Matching Basics

Professor Dr. Petra Mutzel

Lehrstuhl für Algorithm Engineering

Fakultät für Informatik

TU Dortmund

Selected Topics on Combinatorial Optimization

Vienna Graduate School on Computational Optimization

Lectures 3/4 (Part 2), November 7/8, 2018

Maximum Matchings in Graphs

Let $G = (V, E)$ be an undirected simple graph.

Definition (Matching)

A matching is a set $M \subseteq E$ such that every vertex in V is incident to at most one edge in M.

Definition (Maximal matching)

A maximal matching M is a matching that cannot be extended by adding any edge in $E \setminus M$.

Definition (Maximum matching)

A maximum matching M is a matching in G with the largest possible number of edges.

Bipartite Graphs

Definition (Bipartite graph)

Let $G = (V, E)$ be an undirected graph. G is bipartite, if the vertex set can be partitioned into two sets V_1 und V_2 so that all edge have exactly one endnode in V_1 and one in V_2.

Observation

G is bipartite $\Leftrightarrow$ G does not contain cycles of odd length.

Applications of Matching

Matching problems belong to the class of assignment problems.

Many variants: perfect matching, maximum matching, maximum weighted matching, ...

- Travelling Salesman Problem: Christofides-Heuristic
- Chinese Postman Problem
- Maximum cut in planar graphs
- Steganography (hide information in pictures)
- Assignment of students to exercise classes

Two seminal papers — the first 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.

Two seminal papers — the second 1965

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

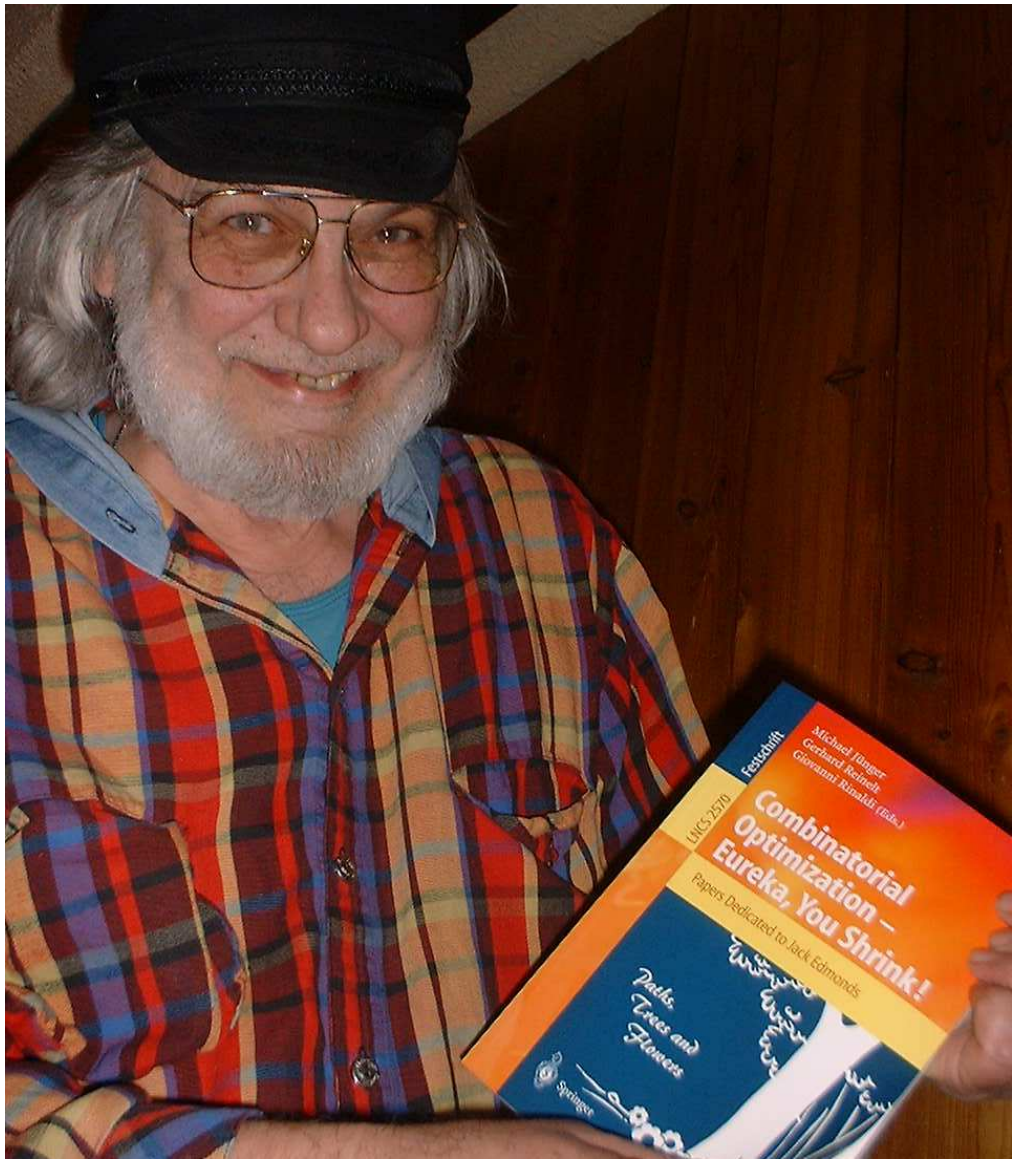
(December 1, 1964)

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



Köln 2004: Jack & Kathie mit Pauline & Paul

Matchings in bipartite graphs

How can be compute maximum matchings in bipartite graphs?

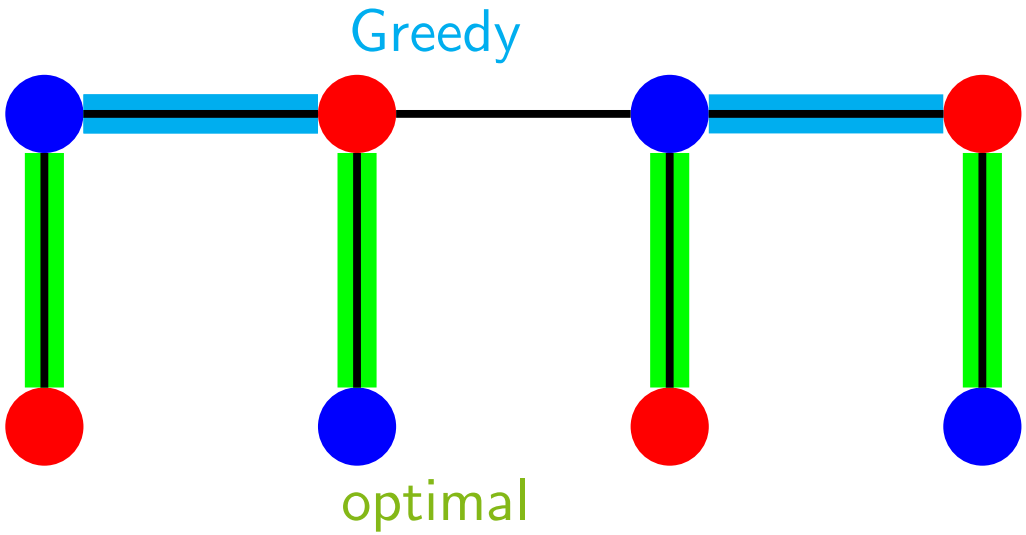
Idee Greedy

Start with $\emptyset$ and iteratively add the next edge, until no edge can be added anymore

Algorithm (Greedy Matching)

1. $M = \emptyset$
2. If $\exists e \in E : M \cup e$ is matching
   Then $M = M \cup e$; goto 2.
3. return M

Matchings in bipartite graphs



Theorem

Let $G = (V, E)$ be an undirected graph, M_{opt} a maximum matching in G. The Greedy algorithm computes a matching M_{greedy} with $|M_{\text{greedy}}| \geq |M_{\text{opt}}|/2$.

Even for bipartite graphs it is possible that $|M_{\text{greedy}}| = |M_{\text{opt}}|/2$. ✓

Proof of the Approximation Guarantee

Proof.

V_{greedy}: Set of vertices incident to M_{greedy}

we have: $|V_{\text{greedy}}| = 2\,|M_{\text{greedy}}|$

Every edge $e \in M_{\text{opt}}$ is incident to a vertex in V_{greedy}, otherwise we could have added e.

Since the matching edges are disjoint, we have:

also $|M_{\text{opt}}| \leq |V_{\text{greedy}}| = 2\,|M_{\text{greedy}}|$

also $|M_{\text{greedy}}| \geq |M_{\text{opt}}|\,/2$ □

From now on we assume that G is connected
(also $n - 1 \leq |E| \leq \binom{n}{2}$)
otherwise we work on the connected components

Central Definitions

Definition (Matching Basics)

$G = (V, E)$ undirected graph, $M \subseteq E$ matching in G.

- $e \in M$ is called matching edge (M-edge).
- $e \notin M$ is called free edge.
- $v \in V$ incident to matching edge $e \in M$ is called matched.
- $v \in V$ not matched is called free.

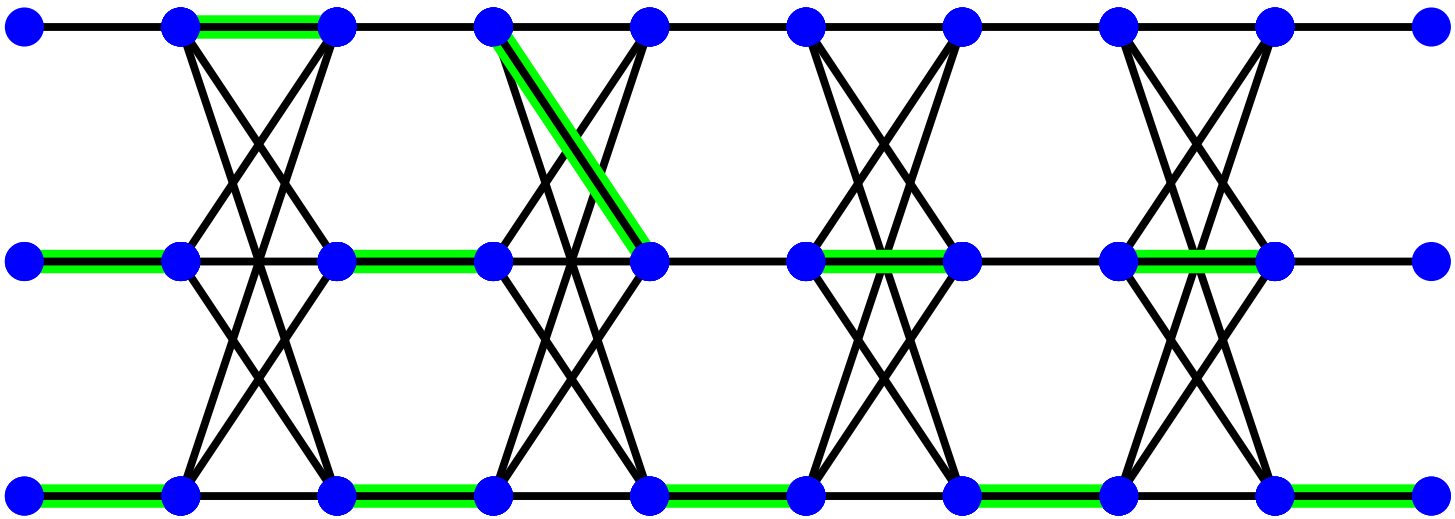
Central Definitions

Definitions (Alternating and M-augmenting paths)

$G = (V, E)$ undirected graph, $M \subseteq E$ matching in G.

- A path P of length k is a series $(v_0, e_1, v_1, e_2, \ldots, e_k, v_k)$ of alternating nodes and edges in G with $e_i = (v_{i-1}, v_i)$ for $i = 1, \ldots, k$.
- We also write: $P = (v_0, v_1, \ldots, v_k)$.
- A simple path is a path in which all vertices are disjoint.
- A non-empty path $P = (v_1, v_2, \ldots, v_k)$ with $k \geq 2$ in which M-edges and free edges are alternating is called M-alternating.
- A cycle-free M-alternating path $P = (v_1, \ldots, v_k)$, $k \geq 2$, with v_1 and v_k free is called M-augmenting.

Example for matching definitions



Remark: An M-augmenting path is simple (node disjoint).

M-augmenting Paths

Theorem (Berge, 1957)

Let G be a graph with matching M. Then we have:
M is maximum $\Leftrightarrow$ There is no M-augmenting path.

Proof.

We prove:

M not maximum $\Leftrightarrow$ There exists an M-augmenting path

„$\Leftarrow$“: Let $P = (v_1, \ldots, v_k)$ M-augmenting path

Observation: k even (thus $k = 2j$ with $j \in \mathbb{N}$)

Change the M-edges on P $\{v_2, v_3\}, \{v_4, v_5\}, \ldots, \{v_{2j-2}, v_{2j-1}\}$ to free edges.

Change free edges on P $\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{2j-1}, v_{2j}\}$ to M-edges.

Observation: M is still a matching.

Observation: size of M grows by 1.

Proof of opposite direction

„M not maximum $\Rightarrow \exists$ M-augmenting path“

Let M be not maximum.

$\exists M' : |M'| > |M|$

Consider $M \oplus M' = \{e \mid e \in M \wedge e \notin M'\} \cup \{e \mid e \notin M \wedge e \in M'\}$

Observation in $M \oplus M'$ all vertex degrees ≤ 2 separates into disjoint paths and cycles

in all these: M-edges and M'-edges are alternating

also all cycles have even length

$|M'| > |M| \Rightarrow \exists$ path P with more M'- than M-edges

P is M-augmenting □

Theorem directly leads to basic bipartite matching algorithm

Basic Matching Algorithm for Bipartite Graphs

From now on: $G = (V, E)$ bipartite

Algorithm

1. $M := \emptyset$
2. Compute M-augmenting path P.
3. If no P found, Then Exit and Return M.
4. $M := M \oplus P$ (Augment M)
5. goto 2.

obviously terminates, since $\leq |V|/2$ Iterationen

thus correctness abvious ✓

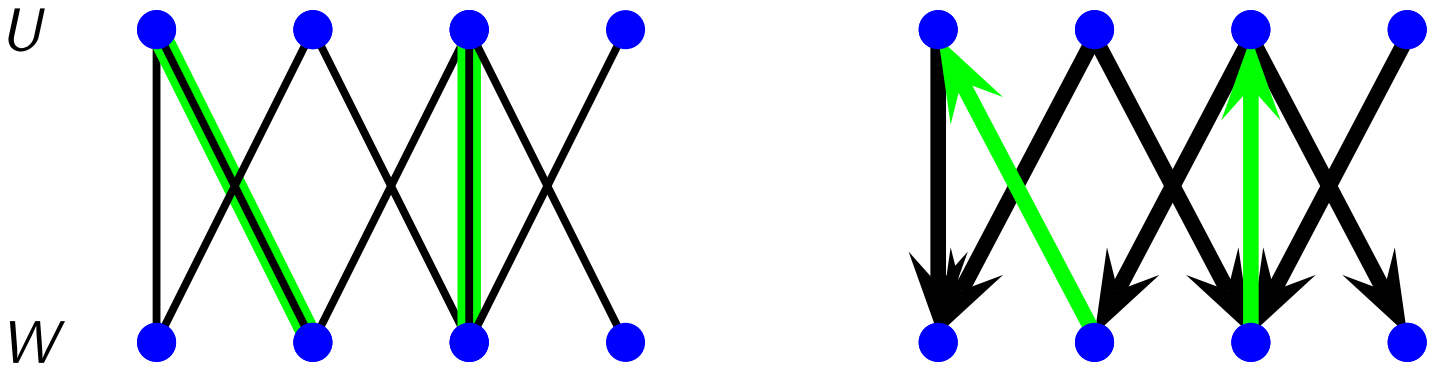
Is it effizient?

How to find M-augmenting paths

Let $G = (U \uplus W, E)$ bipartite graph, M matching in G.

We direct G:
$G_M = (U \uplus W, E_M)$ directed graph with

- $(u, w) \in E_M$ für $\{u, w\} \in E \setminus M$, $u \in U$, $w \in W$
- $(w, u) \in E_M$ für $\{u, w\} \in E \cap M$, $u \in U$, $w \in W$



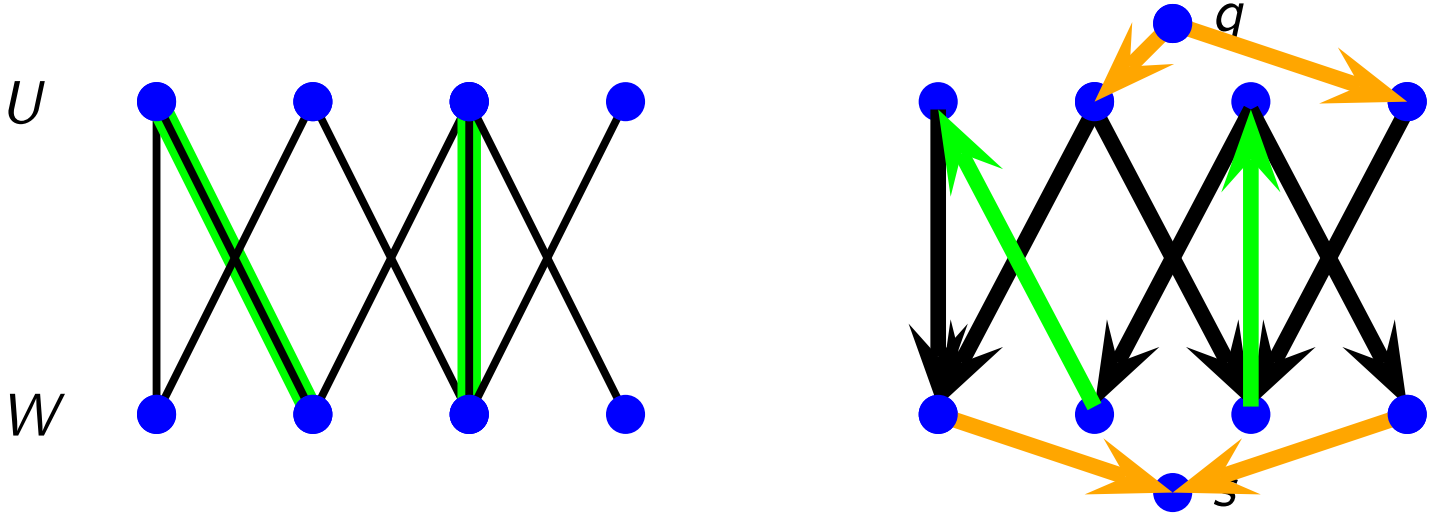
Observations

- every directed path in G_M is M-alternating path in G
- every directed path in G_M from free U-node to free W-node is an M-augmenting path in G

How to find M-augmenting paths

$G_M = (U \uplus W, E_M)$ directed graph with

- $(u, w) \in E_M$ für $\{u, w\} \in E \setminus M$, $u \in U$, $w \in W$
- $(w, u) \in E_M$ für $\{u, w\} \in E \cap M$, $u \in U$, $w \in W$



insert additionally

- nodes q, s
- all edges (q, u) with $u \in U$ free
- all edges (w, s) with $w \in W$ free

Every directed simple (q, s)-path is M-augmenting path

Algorithm

Algorithm (basic matching algorithm)

1. $M := \emptyset$
2. Construct directed graph $G' = (U \uplus W, E')$ with
   $E' = \{(u, w) \mid u \in U, w \in W, \{u, w\} \notin M\}$
   $\cup \{(w, u) \mid u \in U, w \in W, \{u, w\} \in M\}$ with new nodes q, s
   and all edges (q, u) with $u \in U$ free and all edges
   (w, s) with $w \in W$ free.
3. Use BFS to find a directed simple path P from q to s.
4. If P found
   Then $M := M \oplus P$; goto 2.
   Else Return M.

Analysis of the algorithm

Theorem

The above algorithm computes a maximum matching for a bipartite graph $G = (U \uplus W, E)$ with $|U \uplus W| = n$ and $|E| = e$ in time $O(ne) = O(n^3)$.

Beweis.

Running time	One BFS call: $O(e)$
	Augmentation of the M-augmenting path: $O(n)$
	Construction/Update of the directed graph: $O(e)$
	maximum $\leq n/2$ iterations
Correctness	all found directed paths „free $\rightsquigarrow$ free“
	$\Leftrightarrow$ M-augmenting paths
obviously	BFS finds such paths $\square$

Is there a faster algorithm?

Ideas for Improvement

fact a graph traversal needs time $O(e)$

so far a graph traversal needed for matching improvement of 1

wanted matching improvements „in larger jumps“

How does it work?

obviously parallel „addition“
of k vertex disjoint M-augmenting paths
is possible and improves by k

Idea Find many vertex disjoint M-augmenting paths
in one graph traversal.

intuitively short M-augmenting paths seem to be favorable

Algorithm by Hopcroft and Karp

First we show a lower bound on vertex disjoint M-augmenting paths:

Lemma:

Let $G = (V, E)$ be an arbitrary graph, let $M, N \subseteq E$ matchings in G. If $|N| > |M|$, then $(M \oplus N)$ contains at least $|N| - |M|$ vertex disjoint M-augmenting paths.

Proof.

Consider $M \oplus N$. We use similar arguments as for the proof of Berge's Theorem. Each M-augmenting path contains at most one more edge in N than in M.

$\Rightarrow$ Thus there must exist $s := |N| - |M|$ vertex disjoint M-augmenting paths.

Algorithm by Hopcroft and Karp

Lemma: Properties of shortest paths

Let $G = (V, E)$ be an arbitrary graph, $M \subseteq E$ matching in G,
P a shortest M-augmenting path,
P' a $(M \oplus P)$-augmenting path.
Then: $|P'| \geq |P| + |P \cap P'|$.

Reminder: $M \oplus P = \{e \mid e \in M \wedge e \notin P\} \cup \{e \mid e \notin M \wedge e \in P\}$

Proof.

Consider $N := (M \oplus P) \oplus P'$ (new matching)

obviously $|N| = |M| + 2$

We have $M \oplus N = M \oplus ((M \oplus P) \oplus P') = P \oplus P'$

From previous lemma: $M \oplus N$ contains two vertex disjoint M-augmenting paths P_1, P_2

Proof of Lemma ff

We have matching M,
P a shortest M-augmenting path
P' a $(M \oplus P)$-augmenting path
$N = (M \oplus P) \oplus P'$
$M \oplus N = P \oplus P'$ contains two vertex disjoint M-augmenting paths P_1, P_2
$|M \oplus N| = |P \oplus P'| \geq |P_1| + |P_2|$

obviously $|P| \leq |P_1|$ und $|P| \leq |P_2|$
thus $|P \oplus P'| \geq |P_1| + |P_2| \geq 2|P|$
Observation $|P \oplus P'| = |P| + |P'| - 2|P \cap P'|$
also $|P| + |P'| - |P \cap P'| \geq |P \oplus P'|$
alltogether $|P| + |P'| - |P \cap P'| \geq 2|P|$
thus $|P'| \geq |P| + |P \cap P'|$ □

We have shown:

Lemma: Properties of shortest paths

Let $G = (V, E)$ be an arbitrary graph, $M \subseteq E$ matching in G,
P a shortest M-augmenting path,
P' a $(M \oplus P)$-augmenting path.
Then: $|P'| \geq |P| + |P \cap P'|$.

From this we get:

Observation for shortest paths

Let $G = (V, E)$ be an arbitrary graph, $M \subseteq E$ matching in G,
P a shortest M-augmenting path,
P' a $(M \oplus P)$-augmenting path.
If $|P'| = |P|$, then the two paths are vertex disjoint.

A sequence of matchings

Lemma

Let $G = (V, E)$ be an arbitrary graph, $M_0 := \emptyset$, für $i \in \mathbb{N}_0$ let $M_{i+1} := M_i \oplus P_i$, whereas P_i is a shortest M_i-augmenting path.
For all $i, j \in \mathbb{N}_0$ we have:
1. $|P_i| \leq |P_{i+1}|$
2. $|P_i| = |P_j|$ and $i \neq j \Rightarrow P_i$ and P_j are vertex disjoint

Proof (sketch).

1. Observation follows directly from previous lemma ✓
2. by contradiction and using previous lemma (skipped)

Observation: many shortest M-augmenting paths can be augmented in parallel

Algorithm by Hopcroft and Karp (1971)

Algorithm (Hopcroft and Karp)

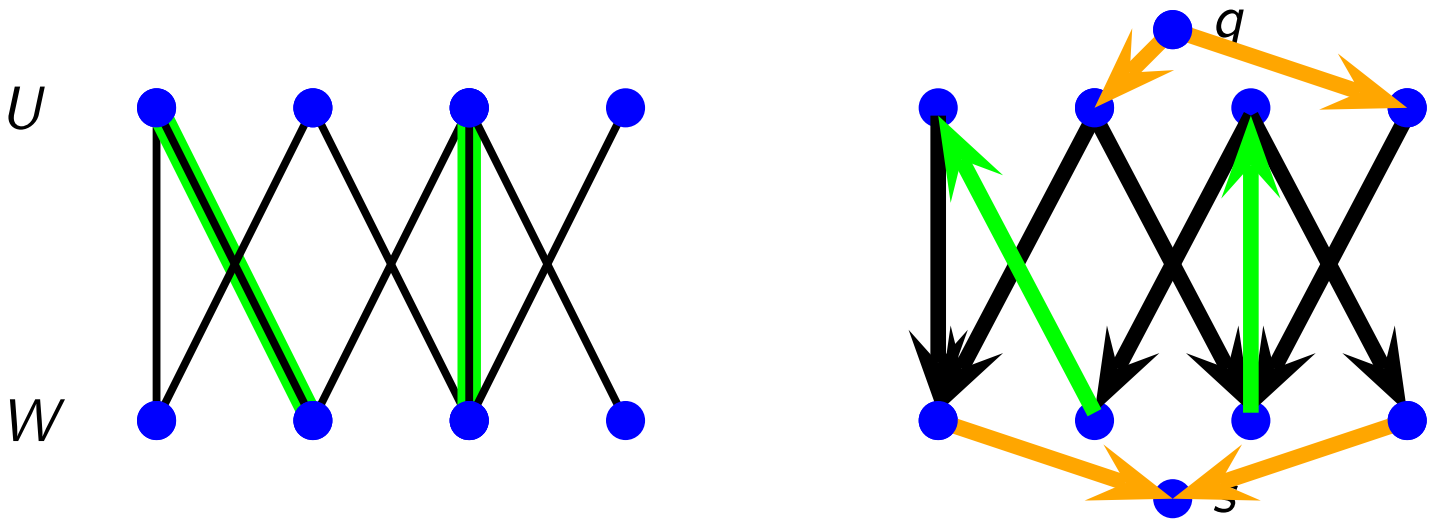
1. $M := \emptyset$
2. Compute a maximal set of shortest vertex disjoint M-augmenting paths $P_1, P_2, \ldots, P_k$.
3. If $k \geq 1$
   Then $M := M \oplus P_1 \oplus P_2 \oplus \cdots \oplus P_k$. goto 2.
   Else Return M.

How to find maximal set of disjoint paths

Basic matching algorithm for bipartite graphs:

$G_M = (U \cup W, E_M)$ directed graph with

- $(u, w) \in E_M$ for $\{u, w\} \in E \setminus M$, $u \in U$, $w \in W$
- $(w, u) \in E_M$ for $\{u, w\} \in E \cap M$, $u \in U$, $w \in W$



additionally inserted

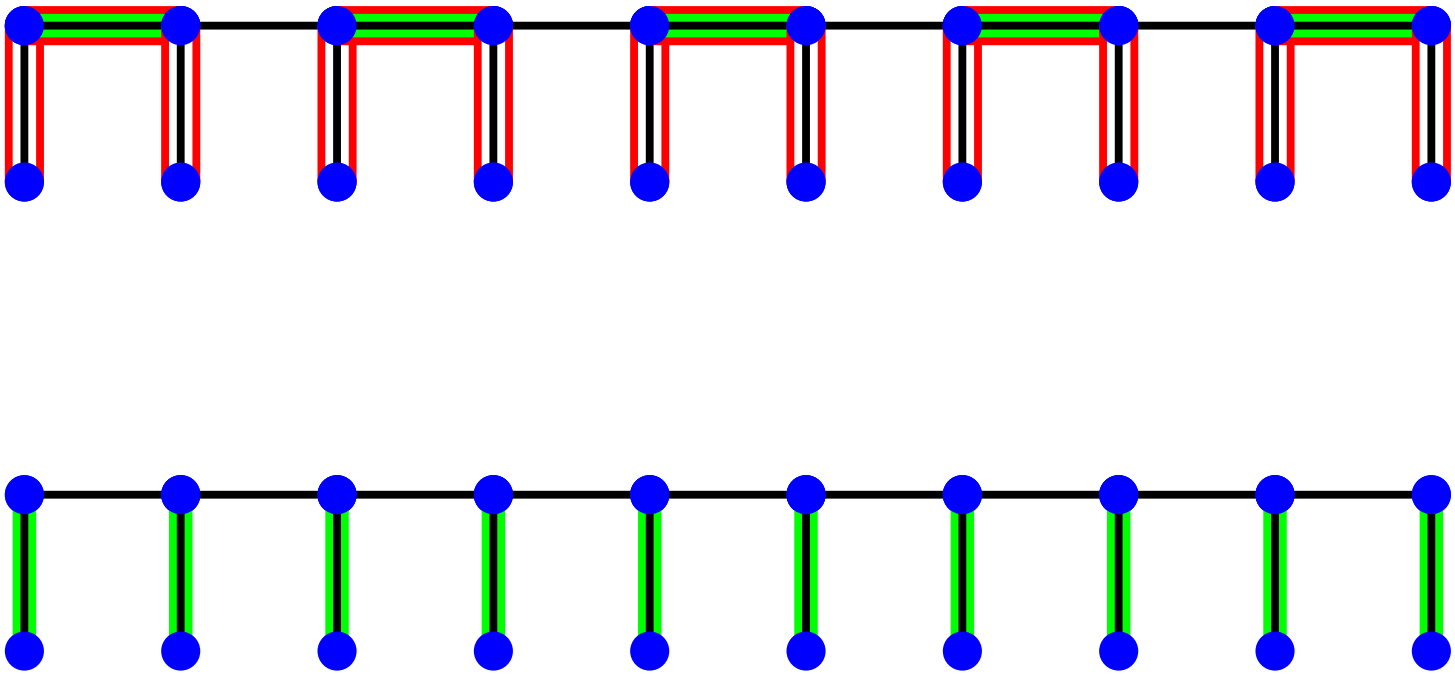
- nodes q, s
- all edges (q, u) with $u \in U$ free
- all vertices (w, s) with $w \in W$ free

Efficient implementation of one round

1. Construct directed graph with new nodes q, s and new edges $\{(q,u) \mid u \in U \text{ free}\}$, $\{(w,s) \mid w \in W \text{ free}\}$.
2. Calculate shortest distances using BFS ($dist()$)-values from q) of reached vertices until s has been reached the first time; then stop BFS
3. Using DFS which uses only edges (u,v) with $\text{dist}(v) - \text{dist}(u) = 1$, extract all shortest q-s-paths, after each extracted path P mark vertices of P as not usable anymore
4. Use all found shortest q-s-paths as M-augmenting paths.

obviously every step runs in time $O(n+e) = O(e)$

Example



Algorithm by Hopcroft and Karp (1971)

Algorithm (Hopcroft and Karp)

1. $M := \emptyset$
2. Compute a maximal set of shortest vertex disjoint M-augmenting paths $P_1, P_2, \ldots, P_k$.
3. If $k \geq 1$
   Then $M := M \oplus P_1 \oplus P_2 \oplus \cdots \oplus P_k$. goto 2.
   Else Return M.

Theorem

For a bipartite graph $G = (U \uplus W, E)$ with $|U \uplus W| = n$ und $|E| = e$ the previous algorithm computes a maximum matching in time $O\left(\sqrt{n} \cdot e\right) = O\left(n^{5/2}\right)$.

Proof of the theorem

Correctness obvious after previous lemmas ✓

We will show

1. Each round needs time $O(e)$ ✓
2. No more than $O\left(\sqrt{n}\right)$ rounds are needed.

useful upper bound for length of shortest M-augmenting paths
Why?

we know Length of shortest M-augmenting paths grows in each round
(because of lemma: if they had the same length, they would be node disjoint and so we would have found them in the previous round)

thus if shortest paths are short $\Rightarrow$ not many rounds needed

M-augmenting paths: number and length

Consider matching M and maximum matching M_{opt}
($|M| \leq |M_{\text{opt}}|$)

Consider $M \oplus M_{\text{opt}}$ connected components
$C_i = (V_i, E_i)$ $(i \in \{1, 2, \dots\})$

Reminder all C_i are either cycles of even length
or simple paths
$\Rightarrow$ there are $\geq |M_{\text{opt}}| - |M|$ vertex disjoint
M-augmenting paths

M-augmenting paths: number and length

we have $\geq |M_{\text{opt}}| - |M|$ vertex disjoint M-augmenting paths
with $\leq |M|$ M-edges

pigeonhole principle $\exists$ M-augmenting path
with $\leq \left\lfloor \frac{|M|}{|M_{\text{opt}}|-|M|} \right\rfloor$ M-edges

klar M-edges and M_{opt}-edges alternate

also path length $\leq 2 \left\lfloor \frac{|M|}{|M_{\text{opt}}|-|M|} \right\rfloor + 1$

Observation shortest M-augmenting paths short,
if $|M_{\text{opt}}| - |M|$ large

Idee Two cases
Case 1: $|M_{\text{opt}}| - |M|$ large $\rightsquigarrow$ only short paths
Case 2: $|M_{\text{opt}}| - |M|$ small $\rightsquigarrow$ only few rounds

Number of rounds of the Hopcroft-Karp algorithm

Define two phases

phase 1 $0 \leq |M| \leq \left\lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \right\rfloor$

phase 2 $\left\lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \right\rfloor < |M| < |M_{\text{opt}}|$

Phase 2

obviously $|M_{\text{opt}}| \leq n/2$ (def. matching)

thus $\sqrt{|M_{\text{opt}}|} = O\left(\sqrt{n}\right)$

thus in Phase 2 only $O\left(\sqrt{n}\right)$ rounds ✓

Length of shortest M-augmenting paths in phase 1

Phase 1 $\quad 0 \leq |M| \leq \left\lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \right\rfloor$

Reminder length of shortest M-augmenting paths
$\leq 2 \left\lfloor \frac{|M|}{|M_{\text{opt}}|-|M|} \right\rfloor + 1$

$$2 \left\lfloor \frac{|M|}{|M_{\text{opt}}| - |M|} \right\rfloor + 1 \leq 2 \left\lfloor \frac{\left\lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \right\rfloor}{|M_{\text{opt}}| - \left\lfloor |M_{\text{opt}}| - \sqrt{|M_{\text{opt}}|} \right\rfloor} \right\rfloor + 1$$

$$= 2 \left\lfloor \frac{|M_{\text{opt}}| - \left\lceil \sqrt{|M_{\text{opt}}|} \right\rceil}{\left\lceil \sqrt{|M_{\text{opt}}|} \right\rceil} \right\rfloor + 1$$

$$= 2 \left\lfloor \frac{|M_{\text{opt}}|}{\left\lceil \sqrt{M_{\text{opt}}} \right\rceil} - 1 \right\rfloor + 1$$

$$\leq 2\sqrt{|M_{\text{opt}}|} + 1$$

Number of rounds

We have
- $O\left(\sqrt{|M_{\text{opt}}|}\right) = O\left(\sqrt{n}\right)$ rounds in phase 2
- length of shortest M-augmenting paths $= O\left(\sqrt{|M_{\text{opt}}|}\right) = O\left(\sqrt{n}\right)$ in phase 1

Reminder length of shortest M-augmenting paths grows by ≥ 1 in each round

thus after $O\left(\sqrt{|M_{\text{opt}}|}\right) = O\left(\sqrt{n}\right)$ rounds phase 1 is finished

thus in total $O\left(\sqrt{|M_{\text{opt}}|}\right) = O\left(\sqrt{n}\right)$ rounds

Since each round can be realized in time $O(e)$, we have:

total running time of algorithm: $O\left(\sqrt{n} \cdot e\right) = O\left(n^{5/2}\right)$

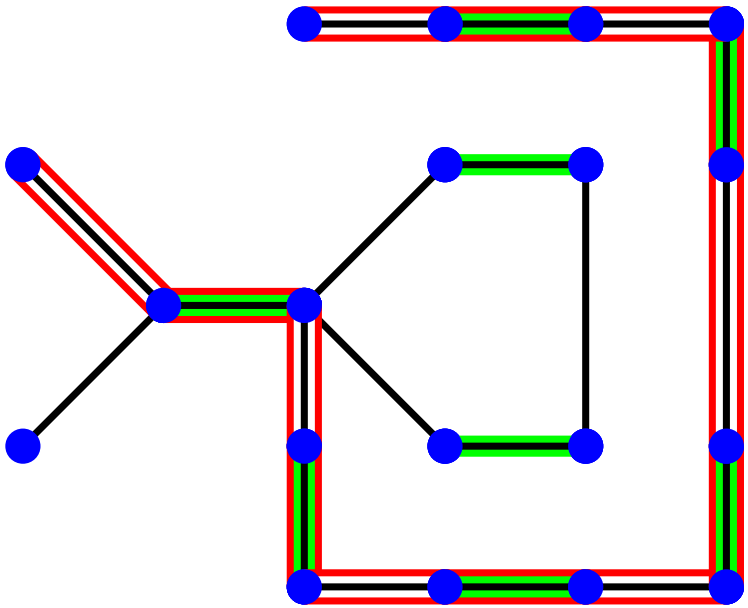
Matchings in general graphs

klar Algorithm by Hopcroft and Karp does not work, because of no sets $U \uplus W = V$ identifiable

Is the general idea still allowed?

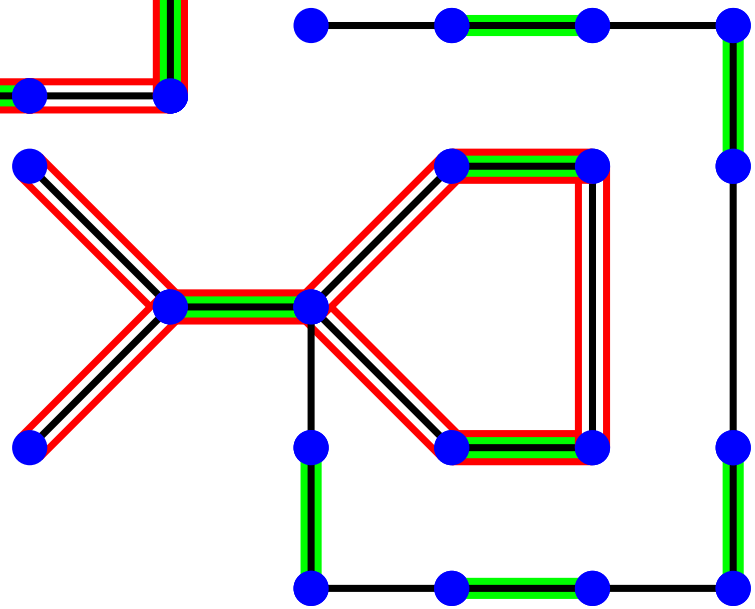
1. $M := \emptyset$
2. Find M-augmenting path P.
3. If P found, Then $M := M \oplus P$; goto 2.
4. Return M

An example



M-augmenting path

of length 13



M-alternating path
of length 9 with
free endvertices
„blossom“

How to deal with „blossoms“

Observation Cycles of odd length can make problems

Remarks Shortest alternating paths
with 2 free endvertices are not automatically
M-augmenting in general graphs;
In bipartite graphs they are,
because all cycles are even

Idea byJack Edmonds: „Heureka, you shrink“

Contraction of the odd cycles („blossoms“) to a node

Remark: repeated contraction may lead to hierarchy of blossoms

This idea led to the first polynomial time algorithm for maximum matching in general graphs

Running time was large: $O(n^4)$

Idea by Micali and Vazirani (1980)

Reminder Algorithm by Hopcroft and Karp
solves the problem in time $O\left(n^{5/2}\right)$ for bipartite graphs

What can we keep from bipartite graphs? – What do we need to change?

Theorem: M maximum $\leftrightarrow \nexists$ M-verbessernder Pfad?

Observation valid for general graphs ✓

Lemma: shortest M-augmenting paths grow,
shortest M-augmenting paths of equal length are disjoint?

Observation valid for general graphs ✓

Hopcroft/Karp for general graphs

What else can we take?

Observation Basis algorithm structure works ✓

1. $M := \emptyset$
2. Compute a maximum set of shortest node disjoint M-augmenting paths $P_1, P_2, \ldots, P_k$.
3. If $k \geq 1$
   Then $M := M \oplus P_1 \oplus P_2 \oplus \cdots \oplus P_k$. goto 2.
   Else Return M.

What about the number of rounds?

Observation Proof works
$O\left(\sqrt{M_{\text{opt}}}\right) = O\left(\sqrt{n}\right)$ rounds

thus only efficient implementation of step 2 open

Matching algorithmus for general graphs

Remark using efficient data structures
$\rightsquigarrow$ running time $O(e)$ for one round

Theorem

The algorithm by Micali und Vazirani computes a maximum matching in a general graph $G = (V, E)$ with $|V| = n$ und $|E| = e$ in time $O(\sqrt{n}e)$.

Proof and algorithm details: skipped

Remarks to bipartite matching algorithms

- The algorithm by Hopcroft-Karp is currently the (theoretically) fastest known algorithm for sparse graphs.
- Alt, Blum, Mehlhorn and Paul suggested 1991 an algorithm based on network-flow which is faster for dense graphs and needs time $O(n^{1.5}\sqrt{e/\log n})$.
- Experimental comparisons (many years ago) give contradictory results concerning the best practical algorithms.

Literature

- Original article: John E. Hopcroft, Richard M. Karp: An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs", SIAM Journal on Computing 2 (4): 225–231, 1973
- Original article: Helmut Alt, Norbert Blum, Kurt Mehlhorn, Markus Paul: Computing a maximum cardinality matching in a bipartite graph in time, Information Processing Letters 37 (4): 237–240, 1991
- Experimental comparison: Kurt Mehlhorn: The Engineering of Some Bipartite Matching Programs, LNCS 1741, Springer, 1–3, 1999
- Experimental comparison: Boris V. Cherkassky, Andrew V. Goldberg, Paul Martin, J.C. Setubal, Joao C. Setubal, Jorge Stolfi: Augment or push: A computational study of bipartite matching and unit capacity flow algorithms, ACM J. Exp. Algorithmics, vol. 3 (8), 1998
- General graphs: Silvio Micali and Vijay V. Vazirani: An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In: 21st Annual Symp. on Foundations of Computer Science, 17–27, 1980.