

Graphalgorithmen

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2023/24

Matchings

Definition Sei $G = (V, E)$ ein Graph und sei $M \subseteq E$.

- M heißt **Matching** in G , falls je zwei Kanten $e \neq e' \in M$ **unabhängig** sind, d.h. $e \cap e' = \emptyset$

- Die **Matchingzahl** von G ist

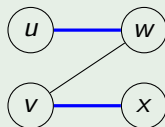
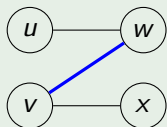
$$\mu(G) = \max\{|M| : M \text{ ist ein Matching in } G\}$$

- Ein Knoten $v \in V$ heißt **M -gebunden**, falls v Endpunkt einer Kante $e \in M$ (also $v \in \cup M$) ist und sonst **M -frei**
- Wir sagen auch, **M bindet v** bzw. **M lässt v frei**
- Ein Matching M heißt **perfekt**, falls alle Knoten in G M -gebunden sind (also $V = \cup M$ ist)
- Ein Matching M heißt **maximal** (engl. *maximum*), falls $|M| = \mu(G)$ ist
- M heißt **gesättigt** (engl. *maximal*), falls es in keinem größeren Matching enthalten ist

Offenbar ist M genau dann ein Matching, wenn $|\cup M| = 2|M|$ ist

Das Ziel besteht nun darin, ein maximales Matching M in G zu finden

Beispiel Ein gesättigtes Matching muss nicht maximal sein:



- $M = \{(v, w)\}$ ist gesättigt, da es sich nicht erweitern lässt
 - M ist jedoch kein maximales Matching, da $M' = \{(u, w), (v, x)\}$ ein größeres Matching ist
-
- Die Greedy-Methode, ausgehend von $M = \emptyset$ solange Kanten zu M hinzuzufügen, bis M gesättigt ist, funktioniert also nicht
 - Wir setzen im Folgenden voraus, dass G keine isolierte Knoten enthält

Matchings

Satz

In einem bipartiten Graphen $G = (U, W, E)$ lässt sich ein maximales Matching in Zeit $O(m\sqrt{n})$ bestimmen

Beweis

- Wir konstruieren zu G das Netzwerk $N = (V, E', s, t, c)$ mit der Knotenmenge $V = U \cup W \cup \{s, t\}$ und der Kantenmenge

$$E' = \{(u, w) \in U \times W \mid \{u, w\} \in E\} \cup \{(s, u), (w, t) \mid u \in U, w \in W\}$$
 wobei $c(e) = 1$ für alle $e \in E'$ gilt
- Da in N alle Knoten $u \in U \cup W$ den Durchsatz $D(u) \leq C = 1$ haben, liefert jeder Fluss f in N ein Matching $M = \{\{u, w\} \in E \mid f(u, w) = 1\}$ in G mit $|M| = |f|$ und umgekehrt
- Daher berechnet Dinitz unter Verwendung von `blockfluss1` in Zeit $O((nC + m)\sqrt{nC}) = O(m\sqrt{n})$ einen maximalen Fluss in N (und somit ein maximales Matching in G) □

- In den Übungen werden wir sehen, dass sich die Laufzeit durch eine verbesserte Analyse sogar durch $O(m\sqrt{\mu})$ begrenzen lässt
- Die Konstruktion im Beweis des vorigen Satzes lässt sich nicht ohne Weiteres auf beliebige Graphen verallgemeinern
- Wir werden jedoch sehen, dass sich manche bei den Flussalgorithmen verwendete Ideen auch für Matchingalgorithmen einsetzen lassen
- So lassen sich Matchings, die nicht maximal sind, ähnlich vergrößern wie dies bei Flüssen durch einen Zunahmepfad möglich ist

Definition Sei $G = (V, E)$ ein Graph und sei M ein Matching in G .

- Ein Pfad $P = (u_0, \dots, u_\ell)$ in G der Länge $\ell \geq 1$ heißt **M -alternierend**, falls für $i = 1, \dots, \ell - 1$ gilt:

$$e_i = \{u_{i-1}, u_i\} \in M \iff e_{i+1} = \{u_i, u_{i+1}\} \notin M$$

- Ein Kreis $C = (u_1, \dots, u_\ell, u_1)$ in G heißt **M -alternierend**, falls der Pfad $P = (u_1, \dots, u_\ell)$ M -alternierend ist und zudem gilt:

$$\{u_1, u_2\} \in M \iff \{u_1, u_\ell\} \notin M$$

- Ein M -alternierender Pfad $P = (u_0, \dots, u_\ell)$ heißt **M -vergrößernder Pfad** (oder einfach **M -Pfad**), falls beide Endpunkte von P M -frei sind

Im bipartiten Fall entsprechen sich M -Pfade und Zunahmepfade

Matchings

Satz (Lemma von Berge)

Ein Matching M ist genau dann maximal, wenn es keinen M -Pfad gibt

Beweis

- Ist $P = (u_0, \dots, u_l)$ ein M -Pfad, so liefert $M' = M \Delta P$ ein Matching der Größe $|M'| = |M| + 1$ in G , wobei wir P als Menge $\{\{u_{i-1}, u_i\} \mid i = 1, \dots, l\}$ seiner Kanten auffassen
- Ist dagegen M nicht maximal und M' ein größeres Matching, so betrachten wir die Kantenmenge $M \Delta M'$
- Da jeder Knoten in dem Graphen $G' = (V, M \Delta M')$ höchstens den Grad 2 hat, lässt sich G' in disjunkte Kreise und Pfade zerlegen
- Da diese Kreise und Pfade sowohl M - als auch M' -alternierend sind, und M' größer als M ist, muss mindestens einer der Pfade mehr Kanten aus M' als aus M enthalten und somit ein M -Pfad sein □

Es genügt also, einen M -Pfad zu finden (sofern er existiert)

Der Algorithmus von Edmonds

- Sei G ein Graph ohne isolierte Knoten und sei M ein Matching in G
- Die Prozedur `FindePfad` gibt einen M -Pfad in G zurück, falls das aktuelle Matching M nicht bereits maximal ist
- Da M nicht mehr als $\lfloor n/2 \rfloor$ Kanten enthalten kann, muss die Prozedur `FindePfad` ausgehend von $M = \emptyset$ höchstens $\lfloor n/2 + 1 \rfloor$ -mal aufgerufen werden, um ein maximales Matching zu finden

Prozedur `FindePfad(V, E, M)`

```

1   $Q := \emptyset$ 
2  for all  $u \in V$  do
3     $\text{parent}(u) := \perp$ 
4    if  $\exists e \in M : u \in e$  then
5       $\text{zustand}(u) := 2$       // unerreicht
6    else
7       $\text{zustand}(u) := 0$       // gerade
8       $\text{root}(u) := u$ 
9       $Q := Q \cup \{(u, v) \mid \{u, v\} \in E\}$ 

```


Prozedur $\text{FindePfad}(V, E, M)$ (Fortsetzung)

```
11 while  $Q \neq \emptyset$  do  
12   entferne eine Kante  $(u, v)$  aus  $Q$   
13   if  $\text{zustand}(v) = 2$  then // Erweiterung von  $W$  gefunden  
14      $\text{parent}(v) := u$ ;  $\text{parent}(M(v)) := v$ ;  $\text{zustand}(v) := 1$  // ungerade  
15      $\text{zustand}(M(v)) := 0$ ;  $\text{root}(M(v)) := \text{root}(v) := \text{root}(u)$   
16      $Q := Q \cup \{(M(v), w) \mid \{M(v), w\} \in E \setminus M\}$   
17   if  $\text{zustand}(v) = 0$  then  
18     if  $\text{root}(u) = \text{root}(v)$  then // Blüte gefunden  
19       kontrahiere in  $W$  die Blüte  $C$  zu ihrer Basis  $b$  und speichere  
20       den Kreis  $C$  unter der Basis  $b$  ab  
21       füge zu  $Q$  für jede Kante  $\{c, a\} \in E$  mit  $c \in C$  ungerade und  
22        $a \notin C$  die Kante  $(b, a)$  hinzu  
23     else //  $M$ -Pfad gefunden  
24       setze die  $\text{parent}$ -Pfade  $P_u$  und  $P_v$  von  $u$  und  $v$  mit Hilfe  
25       der Kante  $\{u, v\}$  zu einem  $r_u$ - $r_v$ -Pfad  $P'$  zusammen  
26       und expandiere  $P'$  zu einem  $M$ -Pfad  $P$  in  $G$ ; return  $P$   
27 return  $\perp$ 
```

Der Algorithmus von Edmonds

- Die Prozedur `FindePfad` sucht wie folgt nach einem M -Pfad in G
- Jeder Knoten u hat einen von 3 Zuständen:
 - gerade (0)
 - ungerade (1)
 - oder unerreicht (2)
- Zu Beginn sind alle M -freien Knoten gerade und alle M -gebundenen Knoten unerreicht
- Dann wird ausgehend von den M -freien Knoten als Wurzeln ein Suchwald W für G aufgebaut
- Hierzu wird Q als Menge aller Kanten (u, v) initialisiert, so dass u gerade (also eine Wurzel) und $\{u, v\} \in E$ ist
- In der `while`-Schleife werden dann die zu Q hinzugefügten Kanten $e = (u, v)$ abgearbeitet, wobei Kanten zu einem ungeraden Knoten v ignoriert werden

Der Algorithmus von Edmonds

- Ist v unerreicht, so wird der aktuelle Suchwald W nicht nur um die Kante $e = (u, v)$, sondern auch um die Matching-Kante $(v, M(v))$ erweitert, wobei $M(v)$ der Matchingpartner von v ist
- Hierzu werden $\text{parent}(v)$ auf u und $\text{parent}(M(v))$ auf v gesetzt
- Zudem wechselt der Zustand von v von unerreicht zu ungerade und der von $M(v)$ von unerreicht zu gerade
- Somit erhält jeder erreichte Knoten u genau dann den Zustand gerade, wenn u in W einen geraden Abstand zu seiner Wurzel r_u hat
- Die entsprechenden Wurzelfade P_u von r_u zu u lassen sich mithilfe der parent -Funktion berechnen
- Für die Suche nach alternierenden Fortsetzungen des Wurzelfads $P_{M(v)}$ werden zu Q noch alle Kanten $(M(v), w)$ mit $\{M(v), w\} \in E$ und $w \neq v$ hinzugefügt

Der Algorithmus von Edmonds

- Ist v dagegen wie u gerade, so gibt es zwei Unterfälle
- Haben u und v verschiedene Wurzeln $r_u \neq r_v$, so lassen sich die beiden Wurzelfade P_u und P_v mit Hilfe der Kante $\{u, v\}$ zu einem Pfad P' zusammensetzen, der die beiden M -freien Wurzeln r_u und r_v verbindet
- Da vor dem Auffinden von P' möglicherweise Blüten kontrahiert wurden (siehe unten), muss P' evtl. noch expandiert werden, um einen M -Pfad P in G zu erhalten

Der Algorithmus von Edmonds

- Im Fall $r_u = r_v$ haben die P_u und P_v dieselbe Wurzel $r = r_u = r_v$, d.h. u und v befinden sich im gleichen Suchbaum von W
- Sei b der Knoten, der sowohl auf P_u als auch auf P_v liegt und unterschiedliche Nachfolger in P_u und P_v hat
- Da b in W (mindestens) 2 Kinder hat und ungerade Knoten in W nur ein Kind haben, muss b gerade sein, und da u und v ebenfalls gerade sind, haben sie in W einen geraden Abstand zu b
- Daher bilden der b - u -Teilpfad von P_u und der b - v -Teilpfad von P_v zusammen mit der Kante $\{u, v\}$ einen ungerichteten Kreis C ungerader Länge, der als **Blüte** mit der **Basis** b bezeichnet wird
- Da nun auch die ungeraden Knoten auf C über einen alternierenden Pfad gerader Länge von r über die Basis b und die Kante $\{u, v\}$ erreichbar sind, werden alle Knoten auf C zu geraden Knoten
- Daher wird C in W zu ihrer Basis b kontrahiert und für jede Kante $\{c, a\}$ in E mit $c \in C$ ungerade wird die Kante (b, a) zu Q hinzugefügt

Der Algorithmus von Edmonds

- Zwar führt die Entdeckung einer Blüte C nicht unmittelbar zu einem M -Pfad
- Sie dehnt aber die Suche nach M -Pfadern auf alle ungeraden Knoten in C aus
- Kontrahieren wir in G die Blüte C zur Basis b , so erbt b in dem kontrahierten Graphen G_C die Nachbarschaften aller Knoten in C zu den Knoten außerhalb von C
- Entfernen wir aus einem Matching M von G alle Kanten, die auf dem Kreis C liegen, so erhalten wir ein Matching M_C in G_C
- Das folgende Lemma zeigt, wie sich ein M_C -Pfad in G_C zu einem M -Pfad in G expandieren lässt

Lemma Sei C eine Blüte in G mit Basis b .

Dann ist jeder M_C -Pfad P_C in G_C zu einem M -Pfad P in G expandierbar

Beweis

- Falls P_C nicht schon selbst ein M -Pfad in G ist, muss P_C eine Kante e enthalten, die in G fehlt
- Da durch die Kontraktion von C zu b in G_C nur solche Kanten neu entstehen, die die Basis b mit einem Knoten a außerhalb der Blüte C verbinden, muss e die Form $e = \{a, b\}$ haben
- Zudem muss a in G einen Nachbarn $c \neq b$ auf der Blüte C haben

Der Algorithmus von Edmonds

Beweis (Fortsetzung)

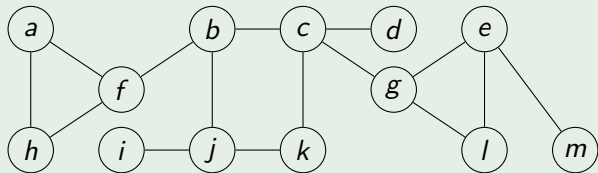
- Falls P_C nicht schon selbst ein M -Pfad in G ist, muss P_C eine Kante e enthalten, die in G fehlt
- Da durch die Kontraktion von C zu b in G_C nur solche Kanten neu entstehen, die die Basis b mit einem Knoten a außerhalb der Blüte C verbinden, muss e die Form $e = \{a, b\}$ haben
- Zudem muss a in G einen Nachbarn $c \neq b$ auf der Blüte C haben
- Von c aus führen auf dem Kreis C genau zwei Pfade zur Basis b , wovon nur einer den Knoten c über eine Matchingkante verlässt (also gerade Länge hat)
- Indem wir diesem Pfad die Kante $\{a, c\}$ hinzufügen, erhalten wir einen M -alternierenden a - b -Pfad P_e in G
- Wir können also P_C zu einem M -Pfad P in G expandieren, indem wir die Kante $e = \{a, b\}$ durch den a - b -Pfad P_e ersetzen □

Der Algorithmus von Edmonds

- Da sich die Anzahl der Knoten bei jeder Kontraktion einer Blüte mindestens um 2 verringert, können höchstens $n/2$ Blüten gefunden werden
- Bei Verwendung geeigneter Datenstrukturen zur Verwaltung der Blüten lässt sich die Prozedur `FindePfad` in Zeit $O(m)$ implementieren, was auf eine Gesamtlaufzeit von $O(nm)$ für den Algorithmus von Edmonds führt
- Tatsächlich lässt sich die Laufzeit noch auf $O(m\sqrt{\mu})$ verringern
- Dazu berechnet man ähnlich wie bei Verwendung von Dinitz im bipartiten Fall pro Runde nicht nur einen M -Pfad, sondern in Zeit $O(m)$ eine maximale Menge knotendisjunkter M -Pfade, die alle eine minimale Länge haben
- Dann kann man wieder zeigen, dass $O(\sqrt{\mu})$ solcher Runden ausreichen, um ein maximales Matching zu finden
- Diese Strategie führt auf den Hopcroft-Karp-Algorithmus im bipartiten Fall und auf den Micali-Vazirani-Algorithmus für beliebige Graphen

Der Algorithmus von Edmonds

Beispiel Wir betrachten den Graphen



i	abgearbeitet	B_i	b_i	abgearbeitet	P_i	M_i
1	af	-	-	-	af	$\{af\}$
2	bc	-	-	-	bc	$\{af, bc\}$
3	dc, bf, ah	-	-	-	$dcbfah$	$\{ah, bf, cd\}$
4	eg	-	-	-	eg	$\{ah, bf, cd, eg\}$
5	ij	-	-	-	ij	$\{ah, bf, cd, eg, ij\}$
6	kc, kj, le, gl	lge	l	ml	$megl$	$\{ah, bf, cd, ij, em, gl\}$
7	kc, kj	-	-	-	-	$\{ah, bf, cd, ij, em, gl\}$

- Für den Beweis der Korrektheit des Edmonds-Algorithmus (genauer: zum Nachweis der Maximalität des berechneten Matchings) benötigen wir den Begriff der Odd Set Cover in einem Graphen G
- Sei M ein Matching und sei C eine Knotenüberdeckung (kurz VC für vertex cover) in G
- Da jede Kante e in M mindestens einen Endpunkt in C hat, aber keine Kanten in M einen gemeinsamen Endpunkt haben, folgt $|M| \leq |C|$
- Wir werden sehen, dass es in jedem bipartiten Graphen sogar ein Matching M und eine Knotenüberdeckung C mit $|M| = |C|$ gibt
- Die Angabe einer Knotenüberdeckung C mit $|C| = |M|$ bietet also eine einfache Möglichkeit, die Maximalität von M nachzuweisen
- Dies geht jedoch nicht in allen Graphen, da z.B. der K_4 nur Matchings der Größe ≤ 2 , aber keine Knotenüberdeckung der Größe 2 hat

Der Algorithmus von Edmonds

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

- Eine Menge $S = \{v_1, \dots, v_k, V_1, \dots, V_\ell\}$ von Knoten $v_1, \dots, v_k \in V$ und Teilmengen $V_1, \dots, V_\ell \subseteq V$ heißt **Odd Set Cover (OSC)** in G , falls
 - es für jede Kante $e \in E$ einen Knoten $v_i \in S$ mit $v_i \in e$ oder eine Menge $V_j \in S$ mit $e \subseteq V_j$ gibt und
 - alle Mengen $V_j \in S$ eine ungerade Größe $n_j = |V_j|$ haben
- Das **Gewicht** von S ist $w(S) = k + \sum_{j=1}^{\ell} (n_j - 1)/2$

Im Fall $\ell = 0$ ist $S = \{v_1, \dots, v_k\}$ also eine **Knotenüberdeckung** oder **vertex cover (VC)** in G

Beispiel

- Der $K_{i,j}$, $i \leq j$, hat die Matchingzahl $\mu(K_{i,j}) = i$ und eine kleinste VC ist $C = \{1, \dots, i\}$, die auch eine OSC vom Gewicht $w(C) = i$ ist
- Der K_n hat die Matchingzahl $\mu(K_n) = \lfloor n/2 \rfloor$ und eine kleinste VC ist $C = \{1, \dots, n-1\}$, während $S = \{1, \{2, \dots, n\}\}$ für gerades n und $S = \{V(K_n)\}$ für ungerades n eine OSC vom Gewicht $w(S) = \lfloor n/2 \rfloor$ ist, wobei $V(G)$ die Knotenmenge eines Graphen G bezeichnet
- Der C_n hat die Matchingzahl $\mu(C_n) = \lfloor n/2 \rfloor$ und eine kleinste VC ist $C = \{1, 3, 5, \dots, n-1\}$ für gerades n und $C = \{1, 3, 5, \dots, n-1, n\}$ für ungerades n , während $S = \{1, 3, 5, \dots, n-1\}$ für gerades n und $S = \{V(C_n)\}$ für ungerades n eine OSC vom Gewicht $w(S) = \mu(C_n)$ ist

Der Algorithmus von Edmonds

Beispiel (Fortsetzung)

- Der vollständige Splitgraph $S_{i,j} = K_i + E_j$ hat im Fall $1 \leq i \leq j$ die Matchingzahl $\mu(S_{i,j}) = i$ und eine kleinste VC ist $C = \{1, \dots, i\}$, die auch eine OSC vom Gewicht $w(C) = i$ ist
- Dagegen hat $S_{i,j}$ im Fall $i > j \geq 1$ die Matchingzahl $\mu(S_{i,j}) = \lfloor n/2 \rfloor$ und eine kleinste VC ist $C = \{1, \dots, i\}$, während $S = \{V(S_{i,j})\}$ für ungerades n und $S = \{1, \{2, 3, \dots, n\}\}$ für gerades n eine OSC mit $w(S) = 1 + \lfloor (n-1)/2 \rfloor = 1 + (n-2)/2 = n/2 = \mu(S_{i,j})$ ist ◀

Lemma

Für jedes Matching M in einem Graphen $G = (V, E)$ und jede OSC $S = \{v_1, \dots, v_k, V_1, \dots, V_\ell\}$ in G gilt $|M| \leq w(S)$

Beweis

M kann für jeden Knoten $v_i \in S$ höchstens eine Kante e mit $v_i \in e$ und für jede Menge $V_j \in S$ höchstens $(n_j - 1)/2$ Kanten $e \subseteq V_j$ enthalten □

Der Algorithmus von Edmonds

Satz Sei M ein Matching in G .

Falls $\text{FindePfad}(G, M)$ keinen M -Pfad findet, ist M maximal

Beweis

- Wir benutzen die Klassifikation der Knoten von G zum Zeitpunkt des Abbruchs der erfolglosen Suche nach einem M -Pfad, um eine OSC S für G mit $w(S) = |M|$ zu finden
- Sei V_0 die Menge der geraden, V_1 die der ungeraden Knoten in W und V_2 die der unerreichten Knoten in G zu diesem Zeitpunkt
- Weiter seien b_1, \dots, b_ℓ die Knoten in V_0 , zu denen die gefundenen Blüten kontrahiert wurden, und für $j = 1, \dots, \ell$ sei $C_j \subseteq V$ die Menge aller Knoten, die zu b_j kontrahiert wurden (d.h. von den Knoten in C_j ist nur noch b_j in $V_0 \cup V_1 \cup V_2$ vorhanden)
- Es ist klar, dass die Größe $n_j = |C_j|$ von C_j ungerade ist, da C_j durch eine Folge von Kontraktionen auf b_j verkleinert wird und dabei jedesmal gerade viele Knoten aus C_j entfernt werden

Der Algorithmus von Edmonds

Beweis (Fortsetzung)

- Zudem sei $V'_0 = V_0 \cup C_1 \cup \dots \cup C_\ell = V \setminus (V_1 \cup V_2)$ die Menge aller geraden und zu einem geraden Knoten kontrahierten Knoten in G
- Dann kann es in G keine Kante $\{u, v\}$ zwischen V'_0 und V_2 geben
- Sonst hätte nämlich FindePfad eine Kante (u', v) mit $u' \in V'_0$ zu Q hinzugefügt und somit wäre v beim Abarbeiten dieser Kante ungerade geworden
- Zudem muss jede Kante $\{u, v\}$ in G mit $u, v \in V'_0$ in einer Menge C_j liegen
- Sonst wäre nämlich zu Q eine Kante (u', v') mit $\{u', v'\} \subseteq V'_0$ und $\{u', v'\} \not\subseteq C_j$ für $j = 1, \dots, l$ hinzugefügt worden, deren Abarbeitung entweder zu einer weiteren Blüte oder zu einem M -Pfad geführt hätte
- Folglich muss jede Kante $e \in E$ entweder
 - einen ungeraden Endpunkt haben (d.h. $e \cap V_1 \neq \emptyset$) oder
 - komplett in einer Menge C_j liegen (d.h. $\exists j : e \subseteq C_j$) oder
 - zwei unerreichte Knoten verbinden (d.h. $e \subseteq V_2$)

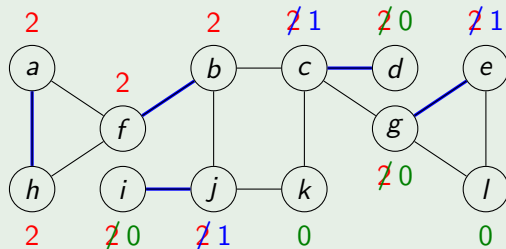
Beweis (Schluss)

- Nun können wir die Menge $S = \{u_1, \dots, u_k, C_1, \dots, C_\ell\}$, wobei $S_1 = \{u_1, \dots, u_k\}$ ist, wie folgt zu einer OSC erweitern:
 - Ist $|V_2| > 0$, so fügen wir einen beliebigen Knoten $u_0 \in V_2$ als Einzelknoten zu S hinzu
 - Ist $|V_2| > 2$, so fügen wir zu S noch die Menge $C_0 = V_2 \setminus \{u_0\}$ hinzu
- Da alle Knoten in V_2 durch eine Matchingkante $e \subseteq V_2$ gebunden sind, ist $|V_2|$ gerade, und somit ist $n_0 = |C_0| = |V_2| - 1 \geq 3$ und ungerade
- Dann ist S eine OSC für G , da jede Kante $e \in E$ entweder einen Endpunkt in S hat oder von einer der Mengen $C_j \in S$ überdeckt wird
- Zudem gilt $w(S) = |M|$ da sich M in $|S|$ Mengen
 - $M_i = \{e \in M \mid u_i \in e\}$ der Größe $|M_i| = 1$ und
 - $M'_j = \{e \in M \mid e \subseteq C_j\}$ der Größen $|M'_j| = (n_j - 1)/2$zerlegen lässt



Der Algorithmus von Edmonds

Beispiel



- In obigem Graphen G mit dem Matching $M = \{ah, bf, cd, eg, ij\}$ arbeitet FindePfad zunächst die Kanten kc , kj und le ab
- Anschließend wird beim Abarbeiten der Kante gl die Blüte $B_1 = lge$ gefunden und zu ihrer Basis $b_1 = l$ kontrahiert
- Nun bricht FindePfad die Suche nach einem M -Pfad erfolglos ab und wir erhalten wegen $V_0 = \{d, i, k, l\}$, $V_1 = \{c, j\}$, $V_2 = \{a, b, f, h\}$ und $C_1 = \{e, g, l\}$ die OSC $S = \{a, c, j, \{b, f, h\}, \{e, g, l\}\}$ in G

Der Algorithmus von Edmonds lässt sich leicht so erweitern, dass er in Zeit $O(nm)$ neben dem berechneten Matching M eine OSC S mit $w(S) = |M|$ ausgibt, um die Maximalität von M nachzuweisen

Korollar

- Für jeden Graphen G gilt

$$\mu(G) = \min\{w(S) : S \text{ ist eine OSC in } G\}$$

- Für bipartite Graphen G gilt (Satz von König)

$$\mu(G) = \min\{|C| : C \text{ ist eine Knotenüberdeckung in } G\}$$

- Im bipartiten Fall lässt sich eine (kleinste) Knotenüberdeckung C der Größe $|C| = \mu(G)$ in Zeit $O(m\sqrt{\mu(G)})$ berechnen

Beweis des Satzes von König

- Sei $G = (A, B, E)$ und sei $W = (V_W, E_W)$ der Suchwald beim Abbruch der erfolglosen Suche nach einem M -Pfad durch den Algorithmus von Edmonds
- Da G bipartit ist, gibt es keine ungeraden Kreise und somit keine Blüten
- Daher hat jede Kante $e \in E$ entweder einen ungeraden Endpunkt oder sie ist in V_2 enthalten
- Da jede Kante $e \in E$ einen Endpunkt in A hat, ist $C = V_1 \cup (V_2 \cap A)$ eine VC in G
- Zudem ist $|C| = |M|$, da keine Matchingkante zwei Endpunkte in C hat
- Um C in Zeit $O(m\sqrt{\mu})$ zu erhalten, berechnen wir zuerst mit dem Algorithmus von Diniz in Zeit $O(m\sqrt{\mu})$ ein maximales Matching M für G und starten danach die Prozedur `FindePfad(G, M)` zum Aufbau des Suchwalds W in Zeit $O(n + m)$, aus dem sich C ablesen lässt \square