WS 2003/04

Diskrete Strukturen I

Ernst W. Mayr

mayr@in.tum.de
Institut für Informatik
Technische Universität München

01-27-2004

Der Floyd-Warshall-Algorithmus für *apsp*

Gegeben sind ein Graph $G = (V, E)$ und eine Gewichtsfunktion $w : E \to \mathbb{R}^+ \cup \{+\infty\}$. Sei o. B. d. A. $V = \{0, \ldots, n-1\}$. Eine Gewichtsmatrix ist wie folgt definiert:

$$D = \big(w(v_i, v_j)\big)_{\substack{0 \le i < n \\ 0 \le j < n}}$$

Ziel ist es, eine $n \times n$-Matrix mit den Einträgen

$$d_{ij} = \text{ Länge eines kürzesten Weges von } i \text{ nach } j$$

zu berechnen. Dazu werden induktiv Matrizen $D^{(k)}$ mit Einträgen

$$d_{ij}^{(k)} = \begin{pmatrix} \text{Länge eines kürzesten Weges von } i \text{ nach } j, \\ \text{so dass alle inneren Knoten} < k \text{ sind} \end{pmatrix}$$

erzeugt.

Der Floyd-Warshall-Algorithmus für *apsp*

Gegeben sind ein Graph $G = (V, E)$ und eine Gewichtsfunktion $w : E \to \mathbb{R}^+ \cup \{+\infty\}$. Sei o. B. d. A. $V = \{0, \ldots, n-1\}$. Eine Gewichtsmatrix ist wie folgt definiert:

$$D = \big(w(v_i, v_j)\big)_{\substack{0 \le i < n \\ 0 \le j < n}}$$

Ziel ist es, eine $n \times n$-Matrix mit den Einträgen

$$d_{ij} = \text{ Länge eines kürzesten Weges von } i \text{ nach } j$$

zu berechnen. Dazu werden induktiv Matrizen $D^{(k)}$ mit Einträgen

$$d_{ij}^{(k)} = \left(\begin{array}{c} \text{Länge eines kürzesten Weges von } i \text{ nach } j, \\ \text{so dass alle inneren Knoten } < k \text{ sind} \end{array} \right)$$

erzeugt.

Der Floyd-Warshall-Algorithmus für *apsp*

Gegeben sind ein Graph $G = (V, E)$ und eine Gewichtsfunktion $w : E \to \mathbb{R}^+ \cup \{+\infty\}$. Sei o. B. d. A. $V = \{0, \ldots, n-1\}$. Eine Gewichtsmatrix ist wie folgt definiert:

$$D = \big(w(v_i, v_j)\big)_{\substack{0 \le i < n \\ 0 \le j < n}}$$

Ziel ist es, eine $n \times n$-Matrix mit den Einträgen

$$d_{ij} = \text{ Länge eines kürzesten Weges von } i \text{ nach } j$$

zu berechnen. Dazu werden induktiv Matrizen $D^{(k)}$ mit Einträgen

$$d_{ij}^{(k)} = \begin{pmatrix} \text{Länge eines kürzesten Weges von } i \text{ nach } j, \\ \text{so dass alle inneren Knoten} < k \text{ sind} \end{pmatrix}$$

erzeugt.

```
algorithm Floyd
  for i=0 to n-1 do
    for j=0 to n-1 do
      D^0[i,j] := w(v_i,v_j)
    od
  od
  for k=0 to n-1 do
    for i=0 to n-1 do
      for j=0 to n-1 do
        D^{k+1}[i,j] := min{D^k[i,j],D^k[i,k]+D^k[k,j]}
      od
    od
  od
end
```

Satz

Der Floyd-Algorithmus berechnet für alle $u, v \in V^2$ die Länge eines kürzesten Weges zwischen u und v, und zwar mit Zeitbedarf $O(n^3)$ und Platzbedarf $O(n^2)$.

Beweis.

Ersichtlich aus Algorithmus. □

Satz

Der Floyd-Algorithmus berechnet für alle $u, v \in V^2$ die Länge eines kürzesten Weges zwischen u und v, und zwar mit Zeitbedarf $O(n^3)$ und Platzbedarf $O(n^2)$.

Beweis.

Ersichtlich aus Algorithmus. □

Bemerkungen:

1. Zur Bestimmung der eigentlichen Pfade (und nicht nur der Entfernungen) muss bei der Minimum-Bestimmung jeweils das k gespeichert werden.
2. Der Algorithmus funktioniert auch, wenn *negative Kantengewichte* vorhanden sind, es jedoch keine *negativen Kreise* gibt.
3. Die Erweiterung auf Digraphen ist offensichtlich.

Bemerkungen:

1. Zur Bestimmung der eigentlichen Pfade (und nicht nur der Entfernungen) muss bei der Minimum-Bestimmung jeweils das k gespeichert werden.
2. Der Algorithmus funktioniert auch, wenn *negative Kantengewichte* vorhanden sind, es jedoch keine *negativen Kreise* gibt.
3. Die Erweiterung auf Digraphen ist offensichtlich.

Bemerkungen:

1. Zur Bestimmung der eigentlichen Pfade (und nicht nur der Entfernungen) muss bei der Minimum-Bestimmung jeweils das k gespeichert werden.
2. Der Algorithmus funktioniert auch, wenn *negative Kantengewichte* vorhanden sind, es jedoch keine *negativen Kreise* gibt.
3. Die Erweiterung auf Digraphen ist offensichtlich.

Dijkstras Algorithmus für *sssp*

Gegeben sind ein Graph $G = (V, E)$, ein Knoten $s \in V$ und eine Gewichtsfunktion $w : E \to \mathbb{R}^+ \cup \{\infty\}$.

```
algorithm Dijkstra
  F:=V\{s}
  for all v ∈ F do d[v] := w(s,v) od
  co d[s]=0 oc
  while F≠∅ do
    bestimme v ∈ F mit d[v] minimal
    F:=F\{v}
    for all w ∈ N(v) do
      d[w] := min{d[w], d[v]+w(v,w)}
    od
  od
end
```

Satz

Dijkstras Algorithmus berechnet $d(s,v)$ für alle $v \in V$; der Zeitaufwand ist $O(n^2)$, der Platzbedarf $O(n+e)$.

Beweis

Zeit- und Platzbedarf sind aus dem Algorithmus ersichtlich. Die Korrektheit zeigen wir mit einem Widerspruchsbeweis: Annahme: v sei der erste Knoten, so dass $d(s,v)$ falsch (d. h. zu groß) berechnet wird.

Satz

Dijkstras Algorithmus berechnet $d(s,v)$ für alle $v \in V$; der Zeitaufwand ist $O(n^2)$, der Platzbedarf $O(n+e)$.

Beweis

Zeit- und Platzbedarf sind aus dem Algorithmus ersichtlich. Die Korrektheit zeigen wir mit einem Widerspruchsbeweis: Annahme: v sei der erste Knoten, so dass $d(s,v)$ falsch (d. h. zu groß) berechnet wird.

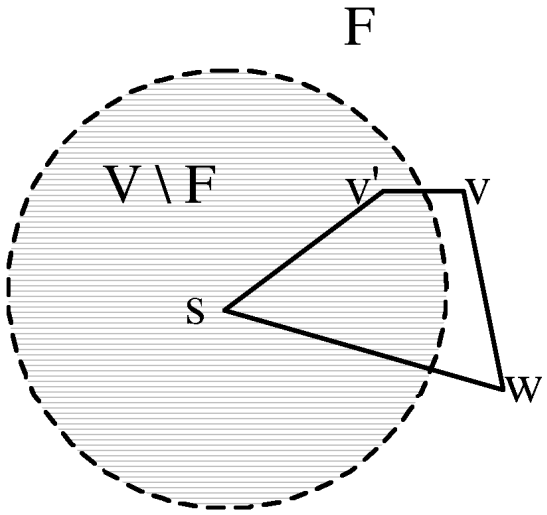
Satz

Dijkstras Algorithmus berechnet $d(s,v)$ für alle $v \in V$; der Zeitaufwand ist $O(n^2)$, der Platzbedarf $O(n+e)$.

Beweis

Zeit- und Platzbedarf sind aus dem Algorithmus ersichtlich. Die Korrektheit zeigen wir mit einem Widerspruchsbeweis: Annahme: v sei der erste Knoten, so dass $d(s,v)$ falsch (d. h. zu groß) berechnet wird.

Diese Situation illustriert folgendes Bild:

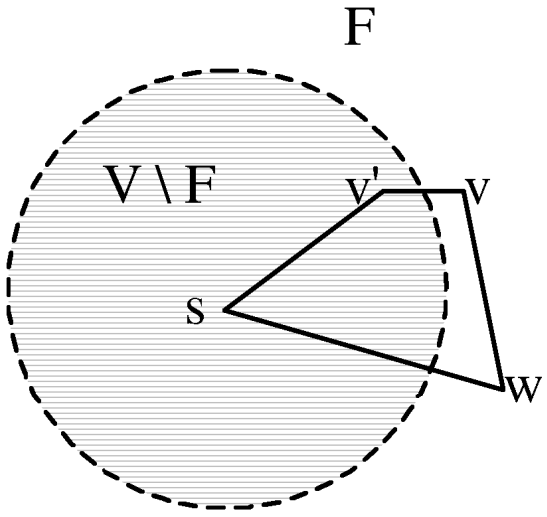


Nach Annahme muss dann gelten:

$$\mathsf{d}(w) + w(w,v) < \mathsf{d}(s,v') + \mathsf{d}(v',v) = \mathsf{d}(v) \ .$$

Damit wäre $\mathsf{d}(w)$ aber kleiner als $\mathsf{d}(v)$, und der Algorithmus hätte w und nicht v gewählt. *q. e. d.*

Diese Situation illustriert folgendes Bild:



Nach Annahme muss dann gelten:

$$\mathsf{d}(w) + w(w,v) < \mathsf{d}(s,v') + \mathsf{d}(v',v) = \mathsf{d}(v) \ .$$

Damit wäre $\mathsf{d}(w)$ aber kleiner als $\mathsf{d}(v)$, und der Algorithmus hätte w und nicht v gewählt. *q. e. d.*

Bemerkung: Mit besseren Datenstrukturen (*priority queues – z. B. Fibonacci heaps) kann Dijkstras Algorithmus so implementiert werden, dass er z. B. in Zeit* $O(e + n \cdot \log n)$ *läuft.*

Matchings

Definition

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

1. $M \subseteq E$ heißt *Matching, falls alle Kanten in M paarweise disjunkt sind.*
2. *M heißt maximales Matching, falls es kein Matching M' in G gibt mit $M \subsetneq M'$.*
3. *M heißt Matching maximaler Kardinalität (aka Maximum Matching), falls es in G kein Matching M' mit $|M'| > |M|$ gibt.*
4. *$\mathrm{m}(G)$ ist die Kardinalität eines Maximum Matchings in G.*

Matchings

Definition

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

1. $M \subseteq E$ heißt *Matching, falls alle Kanten in M paarweise disjunkt sind.*
2. *M heißt maximales Matching, falls es kein Matching M' in G gibt mit $M \subsetneq M'$.*
3. *M heißt Matching maximaler Kardinalität (aka Maximum Matching), falls es in G kein Matching M' mit $|M'| > |M|$ gibt.*
4. *$\mathrm{m}(G)$ ist die Kardinalität eines Maximum Matchings in G.*

Matchings

Definition

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

1. $M \subseteq E$ heißt *Matching, falls alle Kanten in M paarweise disjunkt sind.*
2. *M heißt maximales Matching, falls es kein Matching M' in G gibt mit $M \subsetneq M'$.*
3. *M heißt Matching maximaler Kardinalität (aka Maximum Matching), falls es in G kein Matching M' mit $|M'| > |M|$ gibt.*
4. *$\mathrm{m}(G)$ ist die Kardinalität eines Maximum Matchings in G.*

Matchings

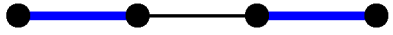
Definition

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

1. $M \subseteq E$ heißt *Matching, falls alle Kanten in M paarweise disjunkt sind.*
2. *M heißt maximales Matching, falls es kein Matching M' in G gibt mit $M \subsetneq M'$.*
3. *M heißt Matching maximaler Kardinalität (aka Maximum Matching), falls es in G kein Matching M' mit $|M'| > |M|$ gibt.*
4. *$\mathrm{m}(G)$ ist die Kardinalität eines Maximum Matchings in G.*

Beispiel

maximales Matching
(aber nicht Maximum)



Maximum-Matching
(natürlich auch maximal)

Matchings in bipartiten Graphen

Satz („Heiratssatz“)

Sei $G = (U, V, E)$ ein bipartiter Graph. Dann ist $\mathrm{m}(G) = |U|$ genau dann, wenn gilt:

$$(\forall A \subseteq U)\big[|A| \leq |N(A)|\big]$$

Beweis

„$\Rightarrow$“

Offensichtlich.

Matchings in bipartiten Graphen

Satz („Heiratssatz“)

Sei $G = (U, V, E)$ ein bipartiter Graph. Dann ist $\mathrm{m}(G) = |U|$ genau dann, wenn gilt:

$$(\forall A \subseteq U)\big[|A| \leq |N(A)|\big]$$

Beweis

„$\Rightarrow$“

Offensichtlich.

„$\Leftarrow$“

Sei M ein Maximum Matching in G.

Annahme: Ein Knoten $u = u_0 \in U$ sei in M *ungematcht*.

Dann existiert ein $v_1 \in N(u_0)$. Fall 1: v_1 ist in M ungematcht. Dann gilt: $M \cup \{u_0, v_1\}$ ist ein Matching, das größer ist als M, was einen Widerspruch darstellt. Fall 2: v_1 ist in M gematcht. Sei $\{u_1, v_1\} \in M$. Dann existiert nach Voraussetzung ein $v_2 \in N\big(\{u_0, u_1\}\big) \setminus \{v_1\}$. Falls v_2 ungematcht ist, stoppe hier. Falls v_2 gematcht ist, iteriere. Da $|M| < \infty$ ist, muss die Iteration mit einem ungematchten v_k enden.

„$\Leftarrow$“

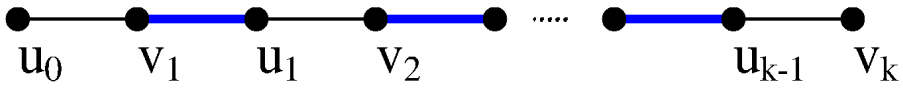
Sei M ein Maximum Matching in G.
Annahme: Ein Knoten $u = u_0 \in U$ sei in M *ungematcht*.
Dann existiert ein $v_1 \in N(u_0)$. Fall 1: v_1 ist in M ungematcht. Dann gilt: $M \cup \{u_0, v_1\}$ ist ein Matching, das größer ist als M, was einen Widerspruch darstellt. Fall 2: v_1 ist in M gematcht. Sei $\{u_1, v_1\} \in M$. Dann existiert nach Voraussetzung ein $v_2 \in N\big(\{u_0, u_1\}\big) \setminus \{v_1\}$. Falls v_2 ungematcht ist, stoppe hier. Falls v_2 gematcht ist, iteriere. Da $|M| < \infty$ ist, muss die Iteration mit einem ungematchten v_k enden.

„$\Leftarrow$“

Sei M ein Maximum Matching in G.
Annahme: Ein Knoten $u = u_0 \in U$ sei in M *ungematcht*.
Dann existiert ein $v_1 \in N(u_0)$. Fall 1: v_1 ist in M ungematcht. Dann gilt: $M \cup \{u_0, v_1\}$ ist ein Matching, das größer ist als M, was einen Widerspruch darstellt. Fall 2: v_1 ist in M gematcht. Sei $\{u_1, v_1\} \in M$. Dann existiert nach Voraussetzung ein $v_2 \in N\big(\{u_0, u_1\}\big) \setminus \{v_1\}$. Falls v_2 ungematcht ist, stoppe hier. Falls v_2 gematcht ist, iteriere. Da $|M| < \infty$ ist, muss die Iteration mit einem ungematchten v_k enden.

„⇐“(Fortsetzung)

Also existiert per Konstruktion ein Pfad wie in folgender Abbildung:

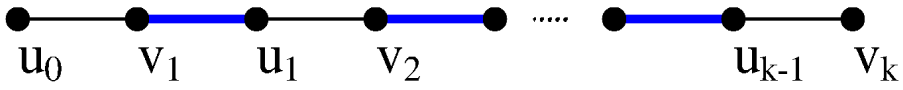


Ein solcher Pfad, bei dem sich gematchte und ungematchte Kanten abwechseln, heißt *alternierender Pfad. Sind, wie hier, Anfangs- und Endknoten ungematcht, heißt der Pfad auch augmentierend. Vertauscht man auf diesem Pfad gematchte und ungematchte Kanten, erhält man dadurch ein Matching* M' *mit* $|M'| = |M| + 1$*, was wiederum einen Widerspruch darstellt:*

q. e. d.

„⇐“(Fortsetzung)

Also existiert per Konstruktion ein Pfad wie in folgender Abbildung:

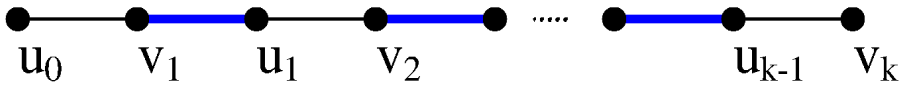


Ein solcher Pfad, bei dem sich gematchte und ungematchte Kanten abwechseln, heißt *alternierender Pfad. Sind, wie hier, Anfangs- und Endknoten ungematcht, heißt der Pfad auch augmentierend. Vertauscht man auf diesem Pfad gematchte und ungematchte Kanten, erhält man dadurch ein Matching* M' *mit* $|M'| = |M| + 1$*, was wiederum einen Widerspruch darstellt:*

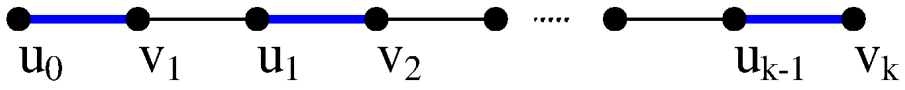
q. e. d.

„⇐“(Fortsetzung)

Also existiert per Konstruktion ein Pfad wie in folgender Abbildung:



Ein solcher Pfad, bei dem sich gematchte und ungematchte Kanten abwechseln, heißt *alternierender Pfad. Sind, wie hier, Anfangs- und Endknoten ungematcht, heißt der Pfad auch augmentierend. Vertauscht man auf diesem Pfad gematchte und ungematchte Kanten, erhält man dadurch ein Matching* M' *mit* $|M'| = |M| + 1$*, was wiederum einen Widerspruch darstellt:*



q. e. d.

Definition

Man definiert für einen bipartiten Graphen $G = (U, V, E)$ die Kenngröße:

$$\delta := \delta(G) := \max_{A \subseteq U} \{|A| - |N(A)|\}$$

Da bei der Maximumsbildung auch $A = \emptyset$ sein kann, ist $\delta \geq 0$.

Satz

Es gilt:

$$\mathrm{m}(G) = |U| - \delta \ .$$

Definition

Man definiert für einen bipartiten Graphen $G = (U, V, E)$ die Kenngröße:

$$\delta := \delta(G) := \max_{A \subseteq U} \{|A| - |N(A)|\}$$

Da bei der Maximumsbildung auch $A = \emptyset$ sein kann, ist $\delta \geq 0$.

Satz

Es gilt:

$$\mathrm{m}(G) = |U| - \delta \ .$$

Beweis

Dass $\mathrm{m}(G) \leq |U| - \delta$ gilt, ist offensichtlich. Wir zeigen nun noch, dass auch $\mathrm{m}(G) \geq |U| - \delta$ gilt, damit ist der Satz bewiesen.

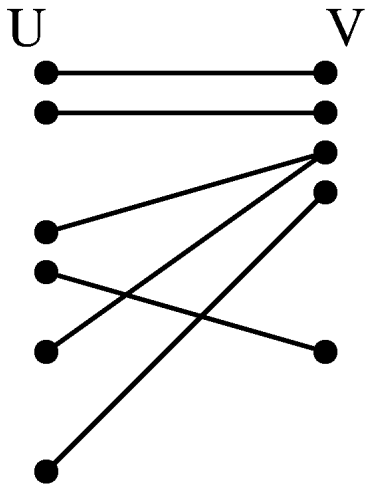
Betrachte folgenden Graphen:

Man fügt nun δ neue Knoten hinzu. Von diesen gehen Kanten zu allen Knoten in U, so dass ein $K_{|U|,\delta}$ entsteht.

Beweis

Dass $\mathrm{m}(G) \leq |U| - \delta$ gilt, ist offensichtlich. Wir zeigen nun noch, dass auch $\mathrm{m}(G) \geq |U| - \delta$ gilt, damit ist der Satz bewiesen.

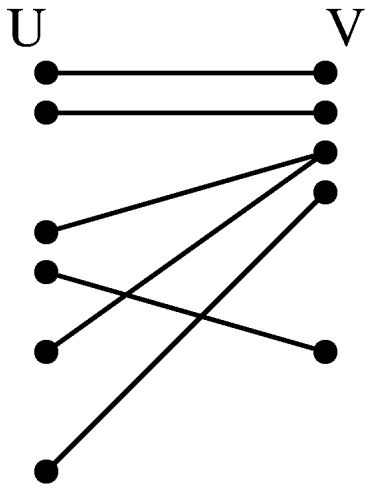
Betrachte folgenden Graphen:



Man fügt nun δ neue Knoten hinzu. Von diesen gehen Kanten zu allen Knoten in U, so dass ein $K_{|U|,\delta}$ entsteht.

Beweis

Dass $\mathrm{m}(G) \leq |U| - \delta$ gilt, ist offensichtlich. Wir zeigen nun noch, dass auch $\mathrm{m}(G) \geq |U| - \delta$ gilt, damit ist der Satz bewiesen.
Betrachte folgenden Graphen:



Man fügt nun δ neue Knoten hinzu. Von diesen gehen Kanten zu allen Knoten in U, so dass ein $K_{|U|,\delta}$ entsteht.

Der neue Graph erfüllt die Voraussetzungen des Heiratssatzes.
Damit gibt es im neuen Graphen ein Matching M' mit $|M'| = |U|$.
Daraus folgt, dass es im alten Graphen ein Matching der Kardinalität $\geq |U| - \delta$ geben muss. *q. e. d.*

Definition

$D \subseteq U \uplus V$ heißt *Träger oder Knotenüberdeckung (vertex cover, VC) von G, wenn jede Kante in G zu mindestens einem $u \in D$ inzident ist.*

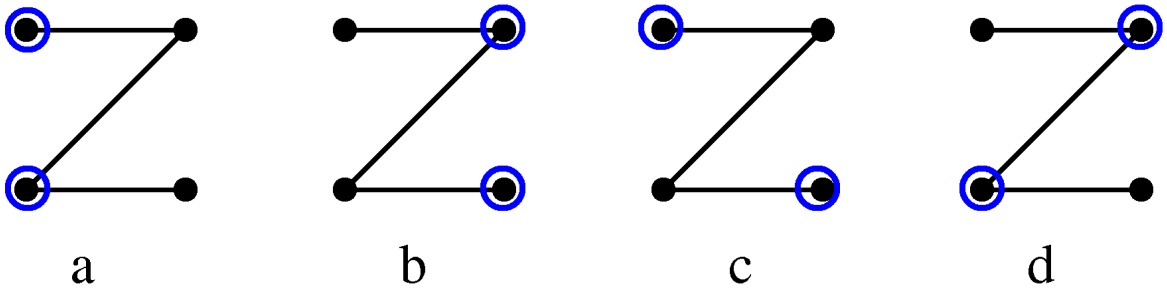
Beispiel

In den Fällen a, b und d sind Träger gezeigt, in c nicht.

Definition

$D \subseteq U \uplus V$ heißt *Träger oder Knotenüberdeckung (vertex cover, VC) von G, wenn jede Kante in G zu mindestens einem $u \in D$ inzident ist.*

Beispiel



In den Fällen a, b und d sind Träger gezeigt, in c nicht.

Satz

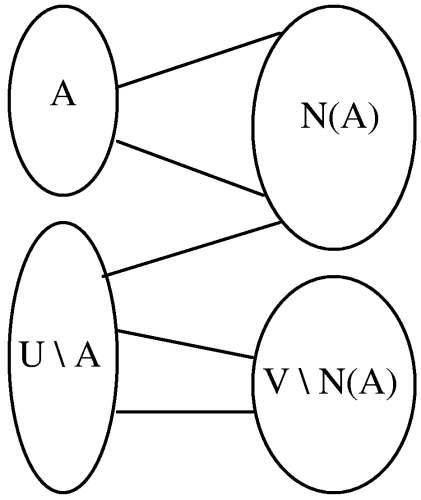
Es gilt:

$$\max\{|M|; M \text{ Matching}\} = \min\{|D|; D \text{ Träger}\}$$

Beweis.

„$\leq$“ Offensichtlich.

„$\geq$“ Für ein geeignetes $A \subseteq U$ gilt
$\mathrm{m}(G) = |U| - \delta(G) = |U \setminus A| + |N(A)|$:



$(U \setminus A) \cup N(A)$ ist Träger von G.

□

Sei

$$M = \left(m_{ij}\right)_{\substack{1 \le i \le n \\ 1 \le j \le n}}$$

eine (quadratische) Matrix mit $m_{ij} \geq 0$. Alle Zeilen- und Spaltensummen von M seien gleich $r > 0$. Man ordnet nun M den bipartiten Graphen $G = (U, V, E)$ zu mit

$$U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\} \text{ und } \{u_1, v_j\} \in E \Leftrightarrow m_{ij} > 0.$$

Ein Matching in G entspricht einer Menge von Positionen in M, die alle in verschiedenen Zeilen und Spalten liegen.

Sei

$$M = \left(m_{ij}\right)_{\substack{1 \le i \le n \\ 1 \le j \le n}}$$

eine (quadratische) Matrix mit $m_{ij} \geq 0$. Alle Zeilen- und Spaltensummen von M seien gleich $r > 0$. Man ordnet nun M den bipartiten Graphen $G = (U, V, E)$ zu mit

$$U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\} \text{ und } \{u_1, v_j\} \in E \Leftrightarrow m_{ij} > 0.$$

Ein Matching in G entspricht einer Menge von Positionen in M, die alle in verschiedenen Zeilen und Spalten liegen.

Sei

$$M = \left(m_{ij}\right)_{\substack{1 \le i \le n \\ 1 \le j \le n}}$$

eine (quadratische) Matrix mit $m_{ij} \geq 0$. Alle Zeilen- und Spaltensummen von M seien gleich $r > 0$. Man ordnet nun M den bipartiten Graphen $G = (U, V, E)$ zu mit

$$U = \{u_1, \ldots, u_n\}, V = \{v_1, \ldots, v_n\} \text{ und } \{u_1, v_j\} \in E \Leftrightarrow m_{ij} > 0.$$

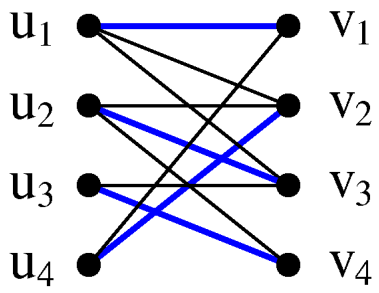
Ein Matching in G entspricht einer Menge von Positionen in M, die alle in verschiedenen Zeilen und Spalten liegen.

Beispiel

Die Matrix

$$\begin{pmatrix} \boxed{3} & 1 & 1 & 0 \\ 0 & 1 & \boxed{2} & 2 \\ 0 & 0 & 2 & \boxed{3} \\ 2 & \boxed{3} & 0 & 0 \end{pmatrix}$$

entspricht dem Graphen



Bemerkung: Ein Träger D von G ist also eine Menge von Zeilen und Spalten von M, die zusammen alle Einträge $m_{ij} > 0$ enthalten.

Definition

Eine Menge von Positionen (in M), die alle in verschiedenen Zeilen und in verschiedenen Spalten liegen, heißt *Diagonale von M.*

Eine Diagonale der Größe n muss in M existieren, denn falls M keine solche Diagonale hat, gibt es nach Satz 10 e Zeilen und f Spalten mit $e + f < n$, die zusammen alle Einträge > 0 von M enthalten. Die Gesamtsumme der Einträge in M wäre dann

$$n \cdot r = \sum_{i,j} m_{ij} \leq (e+f) \cdot r < r \cdot n,$$

was offensichtlich ein Widerspruch ist.

Bemerkung: Ein Träger D von G ist also eine Menge von Zeilen und Spalten von M, die zusammen alle Einträge $m_{ij} > 0$ enthalten.

Definition

Eine Menge von Positionen (in M), die alle in verschiedenen Zeilen und in verschiedenen Spalten liegen, heißt *Diagonale von M.*

Eine Diagonale der Größe n muss in M existieren, denn falls M keine solche Diagonale hat, gibt es nach Satz 10 e Zeilen und f Spalten mit $e + f < n$, die zusammen alle Einträge > 0 von M enthalten. Die Gesamtsumme der Einträge in M wäre dann

$$n \cdot r = \sum_{i,j} m_{ij} \leq (e + f) \cdot r < r \cdot n,$$

was offensichtlich ein Widerspruch ist.

Bemerkung: Ein Träger D von G ist also eine Menge von Zeilen und Spalten von M, die zusammen alle Einträge $m_{ij} > 0$ enthalten.

Definition

Eine Menge von Positionen (in M), die alle in verschiedenen Zeilen und in verschiedenen Spalten liegen, heißt *Diagonale von M.*

Eine Diagonale der Größe n muss in M existieren, denn falls M keine solche Diagonale hat, gibt es nach Satz 10 e Zeilen und f Spalten mit $e + f < n$, die zusammen alle Einträge > 0 von M enthalten. Die Gesamtsumme der Einträge in M wäre dann

$$n \cdot r = \sum_{i,j} m_{ij} \leq (e + f) \cdot r < r \cdot n,$$

was offensichtlich ein Widerspruch ist.

Bemerkung: Ein Träger D von G ist also eine Menge von Zeilen und Spalten von M, die zusammen alle Einträge $m_{ij} > 0$ enthalten.

Definition

Eine Menge von Positionen (in M), die alle in verschiedenen Zeilen und in verschiedenen Spalten liegen, heißt *Diagonale von M.*

Eine Diagonale der Größe n muss in M existieren, denn falls M keine solche Diagonale hat, gibt es nach Satz 10 e Zeilen und f Spalten mit $e + f < n$, die zusammen alle Einträge > 0 von M enthalten. Die Gesamtsumme der Einträge in M wäre dann

$$n \cdot r = \sum_{i,j} m_{ij} \leq (e + f) \cdot r < r \cdot n,$$

was offensichtlich ein Widerspruch ist.

Sei c_1 der minimale Eintrag > 0 in M, und sei P_1 die zu einer Diagonale der Größe n gehörige Permutationsmatrix (d. h. Einträge $= 1$ an den Positionen der Diagonale, 0 sonst). Dann gilt:

$$M_1 := M - c_1 P_1$$

ist eine $n \times n$-Matrix mit allen Zeilen- und Spaltensummen $= r - c_1$. Die Matrix M_1 enthält damit mehr Nullen als M. Damit haben wir gezeigt:

Sei c_1 der minimale Eintrag > 0 in M, und sei P_1 die zu einer Diagonale der Größe n gehörige Permutationsmatrix (d. h. Einträge $= 1$ an den Positionen der Diagonale, 0 sonst). Dann gilt:

$$M_1 := M - c_1 P_1$$

ist eine $n \times n$-Matrix mit allen Zeilen- und Spaltensummen $= r - c_1$. Die Matrix M_1 enthält damit mehr Nullen als M. Damit haben wir gezeigt:

Sei c_1 der minimale Eintrag > 0 in M, und sei P_1 die zu einer Diagonale der Größe n gehörige Permutationsmatrix (d. h. Einträge $= 1$ an den Positionen der Diagonale, 0 sonst). Dann gilt:

$$M_1 := M - c_1 P_1$$

ist eine $n \times n$-Matrix mit allen Zeilen- und Spaltensummen $= r - c_1$. Die Matrix M_1 enthält damit mehr Nullen als M. Damit haben wir gezeigt:

Satz

Sei M wie oben. Dann gibt es für ein geeignetes k Konstanten $c_i > 0$ und Permutationsmatrizen P_i, $i = 1, \ldots, k$, so dass gilt

$$M = \sum_{i=1}^{k} c_i P_i \qquad \sum_{i=1}^{k} c_i = r \ .$$