

MAXIMUM MATCHING IN BIPARTITE GRAPHS (18)

- We exploit the Augmenting Path Th. to design an iterative algorithm.
- During each iteration, we look for a new augmenting path using a modified Breadth First Search starting from the free nodes.
- In this way, nodes are structured in layers.

57

MAXIMUM MATCHING IN BIPARTITE GRAPHS (19)

Idea of the algorithm:

- Let M be an arbitrary matching (possibly empty)
 - Find an augmenting path P
- While there is an augmenting path:
 - Swap in P the role of the edges in and out of the matching
 - Find an augmenting path P

Complexity: it depends on the complexity of finding an augmenting path.

58

MAXIMUM MATCHING IN BIPARTITE GRAPHS (20)

Question: how to decide the existence of an augmenting path and how to find one, if one exists?

If $G=(V_1, V_2, E)$, direct edges in G according to M as follows:

- An edge goes from V_1 to V_2 if it does not belong to the matching M
- an edge goes from V_2 to V_1 if it does.

Call this directed graph D .

Claim. There exists an augmenting path in G w.r.t. M iff there exists a directed path in D between a free node in V_1 and a free node in V_2 .

59

MAXIMUM MATCHING IN BIPARTITE GRAPHS (21)

▪ **Idea:**

- For each free node in V_1
- Run a DFS on D :
 - As soon as a free node in V_2 has been encountered, a new augmenting path has been found.

Complexity: $O(n+m)$

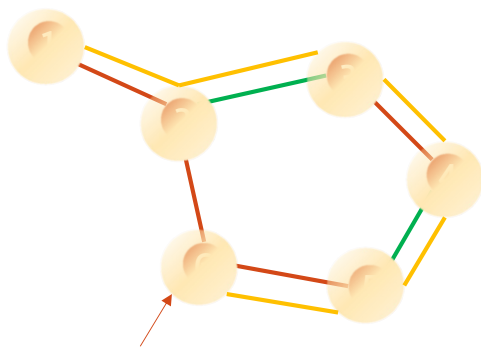
Complexity of the algorithm finding the max matching: $n/2[O(n+m)+O(n)]=O(nm)$

60

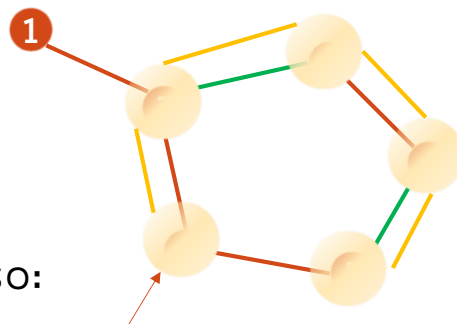
MAXIMUM MATCHING IN BIPARTITE GRAPHS (22)

(a parenthesis)
What if G is
not bipartite?

- For each free node
- Run a modified DFS:
 - Keep trace of the current layer
 - If the layer is even, use an edge in M
 - If the layer is odd, use an edge in $E \setminus M$
 - As soon as a free node has been encountered, a new augmenting path has been found



But also:

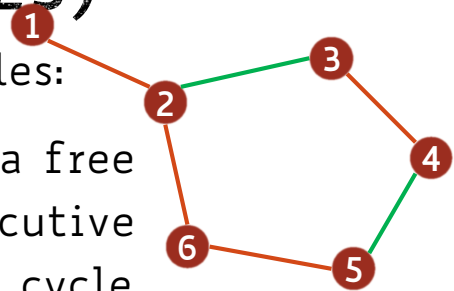


61

MAXIMUM MATCHING IN BIPARTITE GRAPHS (23)

Problem: the presence of odd cycles:

- In an odd cycle, there is always a free node adjacent to two consecutive edges not in M belonging to the cycle
- If the search goes through the cycle along the “wrong” direction, the augmenting path is not detected.



It is necessary to have graphs without odd cycles
= bipartite graphs.

We will handle the general case later...

62

MAXIMUM MATCHING IN BIPARTITE GRAPHS (24)

- The **Hopcroft-Karp algorithm** (1973) finds a max matching in a bipartite graph in $O(m\sqrt{n})$ time (better than the previous $O(mn)$).
- The idea is similar to the previous one, and consists in augmenting the cardinality of the current matching exploiting augmenting paths.
- During each iteration, this algorithm searches not one but a maximal set of augmenting paths.
- In this way, only $O(\sqrt{n})$ iterations are enough.

63

MAXIMUM MATCHING IN BIPARTITE GRAPHS (25)

Hopcroft-Karp Algorithm

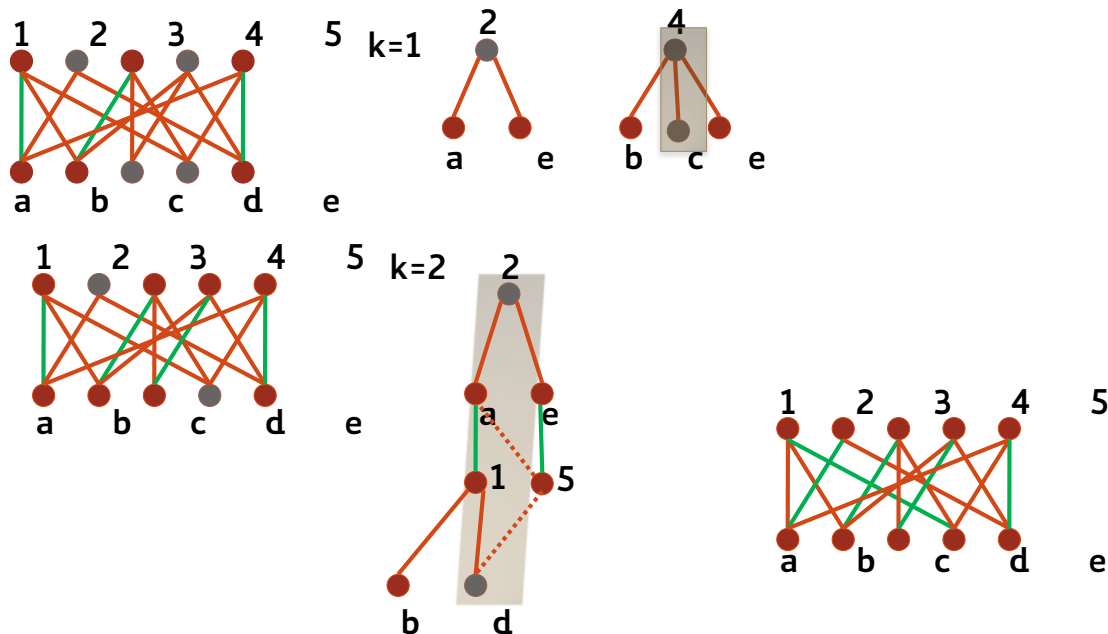
During the k -th step:

- Run a modified **breadth first search** starting from ALL the free nodes in V_1 . The BFS ends when some free nodes in V_2 are reached at layer $2k-1$.
- All the detected free nodes in V_2 at layer $2k-1$ are put in a set F . **Obs.** v is put in F iff it is the endpoint of an aug. path
- Find a maximal set of length $2k-1$ aug. paths node disjoint using a **depth first search** from the nodes in F back to the starting nodes in V_1 (climbing on the BFS tree).
- Each aug. path is used to augment the cardinality of M .
- The algorithm ends when there are no more aug. paths.

64

MAXIMUM MATCHING IN BIPARTITE GRAPHS (26)

Example: Hopcroft-Karp algorithm



65

MAXIMUM MATCHING IN BIPARTITE GRAPHS (27)

Analysis of the Hopcroft-Karp algorithm (sketch)

Each step consists in a BFS and a DFS. Hence it runs in $\Theta(n+m) = \Theta(m)$ time.

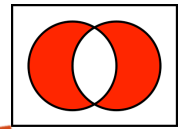
How many steps?

- The first \sqrt{n} steps take $\Theta(m\sqrt{n})$ time.
- Note. At each step, the length of the found aug. paths is larger and larger; indeed, during step k , ALL paths of length $2k-1$ are found and, after that, only longer aug. paths can be in the graph.
- So, after the first \sqrt{n} steps, the shortest aug. path is at least $2\sqrt{n}+1$ long.
- ...

66

MAXIMUM MATCHING IN BIPARTITE GRAPHS (28)

Analysis of the Hopcroft-Karp algorithm (sketch) – contd



- The **symmetric difference** between a maximum matching and the partial matching M found after the first \sqrt{n} steps is a set of vertex-disjoint alternating cycles, alternating paths and augmenting paths.
- Consider the augmenting paths. Each of them must be at least \sqrt{n} long, so there are at most \sqrt{n} such paths. Moreover, the maximum matching is larger than M by at most \sqrt{n} edges.
- ...

67

MAXIMUM MATCHING IN BIPARTITE GRAPHS (29)

Analysis of the Hopcroft-Karp algorithm (sketch) – contd

- ...
- Each step of the algorithm augments the dimension of M by one, so at most \sqrt{n} further steps are enough.
- The whole algorithm executes at most $2\sqrt{n}$ steps, each running in $\Theta(m)$ time, hence the time complexity is $\Theta(m\sqrt{n})$ in the worst case.

68

MAXIMUM MATCHING IN BIPARTITE GRAPHS (30)

- In many cases, this complexity can be improved.
- For example, in the case of random sparse bipartite graphs, it has been proved [Bast et al.'06] that the augmenting paths have an average logarithmic length.
- As a consequence, the Hopcroft–Karp algorithm runs only $O(\log n)$ steps so it can be executed in $O(m \log n)$ time.

69

MINIMUM WEIGHT PERFECT MATCHING IN BIPARTITE GRAPHS

70

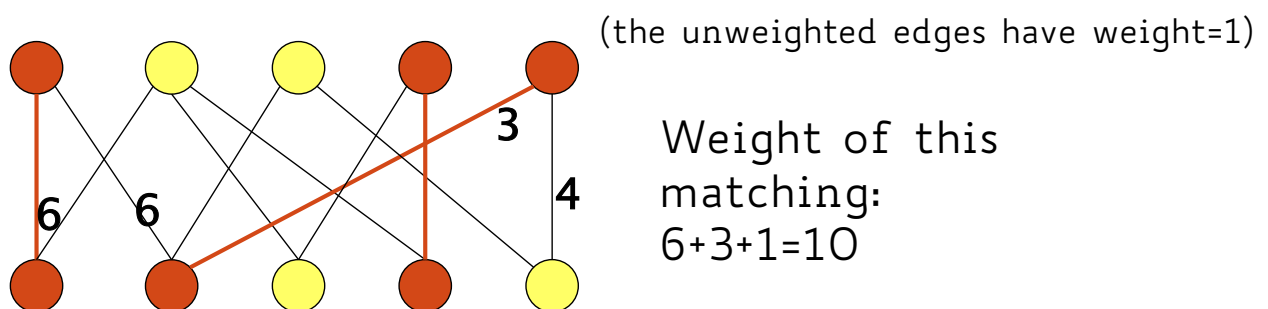


WEIGHTED MATCHING (1)

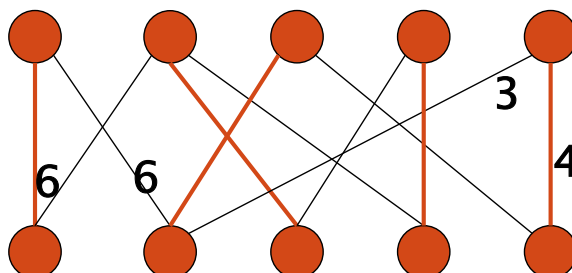
- Each edge has a cost
- The definition of weighted matching is the same as the simple matching (weight does not affect the definition)
- We look for a **minimum weight perfect matching**
- **Note.** This is equivalent to looking for a maximum weight perfect matching, where the weights are all negative.

71

WEIGHTED MATCHING (2)



Max weight matching:
 $6+4+1+1+1=13$

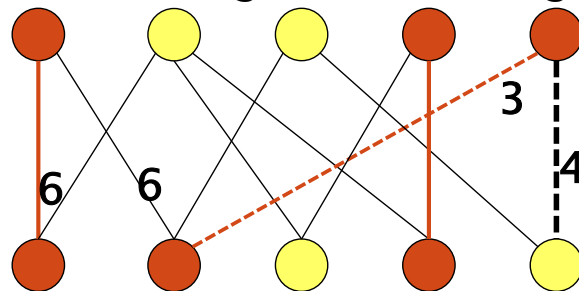


72

WEIGHTED MATCHING (3)

Def. An **augmenting path** (different w.r.t. the previous one!) is any alternating path such that the weight of the edges out of the matching is greater than the weight of the edges in the matching.

Weight of the augmenting path = weight of the edges out of M - weight of the edges in M



Note. In this case, augmenting paths do not need to end at a free node.

73

WEIGHTED MATCHING (4)

Algorithm:

- Start with an empty matching
- Repeat
 - Find an aug. path P with max weight
 - If this weight is positive, swap the role of the edges
 - Else return the found matching (that is, the one of max weight).

- **Complexity:** at least $O(n \cdot m)$.

74

WEIGHTED MATCHING (5)

- It is possible to model the minimum weight perfect matching problem as an ILP problem (**Hungarian method – in honour of Konig and Egevary**):

- Given a matching M , let x be its incidence matrix, where $x_{ij} = 1$ if (i, j) is in M and $x_{ij} = 0$ otherwise.

- The problem can be written as follows:

$$\text{minimize } \sum_{i,j} c_{ij} x_{ij} \quad \text{subject to: } \sum_j x_{ij} = 1, i \in V_1$$

$$\sum_i x_{ij} = 1, j \in V_2$$

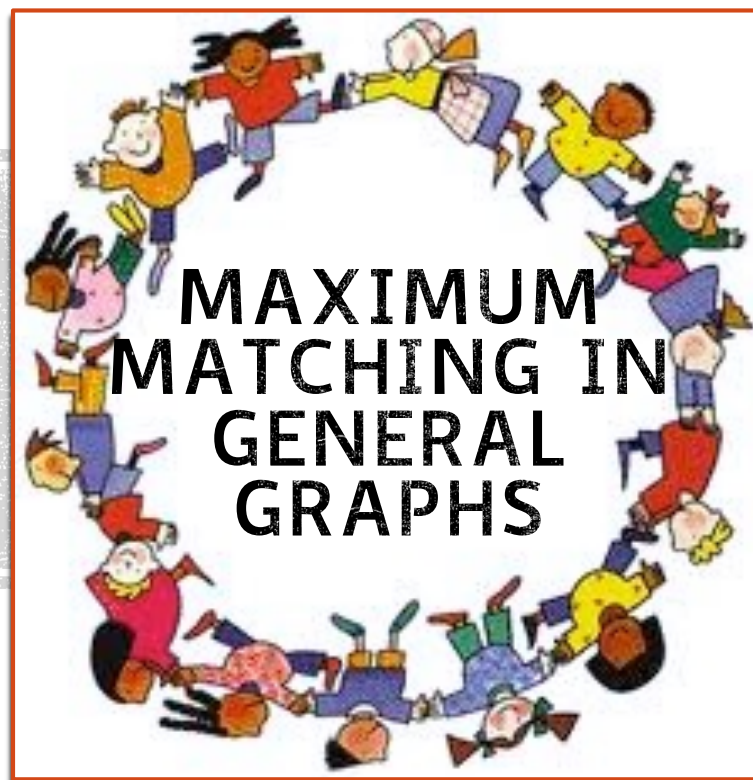
- Complexity: $O(n^3)$.**

$$x_{ij} \geq 0, i \in V_1, j \in V_2$$

$$\forall i \in V_1, \forall j \in V_2, x_{ij} \text{ integer}$$

NOTE: With this definition, the bipartite graph problem is converted into a matrix problem: the rows represent the nodes in V_1 , and the columns represent the nodes in V_2

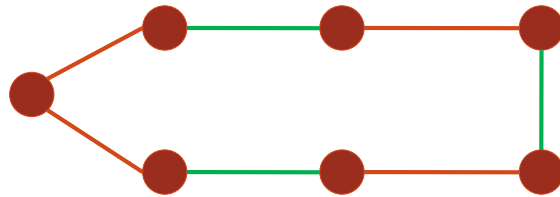
75



76

BLOSSOMS (1)

- We have already noted that the critical point of general graphs are odd cycles containing a maximal number of edges in the matching



- Such cycles are called blossoms

77

BLOSSOMS (2)

Lemma (cycle contraction). Let M be a matching of G , and let B be a blossom. Let B be node disjoint from the rest of M . Let G' be the graph obtained by G contracting B in a single node. Then M' of G' induced by M is maximum in G' iff M is maximum in G .

Proof. M max in $G \Rightarrow M'$ max in G' :

By contradiction. Assume M' not max. Hence, there exists an aug. path P in G' w.r.t. M' .

Let b be the node representing B .

...

78

BLOSSOMS (3)

Proof of the Cycle contraction lemma – contd.

Two cases can hold:

1. P does not cross b \Rightarrow P augmenting for M, too. Contradiction.

Observe that b is free as it represents the node v in B adjacent to two edges out of M. In other words, v is free if we restrict to B.

2. P crosses b \Rightarrow b must be an end-point of P.

Define $P' = P \cup P''$ where P'' is inside B.

P' is augmenting for G. Contradiction.

...

79

BLOSSOMS (4)

Proof of the Cycle contraction lemma – contd.

M' max in $G' \Rightarrow M$ max in G:

By contradiction, M is not max. Let P be an aug. path for M.

Two cases hold:

1. P does not cross b \Rightarrow P is aug. for G' . A contradiction.
2. P crosses b. Since B contains only one free node, at least an end-point of P lies outside B. Let it be w.

Let P' be the sub-path of P joining w with b.

P' is an aug. path for G' . A contradiction. ■

80

MAX MATCHING IN GENERAL GRAPHS (1)

In order to find an aug. path in general graphs, it is “enough” to modify the algorithm on bipartite graphs in order to include blossom search:

- For each found blossom B:
 - B is shrunk in a node, and a new (reduced) graph is generated.
- Each aug. path found in this new graph can be easily “translated” back into an aug. path in G.

Thanks to the previous lemma, if M is max in the new graph, it is max even in G.

81

MAX MATCHING IN GENERAL GRAPHS (2)

This is the Edmonds algorithm [‘65].

The time complexity depends on how blossoms are handled. Varying with the used data structures, it can be either $O(n^3)$ or $O(m n^2)$.

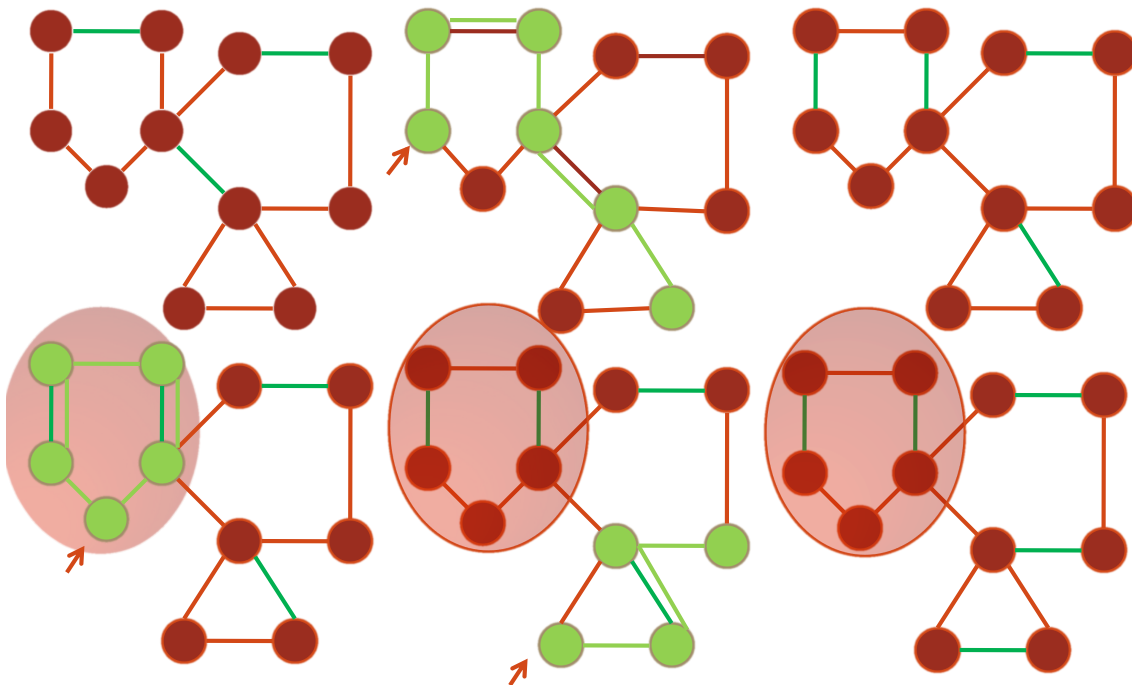
The best-known time complexity is $O(m\sqrt{n})$

[Micali & Vazirani ‘80]

82

MAX MATCHING IN GENERAL GRAPHS (3)

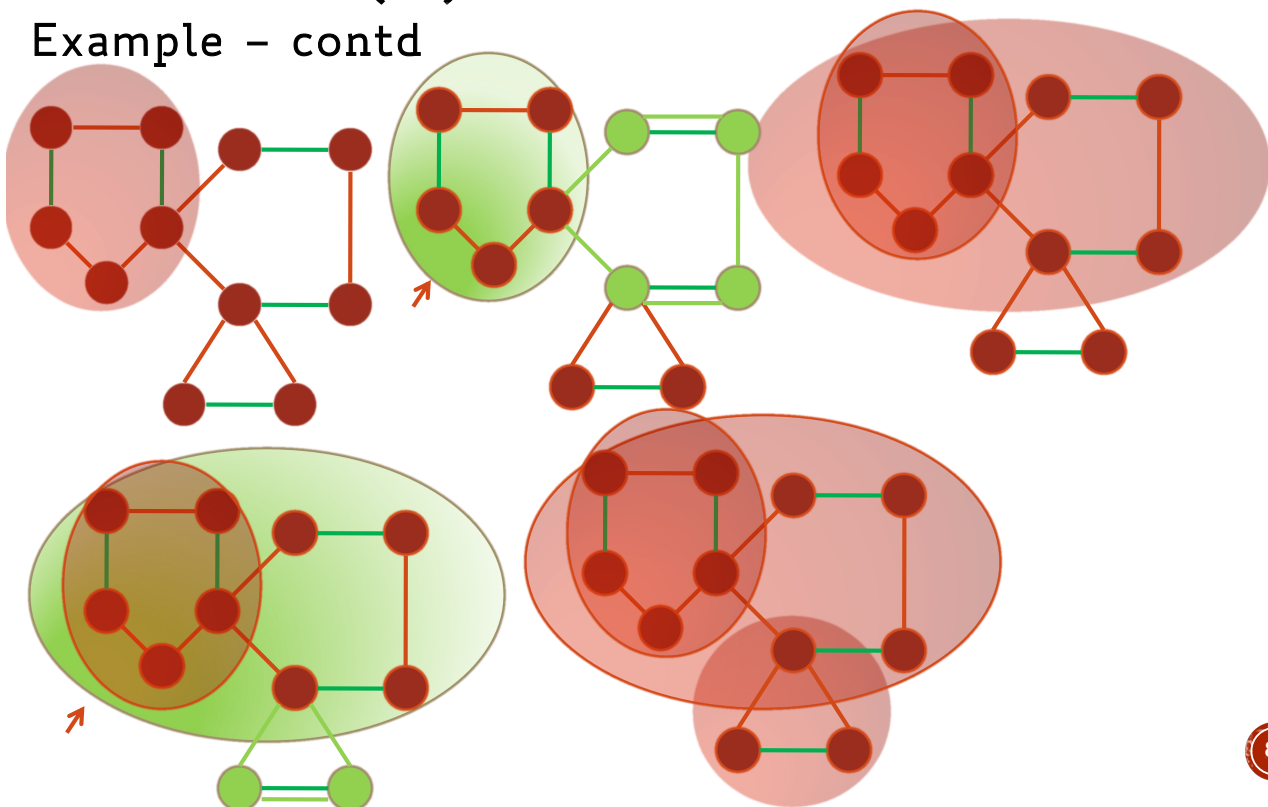
Example:



83

MAX MATCHING IN GENERAL GRAPHS (4)


Example - contd



84

MAX MATCHING IN GENERAL GRAPHS (5)

Edmonds Algorithm ['65]

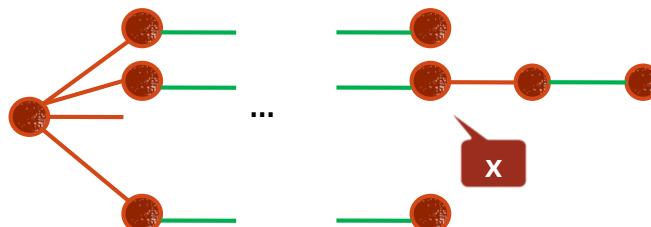
- M matching for G
- L subset of the free nodes (if L empty $\Rightarrow M$ max)
- F forest s.t. each node of L is the root of a tree in F
- Expand F by adding 
- Nodes that are at odd distance from a node of L have degree 2 (1 in M and 1 in $E \setminus M$): we call them **internal nodes**
- The other nodes: **external nodes**
- ...

85

MAX MATCHING IN GENERAL GRAPHS (6)

Edmonds algorithm - contd

- Consider the neighbors of the external nodes.
- 4 possibilities hold:
 1. There exists x external and incident to a node y not in F :
add to F edges (x,y) and (y,z) , and (y,z) is in M .

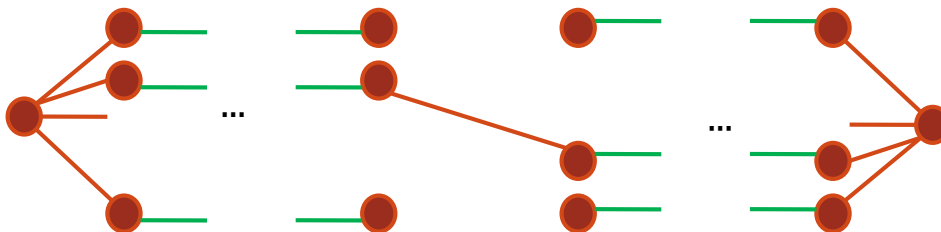


86

MAX MATCHING IN GENERAL GRAPHS (7)

Edmonds algorithm – contd

- Two external nodes lying in two different components of F are adjacent:
augmenting path

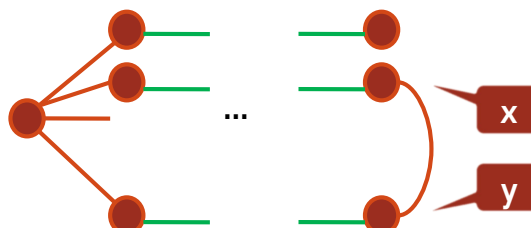


87

MAX MATCHING IN GENERAL GRAPHS (8)

Edmonds algorithm – contd

- Two external nodes x, y in the same component in F are adjacent:
let C be the found cycle. It is possible to move the edges in M around C so that the cycle contraction lemma can be used \Rightarrow reduced graph G'



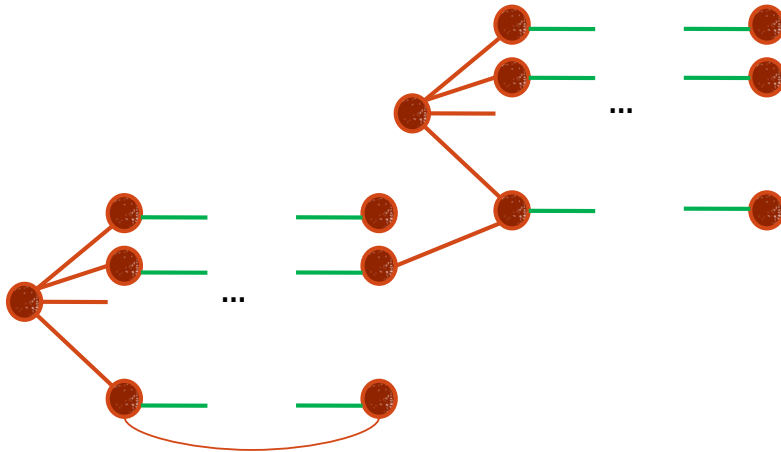
88

MAX MATCHING IN GENERAL GRAPHS (9)

Edmonds algorithm – contd

4. All the external nodes are adjacent to internal nodes:

M is maximum.



89

MAX MATCHING IN GENERAL GRAPHS (10)

Lemma. At each step of the Edmonds algorithm, either the dimension of F increases, or the dimension of G decreases, or an aug. path is found, or M is maximum.

Complexity. Number of iterations \leq

num. of times F is increased (at most n)+

num. of times a blossom is shrunk (at most n)+

num. of found aug. paths (at most $n/2$).

The time complexity depends on how blossoms are handled. Varying with the used data structures, it can be either $O(n^3)$ or $O(mn^2)$.

Best known time complexity: $O(m\sqrt{n})$ [Micali & Vazirani '80]

90