Matroide

Beispiel – Der graphische Matroid

Sei $G = (V, E)$ ein ungerichteter Graph.

Sei $\mathcal{F} = \{\, F \subseteq E \mid (V, F) \text{ ist azyklisch} \,\}$.

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

Zum Beweis machen wir zunächst eine Beobachtung:
Es sei $G = (V, E)$ ein Wald. Dann verbindet die Kante $e \in E$ zwei Bäume in G gdw. $G = (V, E \cup \{e\})$ kreisfrei ist.

Beweis.

- Vererbungseigenschaft: Wegnehmen von Kanten kann keine Kreise schließen.
- Austauscheigenschaft: Seien $G_A = (V, A)$ und $G_B = (V, B)$ Teilwälder von G und $|A| < |B|$
  1. Beobachtung: Ein Wald mit k Kanten besteht aus $|V| - k$ Bäumen.
  2. G_A hat mehr Bäume als G_B
  3. Es gibt einen Baum T in G_B, der zwei Bäume in G_A verbindet
  4. Es gibt eine Kante in T, die keinen Kreis in G_A schließt

$\square$

Matroide

Wir nennen eine unabhängige Menge A maximal, wenn keine echte Obermenge von A unabhängig ist.

Lemma (Lemma G1)

Alle maximalen unabhängigen Mengen eines Matroids haben die gleiche Größe.

Beweis.

Angenommen, daß A und B maximale unabhängige Mengen sind, aber $|A| < |B|$. Nach der Austauscheigenschaft gibt es ein $x \in B$, so daß $A \cup \{x\}$ unabhängig ist. Das widerspricht der Maximalität von A. $\square$

Gewichtete Matroide

- Ein gewichtetes Matroid ist ein Matroid $M = (S, \mathcal{I})$ mit einer Gewichtsfunktion $w \colon S \to \mathbf{Q}$.
- Wir nennen eine Menge maximalen Gewichts unter allen maximalen unabhängigen Mengen optimal.
- Viele Optimierungsprobleme können durch das Finden einer optimalen Menge in einem Matroid gelöst werden.

Beispiel

Ein minimaler Spannbaum ist eine optimale Menge im graphischen Matroid.

Der Greedy-Algorithmus auf gewichteten Matroiden

Sei $M = (S, \mathcal{I})$ ein Matroid mit Gewichten w.

Dieser Algorithmus findet eine optimale Menge.

Algorithmus

function Greedy(S) :
R $:= \emptyset$;
sort S into(s[1], ..., s[n]) with w(s[i]) $\geq$ w(s[i + 1]);
for i = 1, ..., n **do**
  if R $\cup$ { s[i]} $\subseteq$ I **then** R := R $\cup$ { s[i]}**fi**
od;
return R

Die Laufzeit ist $O(n \log n + nf(n))$, wenn ein Vergleich konstante Zeit braucht, und der Unabhängigkeitstest $O(f(n))$.

Korrektheit

Lemma (G2)

Sei $M = (S, \mathcal{I})$ ein Matroid mit Gewichten w. Wenn $x \in S$ maximales Gewicht unter allen x hat, für die $\{x\}$ unabhängig ist, dann gibt eine optimale Menge, die x enthält.

Beweis.

Sei B eine optimale Menge mit $x \notin B$. Wir haben $w(y) \leq w(x)$ für jedes $y \in B$, weil $\{y\}$ nach der Vererbungseigenschaft unabhängig ist. Beginne mit $A = \{x\}$ und füge Elemente von B zu A hinzu (A bleibt unabhängig, wegen der Austauscheigenschaft) bis $|A| = |B|$.
Dann ist $A = B - \{y\} \cup \{x\}$ für ein $y \in B$ und daher gilt $w(A) \geq w(B)$.

□

Kontraktion eines Matroids

Definition

Sei $M = (S, \mathcal{I})$ ein Matroid und $x \in S$.
Dann ist $M' = (S', \mathcal{I}')$ die Kontraktion von M um x, wobei

- $S' = \{\, y \in S \mid \{x, y\} \in \mathcal{I}, x \neq y \,\}$,
- $\mathcal{I}' = \{\, A \subseteq S - \{x\} \mid A \cup \{x\} \in \mathcal{I} \,\}$.

Wir beobachten, daß der Greedy-Algorithmus auf M' arbeitet, nachdem er x gewählt hat.

Lemma (G3)

Sei $M = (S, \mathcal{I})$ ein Matroid mit Gewichten w. Sei x das erste Element, das durch den Greedy-Algorithmus gewählt wird.
Dann ist eine optimale Menge für die Kontraktion M' von M um x zusammen mit x eine optimale Menge für M.

Beweis.

Sei $x \in A$ und $B = A - \{x\}$. Dann ist A maximal unabhängig in M gdw. B maximal unabhängig in M' ist.
Da $w(A) = w(B) + w(x)$ gilt, ist A optimal in M gdw. B optimal in M' ist. $\square$

Korrektheit des Greedy-Algorithmus

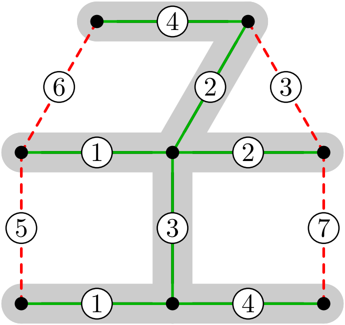
Theorem

Der Greedy-Algorithmus berechnet eine optimale Menge in einem gewichteten Matroid.

Beweis.

Aus $G2$ und $G3$ folgt die Korrektheit mittels Induktion über die Größe der maximalen unabhängigen Menge, die nach $G1$ wohldefiniert ist. $\square$

Kruskal: Minimaler Spannbaum



Kruskals Algorithmus – Implementierung

Algorithmus

function Kruskal(G, w) :
$A := \emptyset$;
for each vertex v in V[G] **do**
  Make_Set(v)
od;
sort the edges of E into nondecreasing order by weight;
for each edge $\{u, v\}$ in E, nondecreasing **do**
  if Find_Set(u) $\neq$ Find_Set(v) **then**
    $A := A \cup \{\{u, v\}\}$;
    Union(u, v)
  fi
od;
return A