Αλγόριθμοι δικτύων και Πολυπλοκότητα

Perfect Matching

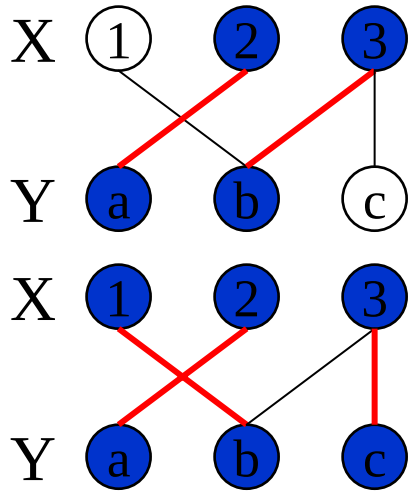
Ν. Αποστόλου
ΜΟΠ083

Contents

- Matchings
- Problem Complexity
- Applications
- Basic definitions
- Main theorem – augmenting path
- Maximum Matching algorithm
- Augmenting paths – bipartite graphs
- Augmenting paths – non bipartite graphs
- Edmond's algorithm
  - Definitions
  - Correctness
  - Analysis
- Historical notes
- References

Matchings

- Matching
  - Set of edges $\mathrm{M} \subseteq \mathrm{E}$ such that no vertex is endpoint of more than one edge or
  - A set of edges which do not share a vertex
- Maximal matching
  - No $e \notin \mathrm{E}$ with $\mathrm{M} \cup \{e\}$ also a matching
- Maximum matching
  - Matching with |M| as large as possible
  - Maximal ≠ Maximum
- Perfect matching
  - |M| = *n*/2: each vertex is an endpoint of edge in M.
  - If there exists such a matching, n must be even.
  - Not every graph of even order has a perfect matching.



Matchings - Cost versions

- Weighted Matching
  - Each edge has cost; look for perfect matching with minimum (maximum) cost
- Max-Min Matching
  - Each edge has cost; look for maximum matching for which the minimum of the weights in the matching is maximum. (bottleneck)

Problem complexity

- Given graph G, find
  - Maximal matching: easy (greedy algorithm)
  - Maximum matching
    - Polynomial time; not easy.
    - Important easier case: bipartite graphs
  - Perfect matching
    - Special case of maximum matching

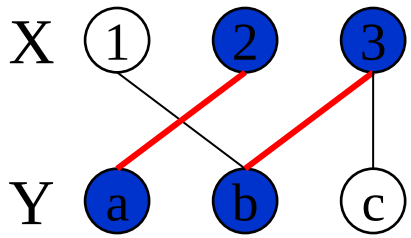
Applications

- Wireless Networks may consist of nodes with single radios, no interference constraint
  - A set of links can be scheduled only if the set is a matching
- Personnel assignment
  - Tasks and competences
- Opponents selection for sport competitions

Matching – basic definitions

Matched vertex: Vertex v of graph G that is incident with an edge of M.

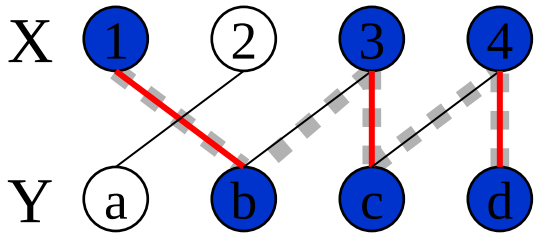
Matched edge: Edge of G that belongs to M. Otherwise it is unmatched.



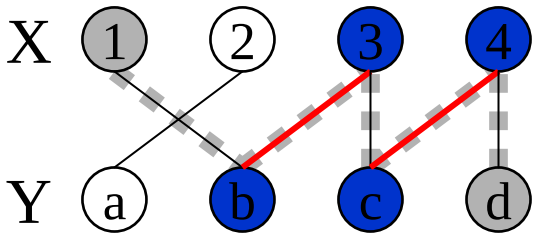
Free node: A node is free if none of its incident edges is in a matching. E.g nodes 1 and c above.

Matching – basic definitions

M-Alternating path: A path in G whose edges are alternatively matched and unmatched.



M-Augmenting path: Augmenting path is a nontrivial path of alternating sequence of matched and unmatched edges with free end nodes.



Size of matching increases if an augmenting path is used

Matching – Main theorem

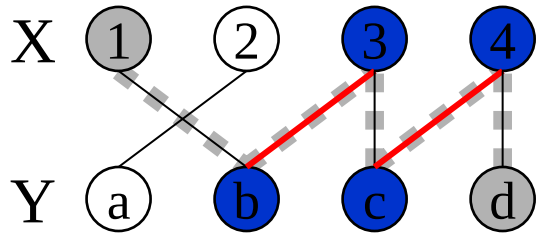
Theorem: A matching is maximum if and only if there is no augmenting path. (Berge - 1957)

(Definition) $A \oplus B = (A - B) \cup (B - A)$

Suppose M is a matching and P is an M-augmenting path.

Then $M \oplus P$ is also a matching

Also, $|M \oplus P| = |M| + 1$



So, if there exists an augmenting path, then clearly the matching is not maximum.

Note: The above holds in the general case of non-bipartite graphs.

Matching – Main theorem - II

Let there be a matching M` of greater cardinality than M.

Consider the graph $M \oplus M`$.

a. A vertex can have one incident edge from M, or one incident edge from M`, or one incident edge from M and another from M` (not more than one edge in M or M`).

b. Clearly, a vertex can have at most 2 incident edges.

So from (a) and (b) the new graph consists of disjoint paths and cycles.

The cycles must be even cycles with alternating edges from M and M`.

Matching – Main Theorem - III

Paths will also have edges alternating between M and M`

Since M` has a cardinality greater than M, there must be at least one alternating path which starts and ends in M`. (M $\oplus$ M` has more edges from M` than from M).

This is an augmenting path for M. (Since it extends between two nodes that are free by M).

Theorem has been proven.

Maximum Matching Algorithm

Start with an empty set.

(1) Search for an augmenting path.

If there is an augmenting path, then update the matching, else terminate.

Go to (1)

Key question: How does one find an augmenting path?

Bipartite Graph – Augmenting path

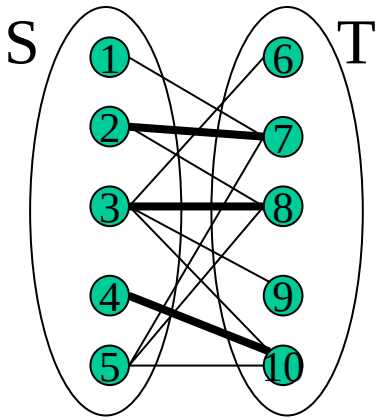
Finding an augmenting path

Augmenting path can be found by Hungarian tree method

(Breadth first search like method)

Definition: alternating tree relative to Matching M is a tree satisfying the following

- It has one free node from S as root
- All paths from root to any node are alternating paths.

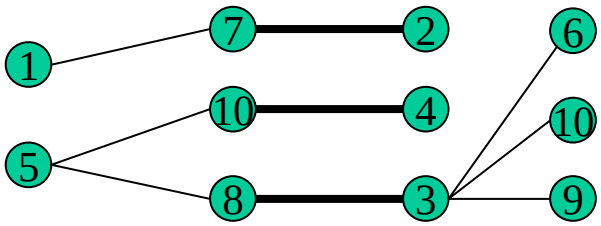


Consider one of the two partitions S and any matching, possibly empty.

Start from the free vertices in S

All the edges from the free vertices are added in the tree.

Bipartite Graph – Augmenting path



If we happen to reach a free vertex in T, then we have found an augmenting path.

Otherwise we reached a matched edge so we follow back the matched edges into S.

Subsequently we follow all free edges from the newly encountered nodes in S.

If it is not possible to add more edges and nodes we have found a Hungarian tree and we have a maximum matching.

Bipartite Graph – Augmenting path

Don’t include free edges from S into already reached vertices in T.

So vertices in T are never repeated

We would never reach the same vertices in S anyway (because we follow matched edges back from T).

Vertices in S are either matched or free.

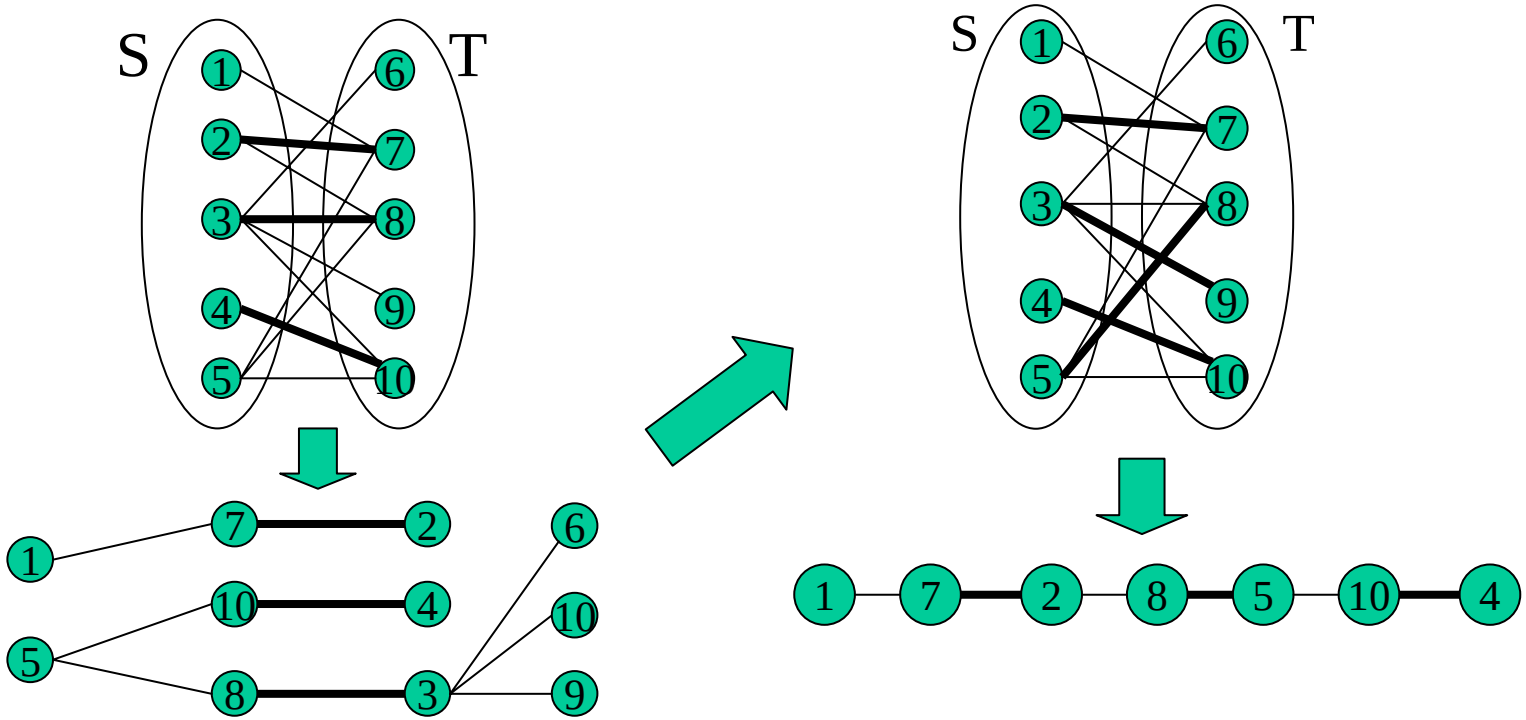
We take only matched edges to reach S from T.

A matched edge can not reach a free vertex.

A matched edge can not reach an already reached matched vertex, because it has been reached using a different matched edge.

Bipartite Graph – Augmenting path

Hungarian tree search finishes in O(E)



Bipartite Graph – Augmenting path

Is it possible that there is an augmenting path, but the Hungarian tree does not yield one?

An augmenting path has odd length.

Any odd length path must have one end in S and another end in T.

An augmenting path must have one free vertex in S and another in T.

Consider an augmenting path from the free vertex in S.

Bipartite Graph – Augmenting path

The first edge is from a free vertex in S.

The second edge is a matched edge from T.

The third edge is a free edge from S.

The fourth is a matched edge from T….

We are looking at **all** such paths in the Hungarian method.

If there is an augmenting path, we would find one!

Complexity Analysis

Once an augmenting path is found, one new vertex is matched.

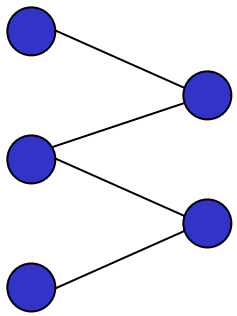
Need to look for augmenting paths at most V times.

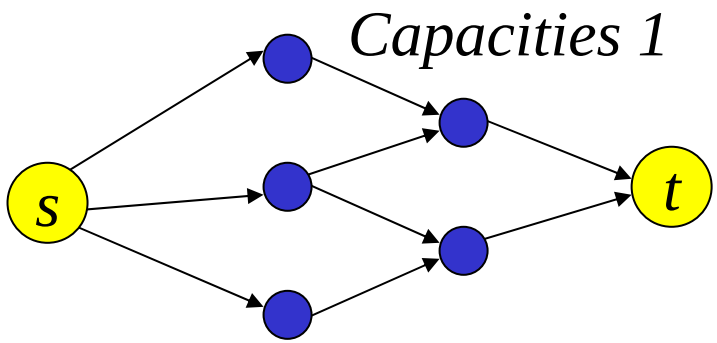
Looking for an augmenting path consume O(E).

So, overall O(VE).

Bipartite graphs – alternative solution: reduction to maximum flow problem

- Finding maximum matching in bipartite graphs:
  - Model as flow problem, and solve it: algorithm finds integral flow (uni-modularity property).





Cost Versions : Alternative solution

- Minimum cost perfect matching in bipartite graphs
  - Model as Min-Cost flow problem

Nonbipartite Graph

– Augmenting path

At one time was thought that theorem for augmenting paths was sufficient for the problem.

However a systematic procedure to find such paths either:

- Contain pitfalls or
- Involves an exponentially growing amount of computation

Nonbipartite Graph

– Augmenting path

Basic algorithm remains the same.

Need to find augmenting paths in different ways.

We can not partition the vertex set into two sets, and start with free vertices in just one set.

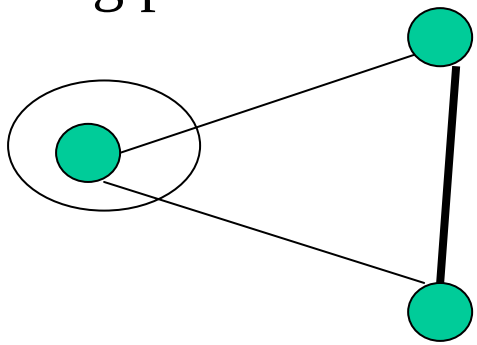
As a result need to consider all free vertices.

In the search for an augmenting path we may clearly reach some free vertex because such a path ends in a free vertex.

Nonbipartite Graph

– Augmenting path

But reaching a free vertex does not mean that we have reached an augmenting path.



So when we reach a free vertex, we do not know whether we have an augmenting path or not! At the same time we can not rule out reaching free vertices (as an augmenting path starts and ends in a free vertex).

Odd cycles are the problems.

Nonbipartite Graph

– Augmenting path

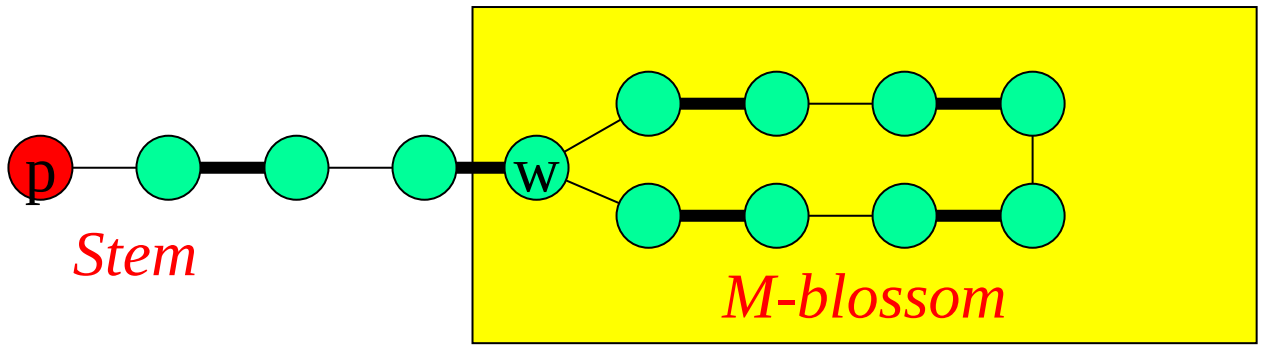
- So for nonbipartite graphs, we start from a single free vertex at a time.
- Also, whenever odd cycles are detected they are shrunk into one vertex, called blossoms.
- After discovering augmenting paths, the blossoms are expanded.

Edmond's Algorithm

- Finds maximum matching in a graph in polynomial time
- Constructs alternating trees (much as bipartite matching)
- Detects certain odd cycles called blossoms
- Shrinks these blossoms by contracting the graph

Definitions

- M-blossom:
  - An **odd** alternating cycle that starts and terminates at terminal node w called base.
- Stem:
  - An **even** alternating path that starts at free node p called root and terminates at terminal node w.
- M-flower
  - M-alternating walk that starts in an unmatched vertex, and ends as:



Algorithmic idea

- Start with some matching M, and find either M-augmenting path or M-blossom.
- If we find an M-augmenting path:
  - Augment M, and obtain matching of one larger size; repeat.
- If we find an M-blossom, we shrink it, and obtain an equivalent smaller problem; recurse.

Finding an M-augmenting path or an M-flower – I

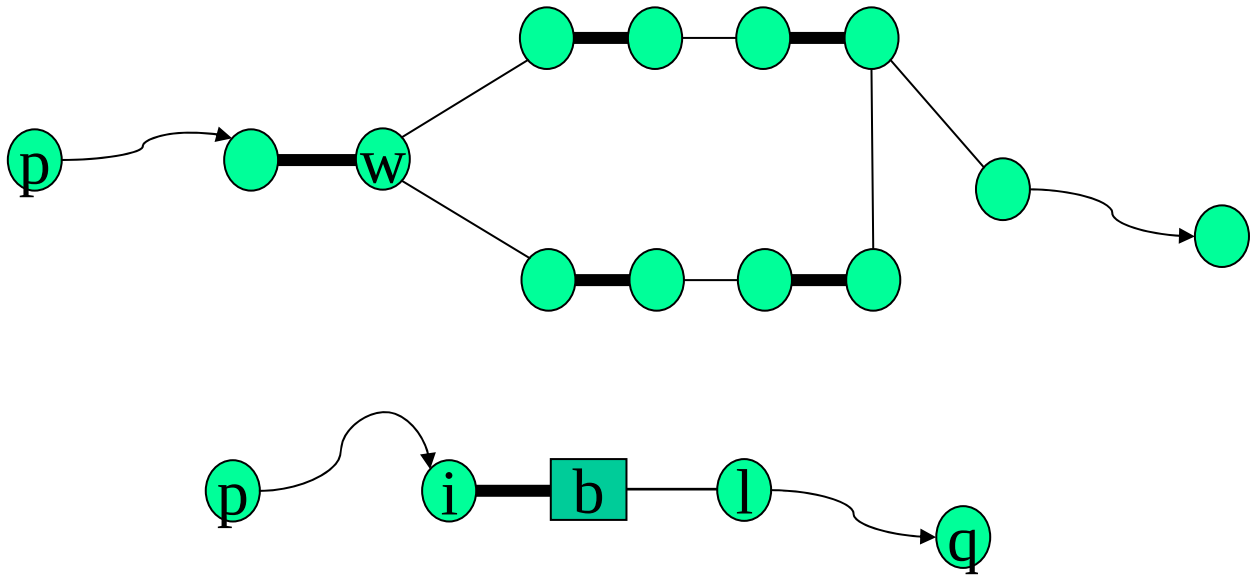
- Let X be the set of unmatched vertices.
- Let Y be the set of vertices with an edge not in M to a vertex in X (neighbours of X).
- Build digraph D = (V,A) with A = { (u,v) | there is an x with $\{u,x\}$ in E-M and $\{x,v\}$ in M}.
  - It can be shown easily that an M-alternating path from X to X in G yields a directed X to Y path in D and vice-versa
- Find a shortest path P from a vertex in X to a vertex in Y of length at least 1. (BFS in D.)
- Take P`: P, followed by an edge to X.
- P` is M-alternating walk between two unmatched vertices.

Finding M-augmenting path or M-flower – II

- Two cases:
  - P` is a simple path: it is an M-augmenting walk
  - P` is not simple. Look to start of P` until the first time a vertex is visited for the second time.
    - This is an M-flower:
      - Cycle-part of walk cannot be of even size, as it then can be removed and we have a shorter walk in D.

Shrinking M-blossoms

- B a set of vertices in G. G/B is the graph, obtained from G by contracting B to a single vertex.
  - M/B: those edges in M that are not entirely on B.



Subroutine

- Given: Graph G, matching M
- Task: Find M-augmenting path if it exists.
  - Let X be the vertices not endpoint of edge in M.
  - Build D, and test if there is an M-alternating walk P from X to X of positive length. (Using Y, etc.)
  - If no such walk exists: M is maximum matching.
  - If P is a path: output P.
  - If P is not a path:
    - Find M-blossom B on P.
    - Shrink B, and recurse on G/B and M/B.
    - If G/B has no M/B-augmenting path, then M is maximum matching.
    - Otherwise, expand M/B-augmenting path to an M-augmenting path.

Algorithm correctness

Lemma 1: If G contains an augmenting path from p to q with respect a matching M, then G/B contains an augmenting path with respect to M/B.

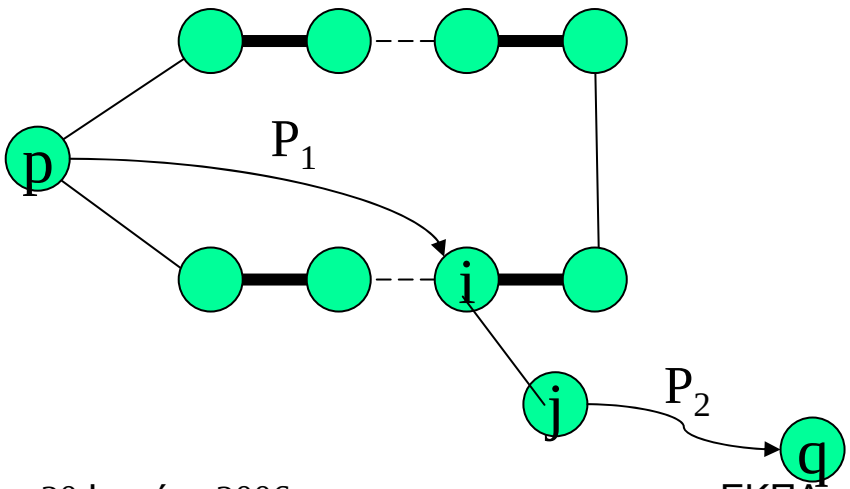
Proof:

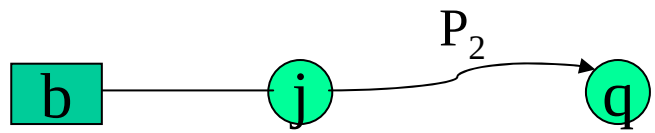
- G contains an M-augmenting path P from p to q
- We can assume with no loss of generality that p and q are the only free nodes in G (we can delete the rest otherwise)
- If P has no node in common with blossom B, then P is an M-augmenting path in G/B as well
- If P has some nodes in common with B then we consider two cases:
  - Case 1: B has an empty stem
  - Case 2: B has a nonempty stem

Algorithm correctness - II

Case 1: empty stem

- p is the base of the blossom
- Path P is of the form $\{P_1,(i , j), P_2 \}$ where i lies in the blossom and (i , j) is unmatched.
- $[(b , j), P_2]$ is an augmenting path in the contracted graph.

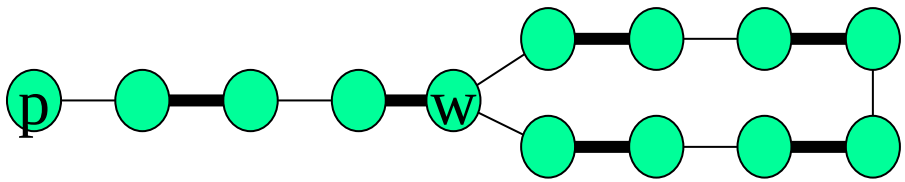


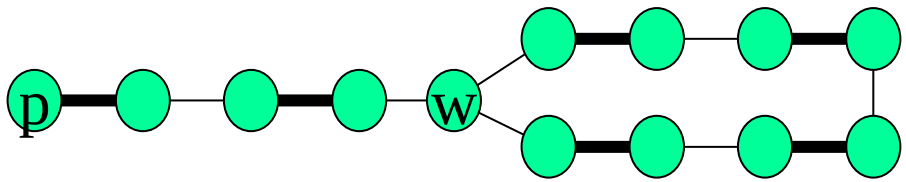


Algorithm correctness - III

Case 2: non empty stem

- Let P_3 the even alternating path from p to w (base)
- In M`=M $\oplus$ P_3 , p is matched and w is free
- |M|=|M`|, so M is maximum iff M` is maximum
- But G contains an M-augmenting path
- Therefore G contains an M`-augmenting path as well
- With respect to M`, w and q are the only free nodes, so G must contain an augmenting path between them.

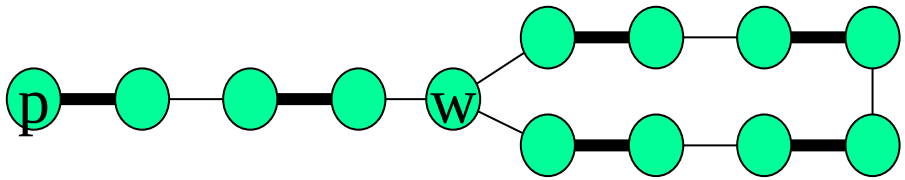




Algorithm correctness - III

Case 2: non empty stem (continued)

- Now in M` blossom B has a an empty stem and from case 1 G/B has an augmenting path with respect to M`
- But |M/B| = |M`/B| so G/B must contain an augmenting path with respect to M.
- Done

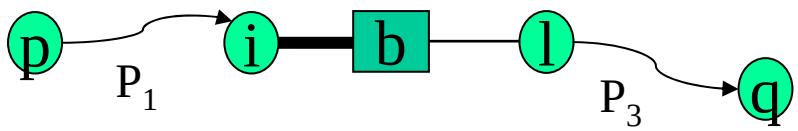


Algorithm correctness - IV

Lemma 2: If G/B contains an augmenting path P from p to q (or from the contracted node containing p) with respect a matching M/B, then G contains an M-augmenting path.

Proof:

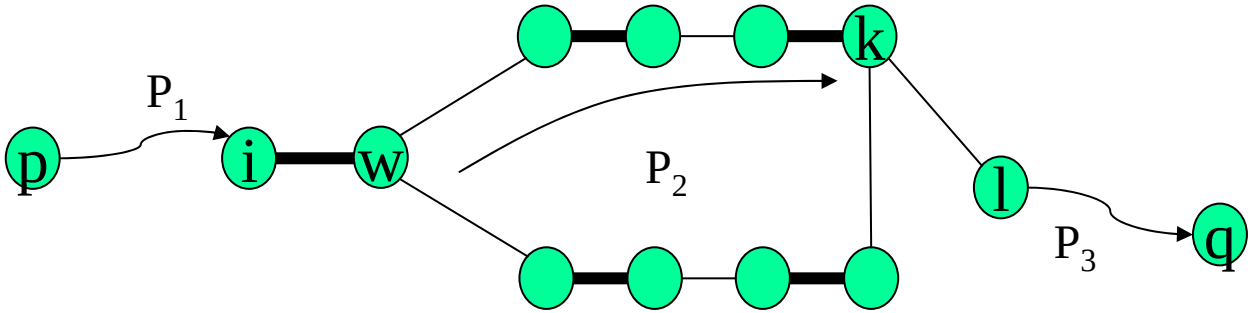
- If P does not traverse node representing p then P is an M-augmenting path in G as well
- P traverses B: construct M-augmenting path in G.
  - Let b be the contracted node. In this case b is an interior node of B
  - The number of edges from p to b is even. Lets expand the graph



Algorithm correctness - V

Lemma 2 : Proof (continued)

- Lets expand the graph



- The new graph has an even alternating path from w to k (either way) ending in a matched edge.
- Path [P_1 , (i , w) , P_2 , (k , l) , P_3] is an augmenting path for G.

- Done

Algorithm correctness - VI

Lemma 1 and Lemma 2 show that the contracted graph has an augmenting path starting at p iff the original graph contains one.

Edmond's algorithm analysis

- A maximum matching can be found in $O(n^2m)$ time.
  - Start with empty (or any) matching, and repeat improving it with M-augmenting paths until this stops.
  - Blossom has at least 3 nodes.
  - $O(n)$ iterations. Recursion depth is $O(n)$; work per recursive call $O(m)$.
- A perfect matching in a graph can be found in $O(n^2m)$ time, if it exists.

Historical notes

Non-bipartite cardinality maximum matching

- 1965 – Edmonds – First polynomial algorithm $O(n^4)$
- 1965 – Witzgall+Zahn – $O(n^2\ m)$
- 1975 – Gabow – $O(n^3)$
- 1975 – Even+Kariv – $O(n^{5/2})$
- 1974 – Kameda+Munro – $O(n \cdot m)$
- 1980 – Micali+Vazirani – $O(n^{1/2}\ m)$
  - **Fastest available algorithm**

References

- Combinatorial Optimization
  - Networks and Matroids, Eugene Lawler
- Combinatorial Optimization
  - Algorithms and Complexity, Papadimitriou-Steiglitz
- Combinatorial Optimization
  - Polyedra and Efficiency, Alexander Schrijver
- Network Flows, Ahuja-Magnati-Orlin

THE END