WS 2003/04

Diskrete Strukturen I

Ernst W. Mayr

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

02-03-2004

Konstruktion optimaler Matchings

Satz

Ein Matching M ist genau dann Maximum, wenn es dazu keinen augmentierenden Pfad gibt.

Beweis.

„$\Rightarrow$“

Offensichtlich.

„$\Leftarrow$“

Sei M ein Matching, zu dem es keinen augmentierenden Pfad gibt. Annahme, M sei kein Maximum Matching, es existiere ein Maximum Matching M'. Betrachte nun $M \Delta M'$. Die Zusammenhangskomponenten dieses Graphen sind alternierende Pfade und Kreise gerader Länge. Da $|M'| > |M|$ gilt, muss es einen alternierenden Pfad mit ungerader Länge geben, der mit Kanten aus M' beginnt und endet. Dies ist aber ein *augmentierender* Pfad.

□

Konstruktion optimaler Matchings

Satz

Ein Matching M ist genau dann Maximum, wenn es dazu keinen augmentierenden Pfad gibt.

Beweis.

„$\Rightarrow$“

Offensichtlich.

„$\Leftarrow$“

Sei M ein Matching, zu dem es keinen augmentierenden Pfad gibt. Annahme, M sei kein Maximum Matching, es existiere ein Maximum Matching M'. Betrachte nun $M \Delta M'$. Die Zusammenhangskomponenten dieses Graphen sind alternierende Pfade und Kreise gerader Länge. Da $|M'| > |M|$ gilt, muss es einen alternierenden Pfad mit ungerader Länge geben, der mit Kanten aus M' beginnt und endet. Dies ist aber ein *augmentierender* Pfad.

Konstruktion optimaler Matchings

Satz

Ein Matching M ist genau dann Maximum, wenn es dazu keinen augmentierenden Pfad gibt.

Beweis.

„$\Rightarrow$“

Offensichtlich.

„$\Leftarrow$“

Sei M ein Matching, zu dem es keinen augmentierenden Pfad gibt. Annahme, M sei kein Maximum Matching, es existiere ein Maximum Matching M'. Betrachte nun $M \Delta M'$. Die Zusammenhangskomponenten dieses Graphen sind alternierende Pfade und Kreise gerader Länge. Da $|M'| > |M|$ gilt, muss es einen alternierenden Pfad mit ungerader Länge geben, der mit Kanten aus M' beginnt und endet. Dies ist aber ein *augmentierender* Pfad.

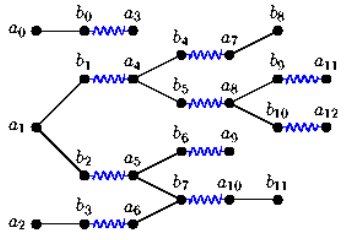
□

Der Algorithmus zur Konstruktion optimaler Matchings ist eine *parallele (simultane) alternierende Breitensuche.*

Beispiel (Konstruktion im bipartiten Graph)

Der Algorithmus zur Konstruktion optimaler Matchings ist eine *parallele (simultane) alternierende Breitensuche.*

Beispiel (Konstruktion im bipartiten Graph)



Ergebnisse und Erweiterungen:

	bipartit	allgemein
ungewichtet	$O(\sqrt{\|V\|} \cdot \|E\|)$	$O(\sqrt{\|V\|} \cdot \|E\|)$
gewichtet	$O(\|V\| \cdot (\|E\| + \|V\| \cdot \log(\|V\|)))$	$O(\|V\| \cdot \|E\| \cdot \log(\|V\|))$

Siehe auch:

Zvi Galil: Efficient algorithms for finding maximum matchings in graphs, ACM Computing Surveys 18 (1986), pp. 23–38

Ergebnisse und Erweiterungen:

	bipartit	allgemein
ungewichtet	$O(\sqrt{\|V\|} \cdot \|E\|)$	$O(\sqrt{\|V\|} \cdot \|E\|)$
gewichtet	$O(\|V\| \cdot (\|E\| + \|V\| \cdot \log(\|V\|)))$	$O(\|V\| \cdot \|E\| \cdot \log(\|V\|))$

Siehe auch:

Zvi Galil: Efficient algorithms for finding maximum matchings in graphs, ACM Computing Surveys 18 (1986), pp. 23–38

Ergebnisse und Erweiterungen:

	bipartit	allgemein
ungewichtet	$O(\sqrt{\|V\|} \cdot \|E\|)$	$O(\sqrt{\|V\|} \cdot \|E\|)$
gewichtet	$O(\|V\| \cdot (\|E\| + \|V\| \cdot \log(\|V\|)))$	$O(\|V\| \cdot \|E\| \cdot \log(\|V\|))$

Siehe auch:

Zvi Galil: Efficient algorithms for finding maximum matchings in graphs, ACM Computing Surveys 18 (1986), pp. 23–38

Reguläre bipartite Graphen

Lemma
Sei $G = (U, V, E)$ ein k-regulärer bipartiter Graph ($k \in \mathbb{N}$). Dann hat G ein perfektes Matching.

Beweis.
Sei $A \subseteq U$ und $B = N(A) \subseteq V$. Dann ist $|A| \leq |B|$, da ja alle von A ausgehenden Kanten in B enden und, falls $|B| < |A|$, es in B damit einen Knoten mit Grad $> k$ geben müsste. □

Korollar 18: S ei $G = (U, V, E)$ ein k-regulärer bipartiter Graph ($k \in \mathbb{N}$). Dann lässt sich E als disjunkte Vereinigung von k perfekten Matchings darstellen.

Reguläre bipartite Graphen

Lemma
Sei $G = (U, V, E)$ ein k-regulärer bipartiter Graph ($k \in \mathbb{N}$). Dann hat G ein perfektes Matching.

Beweis.
Sei $A \subseteq U$ und $B = N(A) \subseteq V$. Dann ist $|A| \leq |B|$, da ja alle von A ausgehenden Kanten in B enden und, falls $|B| < |A|$, es in B damit einen Knoten mit Grad $> k$ geben müsste. □

Korollar 18: S ei $G = (U, V, E)$ ein k-regulärer bipartiter Graph ($k \in \mathbb{N}$). Dann lässt sich E als disjunkte Vereinigung von k perfekten Matchings darstellen.

Reguläre bipartite Graphen

Lemma
Sei $G = (U, V, E)$ ein k-regulärer bipartiter Graph ($k \in \mathbb{N}$). Dann hat G ein perfektes Matching.

Beweis.
Sei $A \subseteq U$ und $B = N(A) \subseteq V$. Dann ist $|A| \leq |B|$, da ja alle von A ausgehenden Kanten in B enden und, falls $|B| < |A|$, es in B damit einen Knoten mit Grad $> k$ geben müsste. □

Korollar 18: S ei $G = (U, V, E)$ ein k-regulärer bipartiter Graph ($k \in \mathbb{N}$). Dann lässt sich E als disjunkte Vereinigung von k perfekten Matchings darstellen.

Transversalen

Definition

Sei $G = (U, V, E)$ ein bipartiter Graph, M ein Matching in G, und $A \subseteq U$ die in M gematchte Teilmenge der Knotenmenge U. Dann heißt A eine *Transversale in G.*

Satz

Sei $G = (U, V, E)$ ein bipartiter Graph, $\mathcal{T} \subseteq 2^U$ die Menge der Transversalen in G. Dann ist $(U, \mathcal{T})$ ein Matroid.

Beweis

Die ersten beiden Bedingungen für ein Matroid sind klarerweise erfüllt:

1. $\emptyset \in \mathcal{T}$
2. $B \subset A, A \in \mathcal{T} \Rightarrow B \in \mathcal{T}$

Transversalen

Definition

Sei $G = (U, V, E)$ ein bipartiter Graph, M ein Matching in G, und $A \subseteq U$ die in M gematchte Teilmenge der Knotenmenge U. Dann heißt A eine *Transversale in G.*

Satz

Sei $G = (U, V, E)$ ein bipartiter Graph, $\mathcal{T} \subseteq 2^U$ die Menge der Transversalen in G. Dann ist $(U, \mathcal{T})$ ein Matroid.

Beweis

Die ersten beiden Bedingungen für ein Matroid sind klarerweise erfüllt:

1. $\emptyset \in \mathcal{T}$
2. $B \subset A, A \in \mathcal{T} \Rightarrow B \in \mathcal{T}$

Transversalen

Definition

Sei $G = (U, V, E)$ ein bipartiter Graph, M ein Matching in G, und $A \subseteq U$ die in M gematchte Teilmenge der Knotenmenge U. Dann heißt A eine *Transversale in G.*

Satz

Sei $G = (U, V, E)$ ein bipartiter Graph, $\mathcal{T} \subseteq 2^U$ die Menge der Transversalen in G. Dann ist $(U, \mathcal{T})$ ein Matroid.

Beweis

Die ersten beiden Bedingungen für ein Matroid sind klarerweise erfüllt:

1. $\emptyset \in \mathcal{T}$
2. $B \subset A, A \in \mathcal{T} \Rightarrow B \in \mathcal{T}$

Seien nun A und A' Transversalen mit den zugehörigen Matchings M und M', und sei $|A'| = |A| + 1$, also auch $|M'| = |M| + 1$. Betrachte $M' \Delta M$. Dann muss $M' \Delta M$ (mindestens) einen Pfad ungerader Länge enthalten, der mit einer Kante in M' beginnt und mit einer Kante in M' endet (und dazwischen abwechselnd Kanten in M bzw. M' enthält). Dieser Pfad ist ein augmentierender Pfad bzgl. M, und einer der beiden Endpunkte liegt in $A' \setminus A$, kann also zu A hinzugenommen werden. *q. e. d.*

Seien nun A und A' Transversalen mit den zugehörigen Matchings M und M', und sei $|A'| = |A| + 1$, also auch $|M'| = |M| + 1$. Betrachte $M'\Delta M$. Dann muss $M'\Delta M$ (mindestens) einen Pfad ungerader Länge enthalten, der mit einer Kante in M' beginnt und mit einer Kante in M' endet (und dazwischen abwechselnd Kanten in M bzw. M' enthält). Dieser Pfad ist ein augmentierender Pfad bzgl. M, und einer der beiden Endpunkte liegt in $A' \setminus A$, kann also zu A hinzugenommen werden. *q. e. d.*

Anwendung: gewichtetes Zuweisungsproblem, Variante 1

n Nutzer wollen jeweils auf eine aus einer nutzerspezifischen Teilmenge von insgesamt m Ressourcen zugreifen. Jede Ressource kann aber nur von höchstens einem Nutzer in Anspruch genommen werden. Der *Wert einer Zuweisung von Ressourcen zu (interessierten) Nutzern ergibt sich als die Summe*

$$\sum_{i \in A} w_i \ ,$$

wobei die Zuweisung einem Matching in dem durch Nutzer, Ressourcen und Zugriffswünsche gegebenen Graphen entspricht, $w_i \in \mathbb{R}^+$ *ein Gewicht für jeden Nutzer* $i \in \{1, \ldots, n\}$ *ist, und* A *die durch die Zuweisung (das Matching) bedachte Teilmenge der Nutzer ist.*

Gewichtetes Matching in bipartiten Graphen

Wir betrachten nun eine zweite Variante eines *Zuweisungsproblems, das durch bipartite Graphen* $G = (U, V, E)$ *mit einer Gewichtsfunktion* $w : E \to \mathbb{R}^+$ *gegeben ist. Das Gewicht eines Matchings* $M \subseteq E$ *ist dann*

$$\sum_{e \in M} w(e) \; .$$

Wir können dann o.B.d.A. annehmen, dass $|U| = |V|$ $(= n)$ *und* G *vollständig bipartit (also* $G = K_{n,n}$*) ist, indem wir zunächst die kleinere der beiden Mengen* U *und* V *mit zusätzlichen Knoten auffüllen und dann die fehlenden Kanten durch Kanten mit Gewicht* 0 *ersetzen.*

Gewichtetes Matching in bipartiten Graphen

Wir betrachten nun eine zweite Variante eines *Zuweisungsproblems, das durch bipartite Graphen* $G = (U, V, E)$ *mit einer Gewichtsfunktion* $w : E \to \mathbb{R}^+$ *gegeben ist. Das Gewicht eines Matchings* $M \subseteq E$ *ist dann*

$$\sum_{e \in M} w(e) \; .$$

Wir können dann o.B.d.A. annehmen, dass $|U| = |V|$ $(= n)$ *und* G *vollständig bipartit (also* $G = K_{n,n}$*) ist, indem wir zunächst die kleinere der beiden Mengen* U *und* V *mit zusätzlichen Knoten auffüllen und dann die fehlenden Kanten durch Kanten mit Gewicht* 0 *ersetzen.*

Damit suchen wir in G *optimale perfekte* Matchings. Wir können das Problem, ein perfektes Matching *maximalen Gewichts reduzieren auf das Problem, ein perfektes Matching minimalen Gewichts zu bestimmen, indem wir jedes Gewicht* $w(e)$ *durch*

$$\max_{e \in E} w(e) \; - \; w(e)$$

ersetzen. Wir nehmen daher an, dass wir o.B.d.A. ein perfektes Matching minimalen Gewichts in (G, w) *suchen. Für die folgende Diskussion nehmen wir zur Vereinfachung weiter an, dass alle Gewichte* $\in \mathbb{N}_0$ *sind.*

Damit suchen wir in G *optimale perfekte* Matchings. Wir können das Problem, ein perfektes Matching *maximalen Gewichts reduzieren auf das Problem, ein perfektes Matching minimalen Gewichts zu bestimmen, indem wir jedes Gewicht* $w(e)$ *durch*

$$\max_{e \in E} w(e) \; - \; w(e)$$

ersetzen. Wir nehmen daher an, dass wir o.B.d.A. ein perfektes Matching minimalen Gewichts in (G, w) *suchen. Für die folgende Diskussion nehmen wir zur Vereinfachung weiter an, dass alle Gewichte* $\in \mathbb{N}_0$ *sind.*

Damit suchen wir in G *optimale perfekte* Matchings. Wir können das Problem, ein perfektes Matching *maximalen Gewichts reduzieren auf das Problem, ein perfektes Matching minimalen Gewichts zu bestimmen, indem wir jedes Gewicht* $w(e)$ *durch*

$$\max_{e \in E} w(e) \; - \; w(e)$$

ersetzen. Wir nehmen daher an, dass wir o.B.d.A. ein perfektes Matching minimalen Gewichts in (G, w) *suchen. Für die folgende Diskussion nehmen wir zur Vereinfachung weiter an, dass alle Gewichte* $\in \mathbb{N}_0$ *sind.*

Sei

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

die zu (G, w) gehörige Gewichtsmatrix, und sei

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

eine Permutationsmatrix (d.h., jede Zeile und jede Spalte von P enthält genau eine 1 und ansonsten nur Einträge 0). Die Permutationsmatrix P entspricht einem perfekten Matching M in G mit Gewicht

$$\sum_{i,j} p_{ij} w_{ij} \ .$$

Sei

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

die zu (G, w) gehörige Gewichtsmatrix, und sei

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

eine Permutationsmatrix (d.h., jede Zeile und jede Spalte von P enthält genau eine 1 und ansonsten nur Einträge 0). Die Permutationsmatrix P entspricht einem perfekten Matching M in G mit Gewicht

$$\sum_{i,j} p_{ij} w_{ij} \ .$$

Beobachtung:

Wenn wir von jedem Element einer Zeile (oder Spalte) in W einen festen Betrag p subtrahieren, verringert sich das Gewicht eines jeden perfekten Matchings M um diesen Betrag p, die relative Ordnung (nach Gewicht) unter den perfekten Matchings bleibt bestehen, insbesondere gehen optimale Matchings wieder in optimale Matchings über. Wir führen nun solche Zeilen- und

Spaltenumformungen durch, um eine *Diagonale mit möglichst vielen Einträgen* $= 0$ *zu erhalten.*

Beobachtung:

Wenn wir von jedem Element einer Zeile (oder Spalte) in W einen festen Betrag p subtrahieren, verringert sich das Gewicht eines jeden perfekten Matchings M um diesen Betrag p, die relative Ordnung (nach Gewicht) unter den perfekten Matchings bleibt bestehen, insbesondere gehen optimale Matchings wieder in optimale Matchings über. Wir führen nun solche Zeilen- und

Spaltenumformungen durch, um eine *Diagonale mit möglichst vielen Einträgen* $= 0$ *zu erhalten.*

Beispiel

Sei

$$W = \begin{pmatrix} 9 & 11 & 12 & 11 \\ 6 & 3 & 8 & 5 \\ 7 & 6 & 13 & 11 \\ 9 & 10 & 10 & 7 \end{pmatrix}$$

Nachdem wir von jeder Zeile das minimale Gewicht subtrahieren, erhalten wir

$$W' = \begin{pmatrix} 0 & 2 & 3 & 2 \\ 3 & 0 & 5 & 2 \\ 1 & 0 & 7 & 5 \\ 2 & 3 & 3 & 0 \end{pmatrix}$$

Beispiel

Sei

$$W = \begin{pmatrix} 9 & 11 & 12 & 11 \\ 6 & 3 & 8 & 5 \\ 7 & 6 & 13 & 11 \\ 9 & 10 & 10 & 7 \end{pmatrix}$$

Nachdem wir von jeder Zeile das minimale Gewicht subtrahieren, erhalten wir

$$W' = \begin{pmatrix} 0 & 2 & 3 & 2 \\ 3 & 0 & 5 & 2 \\ 1 & 0 & 7 & 5 \\ 2 & 3 & 3 & 0 \end{pmatrix}$$

$$W' = \begin{pmatrix} 0 & 2 & 3 & 2 \\ 3 & 0 & 5 & 2 \\ 1 & 0 & 7 & 5 \\ 2 & 3 & 3 & 0 \end{pmatrix}$$

Nachdem wir von jeder Spalte das minimale Gewicht subtrahieren, erhalten wir

$$W'' = \begin{pmatrix} 0 & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & 0 & 4 & 5 \\ 2 & 3 & 0 & 0 \end{pmatrix}$$

$$W' = \begin{pmatrix} 0 & 2 & 3 & 2 \\ 3 & 0 & 5 & 2 \\ 1 & 0 & 7 & 5 \\ 2 & 3 & 3 & 0 \end{pmatrix}$$

Nachdem wir von jeder Spalte das minimale Gewicht subtrahieren, erhalten wir

$$W'' = \begin{pmatrix} 0 & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & 0 & 4 & 5 \\ 2 & 3 & 0 & 0 \end{pmatrix}$$

Diese Matrix enthält eine Diagonale der Größe 3 mit Einträgen $= 0$:

$$W'' = \begin{pmatrix} \boxed{0} & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & \boxed{0} & 4 & 5 \\ 2 & 3 & \boxed{0} & 0 \end{pmatrix}$$

Aus Satz *17 folgt, dass die maximale Länge einer* 0*-Diagonale*

gleich der minimalen Anzahl von Zeilen und Spalten ist, die alle 0*en bedecken. Falls wir noch keine* 0*-Diagonale der Länge* n *haben, iterieren wir folgenden Algorithmus:*

Diese Matrix enthält eine Diagonale der Größe 3 mit Einträgen $= 0$:

$$W'' = \begin{pmatrix} \boxed{0} & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & \boxed{0} & 4 & 5 \\ 2 & 3 & \boxed{0} & 0 \end{pmatrix}$$

Aus Satz *17 folgt, dass die maximale Länge einer* 0*-Diagonale*

gleich der minimalen Anzahl von Zeilen und Spalten ist, die alle 0*en bedecken. Falls wir noch keine* 0*-Diagonale der Länge* n *haben, iterieren wir folgenden Algorithmus:*

Diese Matrix enthält eine Diagonale der Größe 3 mit Einträgen $= 0$:

$$W'' = \begin{pmatrix} \boxed{0} & 2 & 0 & 2 \\ 3 & 0 & 2 & 2 \\ 1 & \boxed{0} & 4 & 5 \\ 2 & 3 & \boxed{0} & 0 \end{pmatrix}$$

Aus Satz *17 folgt, dass die maximale Länge einer* 0*-Diagonale*

gleich der minimalen Anzahl von Zeilen und Spalten ist, die alle 0*en bedecken. Falls wir noch keine* 0*-Diagonale der Länge* n *haben, iterieren wir folgenden Algorithmus:*

1. finde eine minimale Anzahl von e Zeilen und f Spalten $(e + f < n)$, die zusammen alle Einträge $= 0$ enthalten;
2. sei w das Minimum der nicht überdeckten Elemente;
3. subtrahiere w von den $n - e$ nicht überdeckten Zeilen;
4. addiere w zu den f überdeckten Spalten.

Die Gewichte ändern sich also wie folgt:

1. um $-w$, falls (i, j) nicht überdeckt ist;
2. um 0, falls (i, j) von einer Zeile *oder* Spalte überdeckt ist, aber nicht beides;
3. um $+w$, falls (i, j) von einer Zeile *und* einer Spalte überdeckt ist.

1. finde eine minimale Anzahl von e Zeilen und f Spalten $(e + f < n)$, die zusammen alle Einträge $= 0$ enthalten;
2. sei w das Minimum der nicht überdeckten Elemente;
3. subtrahiere w von den $n - e$ nicht überdeckten Zeilen;
4. addiere w zu den f überdeckten Spalten.

Die Gewichte ändern sich also wie folgt:

1. um $-w$, falls (i, j) nicht überdeckt ist;
2. um 0, falls (i, j) von einer Zeile *oder* Spalte überdeckt ist, aber nicht beides;
3. um $+w$, falls (i, j) von einer Zeile *und* einer Spalte überdeckt ist.

1. finde eine minimale Anzahl von e Zeilen und f Spalten $(e + f < n)$, die zusammen alle Einträge $= 0$ enthalten;
2. sei w das Minimum der nicht überdeckten Elemente;
3. subtrahiere w von den $n - e$ nicht überdeckten Zeilen;
4. addiere w zu den f überdeckten Spalten.

Die Gewichte ändern sich also wie folgt:

1. um $-w$, falls (i, j) nicht überdeckt ist;
2. um 0, falls (i, j) von einer Zeile *oder* Spalte überdeckt ist, aber nicht beides;
3. um $+w$, falls (i, j) von einer Zeile *und* einer Spalte überdeckt ist.

1. finde eine minimale Anzahl von e Zeilen und f Spalten $(e + f < n)$, die zusammen alle Einträge $= 0$ enthalten;
2. sei w das Minimum der nicht überdeckten Elemente;
3. subtrahiere w von den $n - e$ nicht überdeckten Zeilen;
4. addiere w zu den f überdeckten Spalten.

Die Gewichte ändern sich also wie folgt:

1. um $-w$, falls (i, j) nicht überdeckt ist;
2. um 0, falls (i, j) von einer Zeile *oder* Spalte überdeckt ist, aber nicht beides;
3. um $+w$, falls (i, j) von einer Zeile *und* einer Spalte überdeckt ist.

1. finde eine minimale Anzahl von e Zeilen und f Spalten $(e + f < n)$, die zusammen alle Einträge $= 0$ enthalten;
2. sei w das Minimum der nicht überdeckten Elemente;
3. subtrahiere w von den $n - e$ nicht überdeckten Zeilen;
4. addiere w zu den f überdeckten Spalten.

Die Gewichte ändern sich also wie folgt:

1. um $-w$, falls (i, j) nicht überdeckt ist;
2. um 0, falls (i, j) von einer Zeile *oder* Spalte überdeckt ist, aber nicht beides;
3. um $+w$, falls (i, j) von einer Zeile *und* einer Spalte überdeckt ist.

1. finde eine minimale Anzahl von e Zeilen und f Spalten $(e + f < n)$, die zusammen alle Einträge $= 0$ enthalten;
2. sei w das Minimum der nicht überdeckten Elemente;
3. subtrahiere w von den $n - e$ nicht überdeckten Zeilen;
4. addiere w zu den f überdeckten Spalten.

Die Gewichte ändern sich also wie folgt:

1. um $-w$, falls (i, j) nicht überdeckt ist;
2. um 0, falls (i, j) von einer Zeile *oder* Spalte überdeckt ist, aber nicht beides;
3. um $+w$, falls (i, j) von einer Zeile *und* einer Spalte überdeckt ist.

1. finde eine minimale Anzahl von e Zeilen und f Spalten $(e + f < n)$, die zusammen alle Einträge $= 0$ enthalten;
2. sei w das Minimum der nicht überdeckten Elemente;
3. subtrahiere w von den $n - e$ nicht überdeckten Zeilen;
4. addiere w zu den f überdeckten Spalten.

Die Gewichte ändern sich also wie folgt:

1. um $-w$, falls (i, j) nicht überdeckt ist;
2. um 0, falls (i, j) von einer Zeile *oder* Spalte überdeckt ist, aber nicht beides;
3. um $+w$, falls (i, j) von einer Zeile *und* einer Spalte überdeckt ist.

Insbesondere sind die resultierenden Gewichte wieder ≥ 0. Die Anzahl der doppelt überdeckten Positionen ist $e \cdot f$, die Anzahl der nicht überdeckten Positionen ist

$$n^2 - n(e+f) + ef \ .$$

Der resultierende Gewichtsunterschied ist daher

$$\begin{aligned} \Delta w &= (ef)w - (n^2 - n(e+f) + ef)w \\ &= (n(e+f) - n^2)w < 0 \end{aligned}$$

Damit muss unsere Iteration enden und wir finden eine 0-Diagonale der Länge n, entsprechend einer optimalen Zuordnung.

Insbesondere sind die resultierenden Gewichte wieder ≥ 0. Die Anzahl der doppelt überdeckten Positionen ist $e \cdot f$, die Anzahl der nicht überdeckten Positionen ist

$$n^2 - n(e+f) + ef \ .$$

Der resultierende Gewichtsunterschied ist daher

$$\begin{aligned} \Delta w &= (ef)w - (n^2 - n(e+f) + ef)w \\ &= (n(e+f) - n^2)w < 0 \end{aligned}$$

Damit muss unsere Iteration enden und wir finden eine 0-Diagonale der Länge n, entsprechend einer optimalen Zuordnung.

Insbesondere sind die resultierenden Gewichte wieder ≥ 0. Die Anzahl der doppelt überdeckten Positionen ist $e \cdot f$, die Anzahl der nicht überdeckten Positionen ist

$$n^2 - n(e+f) + ef \ .$$

Der resultierende Gewichtsunterschied ist daher

$$\begin{aligned} \Delta w &= (ef)w - (n^2 - n(e+f) + ef)w \\ &= (n(e+f) - n^2)w < 0 \end{aligned}$$

Damit muss unsere Iteration enden und wir finden eine 0-Diagonale der Länge n, entsprechend einer optimalen Zuordnung.

Insbesondere sind die resultierenden Gewichte wieder ≥ 0. Die Anzahl der doppelt überdeckten Positionen ist $e \cdot f$, die Anzahl der nicht überdeckten Positionen ist

$$n^2 - n(e+f) + ef \ .$$

Der resultierende Gewichtsunterschied ist daher

$$\begin{aligned} \Delta w &= (ef)w - (n^2 - n(e+f) + ef)w \\ &= (n(e+f) - n^2)w < 0 \end{aligned}$$

Damit muss unsere Iteration enden und wir finden eine 0-Diagonale der Länge n, entsprechend einer optimalen Zuordnung.

In unserem Beispiel ergibt sich

$$W'' = \begin{pmatrix} \boxed{0} & \boxed{2} & \boxed{0} & \boxed{2} \\ 3 & \boxed{0} & 2 & 2 \\ 1 & \boxed{0} & 4 & 5 \\ \boxed{2} & \boxed{3} & \boxed{0} & \boxed{0} \end{pmatrix}$$

Der Algorithmus bestimmt $w = 1$:

$$\Rightarrow \begin{pmatrix} 0 & 3 & \boxed{0} & 2 \\ 2 & \boxed{0} & 1 & 1 \\ \boxed{0} & 0 & 3 & 4 \\ 2 & 4 & 0 & \boxed{0} \end{pmatrix}$$

In unserem Beispiel ergibt sich

$$W'' = \begin{pmatrix} \boxed{0} & \boxed{2} & \boxed{0} & \boxed{2} \\ 3 & \boxed{0} & 2 & 2 \\ 1 & \boxed{0} & 4 & 5 \\ \boxed{2} & \boxed{3} & \boxed{0} & \boxed{0} \end{pmatrix}$$

Der Algorithmus bestimmt $w = 1$:

$$\Rightarrow \begin{pmatrix} 0 & 3 & \boxed{0} & 2 \\ 2 & \boxed{0} & 1 & 1 \\ \boxed{0} & 0 & 3 & 4 \\ 2 & 4 & 0 & \boxed{0} \end{pmatrix}$$