

Grundlagen Theoretischer Informatik 3

SoSe 2010 in Trier

Henning Fernau

Universität Trier

fernau@uni-trier.de

Grundlagen Theoretischer Informatik 3 Gesamtübersicht

- Organisatorisches; Einführung
- Algorithmenanalyse: Komplexität und Korrektheit
- Zur Behandlung NP-schwerer Probleme: Ausblicke
- Randomisierte Algorithmen und ihre Analyse

Organisatorisches

Vorlesungen FR 8.30-10.00 im HS 13

Übungsbetrieb in Form von einer Übungsgruppe

BEGINN: in der zweiten Semesterwoche

FR 10-12, HS 13

Dozentensprechstunde DO 13-14 in meinem Büro H 410 (4. Stock)

Mitarbeitersprechstunde (Stefan Gulan) DO 13-14 H 413

Tutorensprechstunde DI 13.30-14.30 & MI 14-15 H 407/H412

Ausnahme in diesen zwei Wochen

In der nächsten Woche bin ich nicht da.

~> Wir machen in dieser Woche zwei Vorlesungstermine, MI und FR.
Der Übungsbetrieb bleibt durchgehend wie bisher (FR).

In der nächsten Woche muss auch meine Sprechstunde ausfallen.

Zum Umgang mit *NP*-harten Problemen

- In manchen Anwendungen ist das garantierte Auffinden exakter Lösungen unabdingbar.
- Das bedeutet, dass man (theoretisch) mit nicht-polynomiellen Algorithmen auskommen muss.
- In manchen Anwendungsfällen ist aber das schnelle Auffinden von (irgendwelchen zulässigen, aber möglichst guten) Lösungen wichtig.
Heute: *(Meta-)Heuristiken* zum schnellen Lösungs-Finden.

Greedy-Algorithmen

- Ziel: exakte/approximative Lösung von Optimierungsproblemen.
- wie bei Verzweigungsalgorithmen und beim dynamischen Programmieren: Lösung durch Erweiterung von Lösungsansätzen
- dynamisches Programm: jeweils beste Erweiterung (\rightsquigarrow Tabelle...)
- Greedy: gute, leicht findbare Erweiterung (ohne Tabelle...)
- Grundidee: **Nimm immer das beste Stück!**

Beispiel: 0/1-Rucksackproblem,

$$n = 3, \quad (v_1, v_2, v_3) = (3, 5, 6), \quad (g_1, g_2, g_3) = (1, 2, 3), \quad G = 5$$

Greedy-Heuristik hier:

Nimm das Stück mit dem höchsten Wert pro Gewichtsanteil!

Reihenfolge für Beispiel:

$$\frac{3}{1} \geq \frac{5}{2} \geq \frac{6}{3}$$

also: erst Objekt 1 nehmen, dann Objekt 2, dann ... bis G überschritten.

Resultat: Lösung $\{1, 2\}$ mit Wert $3 + 5$ (nicht optimal...)

Bruchteil-Rucksackproblem

- Gegeben: $n \in \mathbb{N}$, $G \in \mathbb{N}$, Vektoren $(v_1, \dots, v_n), (g_1, \dots, g_n) \in \mathbb{N}^n$
- Gesucht: Wie groß kann der 'Wert' von Objekten werden, deren Gewicht $\sum_{i \in I} g_i$ nicht größer als die Schranke G ?
- Aber: **Jetzt dürfen Objekte zerteilt werden!** \rightsquigarrow Relaxation
- als Lösung daher jetzt ein Vektor (a_1, \dots, a_n) mit $a_i \in \mathbb{Q}$ und $0 \leq a_i \leq 1$ (statt einer Menge I)
- formal: bestimme

$$\max\left\{\sum_{i=1}^n a_i v_i \mid a_i \in [0, 1] \text{ mit } \sum_{i=1}^n a_i g_i \leq G\right\}$$

Greedy-Ansatz für das Bruchteil-Rucksackproblem:

Sortiere alle Objekte nach Wert v_i/g_i , d.h. $\frac{v_1}{g_1} \geq \frac{v_2}{g_2} \geq \dots \geq \frac{v_n}{g_n}$
dann nimm soviel von den hochwertigen Objekten, wie möglich...

Algorithmische Lösung:

$k := 0$

WHILE $\sum_{i=1}^{k+1} g_i \leq G$ DO $k := k + 1$ ENDWHILE

$b := (G - \sum_{i=1}^k g_i) / g_{k+1}$

Optimale Lösung ist

$(a_1, a_2, \dots, a_n) = (\underbrace{1, 1, \dots, 1}_k, b, \underbrace{0, 0, \dots, 0}_{n-k-1})$

Grundidee zu Nachweis der Optimalität der Lösung:

- Sei $(a'_1, a'_2, \dots, a'_n)$ eine andere Lösung.
- Betrachte $j := \min\{i \mid a'_i \neq a_i\}$
- Dann $\sum_{i=1}^j g_i < G$, also gibt es $m > j$ mit $a'_m > 0$.
- Transferiere dann Gewicht von Objekt m zu Objekt j ,
(d.h. verkleinere a'_m und vergrößere a'_j soweit wie möglich)
- die entstehende Lösung wäre nicht schlechter...,
aber 'näher' an (a_1, a_2, \dots, a_n)

Allgemeine Form der Probleme, bei denen Greedy anwendbar ist:

E sei endliche Menge, \mathcal{U} sei eine Menge von Teilmengen von E .
Die Struktur (E, \mathcal{U}) heißt *Teilmengensystem*, wenn

- $\emptyset \in \mathcal{U}$
- $A \subseteq B \wedge B \in \mathcal{U} \Rightarrow A \in \mathcal{U}$

$w : E \rightarrow \mathbb{Q}$ sei eine Kostenfunktion,
gesucht wird eine in \mathcal{U} maximale Menge T (bzgl. \subseteq) mit maximalen Gesamtkosten

$$w(T) = \sum_{e \in T} w(e)$$

- Die Rolle der Gewichtsschranke G übernimmt das Teilmengensystem \mathcal{U} .
- Dabei ist $T \in \mathcal{U}$ maximal in \mathcal{U} , wenn kein $T' \in \mathcal{U}$ mit $T \subset T'$ existiert.

Kanonischer Greedy-Algorithmus für Teilmengensysteme:

Ordne die Elemente in $E = \{e_1, \dots, e_n\}$,

so dass $w(e_1) \geq w(e_2) \geq \dots \geq w(e_n)$

Setze $T := \emptyset$.

FOR $k := 1$ TO n DO

 IF $(T \cup \{e_k\}) \in \mathcal{U}$ THEN $T := T \cup \{e_k\}$

ENDFOR

Ausgabe von T als Lösung

Beispiel: Eine Menge $I \subseteq V$ heißt *unabhängig* im Graphen $G = (V, E)$ gdw. zwei verschiedene $x, y \in I$ sind nicht durch eine Kante verbunden.

Die unabhängigen Knotenmengen bilden ein Teilmengensystem.

Charakterisierung der Probleme, die Greedy-Algorithmus optimal löst:

Ein Teilmengensystem (E, \mathcal{U}) heißt *Matroid*,
wenn zusätzlich die *Austauscheigenschaft* gilt:

$$A, B \in \mathcal{U} \wedge |A| < |B| \Rightarrow (\exists x \in B \setminus A) A \cup \{x\} \in \mathcal{U}$$

- ‘Matroid’ verallgemeinert Begriff ‘(lineare) Unabhängigkeit’
- In Matroiden haben maximale Mengen gleiche Mächtigkeit!
- Unabhängige Knotenmengen in Graphen bilden i.Allg. kein Matroid.

Beispiel für Matroide:

- Seien $n, k \in \mathbb{N}$ gegeben, $k \leq n$
- Betrachte $E = \{1, 2, \dots, n\}$
- Setze $\mathcal{U} = \{A \subseteq E : |A| \leq k\}$

Maximale Menge sind hier die k -elementigen Mengen, die Austauscheneigenschaft ist offensichtlich erfüllt.

Weitere Beispiele für Matroide (E, \mathcal{U})

- E : endliche Menge von Vektoren,
 \mathcal{U} : linear unabhängige Teilmengen von E
(z.B.: E besteht aus Spalten einer Matrix, daher auch 'Matroid')
- E : Kantenmenge eines endlichen Graphen G ,
 \mathcal{U} : kreisfreie Teilmengen von E
(auch *graphisches Matroid* genannt)
Maximale Elemente von \mathcal{U} sind aufspannende Wälder von G .

Bedeutung der Austauscheneigenschaft:

- A und B seien maximale Elemente im Matroid (E, \mathcal{U})
- Damit $|A| = |B|$
- Falls $A \neq B$:
 - Wähle $a \in A \setminus B$.
 - Zu $A \setminus \{a\}$ gibt es $b \in B \setminus A$ mit $A' := A \setminus \{a\} \cup \{b\} \in \mathcal{U}$
 - $A' \setminus B$ ist kleiner als $A \setminus B$, $A' \cap B$ ist größer als $A \cap B$
- Also: Man kann A durch Austausch einzelner Elemente schrittweise in B umwandeln, ohne dabei \mathcal{U} zu verlassen!

Satz: Sei (E, \mathcal{U}) ein Teilmengensystem.

Der kanonische Greedy-Algorithmus liefert beim Optimierungsproblem genau dann **für jede beliebige** Kostenfunktionen $w : E \rightarrow \mathbb{Q}$ die optimale Lösung, wenn (E, \mathcal{U}) ein Matroid ist.

Beweis von “ \Leftarrow ”:

- gegeben: (E, \mathcal{U}) Matroid, $w : E \rightarrow \mathbb{Q}$ Gewichtsfunktion
- O.B.d.A.: bereits Ordnung $w(e_1) \geq w(e_2) \dots \geq w(e_n)$ vorhanden
- $T = \{e_{i_1}, \dots, e_{i_k}\}$ sei Lösung durch Greedy-Algorithmus.
- Annahme: Greedy-Algorithmus liefert nicht die optimale Lösung.
- $T' = \{e_{j_1}, \dots, e_{j_k}\}$ sei bessere Lösung, d.h. $w(T') > w(T)$.
- O.B.d.A.: $i_1 < i_2 < \dots < i_k$ und $j_1 < j_2 < \dots < j_k$
- Also existiert minimales μ mit $w(e_{j_\mu}) > w(e_{i_\mu})$, insbesondere dabei $j_\mu < i_\mu$
- Wende Austauscheneigenschaft auf $A = \{e_{i_1}, \dots, e_{i_{\mu-1}}\}$ und $B = \{e_{j_1}, \dots, e_{j_\mu}\}$ an:
Es gibt $e_{j_\sigma} \in B \setminus A$ mit $A \cup \{e_{j_\sigma}\} \in \mathcal{U}$
- Mit $\sigma \leq \mu$ jedoch $w(e_{j_\sigma}) \geq w(e_{j_\mu}) > w(e_{i_\mu})$, d.h. Greedy-Algorithmus hätte e_{j_σ} vor e_{i_μ} in T aufnehmen müssen. Widerspruch!

Indirekter Beweis von “ \Rightarrow ”:

- Annahme: Austausch Eigenschaft gilt nicht, aber Greedy liefert für jedes w optimale Lösung.
- Also gibt es $A, B \in \mathcal{U}$ mit $(\forall b \in B \setminus A) A \cup \{b\} \notin \mathcal{U}$.
- Setze $r := |B|$ und betrachte folgende Kostenfunktion w :

$$w(e) := \begin{cases} r+1, & e \in A \\ r, & e \in B \setminus A \\ 0, & \text{sonst} \end{cases}$$

- Greedy-Algorithmus: Lösung T mit $A \subseteq T$ und $T \cap (B \setminus A) = \emptyset$.
- Wegen $B \in \mathcal{U}$ gibt es auch eine Lösung T' mit $B \subseteq T'$
- Dann jedoch

$$\begin{aligned} w(T) &= (r+1) \cdot |A| \leq (r+1) \cdot (r-1) = r^2 - 1 \\ w(T') &\geq r \cdot |B| = r^2 \end{aligned}$$

Also $w(T) < w(T')$, d.h. Greedy versagt bei diesem w . Widerspruch!

Folgerungen

- Teilmengensysteme erlauben die Anwendung von Greedy-Algorithmen
- Matroide führen zu optimaler Lösung
- Oft jedoch: Austausch Eigenschaft nicht vollständig erfüllt
(z.B.: \exists mehrere maximale Mengen mit unterschiedlicher Mächtigkeit)
also Greedy-Lösung nicht optimal, sondern nur approximativ
- mögliches anderes Problem:
nicht-additive Kostenfunktion, $w(T) \neq \sum_{e \in T} w(e)$
- aber: Greedy liefert oft gute Heuristik!

Im Folgenden: Beispiele für 'gute' Greedy-Heuristiken bei Färbbarkeit und TSP.

Färbbarkeit

- Gegeben Graph $G = (V, E)$
- Ziel: Färbe Knoten mit möglichst wenigen Farben, d.h. suche 'kleines' $k \in \mathbb{N}$ und Funktion $f : V \rightarrow \{0, \dots, k-1\}$, so dass für Kanten $(u, v) \in E$ stets $f(u) \neq f(v)$.

- Greedy-Heuristik:

- Sortiere V nach Grad $\delta(v)$ der Knoten $v \in V$, also o.B.d.A.: $V = \{v_1, \dots, v_n\}$ mit $\delta(v_i) \geq \delta(v_{i+1})$.
- Wiederhole für $\mu = 1, 2, \dots, n$: Setze

$$f(\mu) := \min(\mathbb{N} \setminus \{f(i) \mid i < \mu, (v_i, v_\mu) \in E\})$$

- Jeder Knoten erhält also, in der Reihenfolge absteigender Grade, die jeweils kleinstmögliche Farbe.
- Jeder Graph mit maximalem Grad Δ wird dabei mit höchstens $\Delta + 1$ Farben gefärbt!

“Klar”: 2-färbbare (d.h. bipartite) Graphen sind in Polynomzeit färbbar! Damit:
Lemma: Jeder 3-färbbare Graph mit n Knoten und Grad Δ kann in Polynomzeit mit $\mathcal{O}(\min(\Delta, \sqrt{n}))$ Farben gefärbt werden.

Beweis dazu:

- Ist G 3-färbbar, dann ist bei jedem Knoten v die Menge $N(v)$ seiner Nachbarn 2-färbbar!
- Für jeden Knoten $v \in V$ mit $\delta(v) \geq \sqrt{n}$ ist $\bar{N}(v) := N(v) \cup \{v\}$ also schnell mit 3 (neuen) Farben färbbar.
Danach entferne $\bar{N}(v)$ aus dem Graphen.
- Da $|N(v)| \geq \sqrt{n}$: Nach maximal \sqrt{n} Iterationen kein Knoten mehr in G mit $\delta(v) \geq \sqrt{n}$.
- Rest des Graphen dann mit $\leq \sqrt{n}$ weiteren Farben färbbar!

Bestes bekanntes Resultat (Baumann, 2004): Färbung 3-färbbarer Graphen in Polynomzeit mit $\mathcal{O}(n^{3/14})$ Farben. <http://archiv.tu-chemnitz.de/pub/2004/0142/data/Diplomarbeit-TobiBaumann.pdf>

Greedy-Algorithmen für TSP

- Grundidee: Konstruiere sukzessive immer längere Strecken, bis schließlich eine Rundreise entsteht...
- Heuristik 'Nearest Neighbor':
 - Gehe vom aktuellen Tour-Ende immer zur nächstliegenden neuen Stadt
- Heuristik 'Tourerweiterungen':
 - Starte mit kurzer Tour aus zwei Städten.
 - Erweitere Tour iterativ um je einen Knoten; Heuristiken dabei:
 - 'Random Insertion': zufällige Knotenwahl
 - 'Farthest Insertion': Maximiere Minimalabstand zur aktuellen Route
 - Gewählter Knoten wird an optimaler Position in die Tour eingefügt.
- 'Nearest Neighbor' ist nur mäßig erfolgreich: Anfangs viele kurze Strecken, aber am Ende kostspieliges Einsammeln übersehener, weit auseinanderliegender Städte (aber mit Nachverbesserungen evtl. brauchbar, s. lokale Suche)
- 'Farthest Insertion' scheint die bessere Heuristik zu sein: Grobstruktur der Route wird relativ schnell festgelegt!

Lokale Suche zur Verbesserung von gefundenen Lösungen:

- Greedy-Algorithmus liefert Startwert für Maximumsuche...
- Ziel wie bei der Austausch Eigenschaft: teste lokale Änderungen
d.h.: tausche beliebig $a \in A$ gegen $b \in B \setminus A$
- z.B.: nach Konstruktion einer Lösung: entferne Teile der Lösung und suche eine 'Zeitlang' nach Verbesserungen

Grundstruktur:

Erzeuge eine Anfangslösung L .

REPEAT

 Modifiziere L zufällig zu L'

 IF L' ist besser als L THEN $L = L'$ ENDIF

UNTIL längere Zeit keine Verbesserung gefunden

AUSGABE von L

Prinzipieller Nachteil:

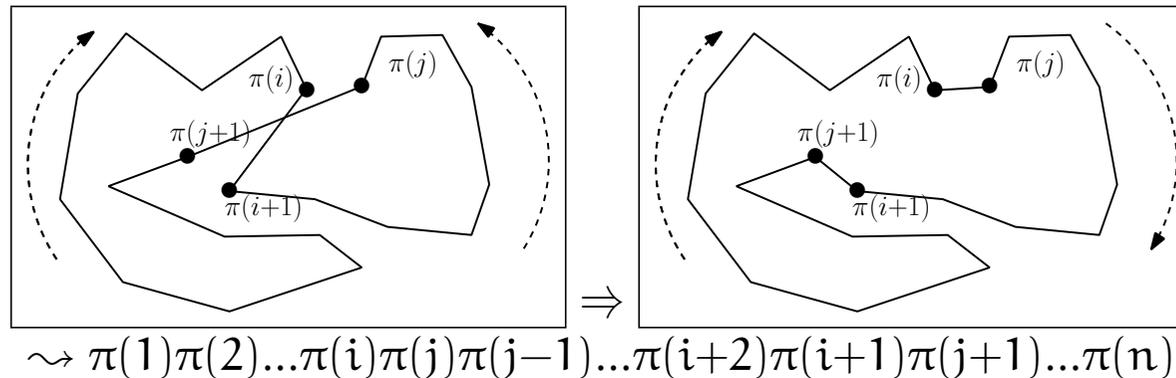
- die gefundene Lösung ist ein 'lokales Optimum';
- das globale Optimum kann beliebig weit entfernt sein!

2-Opt-Heuristik bei TSP

- Betrachte TSP mit symmetrischen Entfernungen $m_{i,j} = m_{j,i}$ (ungerichteter Graph)
- Lösungen mit unendlichen Kosten seien erlaubt!
- Beginne mit beliebiger Permutation π der Knoten (zulässig!)
- Wähle zufällig zwei Knoten $\pi(i)$, $\pi(j)$ (mit $i < j$). Falls

$$m_{\pi(i),\pi(i+1)} + m_{\pi(j),\pi(j+1)} > m_{\pi(i),\pi(j)} + m_{\pi(i+1),\pi(j+1)}$$

gilt, dann gibt es eine kürzere Route / bessere Permutation:



- Animation: www-e.uni-magdeburg.de/mertens/TSP/TSP.html
- Laufzeit und Güte der Heuristik sind unbekannt!

Weitere Meta-Heuristiken:

- Grundidee des *Simulated Annealing* (simulierte Abkühlung):
 - Erlaube bei lokaler Suche auch Zwischenlösungen, die um $c(t)$ schlechter sind als die aktuelle Lösung;
 - aber: $c(t)$ geht mit wachsender Rechenzeit gegen 0, etwa

$$c(t) = \mathcal{O}(1/\log t)$$

- Grundidee der *genetischen Algorithmen*:
 - Speichere mehrere verschiedenartige Lösungen.
 - Kombiniere jeweils mehrere Lösungen und verwende die besseren Resultate.