

# Algorithmen für Maximum Matchings

Christine Heller      Berthold Vöcking

5. Januar 2006

## 1 Einleitung

Matchingprobleme sind Zuordnungsprobleme. Es geht darum z.B. Studierenden Plätze in Seminaren zuzuordnen, Bewerber auf freie Stellen zu verteilen oder auch Paare von Männern und Frauen zu ermitteln, die einander sympathisch sind. Dies sind alles Beispiele für bipartite Matchingprobleme. Das allgemeine, nicht-bipartite Matchingproblem erschließt auch Anwendungen mit gleichartigen Zuordnungen bzw. gleichgeschlechtlichen Paarungen.

**Definition 1** Sei  $G = (V, E)$  ein ungerichteter Graph. Eine Kantenmenge  $M \subseteq E$  ist ein Matching, wenn kein Knoten aus  $V$  zu mehr als einer Kante aus  $M$  inzident ist. Ein Matching  $M$  ist (inklusions-)maximal, wenn zu  $M$  keine Kante mehr hinzugefügt werden kann, ohne die Matchingeigenschaft zu zerstören.  $M$  ist ein maximum Matching, wenn es kein Matching  $N$  mit  $|N| > |M|$  gibt.

Beim *Matchingproblem* geht es darum, zu einem gegebenen Graphen ein maximum Matching zu berechnen. Beim *bipartiten Matchingproblem* ist der zugrundeliegende Graph bipartit. Beachte, ein maximales Matching kann man durch einen einfachen Greedy-Algorithmus berechnen, der startend mit dem leeren Matching, solange Kanten hinzufügt, bis keine Kante mehr hinzugefügt werden kann, ohne die Matchingeigenschaft zu zerstören. Die Berechnung eines maximum Matching ist hingegen ein anspruchsvolles kombinatorisches Optimierungsproblem.

Die wichtigste Botschaft dieses Kapitels ist, dass das Matchingproblem in seiner allgemeinen Form effizient – also in Polynomialzeit – gelöst werden kann.

Im Folgenden bezeichne  $n$  die Anzahl der Knoten und  $m$  die Anzahl der Kanten im Eingabegraphen  $G$ . Wir gehen davon aus, dass  $G$  zusammenhängend ist, und somit gilt  $n - 1 \leq m \leq \binom{n}{2}$ .

**Satz 2** [3] *Es gibt einen Algorithmus mit Laufzeit  $O(m\sqrt{n})$  für das Matchingproblem auf allgemeinen Graphen.*

Das Matchingproblem gibt es auch in verschiedenen gewichteten Varianten.

- *Gewichtetes maximum Matching:* Jede Kante hat ein Gewicht aus  $\mathbb{N}_0$ . Gesucht ist ein Matching, das die Summe der Kantengewichte maximiert.
- *Min-cost Matching:* Jeder Kante ist ein Kostenwert aus  $\mathbb{N}_0$  zugeordnet. Berechnet werden soll ein maximum Matching, welches die Summe der Kantenkosten im Matching minimiert. Gesucht ist also das kostengünstigste Matching unter allen maximum Matchings.

Tatsächlich gibt es Polynomialzeitreduktionen zwischen diesen beiden Problemen. Dabei handelt es sich um nicht ganz triviale Eingabetransformationen, die in  $O(n^2)$  Schritten auf einer uniformen RAM durchgeführt werden können (vgl. Übung). Deshalb spricht man häufig allgemein nur vom *gewichteten Matchingproblem*, ohne die Problemvariante genau zu spezifizieren. Auch das gewichtete Matchingproblem kann effizient gelöst werden.

**Satz 3** [2] *Es gibt einen  $O(n^3)$ -Algorithmus für das gewichtete Matchingproblem auf allgemeinen Graphen.*

Wir werden diese beiden Sätze nicht in ihrer Allgemeinheit beweisen. Statt dessen werden wir uns auf Matchings in bipartiten Graphen beschränken, weil diese wegen ihrer Nähe zu maximalen Flüssen wesentlich einfacher zu handhaben sind. Die meisten Anwendungen für Matchingalgorithmen beziehen sich tatsächlich auf bipartite Graphen. Außerdem werden wir uns auf ungewichtete Matchings beschränken. Trotz dieser Vereinfachungen werden wir das wichtigste Konzept, das allen uns bekannten Matchingalgorithmen zugrunde liegt, kennenlernen, nämlich sogenannte „verbessernde Pfade“. Mehr Details zu den anderen oben erwähnten Problemvarianten finden sich z.B. in [2, 4].

## 2 Bipartite Matchings und maximale Flüsse

Die Eingabe des bipartiten Matchingproblems ist ein bipartiter Graph  $G = (U \cup W, E)$ . In der Literatur spricht man tatsächlich häufig von  $U$  als Menge der *Jungen* und  $W$  als Menge der *Mädchen*. Kanten stellen beidseitige Sympathien zwischen Jungen und Mädchen dar. Nur durch eine Sympathiekante verbundene Jungen und Mädchen können zu Paaren verbunden werden. Gesucht ist eine möglichst große Menge von Paaren; in anderen Worten, ein maximum Matching auf  $G$ .

**Satz 4** *Das bipartite Matchingproblem kann in Polynomialzeit auf das ganzzahlige Max-Fluss-Problem reduziert werden.*

### Beweis

- Der Graph  $G = (U \cup W, E)$  wird in ein Flussnetzwerk  $G' = (V, E')$  mit Kapazität Funktion  $c : E' \rightarrow \mathbb{N}$  transformiert, wobei

$$\begin{aligned} V &= U \cup W \cup \{q, s\} \\ E' &= \{(u, w) \mid u \in U, w \in W, \{u, w\} \in E\} \cup \{q\} \times U \cup W \times \{s\} \\ c(e) &= 1 \quad \forall e \in E' \end{aligned}$$

- Wir beobachten, dass es unter dieser Transformation eine bijektive Abbildung zwischen ganzzahligen Flüssen und Matchings gibt.
- Insbesondere stimmt dabei der Wert des Flusses mit der Kardinalität des jeweiligen Matchings überein.
- Also liefert ein maximaler ganzzahliger Fluss ein maximum Matching.

□

Die beschriebene Transformation kann offensichtlich in  $O(n + m)$  Schritten berechnet werden. Wir können also das bipartite Matchingproblem mit Hilfe eines Flussalgorithmus lösen. Beispielsweise erhalten wir dadurch bei Verwendung von „Forward-Backward-Propagation“ einen  $O(n^3)$ -Algorithmus für bipartite Matchings. Im Folgenden werden wir Algorithmen kennenlernen, die noch wesentlich effizienter sind. Diese Algorithmen basieren auf der Berechnung von „verbessernden Pfaden“. Letztendlich werden wir eine Laufzeit von  $O(m\sqrt{n})$  erreichen. Insbesondere für *dünn-besetzte* Graphen, also Graphen mit nur  $O(n)$  vielen Kanten, ist dies eine signifikante Verbesserung von  $O(n^3)$  auf  $O(n^{1.5})$ .

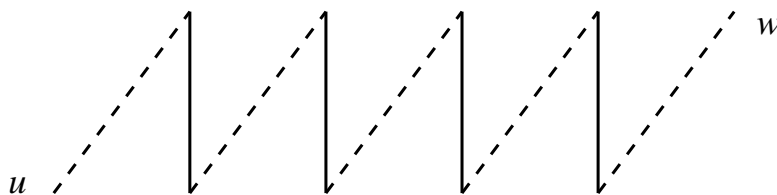
### 3 Verbessernde Pfade

Sei  $M$  ein Matching auf dem Graphen  $G = (V, E)$ . Ein Knoten aus  $V$  heißt *frei*, wenn er nicht inzident zu einer Kante aus  $M$  ist. Ein Pfad in  $G$  heißt  *$M$ -alternierend*, wenn seine Kanten abwechselnd aus  $M$  und  $E \setminus M$  sind. Ein Pfad heißt  *$M$ -verbessernd*, wenn er  $M$ -alternierend ist, seine Knoten alle verschieden und seine Endknoten frei sind.

Im Folgenden wird ein Pfad in einem Graphen auch als die Menge seiner Kanten interpretiert. Mit  $M \oplus N$  bezeichnen wir die symmetrische Differenz der Mengen  $M$  und  $N$ , d.h.  $M \oplus N = (M \cup N) \setminus (M \cap N)$ .

**Beobachtung 5** Wenn  $M$  ein Matching und  $P$  ein  $M$ -verbessernder Pfad ist, dann ist  $M \oplus P$  ein Matching und es gilt  $|M \oplus P| = |M| + 1$ .

Wir betrachten als Beispiel den unten abgebildeten Graphen, der nur aus einem Pfad  $P$  besteht. Wenn  $M$  das Matching bezeichnet, das aus den durchgezogenen Kanten besteht, dann ist  $P$  ein  $M$ -verbessernder Pfad und  $u$  und  $w$  sind die freien Knoten.



Die gestrichelten Kanten bilden das Matching  $M \oplus P$ . Offensichtlich gilt  $|M \oplus P| = |M| + 1$ . Mit Hilfe eines  $M$ -verbessernden Pfades lässt sich also leicht ein neues Matching berechnen, das eine Kante mehr als  $M$  enthält.

**Lemma 6** Seien  $M$  und  $N$  Matchings auf dem Graphen  $G = (V, E)$  und  $|N| > |M|$ . Dann enthält der Teilgraph  $G' = (V, M \oplus N)$  mindestens  $|N| - |M|$  knotendisjunkte  $M$ -verbessernde Pfade.

**Beweis** Seien  $C_1 \dots C_g$  mit  $C_i = (V_i, E_i)$  für  $1 \leq i \leq g$ , die Zusammenhangskomponenten von  $G'$ . Da  $M$  und  $N$  Matchings sind, ist jeder Knoten aus  $V$  höchstens zu einer Kante aus  $M$  und höchstens zu einer Kante aus  $N$  inzident. Daher ist  $C_i$  für  $1 \leq i \leq g$  entweder

- ein isolierten Knoten,
- ein Kreis gerader Länge, dessen Kanten abwechselnd in  $M$  und in  $E \setminus M$  sind oder
- ein  $M$ -alternierenden Pfad, dessen Knoten alle verschieden sind.

Sei  $\delta(C_i) = |E_i \cap N| - |E_i \cap M|$ . Dann gilt:

- $\delta(C_i) \in \{-1, 0, 1\}$ .
- $\delta(C_i) = 1$  genau dann, wenn  $C_i$  ein  $M$ -verbessernder Pfad ist.
- $\sum_{i=1}^g \delta(C_i) = |N \setminus M| - |M \setminus N| = |N| - |M|$ .

Aus a) und c) folgt, dass es mindestens  $|N| - |M|$  Komponenten  $C_i$  mit  $\delta(C_i) = 1$  gibt. Aus b) folgt dann, dass es mindestens  $|N| - |M|$  viele  $M$ -verbessernde Pfade gibt. Nach Definition sind die Komponenten  $C_i$  und damit auch die  $M$ -verbessernden Pfade knotendisjunkt.  $\square$

Insbesondere gilt, wenn  $M$  kein maximum Matching ist, so gibt es immer einen  $M$ -verbessernden Pfad. Daraus folgt

**Korollar 7** *Sei  $M$  ein Matching auf einem Graphen  $G = (V, E)$ . Es gibt keinen  $M$ -verbessernden Pfad, genau dann wenn  $M$  ein maximum Matching ist.*

## 4 Ein einfacher Algorithmus

Aus obigem Korollar leiten wir den folgenden Algorithmus ab.

### Algorithmus Maximum-Matching-1

- Schritt 0:  $M \leftarrow \emptyset$ .  
 Schritt 1: Berechne einen  $M$ -verbessernden Pfad  $P$ .  
 Falls kein solcher Pfad existiert, stopp.  
 Schritt 2:  $M \leftarrow M \oplus P$ . Gehe zu Schritt 1.

Da ein Matching nicht mehr als  $\lfloor n/2 \rfloor$  Kanten enthalten kann, haben wir spätestens nach  $\lfloor n/2 \rfloor$  Ausführungen von Schritt 1 ein maximum Matching gefunden. Wenn wir also in der Lage sind, Schritt 1 des Algorithmus effizient

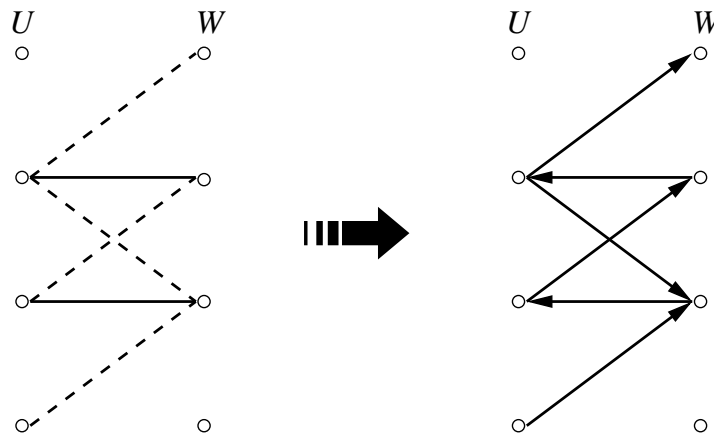
auszuführen, können wir auch effizient ein maximum Matching berechnen. Wir überlegen uns, wie wir einen  $M$ -verbessernden Pfad in einem bipartiten Graphen berechnen können.

Sei  $G = (U \cup W, E)$  ein bipartiter Graph und  $M$  ein Matching auf  $G$ .  $M$ -verbessernde Pfade haben ungerade Länge. Daher führen sie von einem Knoten aus  $U$  zu einem Knoten aus  $W$ . O.B.d.A. suchen wir  $M$ -verbessernde Pfade ausgehend von freien Knoten in der Menge  $U$ .

Zunächst konstruieren wir in Zeit  $O(m)$  einen gerichteten Graphen  $G_M$ . Dazu geben wir den Kanten in  $G$  eine Richtung. Sei  $u \in U$ ,  $w \in W$  und  $\{u, w\}$  eine Kante in  $G$ .

- Wenn  $\{u, w\} \notin M$ , so enthält  $G_M$  die gerichtete Kante  $(u, w)$ .
- Wenn  $\{u, w\} \in M$ , so enthält  $G_M$  die gerichtete Kante  $(w, u)$ .

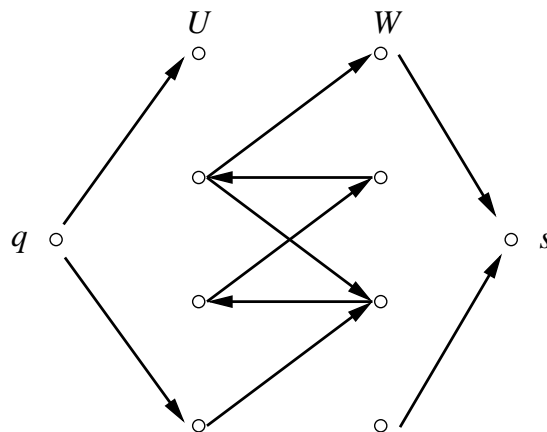
Graphisch lässt sich diese Transformation wie folgt darstellen. Die durchgezogenen Linien stellen wieder die Matchingkanten dar.



Wir beobachten, jeder gerichtete Weg in  $G_M$  ist ein  $M$ -alternierender Pfad und jeder  $M$ -verbessernde Pfad ist ein gerichteter Weg von einem freien  $U$ -Knoten zu einem freien  $W$ -Knoten in  $G_M$ . Jetzt können wir in einer Breiten- oder Tiefensuche, ausgehend von den freien  $U$ -Knoten, nach einen Weg zu einem freien  $W$ -Knoten suchen. Auf diese Art und Weise finden wir, falls  $M$  kein maximum Matching ist, einen  $M$ -verbessernden Weg. Die Laufzeit für diese Suche ist  $O(n + m) = O(m)$ .

**Satz 8** *Algorithmus Maximum-Matching-1 löst das bipartite Matchingproblem in Zeit  $O(nm)$ .*  $\square$

In welcher Beziehung steht dieses Ergebnis zu der in Kapitel 2 beschriebenen Reduktion von Matchings auf Flussprobleme? – Wenn wir dem bipartiten Graphen  $G_M$  weitere Knoten  $q$  und  $s$  hinzufügen sowie für jeden freien Knoten  $u \in U$  die Kante  $(q, u)$  und für jeden freien Knoten  $w \in W$  die Kante  $(w, s)$ , so entspricht der so entstehende Graph  $R_M$  genau dem Restnetzwerk gemäß der Ford-Fulkerson-Methode. Alle Kanten haben dabei Restkapazität 1.



Die  $M$ -verbessernden Wege ergänzt um die Start- und Zielknoten  $q$  und  $s$  entsprechen sind flussvergrößernde Wege im Restnetzwerk  $R_M$  und umgekehrt. Dies bedeutet aber, Algorithmus Maximum-Matching-1 entspricht der Ford-Fulkerson-Methode. Für allgemeine Flussprobleme hatte diese Methode eine pseudopolynomielle Laufzeitschranke bzw. Laufzeit  $O(nm^2)$  bei Verwendung von Breitensuche. Unsere obige Analyse hat gezeigt, dass die Ford-Fulkerson-Methode nur Zeit  $O(nm)$  benötigt, wenn sie auf das bipartite Matchingproblem angewendet wird.

## 5 Kürzeste verbessernde Pfade

Ein *kürzester  $M$ -verbessernder Pfad* ist ein  $M$ -verbessernder Pfad minimaler Länge. Wir möchten nun zeigen, dass wir die Laufzeit noch weiter verbessern

können, in dem wir in einer Iteration mehrere kürzeste verbessernde Pfade berechnen. Dazu benötigen wir zunächst einige Eigenschaften kürzester verbessernder Pfade.

**Lemma 9** *Sei  $M$  ein Matching,  $P$  ein kürzester  $M$ -verbessernder Pfad und  $P'$  ein  $M \oplus P$ -verbessernder Pfad. Dann gilt*

$$|P'| \geq |P| + |P \cap P'|.$$

**Beweis**

- Sei  $N = (M \oplus P) \oplus P'$ . Aus Beobachtung 5 folgt,  $N$  ist ein Matching und es gilt  $|N| = |M| + 2$ .
- Aus Lemma 6 folgt, dass  $M \oplus N$  zwei knotendisjunkte  $M$ -verbessernde Pfade enthält. Jeder dieser Wege hat mindestens die Länge des kürzesten  $M$ -verbessernden Pfades  $P$ . Also gilt  $M \oplus N \geq 2|P|$ .
- Wir nutzen die Assoziativität der symmetrischen Differenz und erhalten  $M \oplus N = M \oplus (M \oplus P \oplus P') = (M \oplus M) \oplus (P \oplus P') = P \oplus P'$ . Somit folgt  $P \oplus P' \geq 2|P|$ .
- Für die symmetrische Differenz gilt nun  $|P \oplus P'| = |P \cup P'| - |P \cap P'| \leq |P| + |P'| - |P \cap P'|$ . Somit erhält man  $|P| + |P'| - |P \cap P'| \geq 2|P|$ . Einfaches Umformen liefert jetzt die Behauptung.

□

Wir nehmen nun an, dass wir nacheinander jeweils kürzeste verbessernde Wege berechnen und anwenden. Sei  $M_0 = \emptyset$  ein leeres Matching,  $P_i$  ein kürzester  $M_i$ -verbessernder Pfad und  $M_{i+1} = M_i \oplus P_i$ .

**Lemma 10** *Fr  $i, j \geq 0$  gilt*

- 1)  $|P_i| \leq |P_{i+1}|$ , und
- 2)  $|P_i| = |P_j| \Rightarrow P_i$  und  $P_j$  sind knotendisjunkt.

**Beweis** Aussage 1) folgt direkt aus Lemma 9.

Aussage 2) zeigen wir per Widerspruch. Angenommen  $|P_i| = |P_j|$ ,  $i < j$  und  $P_i, P_j$  sind nicht knotendisjunkt. O.B.d.A. können wir zudem annehmen, dass

für alle  $k \in \{i+1, \dots, j-1\}$  gilt,  $P_k$  ist knotendisjunkt sowohl zu  $P_i$  als auch zu  $P_j$ , denn ansonsten könnten wir statt des Paares  $P_i, P_j$  das Paar  $P_k, P_j$  bzw.  $P_i, P_k$  betrachten und dieses Argument rekursiv fortsetzen bis wir die gewünschte Eigenschaft haben. Jetzt können wir wie folgt einen Widerspruch herleiten.

- Da alle  $P_k$ ,  $i < k < j$  knotendisjunkt zu  $P_i$  und  $P_j$  sind, ist  $P_j$  ein  $M_i \oplus P_i$ -verbessernder Pfad.
- Somit folgt, wenn  $v$  ein gemeinsamer Knoten von  $P_i$  und  $P_j$  ist, enthalten sowohl  $P_i$  als auch  $P_j$  die zu  $v$  inzidente Kante aus dem Matching  $M_i \oplus P_i$ . Damit gilt  $|P_i \cap P_j| \geq 1$ .
- Aus Lemma 9 folgt nun  $|P_j| \geq |P_i| + |P_i \cap P_j| \geq |P_i| + 1$ . Ein offensichtlicher Widerspruch zu unserer Annahme  $|P_i| = |P_j|$ .

□

Aus der ersten Aussage des Lemmas folgt, dass der unmittelbar vor dem Lemma skizzierte Algorithmus zunächst einige kürzeste verbessernde Pfade der Länge  $\ell_1$ , für eine ungerade Zahl  $\ell_1 \geq 1$  berechnet, dann kürzeste verbessernde Pfade der Länge  $\ell_2$ , für eine andere ungerade Zahl  $\ell_2 > \ell_1$ , dann kürzeste verbessernde Pfade der Länge  $\ell_3$ , für ein ungerades  $\ell_3 > \ell_2$  usw. bis keine verbessernden Pfade mehr vorhanden sind. Aus der zweiten Aussage des Lemmas folgt, dass alle berechneten Pfade derselben Länge untereinander knotendisjunkt sind.

## 6 Der Algorithmus von Hopcroft und Karp

Die Idee von Hopcroft und Karp [1] ist es nun, die in Lemma 10 beschriebenen Eigenschaften auszunutzen und in einer Iteration mehrere kürzeste verbessernde Pfade zu berechnen und anzuwenden, so dass in der nächsten Iteration keine Pfade dieser Länge mehr gefunden werden. Dazu genügt es eine inklusions-maximale Menge knotendisjunkter kürzester verbessernder Pfade zu berechnen. Da diese Pfade knotendisjunkt sind können sie in beliebiger Reihenfolge auf das Ausgangsmatching angewandt werden. Aus Lemma 10 folgt, dass nach dem Anwenden dieser Pfade die Länge der kürzesten

verbessernden Pfade anwächst, und zwar mindestens um den Wert 2, da verbessernde Pfade immer ungerade Länge haben.

### Algorithmus Maximum-Matching-2

Schritt 0:  $M \leftarrow \emptyset$ .

Schritt 1: Sei  $\ell$  die Länge eines kürzesten  $M$ -verbessernden Pfades. Berechne eine inklusions-maximale Menge knotendisjunkter  $M$ -verbessernder Pfade  $P_1, \dots, P_i$  mit Länge  $\ell$ . Falls kein solcher Pfad existiert, stopp.

Schritt 2:  $M \leftarrow M \oplus P_1 \oplus \dots \oplus P_i$ . Gehe zu Schritt 1.

Wieviele Iterationen benötigt der Algorithmus? – Sei  $s \leq n$  die Kardinalität eines maximum Matchings und  $M$  ein beliebiges Matching. Wir zeigen, dass  $O(\sqrt{s})$  Iterationen ausreichen. Das Schlüsselargument ist, dass es immer relativ kurze  $M$ -verbessernde Pfade gibt.

**Lemma 11** *Es gibt einem  $M$ -verbessernden Pfad mit Länge höchstens*

$$2 \cdot \left\lfloor \frac{|M|}{s - |M|} \right\rfloor + 1 .$$

**Beweis**

- Aus Lemma 6 folgt, dass es mindestens  $s - |M|$  viele  $M$ -verbessernde Pfade gibt.
- Diese  $s - |M|$  verbessernden Pfade enthalten zusammen höchstens  $|M|$  viele Kanten aus  $M$ .
- Nach dem Durchschnittsprinzip muss es also einen verbessernden Pfad geben, der höchstens  $\lfloor |M|/(s - |M|) \rfloor$  Kanten aus  $M$  enthält.
- Dieser Pfad hat höchstens die Länge  $2 \lfloor |M|/(s - |M|) \rfloor + 1$ .

□

Wir zerlegen jetzt die Rechnung in zwei Phasen. Wir werden zeigen, dass in jeder dieser Phasen jeweils höchstens  $\sqrt{s} \leq \sqrt{n}$  Iterationen durchgeführt werden.

In Phase 1 gelte  $|M| \leq \lfloor s - \sqrt{s} \rfloor$ . Aus Lemma 11 folgt dann, es gibt einen  $M$ -verbessernden Weg mit Länge höchstens

$$2 \cdot \frac{\lfloor s - \sqrt{s} \rfloor}{s - \lfloor s - \sqrt{s} \rfloor} + 1 = 2 \cdot \frac{s - \lceil \sqrt{s} \rceil}{\lceil \sqrt{s} \rceil} + 1 \leq 2\sqrt{s} - 1 .$$

In dieser Phase nimmt die Variable  $\ell$  in Schritt 1 des Algorithmus also höchstens den Wert  $2\sqrt{s} - 1$  an. Da  $\ell$  mindestens in Zweierschritten von Iteration zu Iteration anwächst, bedeutet dies, es gibt höchstens  $\sqrt{s}$  Iterationen in Phase 1.

In Phase 2 enthält das Matching  $M$  bereits mehr als  $\lfloor s - \sqrt{s} \rfloor$  Kanten. Es fehlen also höchstens noch  $\sqrt{s}$  Kanten. Spätestens nach  $\sqrt{s}$  weiteren Iterationen ist somit ein maximum Matching berechnet.

Somit gibt es insgesamt höchstens  $2\sqrt{s} = O(\sqrt{n})$  Iterationen.

**Lemma 12** *Die Berechnung einer inklusions-maximalen Menge kürzester  $M$ -verbessernder Pfade kann in Zeit  $O(m)$  durch geführt werden.*

**Beweis** Der folgende Algorithmus berechnet eine derartige Menge von Pfaden in Zeit  $O(n + m) = O(m)$ .

1. Wie auf Seite 6 beschrieben, berechnen wir zunächst den gerichteten Graphen  $G_M$ . Durch Hinzufügen einer Quelle  $q$  und einer Senke  $s$ , wie auf Seite 7 beschrieben, erhalten wir das Restnetzwerk  $R_M$ .
2. Als nächstes streichen wir in einer von  $q$  ausgehenden Breitensuche alle Kanten, die nicht auf kürzesten Pfaden zwischen  $q$  und  $s$  liegen, und erhalten dadurch ein *Niveaunetzwerk*, das wir  $R'_M$  nennen. Dieses Netzwerk enthält alle kürzesten  $M$ -verbessernden Wege. Beachte, die freien  $U$ -Knoten sind die Nachfolger der Quelle und die freien  $W$ -Knoten die Vorgänger der Senke.
3. Nacheinander, von jedem freien  $U$ -Knoten ausgehend, versuchen wir jetzt in einer Tiefensuche in  $R'_M$  einen Pfad zu den freien  $W$ -Knoten zu finden. Einmal überschrittene Knoten und Kanten werden dabei aus dem Niveaunetzwerk entfernt. Dadurch garantieren wir zum einen, dass die gefundenen Pfade knotendisjunkt sind, und zum anderen, dass die Laufzeit durch  $O(m + n)$  beschränkt ist.

□

Insgesamt ergibt sich folgende Laufzeitschranke für den Algorithmus von Hopcroft und Karp.

**Satz 13** *Algorithmus Maximum-Matching-2 hat Laufzeit  $O(m\sqrt{n})$ .* □

Auch der Algorithmus von Hopcroft und Karp steht in direkter Beziehung zu einem der uns bekannten Flussalgorithmen, nämlich dem Algorithmus von Dinic. Die Berechnung der knotendisjunkten Wege im Niveaunetzwerk entspricht tatsächlich der Berechnung eines Sperrflusses (vgl. Übung). Für allgemeine Flussprobleme benötigt Dinics Algorithmus  $O(nm)$  Schritte zur Berechnung eines Sperrflusses, und diese Sperrflussberechnung wird  $O(n)$  mal iteriert. Im Falle von maximum Matchings dauert die Berechnung des Sperrflusses, also der maximalen Menge knotendisjunkter  $M$ -verbessernder Wege, nur  $O(m)$  Schritte und der Algorithmus benötigt, wie gesehen, nur  $O(\sqrt{n})$  Iterationen.

## Literatur

- [1] J. E. Hopcroft and R. M. Karp. An  $n^{5/2}$  Algorithm for Maximum Matching in Bipartite Graphs. *SIAM Journal on Computing*, Vol. 2, pp. 225-231, 1973.
- [2] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. New York: Holt, Rinehart & Winston, 1976.
- [3] S. Micali and V. V. Vazirani. An  $O(\sqrt{|V|} \cdot |E|)$  Algorithm for Finding Maximum Matching in General Graphs. *Proc. Twenty-first Annual Symposium on the Foundations of Computer Science (FOCS)*, pp. 17-21, 1980.
- [4] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization*. Prentice-Hall, Englewood Cliffs, NJ, 1982.