Kap. I.2: Matching in Graphen

Teil 2: Allgemeine Graphen

Professor Dr. Petra Mutzel

Abteilung für Computational Analytics

Institut für Informatik (Abt. 1)

Universität Bonn

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

INSTITUT FÜR INFORMATIK DER UNIVERSITÄT BONN

SoSe 2024 VO 4/5 30. April / 7. Mai 2024

Outline

1. **Perfektes Matching für allgemeine Graphen**
2. **Maximum Matching**
   - Algorithmus für maximum Matching
   - Sätze von Tutte und Berge
3. **Historische Anmerkungen**

Table of Contents

Wdhlg.: Perfektes Matching für bipartite Graphen

Perfektes Matching Algorithmus für bipartite Graphen (PMB)

$M \longleftarrow \emptyset$;
Wähle beliebigen Knoten $r \in V(G)$;
$T \longleftarrow (\{r\}, \emptyset)$;
while ($\exists$ $vw \in E$ mit $v \in B(T)$, $w \notin V(T)$) {
    if (w ist M-exponiert) {
        Benutze vw zur M-Augmentierung;
        if ($\nexists$ M-exponierter Knoten in G)
            STOP „M ist ein perfektes Matching";
        else $T \longleftarrow (\{r\}, \emptyset)$ für einen M-exponierten Knoten r;
    }
    else Benutze vw zur Baumerweiterung;
}
STOP „G hat kein perfektes Matching";

Wichtig: Wir suchen immer nur von B-Knoten aus

Korrektheit im bipartiten Fall

Lemma (Korrektheit des Algorithmus PMB (Lemma 3))

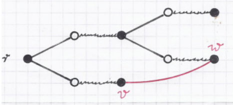
Ist im Algorithmus PMB die **while**-*Bedingung nicht erfüllt, so hat G kein perfektes Matching.*

Beweis Wir zeigen, dass T frustriert ist (d.h., jede Kante in $E(G)$ mit einem Ende in $B(T)$ hat das andere Ende in $A(T)$), dann folgt mit Lemma 2, dass kein perfektes Matching existiert.

Die **while**-Bedingung sei nicht erfüllt: Es gibt kein $vw \in E$ mit $v \in B(T)$ und $w \notin V(T)$.

Also gilt für alle $vw \in E$ mit $v \in B(T)$: $w \in A(T)$ oder $w \in B(T)$.

Wäre $w \in B(T)$, so hätten wir einen ungeraden Kreis in G und G wäre nicht bipartit:



Also gilt $w \in A(T)$ für alle $vw \in E$ mit $v \in B(T)$.

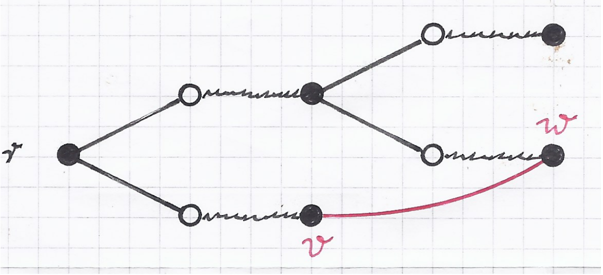
Deshalb ist T frustriert und wir können mit Lemma 2 schließen, dass G kein perfektes Matching besitzt. $\square$

Perfektes Matching für allgemeinen Fall

Kreise ungerader Länge können ein Problem sein!

Besonders dann, wenn für alle $vw \in E$ mit $v \in B(T)$: $w \in B(T)$

$\Rightarrow$ Baum ist nicht frustriert, kann aber auch nicht erweitert werden



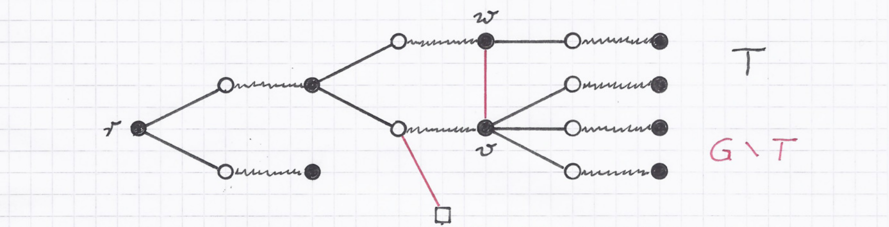
Vielleicht existiert kein größeres Matching in diesen Fällen?

Perfektes Matching für allgemeinen Fall

Kreise ungerader Länge können ein Problem sein!

Besonders dann, wenn für alle $vw \in E$ mit $v \in B(T)$: $w \in B(T)$

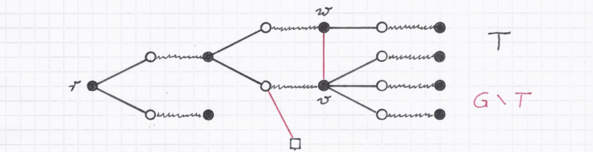
$\Rightarrow$ Baum ist nicht frustriert, kann aber auch nicht erweitert werden



Die Baumkanten sind schwarz, die restlichen Kanten rot gezeichnet.

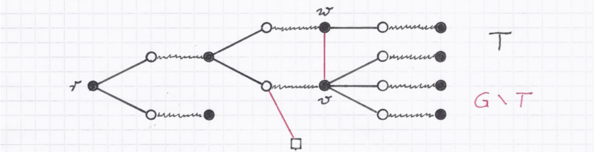
Perfektes Matching für allgemeine Graphen

Kreise ungerader Länge können ein Problem sein!



Perfektes Matching für allgemeine Graphen

Kreise ungerader Länge können ein Problem sein!



Idee von Jack Edmonds (1965): „Heureka, you shrink"

- Kontraktion der ungeraden Kreise („Blüten") zu einem Knoten
- Achtung: wiederholte Kontraktion führt zu „Blütenhierarchie"
- Diese Idee führte zu dem ersten polynomiellen Matching-Algorithmus in allgemeinen Graphen

Schrumpfen ungerader Kreise

Eine Schlüsseloperation ist das Schrumpfen ungerader Kreise:

- Sei C ungerader Kreis in G.

Schrumpfen ungerader Kreise

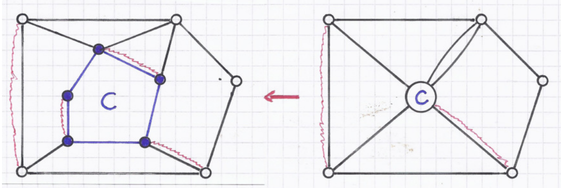
Eine Schlüsseloperation ist das Schrumpfen ungerader Kreise:

- Sei C ungerader Kreis in G.
- Sei $G' := G \times C$ der Subgraph von G, der durch das Schrumpfen des Kreises C entsteht.

Schrumpfen ungerader Kreise

Eine Schlüsseloperation ist das Schrumpfen ungerader Kreise:

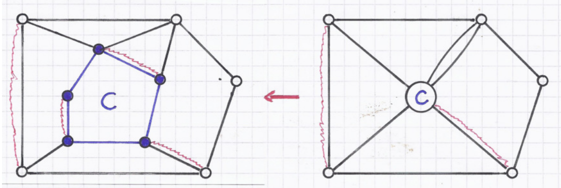
- Sei C ungerader Kreis in G.
- Sei $G' := G \times C$ der Subgraph von G, der durch das Schrumpfen des Kreises C entsteht.
- Dann erhalten wir $V(G') = (V \setminus V(C)) \cup \{c\}$, wobei c als neuer Knoten („Pseudoknoten") aufgefasst wird, sowie $E(G') = E \setminus E(V(C))$ wobei die Endknoten in $V(C)$ durch c ersetzt sind:



Schrumpfen ungerader Kreise

Eine Schlüsseloperation ist das Schrumpfen ungerader Kreise:

- Sei C ungerader Kreis in G.
- Sei $G' := G \times C$ der Subgraph von G, der durch das Schrumpfen des Kreises C entsteht.
- Dann erhalten wir $V(G') = (V \setminus V(C)) \cup \{c\}$, wobei c als neuer Knoten („Pseudoknoten") aufgefasst wird, sowie $E(G') = E \setminus E(V(C))$ wobei die Endknoten in $V(C)$ durch c ersetzt sind:

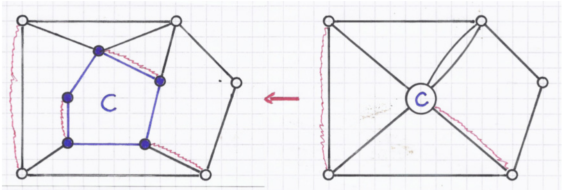


- Offensichtlich induziert ein Matching im rechten Bild eines mit dem gleichen Defizit im linken Bild und umgekehrt.

Entschrumpfen der Pseudoknoten

Entschrumpfen des Pseudoknoten c und Erweitern des Matchings M:

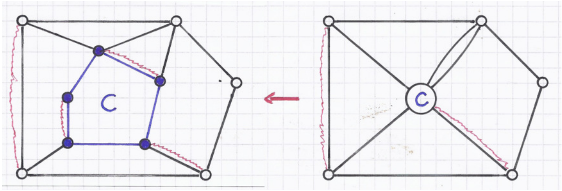
- Sei c der Pseudoknoten, der durch Schrumpfen eines ungeraden Kreises $C = (c_1, \ldots, c_k)$ entstanden ist; nach der Augmentierung ist c nicht M-exponiert, besitzt also genau eine Matchingkante;



Entschrumpfen der Pseudoknoten

Entschrumpfen des Pseudoknoten c und Erweitern des Matchings M:

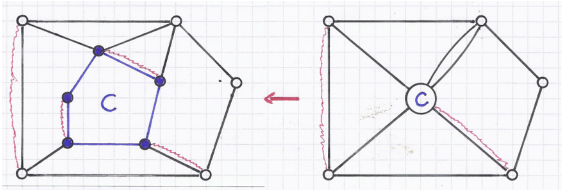
- Sei c der Pseudoknoten, der durch Schrumpfen eines ungeraden Kreises $C = (c_1, \ldots, c_k)$ entstanden ist; nach der Augmentierung ist c nicht M-exponiert, besitzt also genau eine Matchingkante;



Entschrumpfen der Pseudoknoten

Entschrumpfen des Pseudoknoten c und Erweitern des Matchings M:

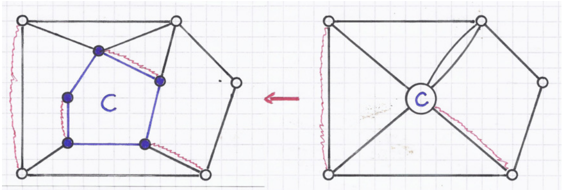
- Sei c der Pseudoknoten, der durch Schrumpfen eines ungeraden Kreises $C = (c_1, \ldots, c_k)$ entstanden ist; nach der Augmentierung ist c nicht M-exponiert, besitzt also genau eine Matchingkante;
- Wir ersetzen c durch den Originalkreis C und restaurieren die Originalkanten zu dem Restgraphen $G \setminus C$;



Entschrumpfen der Pseudoknoten

Entschrumpfen des Pseudoknoten c und Erweitern des Matchings M:

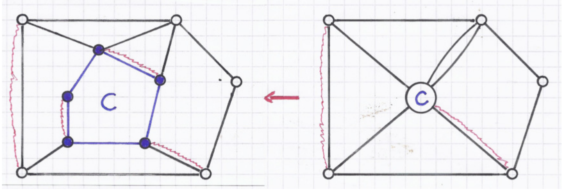
- Sei c der Pseudoknoten, der durch Schrumpfen eines ungeraden Kreises $C = (c_1, \ldots, c_k)$ entstanden ist; nach der Augmentierung ist c nicht M-exponiert, besitzt also genau eine Matchingkante;
- Wir ersetzen c durch den Originalkreis C und restaurieren die Originalkanten zu dem Restgraphen $G \setminus C$;
- Sei c_1 der einzige Knoten in $V(C)$, der mit einer Kante zu $w \in G \setminus C$ gematcht ist



Entschrumpfen der Pseudoknoten

Entschrumpfen des Pseudoknoten c und Erweitern des Matchings M:

- Sei c der Pseudoknoten, der durch Schrumpfen eines ungeraden Kreises $C = (c_1, \ldots, c_k)$ entstanden ist; nach der Augmentierung ist c nicht M-exponiert, besitzt also genau eine Matchingkante;
- Wir ersetzen c durch den Originalkreis C und restaurieren die Originalkanten zu dem Restgraphen $G \setminus C$;
- Sei c_1 der einzige Knoten in $V(C)$, der mit einer Kante zu $w \in G \setminus C$ gematcht ist
- Die beiden Nachbarkanten von c_1 in C bleiben frei; nun besetzen wir abwechselnd jede zweite Kante des Kreises mit einer Matchingkante, es werden also die Kanten $(c_2, c_3), (c_4, c_5) \ldots, (c_{k-1}, c_k)$ gematcht;



Kreisschrumpfung und Matchings

D.h. wir haben:

Lemma (Matchings nach Kreisschrumpfung (Lemma 4))

Sei C ein ungerader Kreis in G, $G' = G \times C$ und M' ein Matching in G'. Dann existiert ein Matching M in G mit $M \subseteq M' \cup E(C)$ und die Anzahl M-exponierter Knoten in G ist dieselbe wie die der M-exponierten Knoten in G'.

Kreisschrumpfung und Matchings

D.h. wir haben:

Lemma (Matchings nach Kreisschrumpfung (Lemma 4))

Sei C ein ungerader Kreis in G, $G' = G \times C$ und M' ein Matching in G'. Dann existiert ein Matching M in G mit $M \subseteq M' \cup E(C)$ und die Anzahl M-exponierter Knoten in G ist dieselbe wie die der M-exponierten Knoten in G'.

Es gilt also:

$$\mathrm{def}(G) = \mathrm{def}(G \times C)$$

bzw.

$$\nu(G) \geq \nu(G \times C) + \frac{|V(C)| - 1}{2}$$

Der Blütenschrumpfalgorithmus

Der Blütenschrumpfalgorithmus - im Original „blossom shrinking algorithm“ wurde in dem bahnbrechenden Artikel

> Jack Edmonds: Path, trees, and flowers, Canadian Journal of Mathematics, 17 (1965) 449–467.

veröffentlicht.

Der Blütenschrumpfalgorithmus

Der Blütenschrumpfalgorithmus - im Original „blossom shrinking algorithm" wurde in dem bahnbrechenden Artikel

> Jack Edmonds: Path, trees, and flowers, Canadian Journal of Mathematics, 17 (1965) 449–467.

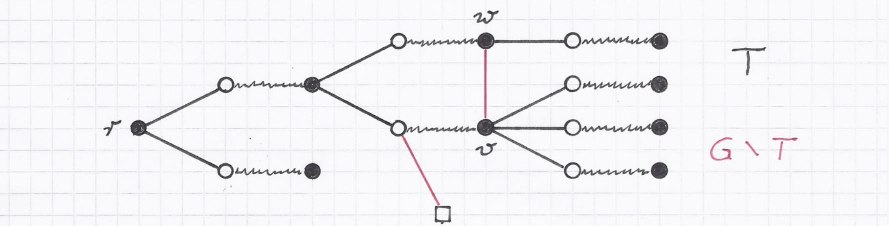
veröffentlicht.

Aus dem Graphen G entsteht durch wiederholtes Schrumpfen ungerader Kreise der Graph G'. G' hat

- Originalknoten von G und
- Pseudoknoten, die durch Schrumpfen entstanden sind.

Welche Kreise schrumpfen wir?

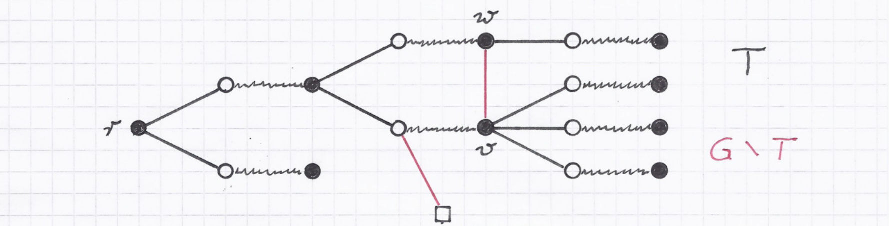
Für aus G abgeleitete Graphen G' haben wir:



Welche Kreise schrumpfen wir?

Für aus G abgeleitete Graphen G' haben wir:

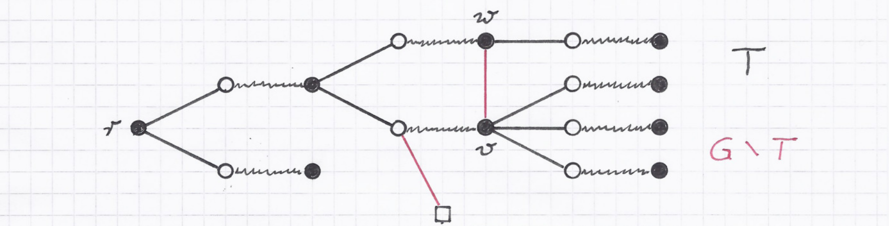
- Hat G' ein perfektes Matching, so können wir daraus auch eines für G produzieren.



Welche Kreise schrumpfen wir?

Für aus G abgeleitete Graphen G' haben wir:

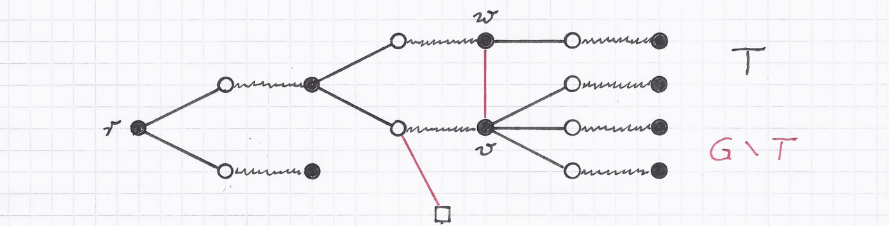
- Hat G' ein perfektes Matching, so können wir daraus auch eines für G produzieren.
- Gewisse frustrierte Bäume in G' implizieren, dass kein perfektes Matching existiert. ← siehe später bei Korrektheit



Welche Kreise schrumpfen wir?

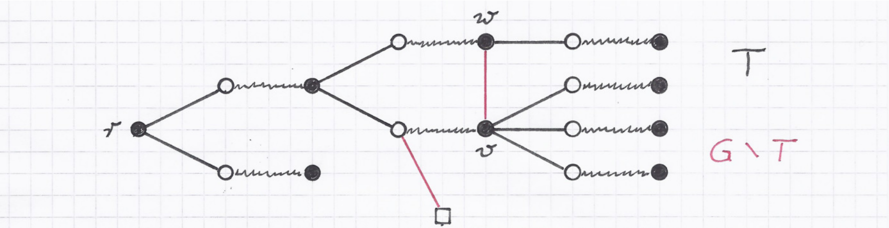
Für aus G abgeleitete Graphen G' haben wir:

- Hat G' ein perfektes Matching, so können wir daraus auch eines für G produzieren.
- Gewisse frustrierte Bäume in G' implizieren, dass kein perfektes Matching existiert. ← siehe später bei Korrektheit

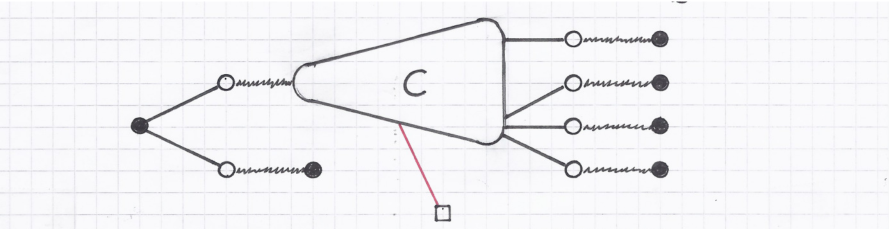


Hier ist weder eine Baumerweiterung noch eine M-Augmentierung möglich. Die Kante vw schließt einen ungeraden Kreis – eine Blüte (Jack Edmonds: „blossom“).

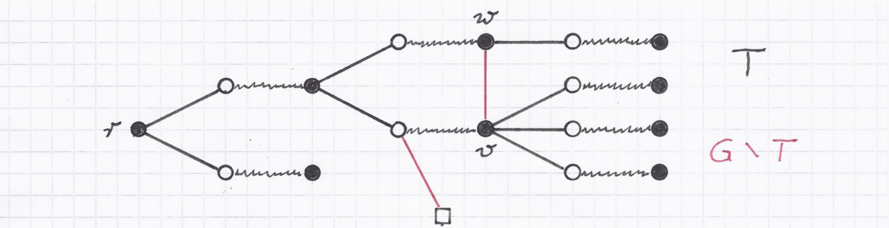
Schrumpfen von Blüten



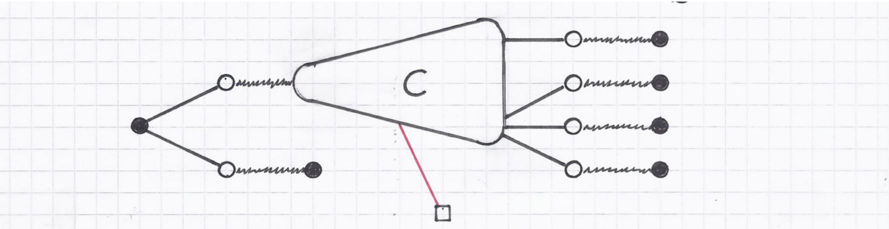
Wir schrumpfen die Blüte . . .



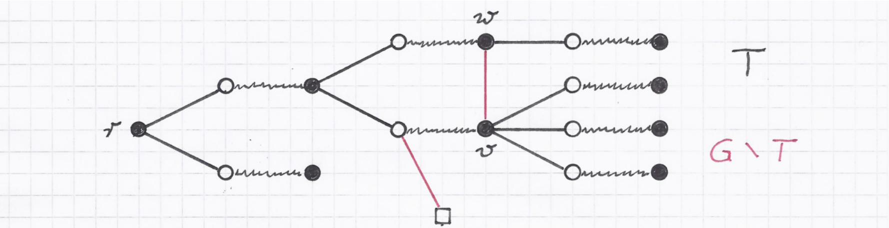
Schrumpfen von Blüten



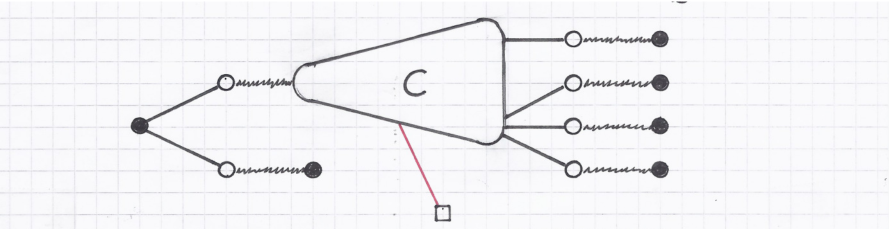
Wir schrumpfen die Blüte ...



Schrumpfen von Blüten



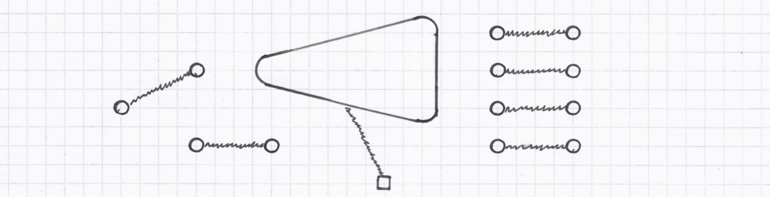
Wir schrumpfen die Blüte ...



... und „behalten“, was von T und M übrig bleibt.

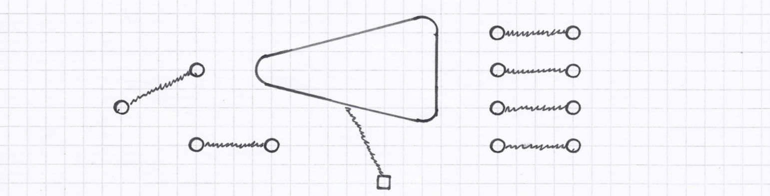
Schrumpfen von Blüten

Jetzt können wir augmentieren . . .



Schrumpfen von Blüten

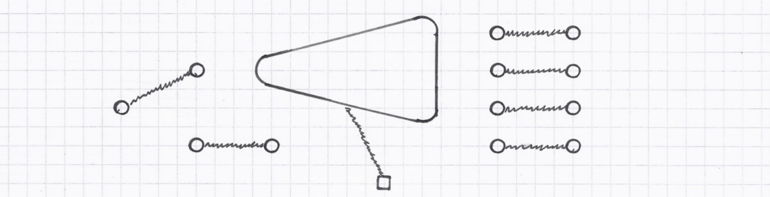
Jetzt können wir augmentieren ...



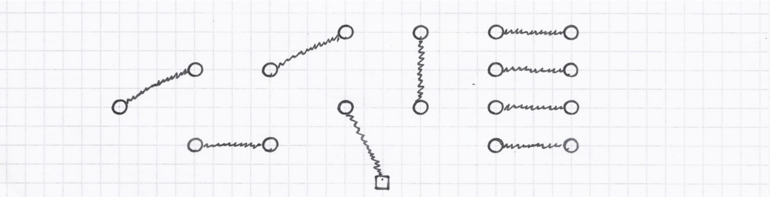
... und erhalten ein perfektes Matching.

Schrumpfen von Blüten

Jetzt können wir augmentieren ...



... und erhalten ein perfektes Matching.
Nach Entschrumpfen erhalten wir ein perfektes Matching im Originalgraphen G:



Unterprogramm zum Schrumpfen

Benutze vw zum Schrumpfen und adaptiere M' und T

Eingabe: Matching M' von G',
M'-alternierender Baum T,
$vw \in E(G')$ mit $v, w \in B(T)$

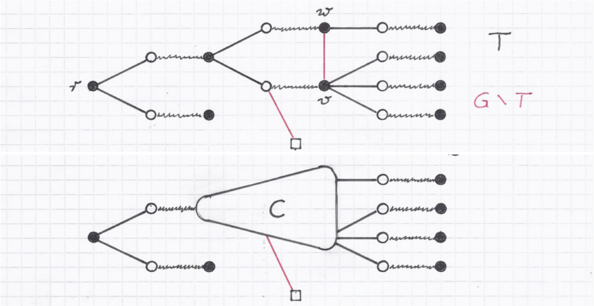
Sei C der Kreis gebildet aus vw und dem v-w-Pfad in T.
$G' \longleftarrow G' \times C$;
$M' \longleftarrow M' \setminus E(C)$;
$E(T) \longleftarrow E(T) \setminus E(C)$;

Unterprogramm zum Schrumpfen

Man kann sich leicht überzeugen, dass Folgendes gilt:

Lemma (Korrektheit des Unterprogramms (Lemma 5))

Nach Anwendung des obigen Unterprogramms ist M' ein Matching in G', T ein M'-alternierender Baum in G' und $c \in B(T)$. $\square$



Blüten-Schrumpf-Algorithmus für perfektes Matching (PMS)

Eingabe: Graph G und Matching M in G

if (M ist perfekt) STOP „M ist ein perfektes Matching";
$M' \longleftarrow M$; $G' \longleftarrow G$;
Wähle M'-exponierten Knoten r in G';
$T \longleftarrow (\{r\}, \emptyset)$;
while ($\exists\ vw \in E'$ mit $v \in B(T)$, $w \notin A(T)$) {
    case ($w \notin V(T)$ und w ist M'-exponiert) {
        Benutze vw zur M'-Augmentierung;
        Erweitere M' zu einem Matching M von G;
        $M' \longleftarrow M$;
        $G' \longleftarrow G$;
        if ($\nexists\ M'$-exponierter Knoten in G')
            STOP „M' ist ein perfektes Matching";
        else $T \longleftarrow (\{r\}, \emptyset)$ für einen M-exponierten Knoten r;
    }
    case ($w \notin V(T)$ und w ist M'-überdeckt) {
        Benutze vw zur Baumerweiterung;
    }
    case ($w \in B(T)$) {
        Benutze vw zum Schrumpfen und adaptiere M' und T;
    }
}
STOP „G hat kein perfektes Matching";

Terminierung des Algorithmus PMS

Theorem (Terminierung des Blütenschrumpfalgorithmus)

Sei $n = |V|$. Der Algorithmus PMS terminiert nach

- $O(n)$ Augmentierungen,
- $O(n^2)$ Schrumpf-Schritten,
- $O(n^2)$ Baumerweiterungs-Schritten.

Terminierung des Algorithmus PMS

Theorem (Terminierung des Blütenschrumpfalgorithmus)

Sei $n = |V|$. Der Algorithmus PMS terminiert nach

- $O(n)$ Augmentierungen,
- $O(n^2)$ Schrumpf-Schritten,
- $O(n^2)$ Baumerweiterungs-Schritten.

Beweis

Terminierung des Algorithmus PMS

Theorem (Terminierung des Blütenschrumpfalgorithmus)

Sei $n = |V|$. Der Algorithmus PMS terminiert nach

- $O(n)$ Augmentierungen,
- $O(n^2)$ Schrumpf-Schritten,
- $O(n^2)$ Baumerweiterungs-Schritten.

Beweis

Mit jeder Augmentierung erniedrigt sich die Anzahl der exponierten Knoten.
Also gibt es $O(n)$ Augmentierungen.

Beweis der Terminierung ff

Jeder Schrumpfschritt

Beweis der Terminierung ff

Jeder Schrumpfschritt

- erniedrigt die Anzahl der Knoten in G',

Beweis der Terminierung ff

Jeder Schrumpfschritt

- erniedrigt die Anzahl der Knoten in G',
- erhält die Anzahl der Nicht-Baumknoten.

Beweis der Terminierung ff

Jeder Schrumpfschritt

- erniedrigt die Anzahl der Knoten in G',
- erhält die Anzahl der Nicht-Baumknoten.

Jeder Baumerweiterungsschritt

Beweis der Terminierung ff

Jeder Schrumpfschritt

- erniedrigt die Anzahl der Knoten in G',
- erhält die Anzahl der Nicht-Baumknoten.

Jeder Baumerweiterungsschritt

- erniedrigt die Anzahl der Nicht-Baumknoten,

Beweis der Terminierung ff

Jeder Schrumpfschritt

- erniedrigt die Anzahl der Knoten in G',
- erhält die Anzahl der Nicht-Baumknoten.

Jeder Baumerweiterungsschritt

- erniedrigt die Anzahl der Nicht-Baumknoten,
- erhält die Anzahl der Knoten in G'.

Beweis der Terminierung ff

Jeder Schrumpfschritt

- erniedrigt die Anzahl der Knoten in G',
- erhält die Anzahl der Nicht-Baumknoten.

Jeder Baumerweiterungsschritt

- erniedrigt die Anzahl der Nicht-Baumknoten,
- erhält die Anzahl der Knoten in G'.

Also gibt es zwischen zwei Augmentierungen je $O(n)$ Schrumpf-/Baumerweiterungsschritte.

Beweis der Terminierung ff

Jeder Schrumpfschritt

- erniedrigt die Anzahl der Knoten in G',
- erhält die Anzahl der Nicht-Baumknoten.

Jeder Baumerweiterungsschritt

- erniedrigt die Anzahl der Nicht-Baumknoten,
- erhält die Anzahl der Knoten in G'.

Also gibt es zwischen zwei Augmentierungen je $O(n)$ Schrumpf-/Baumerweiterungsschritte.

Insgesamt sind das $O(n^2)$ Schrumpf-/Baumerweiterungsschritte. $\square$

Korrektheit: Ein Analogon zu Lemma 2

Für $v \in V(G')$ sei $S(v)$ die Menge der zu v gehörigen Originalknoten. (Diese können rekursiv ermittelt werden.)

Korrektheit: Ein Analogon zu Lemma 2

Für $v \in V(G')$ sei $S(v)$ die Menge der zu v gehörigen Originalknoten. (Diese können rekursiv ermittelt werden.)

- $|S(v)|$ ist immer ungerade. ($S(v) = \{v\}$ für Originalknoten)

Korrektheit: Ein Analogon zu Lemma 2

Für $v \in V(G')$ sei $S(v)$ die Menge der zu v gehörigen Originalknoten. (Diese können rekursiv ermittelt werden.)

- $|S(v)|$ ist immer ungerade. ($S(v) = \{v\}$ für Originalknoten)
- $\{S(v) \mid v \in V(G')\}$ ist eine Partition von $V(G)$.

Korrektheit: Ein Analogon zu Lemma 2

Für $v \in V(G')$ sei $S(v)$ die Menge der zu v gehörigen Originalknoten. (Diese können rekursiv ermittelt werden.)

- $|S(v)|$ ist immer ungerade. ($S(v) = \{v\}$ für Originalknoten)
- $\{S(v) \mid v \in V(G')\}$ ist eine Partition von $V(G)$.

Lemma (Frustrierter M'-alternierender Baum (Lemma 4))

Sei G' von G abgeleitet, M' ein Matching in G', T ein M'-alternierender Baum in G' und kein Pseudoknoten in $A(T)$. Ist T frustriert, so hat G kein perfektes Matching.

Korrektheit: Ein Analogon zu Lemma 2

Für $v \in V(G')$ sei $S(v)$ die Menge der zu v gehörigen Originalknoten. (Diese können rekursiv ermittelt werden.)

- $|S(v)|$ ist immer ungerade. ($S(v) = \{v\}$ für Originalknoten)
- $\{S(v) \mid v \in V(G')\}$ ist eine Partition von $V(G)$.

Lemma (Frustrierter M'-alternierender Baum (Lemma 4))

Sei G' von G abgeleitet, M' ein Matching in G', T ein M'-alternierender Baum in G' und kein Pseudoknoten in $A(T)$. Ist T frustriert, so hat G kein perfektes Matching.

Beweis.

Die Entfernung von $A(T)$ aus $V(G)$ ergibt für alle $v \in B(T)$ ungerade Komponenten mit Knotenmenge $S(v)$.
Wir erhalten $\mathrm{oc}(G \setminus A(T)) \geq |B(T)| > |A(T)|$.

$\square$

Korrektheit: Ein Analogon zu Lemma 2

Für $v \in V(G')$ sei $S(v)$ die Menge der zu v gehörigen Originalknoten. (Diese können rekursiv ermittelt werden.)

- $|S(v)|$ ist immer ungerade. ($S(v) = \{v\}$ für Originalknoten)
- $\{S(v) \mid v \in V(G')\}$ ist eine Partition von $V(G)$.

Lemma (Frustrierter M'-alternierender Baum (Lemma 4))

Sei G' von G abgeleitet, M' ein Matching in G', T ein M'-alternierender Baum in G' und kein Pseudoknoten in $A(T)$. Ist T frustriert, so hat G kein perfektes Matching.

Beweis.

Die Entfernung von $A(T)$ aus $V(G)$ ergibt für alle $v \in B(T)$ ungerade Komponenten mit Knotenmenge $S(v)$.
Wir erhalten $\mathrm{oc}(G \setminus A(T)) \geq |B(T)| > |A(T)|$.
Mit der Tutte-Bedingung folgt, dass G kein perfektes Matching besitzt. □

Korrektheit und Laufzeit

Theorem (Korrektheit und Laufzeit von PMS)

Sei $G = (V, E)$ ein Graph. Der Blütenschrumpfalgorithmus PMS berechnet ein perfektes Matching, wenn G ein solches besitzt. Andernfalls stoppt er mit einem frustrierten Baum als Beweis, dass kein solches existiert.

Korrektheit und Laufzeit

Theorem (Korrektheit und Laufzeit von PMS)

Sei $G = (V, E)$ ein Graph. Der Blütenschrumpfalgorithmus PMS berechnet ein perfektes Matching, wenn G ein solches besitzt. Andernfalls stoppt er mit einem frustrierten Baum als Beweis, dass kein solches existiert. Der Blütenschrumpfalgorithmus kann in Laufzeit $O(|V||E| \log |V|)$ sowie $O(|V|^3)$ realisiert werden.

Korrektheit und Laufzeit

Theorem (Korrektheit und Laufzeit von PMS)

Sei $G = (V, E)$ ein Graph. Der Blütenschrumpfalgorithmus PMS berechnet ein perfektes Matching, wenn G ein solches besitzt. Andernfalls stoppt er mit einem frustrierten Baum als Beweis, dass kein solches existiert. Der Blütenschrumpfalgorithmus kann in Laufzeit $O(|V||E| \log |V|)$ sowie $O(|V|^3)$ realisiert werden.

Beweis

M' ist stets ein Matching. Jedes G' ist aus G abgeleitet.

Korrektheit und Laufzeit

Theorem (Korrektheit und Laufzeit von PMS)

Sei $G = (V, E)$ ein Graph. Der Blütenschrumpfalgorithmus PMS berechnet ein perfektes Matching, wenn G ein solches besitzt. Andernfalls stoppt er mit einem frustrierten Baum als Beweis, dass kein solches existiert. Der Blütenschrumpfalgorithmus kann in Laufzeit $O(|V||E|\log|V|)$ sowie $O(|V|^3)$ realisiert werden.

Beweis

M' ist stets ein Matching. Jedes G' ist aus G abgeleitet.
Hält der Algorithmus mit „G hat kein perfektes Matching“, so haben wir einen frustrierten Baum T ohne Pseudoknoten in $A(T)$, der die Voraussetzungen von Lemma 4 erfüllt.

Korrektheit und Laufzeit

Theorem (Korrektheit und Laufzeit von PMS)

Sei $G = (V, E)$ ein Graph. Der Blütenschrumpfalgorithmus PMS berechnet ein perfektes Matching, wenn G ein solches besitzt. Andernfalls stoppt er mit einem frustrierten Baum als Beweis, dass kein solches existiert. Der Blütenschrumpfalgorithmus kann in Laufzeit $O(|V||E| \log |V|)$ sowie $O(|V|^3)$ realisiert werden.

Beweis

M' ist stets ein Matching. Jedes G' ist aus G abgeleitet.
Hält der Algorithmus mit „G hat kein perfektes Matching“, so haben wir einen frustrierten Baum T ohne Pseudoknoten in $A(T)$, der die Voraussetzungen von Lemma 4 erfüllt.
Realisierung der einzelnen Schritte mit Hilfe von Datenstrukturen: Geht einfach in Zeit $O(|V|^4)$ oder auch in $O(|V|^3)$.

Korrektheit und Laufzeit

Theorem (Korrektheit und Laufzeit von PMS)

Sei $G = (V, E)$ ein Graph. Der Blütenschrumpfalgorithmus PMS berechnet ein perfektes Matching, wenn G ein solches besitzt. Andernfalls stoppt er mit einem frustrierten Baum als Beweis, dass kein solches existiert. Der Blütenschrumpfalgorithmus kann in Laufzeit $O(|V||E| \log |V|)$ sowie $O(|V|^3)$ realisiert werden.

Beweis

M' ist stets ein Matching. Jedes G' ist aus G abgeleitet.
Hält der Algorithmus mit „G hat kein perfektes Matching“, so haben wir einen frustrierten Baum T ohne Pseudoknoten in $A(T)$, der die Voraussetzungen von Lemma 4 erfüllt.
Realisierung der einzelnen Schritte mit Hilfe von Datenstrukturen: Geht einfach in Zeit $O(|V|^4)$ oder auch in $O(|V|^3)$. Benutzt man die Datenstruktur Union-Find, dann geht dies in Zeit $O(|V||E| \log |V|)$ (s. Übung).

Table of Contents

Table of Contents

Algorithmus für Maximum Matching (MMS)

Idee: Ableitung aus Algorithmus PMS für perfektes Matching:

1. Starte Algorithmus PMS für aktuelles G'

Algorithmus für Maximum Matching (MMS)

Idee: Ableitung aus Algorithmus PMS für perfektes Matching:

1. Starte Algorithmus PMS für aktuelles G'
2. Stoppt dieser mit perfektem Matching $\Rightarrow$ fertig.

Algorithmus für Maximum Matching (MMS)

Idee: Ableitung aus Algorithmus PMS für perfektes Matching:

1. Starte Algorithmus PMS für aktuelles G'
2. Stoppt dieser mit perfektem Matching $\Rightarrow$ fertig.
3. Stoppt dieser mit Matching M' in G' und einem frustrierten Baum T, so ist das zu M' korrespondierende Matching M in G nicht notwendigerweise ein maximum Matching in G. Dann:

Algorithmus für Maximum Matching (MMS)

Idee: Ableitung aus Algorithmus PMS für perfektes Matching:

1. Starte Algorithmus PMS für aktuelles G'
2. Stoppt dieser mit perfektem Matching $\Rightarrow$ fertig.
3. Stoppt dieser mit Matching M' in G' und einem frustrierten Baum T, so ist das zu M' korrespondierende Matching M in G nicht notwendigerweise ein maximum Matching in G. Dann:
    - Wir entfernen $V(T)$ aus G'.

Algorithmus für Maximum Matching (MMS)

Idee: Ableitung aus Algorithmus PMS für perfektes Matching:

1. Starte Algorithmus PMS für aktuelles G'
2. Stoppt dieser mit perfektem Matching $\Rightarrow$ fertig.
3. Stoppt dieser mit Matching M' in G' und einem frustrierten Baum T, so ist das zu M' korrespondierende Matching M in G nicht notwendigerweise ein maximum Matching in G. Dann:
    - Wir entfernen $V(T)$ aus G'.
    - Falls ein exponierter Knoten übrig ist, wenden wir den Blütenschrumpfalgorithmus auf das neue G' an. (Achtung: G' hat keine Pseudoknoten.) $\Rightarrow$ Gehe zu (1)

Algorithmus für Maximum Matching (MMS)

Idee: Ableitung aus Algorithmus PMS für perfektes Matching:

1. Starte Algorithmus PMS für aktuelles G'
2. Stoppt dieser mit perfektem Matching $\Rightarrow$ fertig.
3. Stoppt dieser mit Matching M' in G' und einem frustrierten Baum T, so ist das zu M' korrespondierende Matching M in G nicht notwendigerweise ein maximum Matching in G. Dann:
   - Wir entfernen $V(T)$ aus G'.
   - Falls ein exponierter Knoten übrig ist, wenden wir den Blütenschrumpfalgorithmus auf das neue G' an. (Achtung: G' hat keine Pseudoknoten.) $\Rightarrow$ Gehe zu (1)
4. Wiederhole (1)-(3), bis kein exponierter Knoten übrig bleibt.

Algorithmus für Maximum Matching (MMS)

Idee: Ableitung aus Algorithmus PMS für perfektes Matching:

1. Starte Algorithmus PMS für aktuelles G'
2. Stoppt dieser mit perfektem Matching $\Rightarrow$ fertig.
3. Stoppt dieser mit Matching M' in G' und einem frustrierten Baum T, so ist das zu M' korrespondierende Matching M in G nicht notwendigerweise ein maximum Matching in G. Dann:
   - Wir entfernen $V(T)$ aus G'.
   - Falls ein exponierter Knoten übrig ist, wenden wir den Blütenschrumpfalgorithmus auf das neue G' an. (Achtung: G' hat keine Pseudoknoten.) $\Rightarrow$ Gehe zu (1)
4. Wiederhole (1)-(3), bis kein exponierter Knoten übrig bleibt.
5. Wir restaurieren das Original G mit allen produzierten Matchingkanten.

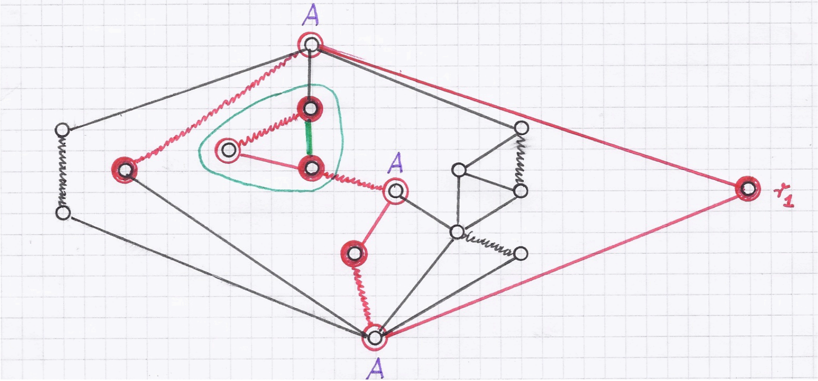
Algorithmus für Maximum Matching (MMS)

Idee: Ableitung aus Algorithmus PMS für perfektes Matching:

1. Starte Algorithmus PMS für aktuelles G'
2. Stoppt dieser mit perfektem Matching $\Rightarrow$ fertig.
3. Stoppt dieser mit Matching M' in G' und einem frustrierten Baum T, so ist das zu M' korrespondierende Matching M in G nicht notwendigerweise ein maximum Matching in G. Dann:
    - Wir entfernen $V(T)$ aus G'.
    - Falls ein exponierter Knoten übrig ist, wenden wir den Blütenschrumpfalgorithmus auf das neue G' an. (Achtung: G' hat keine Pseudoknoten.) $\Rightarrow$ Gehe zu (1)
4. Wiederhole (1)-(3), bis kein exponierter Knoten übrig bleibt.
5. Wir restaurieren das Original G mit allen produzierten Matchingkanten.
6. Wir bestimmen das korrespondierende Matching M in G.

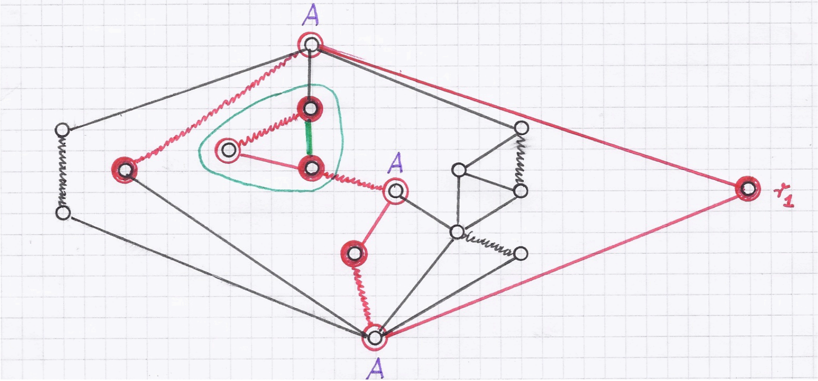
Der Blütenschrumpfalgorithmus: Beispiel

Nach Augmentierungsschritten erreicht man folgendes Matching M mit $|M| = 7$ in $G = (V, E)$ mit $|V| = 16$ (T ist rot gezeichnet):



Der Blütenschrumpfalgorithmus: Beispiel

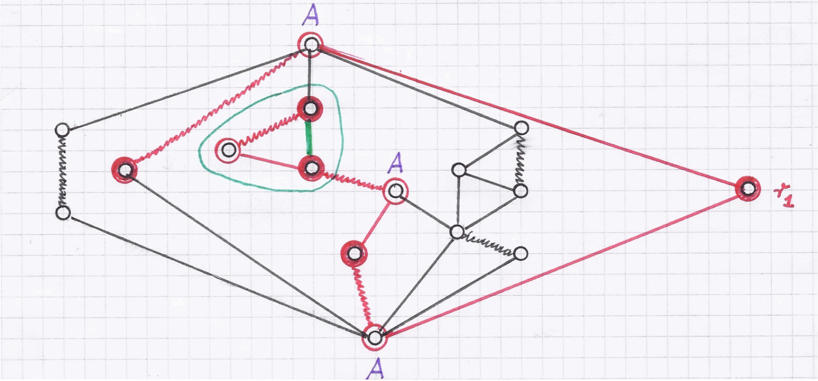
Nach Augmentierungsschritten erreicht man folgendes Matching M mit $|M| = 7$ in $G = (V, E)$ mit $|V| = 16$ (T ist rot gezeichnet):



Der alternierende Baum mit Wurzel r_1 führt zur grünen Blüte, die geschrumpft wird.

Der Blütenschrumpfalgorithmus: Beispiel

Nach Augmentierungsschritten erreicht man folgendes Matching M mit $|M| = 7$ in $G = (V, E)$ mit $|V| = 16$ (T ist rot gezeichnet):

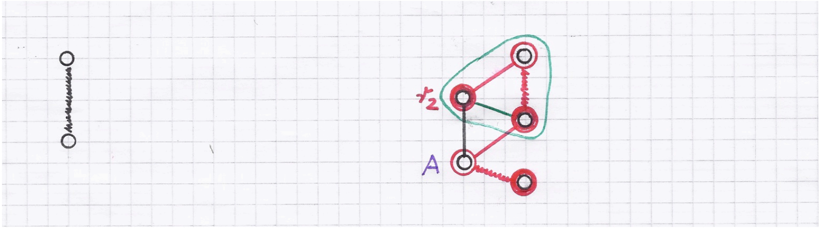


Der alternierende Baum mit Wurzel r_1 führt zur grünen Blüte, die geschrumpft wird.

Wir erhalten einen frustrierten Baum.

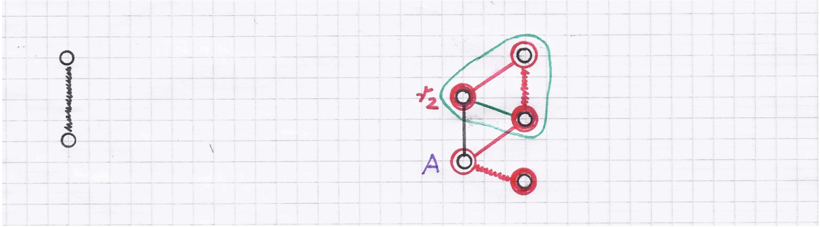
Der Blütenschrumpfalgorithmus: Beispiel

Nach Entfernen von $V(T)$ erhalten wir:



Der Blütenschrumpfalgorithmus: Beispiel

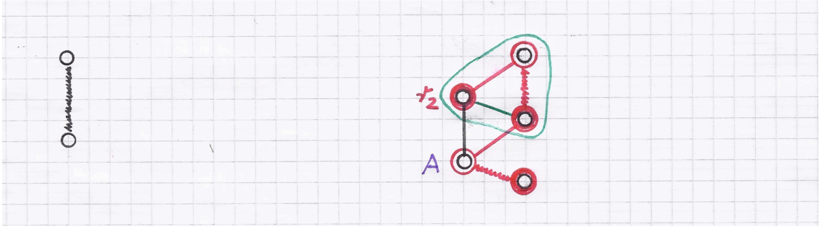
Nach Entfernen von $V(T)$ erhalten wir:



Der alternierende Baum mit Wurzel r_2 führt zur grünen Blüte, die geschrumpft wird.

Der Blütenschrumpfalgorithmus: Beispiel

Nach Entfernen von $V(T)$ erhalten wir:

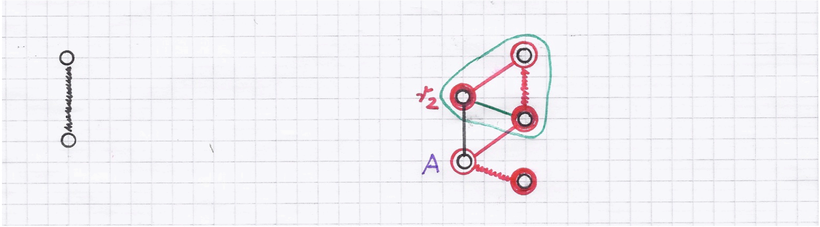


Der alternierende Baum mit Wurzel r_2 führt zur grünen Blüte, die geschrumpft wird.

Wir erhalten wieder einen frustrierten Baum.

Der Blütenschrumpfalgorithmus: Beispiel

Nach Entfernen von $V(T)$ erhalten wir:



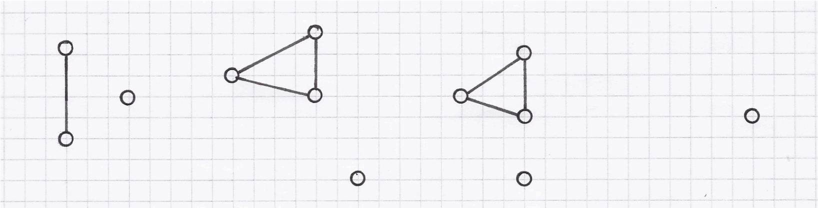
Der alternierende Baum mit Wurzel r_2 führt zur grünen Blüte, die geschrumpft wird.

Wir erhalten wieder einen frustrierten Baum.

Nach Entfernung von $V(T)$ ist kein exponierter Knoten mehr übrig.

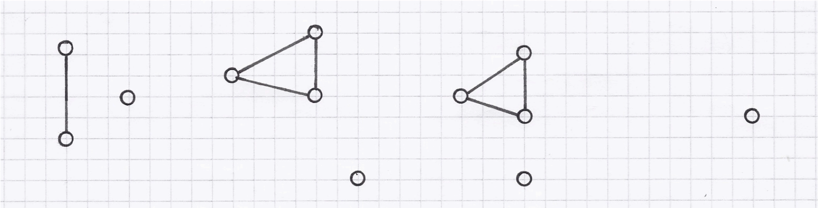
Korrektheit anhand des Beispiels

Nach Entfernung der Menge der A-Knoten aller frustrierten Bäume (4 A-Knoten) ergibt sich:



Korrektheit anhand des Beispiels

Nach Entfernung der Menge der A-Knoten aller frustrierten Bäume (4 A-Knoten) ergibt sich:



D.h., wir haben 6 ungerade Komponenten. Mit $|V| = 16$, $|M| = 7$, $|A| = 4$ und $\mathrm{oc}(G \setminus A) = 6$ erhalten wir die Tutte-Berge-Schranke:

$$\frac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|) = \frac{1}{2}(16 - 6 + 4) = 7$$

Korrektheit von MMS für Maximum Matching

Seien $T_1, T_2, \ldots, T_k$ die generierten frustrierten Bäume.

Korrektheit von MMS für Maximum Matching

Seien $T_1, T_2, \ldots, T_k$ die generierten frustrierten Bäume.

Die Wurzeln dieser Bäume sind k exponierte Knoten in G' und G.

Korrektheit von MMS für Maximum Matching

Seien $T_1, T_2, \ldots, T_k$ die generierten frustrierten Bäume.

Die Wurzeln dieser Bäume sind k exponierte Knoten in G' und G.

Also gilt $|M| = \frac{1}{2}(|V| - k)$.

Korrektheit von MMS für Maximum Matching

Seien $T_1, T_2, \ldots, T_k$ die generierten frustrierten Bäume.

Die Wurzeln dieser Bäume sind k exponierte Knoten in G' und G.

Also gilt $|M| = \frac{1}{2}(|V| - k)$.

Sei $A := \bigcup_{i=1}^{k} A(T_i)$. Die Entfernung von A aus G ergibt eine ungerade Komponente für jeden Knoten aus $B(T_i)$ für alle $i \in \{1, 2, \ldots, k\}$.

Korrektheit von MMS für Maximum Matching

Seien $T_1, T_2, \ldots, T_k$ die generierten frustrierten Bäume.

Die Wurzeln dieser Bäume sind k exponierte Knoten in G' und G.

Also gilt $|M| = \frac{1}{2}(|V| - k)$.

Sei $A := \bigcup_{i=1}^{k} A(T_i)$. Die Entfernung von A aus G ergibt eine ungerade Komponente für jeden Knoten aus $B(T_i)$ für alle $i \in \{1, 2, \ldots, k\}$.

Für alle $i \in \{1, 2, \ldots, k\}$ haben wir: $|B(T_i)| = |A(T_i)| + 1$

$$\begin{aligned} \mathrm{oc}(G \setminus A) &\geq \sum_{i=1}^{k} |B(T_i)| \\ &= \sum_{i=1}^{k} |A(T_i) + 1| \\ &= |A| + k \end{aligned}$$

Korrektheit für Maximum Matching ff

Mit $\mathrm{oc}(G \setminus A) \geq |A| + k$ folgt:

$$\begin{aligned} \tfrac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|) &\leq \tfrac{1}{2}(|V| - |A| - k + |A|) \\ &= \tfrac{1}{2}(|V| - k) \\ &= |M| \end{aligned}$$

Korrektheit für Maximum Matching ff

Mit $\mathrm{oc}(G \setminus A) \geq |A| + k$ folgt:

$$\begin{aligned} \tfrac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|) &\leq \tfrac{1}{2}(|V| - |A| - k + |A|) \\ &= \tfrac{1}{2}(|V| - k) \\ &= |M| \end{aligned}$$

Mit der Tutte-Berge-Schranke $|M| \leq \frac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|)$ erhalten wir

$$|M| = \tfrac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|).$$

Korrektheit für Maximum Matching ff

Mit $\mathrm{oc}(G \setminus A) \geq |A| + k$ folgt:

$$\begin{aligned} \tfrac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|) &\leq \tfrac{1}{2}(|V| - |A| - k + |A|) \\ &= \tfrac{1}{2}(|V| - k) \\ &= |M| \end{aligned}$$

Mit der Tutte-Berge-Schranke $|M| \leq \frac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|)$ erhalten wir

$$|M| = \tfrac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|).$$

Deshalb ist M ein maximum Matching in G. $\square$

Laufzeit des Algorithmus MMS

Seien $n = |V|$ und $m = |E|$. In der Übung werden wir sehen:

Satz (Laufzeit)

Der Blütenschrumpfalgorithmus für maximale Matchings MMS kann in Laufzeit $O(nm \log n)$ sowie $O(n^3)$ realisiert werden. $\square$

Laufzeit des Algorithmus MMS

Seien $n = |V|$ und $m = |E|$. In der Übung werden wir sehen:

Satz (Laufzeit)

Der Blütenschrumpfalgorithmus für maximale Matchings MMS kann in Laufzeit $O(nm \log n)$ sowie $O(n^3)$ realisiert werden. $\square$

Ohne Beweis fügen wir hinzu:

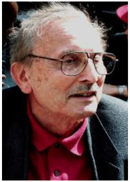
Satz (Micali und Vazirani [1980])

Es gibt eine Implementierung des Blütenschrumpfalgorithmus für maximale Matchings mit Laufzeit $O(\sqrt{n}m)$. $\square$

Table of Contents

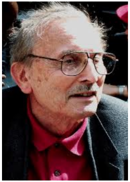
Sätze von Tutte und Berge

Somit haben wir algorithmisch den Satz von Berge [1958] bewiesen:



Sätze von Tutte und Berge

Somit haben wir algorithmisch den Satz von Berge [1958] bewiesen:

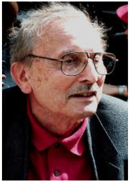


Satz (Tutte-Berge-Formel)

Für $G(V,E)$ gilt $\nu(G) = \min_{A \subseteq V} \left\{ \frac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|) \right\}$. $\square$

Sätze von Tutte und Berge

Somit haben wir algorithmisch den Satz von Berge [1958] bewiesen:



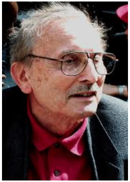
Satz (Tutte-Berge-Formel)

Für $G(V, E)$ gilt $\nu(G) = \min_{A \subseteq V} \left\{ \frac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|) \right\}$. □

Daraus folgt ein älteres Resultat von Tutte [1947]:

Sätze von Tutte und Berge

Somit haben wir algorithmisch den Satz von Berge [1958] bewiesen:



Satz (Tutte-Berge-Formel)

Für $G(V,E)$ gilt $\nu(G) = \min_{A \subseteq V} \left\{ \frac{1}{2}(|V| - \mathrm{oc}(G \setminus A) + |A|) \right\}$. $\square$

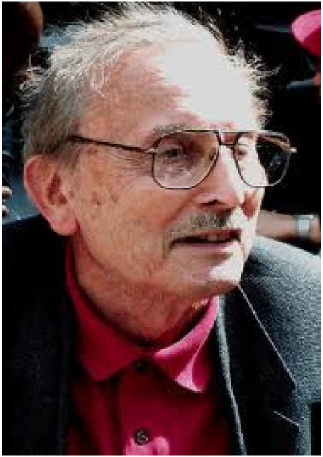
Daraus folgt ein älteres Resultat von Tutte [1947]:

Korollar (Tutte's Matching Theorem)

$G = (V,E)$ hat ein perfektes Matching genau dann, wenn für jede Teilmenge $A \subseteq V$ gilt $\mathrm{oc}(G \setminus A) \le |A|$. $\square$

Table of Contents

Claude Berge



Claude Berge (1926–2002)

William Thomas Tutte

William Thomas Tutte (1917–2002)

Jack Edmonds

Jack Edmonds

Köln 2004: Jack & Kathie mit Pauline & Paul

Jack Edmonds

Jack Edmonds

Zwei bahnbrechende Aufsätze - der erste 1965:

PATHS, TREES, AND FLOWERS

JACK EDMONDS

1. Introduction. A *graph* G for purposes here is a finite set of elements called *vertices* and a finite set of elements called *edges* such that each edge *meets* exactly two vertices, called the *end-points* of the edge. An edge is said to *join* its end-points.

A *matching* in G is a subset of its edges such that no two meet the same vertex. We describe an efficient algorithm for finding in a given graph a matching of maximum cardinality. This problem was posed and partly solved by C. Berge; see Sections 3.7 and 3.8.

Maximum matching is an aspect of a topic, treated in books on graph theory, which has developed during the last 75 years through the work of about a dozen authors. In particular, W. T. Tutte **(8)** characterized graphs which do not contain a *perfect* matching, or *1-factor* as he calls it—that is a set of edges with exactly one member meeting each vertex. His theorem prompted attempts at finding an efficient construction for perfect matchings.

Zwei bahnbrechende Aufsätze - der zweite:

JOURNAL OF RESEARCH of the National Bureau of Standards—B. Mathematics and Mathematical Physics
Vol. 69B, Nos. 1 and 2, January–June 1965

Maximum Matching and a Polyhedron With 0,1-Vertices[1]

Jack Edmonds

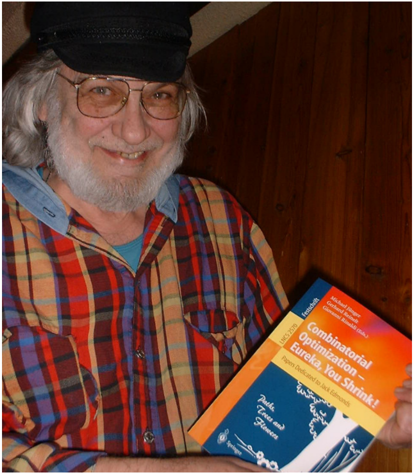
(December 1, 1964)

A matching in a graph G is a subset of edges in G such that no two meet the same node in G. The convex polyhedron C is characterized, where the extreme points of C correspond to the matchings in G. Where each edge of G carries a real numerical weight, an efficient algorithm is described for finding a matching in G with maximum weight-sum.

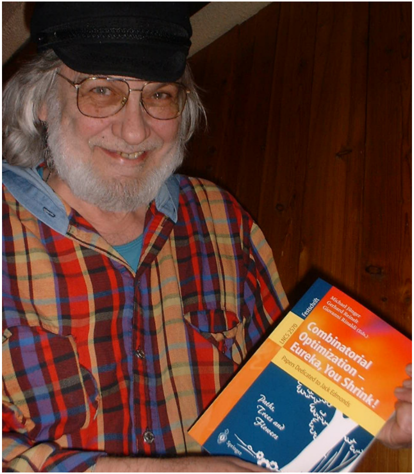
Section 1

An algorithm is described for optimally pairing a finite set of objects. That is, given a real numerical weight for each unordered pair of objects in a set Y, to select a family of mutually disjoint pairs the sum of whose weights is maximum. The well-known optimum assignment problem [5][2] is the special case where Y partitions into two sets A and B such that inequalities. In particular, we prove a theorem analogous to one of G. Birkhoff [1] and J. von Neuman [5] which says that the extreme points of the convex set of doubly stochastic matrices (order n by n) are the permutation matrices (order n by n). That theorem and the Hungarian method are based on Konig's theorem about matchings in bipartite graphs. Our work is related to results on graphs due to Tutte [4].

Aussois 2002: Jack Edmonds



Aussois 2002: Jack Edmonds



Jetzt: Videodokumente: Jack Edmonds im Interview (Aussois 2008)