Algorithmische Graphentheorie

Felix Brandt, Jan Johannsen

Vorlesung im Wintersemester 2008/09

Übersicht

Übersicht

Matchings

Definition

Ein Matching in $G = (V, E)$ ist eine Menge $M \subseteq E$ mit

$$e_1 \cap e_2 = \emptyset \quad \text{für } e_1, e_2 \in M,\ e_1 \neq e_2$$

Ein Matching M ist perfekt, falls $|M| = |V|/2$ ist.

$v \in V$ heißt M-frei, wenn v mit keiner Kante in M inzidiert.

Probleme:

- Entscheide ob G ein perfektes Matching enthält.
- Finde ein Matching maximaler Größe in G.

$\nu(G) := \max\{|M|,\ M \text{ Matching in } G\}$

Ungerade Komponenten

$uk(G)$ bezeichnet die Anzahl der ungeraden Zusammenhangskomponenten von G.

Lemma

Für alle $S \subseteq V$ gilt

$$\nu(G) \leq \frac{1}{2}\big(|V| - uk(G - S) + |S|\big)$$

Korollar

Falls es $S \subseteq V$ gibt mit $uk(G - S) > |S|$,
dann hat G kein perfektes Matching.

Augmentierende Wege

Ein Weg $W = (v_1, \ldots, v_k)$ ist M-alternierend, wenn gilt

$$\{v_i, v_{i+1}\} \in M \leftrightarrow \{v_{i+1}, v_{i+2}\} \notin M \text{ für alle } i$$

W ist M-augmentierend, wenn v_1 und v_k M-frei sind.

Dann ist $M' = M \triangle E(W)$ ein Matching mit $|M'| = |M| + 1$.

Satz

Ein Matching M ist maximal
gdw. es keinen M-augmentierenden Weg gibt.

Generischer Algorithmus

Generischer Algorithmus:

$M := \emptyset$
`while` $\exists$ augmentierender Weg W
    $M := M \triangle E(W)$

Berechnet ein maximales Matching.

Gebraucht wird also nur Algorithmus für:

- Stelle fest, ob augmentierender Weg existiert,
- falls ja, finde einen.

Einfacher Algorithmus

Konvention: In bipartiten Graphen ist $V = A \uplus B$ mit $|A| \leq |B|$ und $|e \cap A| = |e \cap B| = 1$ für alle $e \in E$.

Matching M ist perfekt, falls $|M| = |A|$.

Gegeben: bipartiter Graph $G = (V, E)$, Matching M.

Konstruiere gerichteten Graphen G' mit Kanten:

$$\begin{array}{ll} (b, a) & \text{falls } \{a, b\} \in M \\ (a, b) & \text{falls } \{a, b\} \in E \setminus M \end{array}$$

für $a \in A$ und $b \in B$.

Finde ein M-freies $s \in A$.

Breitensuche ausgehend von s in G' findet kürzesten augmentierenden Weg, falls es einen gibt.

Alternierende Bäume

Sei $s \in V$ M-frei.

Ein alternierender Baum ist ein Baum T mit Wurzel s und:

- jeder Pfad in T ist ein alternierender Weg,
- kein Knoten $v \neq s$ in T ist M-frei.

Seien $even(T)$ und $odd(T)$ die Mengen der $v \in T$ gerader bzw. ungerader Tiefe.

Es gilt:

$$|even(T)| = |odd(T)| + 1$$

Algorithmus

Sei G bipartit, M Matching in G und T ein alternierender Baum.

Es gibt zwei Fälle:

- Fall 1: es gibt $u \in \mathit{even}(T)$ und $\{u, v\} \in E$ mit $v \notin T$.
  - Fall 1a: v ist M-frei.

    $\rightsquigarrow$ augmentierender Weg.
  - Fall 1b: v ist von M überdeckt.

    $\rightsquigarrow$ T kann vergrößert werden.
- Fall 2: für alle $u \in \mathit{even}(T)$ und $\{u, v\} \in E$ gilt $v \in \mathit{odd}(T)$.

  In diesem Fall heißt T verkümmert.

Satz

Ist T ein verkümmerter Baum in T,
dann gibt es in T kein perfektes Matching.

Hall's Theorem

Der obige Algorithmus findet ein perfektes Matching oder stellt fest, dass keines existiert, in Zeit $O(nm)$.

Er liefert auch einen einfachen Beweis für den Satz von Hall:

Satz

Sei G bipartit. Es gibt genau dann ein perfektes Matching in G, wenn für alle $S \subseteq A$ gilt $|N(S)| \geq |S|$.

Ungerade Kreise

In nicht-bipartiten Graphen kann ein weiterer Fall eintreten:

- Fall 2b: es gibt $u \in \mathit{even}(T)$ und $v \in \mathit{even}(T)$ mit $\{u, v\} \in E$.

Dann existiert einen Kreis ungerader Länge!

Problem: augmentierende Wege werden dann nicht immer gefunden.

Kontraktion ungerader Kreise

Lösung: Ungerade Kreise werden kontrahiert.

Definition

Sei C ein ungerader Kreis in G

G/C ist definiert als $G[V \setminus C]$ plus

- neuer Knoten $\{v_C\}$
- neue Kanten: $\{u, v_C\}$ für jede Kante $\{u, v\}$ mit $u \notin C$ und $v \in C$

Knoten v_C heißt Superknoten in G/C.

Augmentierende Wege in G/C

Satz

Ist M' ein Matching in G/C, dann gibt es ein Matching M in G so daßs

- *Anzahl der M-freien Knoten in G*
  $=$ *Anzahl der M'-freien Knoten in G/C.*

Verkümmerte Bäume

G' ist abgeleitet von G, wenn es $G = G_1, \ldots, G_k = G'$ gibt mit

$$G_{i+1} = G_i / C_i \qquad \text{für alle } i$$

wobei C_i ungerader Kreis in G_i ist.

Satz

Sei G' abgeleitet von G, M' Matching in G'
und T ein alternierender Baum in G',
so dass $\mathrm{odd}(T)$ keine Superknoten enthält.

Ist T verkümmert, so hat G kein perfektes Matching.

Der Algorithmus

Input: Graph G, Matching M in G

1 $G' := G$
2 `if` M perfekt `return` M
3 wähle $s \in V$ M-frei, $T := (\{s\}, \emptyset)$
4 `if` $\exists \{u, v\} \in E'$ mit $u \in even(T)$ und $v \notin odd(T)$
5 `if` v M-frei
6 vergrößere M entlang des Pfades von s zu v
7 erweitere M zu Matching auf G; goto 2
8 `if` $\exists \{v, w\} \in M$
9 erweitere T um $\{u, v\}$ und $\{v, w\}$; goto 4
10 `if` $v \in even(T)$
11 $C :=$ ungerader Kreis in $T + \{u, v\}$
12 $G' := G/C$; $M := M \setminus E(C)$
13 $T := (T + \{u, v\})/C$; goto 4
14 return "G hat kein perfektes Matching"

Komplexität des Algorithmus

Satz

Der obige Algorithmus entscheidet korrekt,ob G ein perfektes Matching hat.
Er führt $O(n)$ Vergröerungen des Matchings und $O(n^2)$ Kontraktionen und Baumwachstumsschritte durch.

Der Algorithmus kann mit Laufzeit $O(nm\alpha(n))$ implementiert werden.

$\alpha(n)$ ist die inverse Ackermannfunktion.

Es ist $\alpha(n) \leq 4$ für $n \leq 10^{600}$.

Dazu: UNION-FIND-Datenstruktur

Exkurs: Union-Find-Datenstruktur

Es soll eine Familie von disjunkten Mengen verwaltet werden:

$$M_1, \ldots, M_k \quad \text{mit} \quad M_i \cap M_j = \emptyset \text{ für } i \neq j$$

Dabei sollen folgende Operationen zur Verfügung stehen:

- Make-Set(x) fügt die Menge $\{x\}$ zur Familie hinzu.
- Find-Set(x) liefert ein kanonisches Element der Menge M_i mit $x \in M_i$, also Find-Set$(x) =$ Find-Set(y), falls x und y in der gleichen Menge M_i liegen.
- Union(x, y) ersetzt die Mengen M_i und M_j mit $x \in M_i$, $y \in M_j$ durch ihre Vereinigung $M_i \cup M_j$.

Exkurs: Union-Find-Datenstruktur

Union-Find-Datenstruktur kann so implementiert werden, dass gilt:

Satz

Jede Folge von m Make-Set-, Union- und Find-Set-Operationen, von denen n Make-Set sind, hat eine Laufzeit von $O(m\alpha(n))$.

Maximale Matchings

Input: Graph G

1 $G^* := G$; $M, M^* := \emptyset$
2 $(M', G', T) := Algorithmus(G^*, M^*)$
3 if M' nicht perfekt
4 $\quad G^* := G' \setminus T$
5 $\quad$ if $V^* \neq \emptyset$
6 $\quad\quad M^* := M' \setminus E(T)$
7 $\quad\quad M := M \cup (M' \cap E(T))$
8 $\quad\quad$ goto 2
9 $M := M \cup M'$
10 return M

Maximale Matchings

Satz

Der obige Algorithmus berechnet ein maximales Matching in Laufzeit $O(nm\alpha(n))$.

Korollar

Tutte-Berge-Formel:

$$\nu(G) = \min_{S \subseteq V} \frac{1}{2}\big(|V| - uk(G - S) + |S|\big)$$