimalität):
Der kanonische Greedy-Algorithmus liefert auf $(E, \mathcal{M})$ eine optimale Lösung für beliebige Gewichtsfunktion $w : E \to \mathbb{R}$

$$\Updownarrow$$

$(E, \mathcal{M})$ ist ein Matroid.

Auftragsplanung

Gegeben:

- Menge E von n Aufträgen, jeweils in einer Zeiteinheit (an einem Tag) zu bewältigen.
- Zu jedem Auftrag $x \in E$ gibt es Gewinn $w(x) > 0$ (Gewichtsfunktion).
- Termin $d(x)$, bis zu dem der Auftrag x abgeschlossen sein muss (sonst kein Gewinn).

Gesucht: Menge $L \subseteq E$ von Aufträgen, so dass

a) der Gesamtgewinn $\sum_{x \in L} w(x)$ maximal ist, und

b) L sich so sortieren lässt, dass jeder Auftrag vor seinem Abschlusstermin erledigt wird. Menge $\mathcal{M}$ zulässiger Auftragsmengen ist also gegeben durch

$$A \in \mathcal{M} \Leftrightarrow \forall s \in \mathbb{N} : |\{y \in A : d(y) \leq s\}| \leq s$$

Übungsaufgabe 15: Zeigen Sie, daß $(E, \mathcal{M})$ ein Matroid ist.

Auftragsplanung: Beispiel

Auftragsmenge $E = \{a, b, c, d, e\}$

Auftrag x	Wert w(x)	Termin d(x)
a	13	2
b	7	1
c	9	1
d	3	2
e	5	1

Lösung	Gewinn
$\emptyset$	0
$\{a\}$	13
$\{b\}$	7
$\{c\}$	9
$\{d\}$	3
$\{e\}$	5
$\{a, b\}$	20
$\{a, c\}$	22
$\{a, d\}$	16
$\{a, e\}$	18
$\{b, d\}$	10
$\{c, d\}$	12
$\{d, e\}$	8

... und noch mal der Kruskal-Algorithmus

Kruskal-Algorithmus (1956) für minimalen Spannbaum auf ungerichtetem zusammenhängendem Graph (V, E) mit Kantengewicht $w(e)$ für $e \in E$

1. Erzeuge eine PriorityQueue Q mit der Kantenmenge E sortiert nach Gewicht (kleinstes zuerst). Initialisiere F als leere Menge.
2. Entferne Kante $e = \{u, v\}$ aus Q
3. Falls (V, F) keinen Pfad zwischen u und v enthält:

$$F := F \cup \{e\}$$

   (sonst: tue nichts)
4. Falls Q nicht leer, zurück zu 2.
5. Ausgabe F.

Greedy-Algorithmus zur *Minimierung* der Summe der Kantengewichte

Kruskal: Korrektheit (1)

- E = Kantenmenge
- Gewichtsfunktion $w : E \to \mathbb{R}$ = Kantengewichte
- $\mathcal{M}$ enthält $F \subseteq E$ genau dann wenn (V, F) azyklisch (Wald) ist.

Wende Satz über Greedy-Optimalität an:

Kruskal findet optimale Lösung für beliebige Kantengewichte

$$\Updownarrow$$

$(E, \mathcal{M})$ ist ein Matroid.

Der Kruskal *minimiert*, während der kanonische Greedy-Algorithmus *maximiert*. Der Algorithmus ist direkt anwendbar nach Transformation der Kantengewichte $w'(e) = c - w(e)$. Die Konstante c ist so groß zu wählen, dass alle $w'(e)$ positiv werden.

Kruskal: Korrektheit (2)

Behauptung: $(E, \mathcal{M})$ ist ein Matroid.

Beweis(skizze):

$(E, \mathcal{M})$ ist *Unabhängigkeitssystem*, denn Teilgraphen von azyklischen Graphen sind azyklisch.

Austauscheigenschaft: Seien $A, B \in \mathcal{M}$ mit $|A| > |B|$, also (V, A) azyklisch und (V, B) azyklisch. (V, A) ist disjunkte Vereinigung von $|V| - |A|$ Bäumen, (V, A) ist disjunkte Vereinigung von $|V| - |B|$ Bäumen. Somit besteht (V, B) aus mehr Bäumen als (V, A), und es gibt Knoten $u, v \in V$, die in demselben Baum von (V, A) liegen, aber in verschiedenen von (V, B). Insbesondere können u und v so gewählt werden, dass die Kante $\{u, v\} := e$ in A liegt aber nicht in B. Ausserdem ist $(V, B \cup \{e\})$ azyklisch nach Wahl von u und v aus verschiedenen Bäumen von (V, B). Also $e \in A \setminus B$ und $B \cup \{e\} \in \mathcal{M}$, q.e.d.

$(E, \mathcal{M})$ heißt graphisches Matroid.