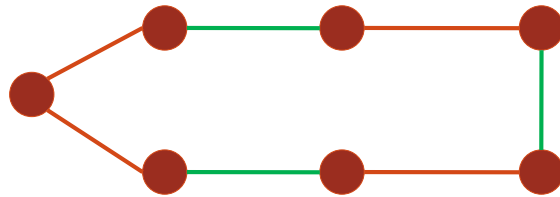# BLOSSOMS (1)

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

   o Such cycles are called blossoms

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 $\implies$ 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.

…

BLOSSOMS (3)
Proof of the Cycle contraction lemma – contd.

Two cases can hold:

1. <u>P does not cross b</u> $\Rightarrow$ 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. <u>P crosses b</u> $\Rightarrow$ b must be an end-point of P.

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

P' is augmenting for G. Contradiction.

…

BLOSSOMS (4)
Proof of the Cycle contraction lemma – contd.

<u>M' max in G' $\Rightarrow$ M max in G</u>:

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. ■

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.

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 $O(m\,n^2)$.
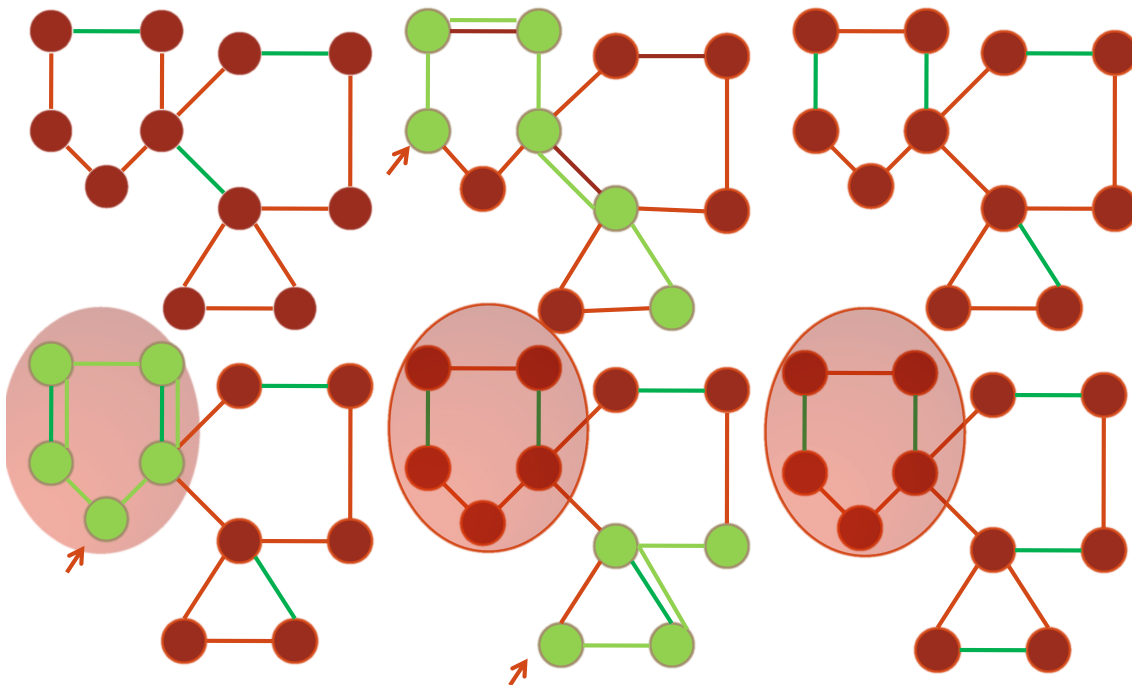
The best-known time complexity is $O(m\sqrt{n})$

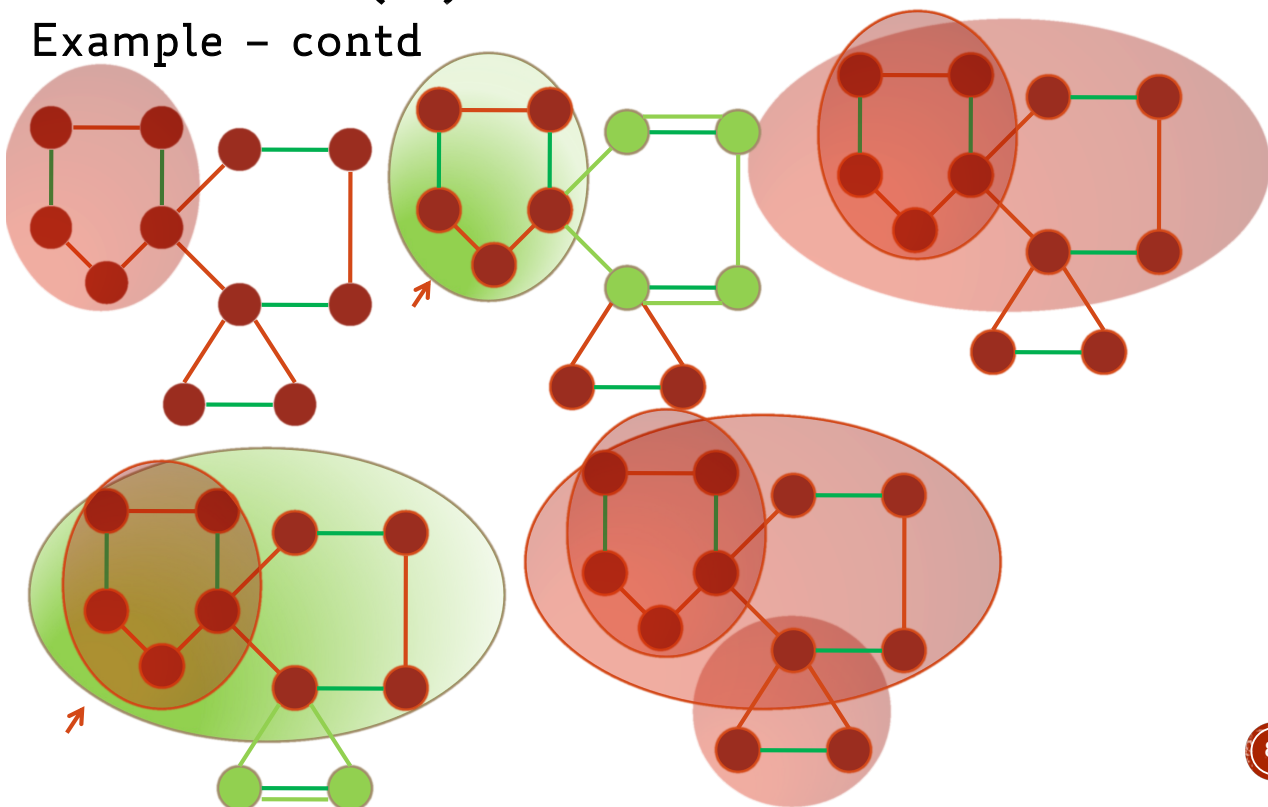[Micali & Vazirani '80]

MAX MATCHING IN GENERAL GRAPHS (3)

Example:

MAX MATCHING IN GENERAL GRAPHS (4)

Example – contd

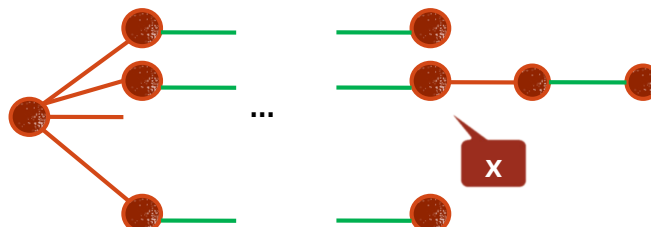# MAX MATCHING IN GENERAL GRAPHS (5)

Edmonds Algorithm ['65]

- M matching for G

- L subset of the free nodes (if L empty => 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\M): we call them internal nodes

- The other nodes: external nodes

- ...

MAX MATCHING IN GENERAL GRAPHS (6)

Edmonds algorithm – contd

- Consider the neighbors of the external nodes.

- 4 possibilities hold:

  1. There esists x external and incident to a node y not in F:
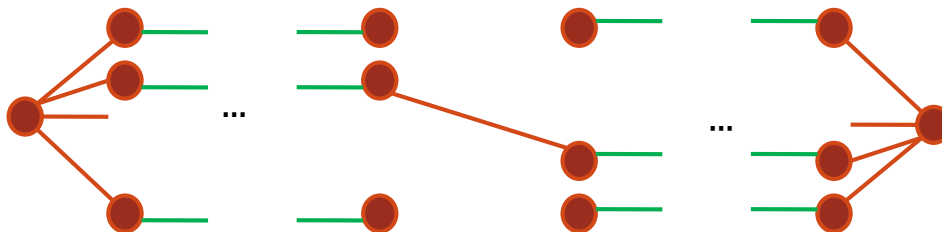
     add to F edges (x,y) and (y,z), and (y,z) is in M.

MAX MATCHING IN GENERAL GRAPHS (7)

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

MAX MATCHING IN GENERAL GRAPHS (8)
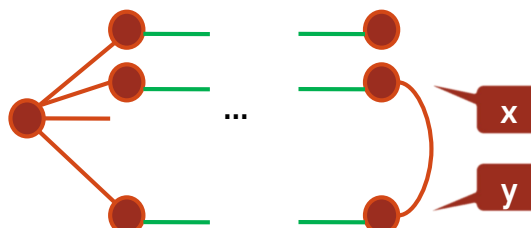
3. Two external nodes x, y in the same component in F are adjacent:

   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 => reduced graph G'

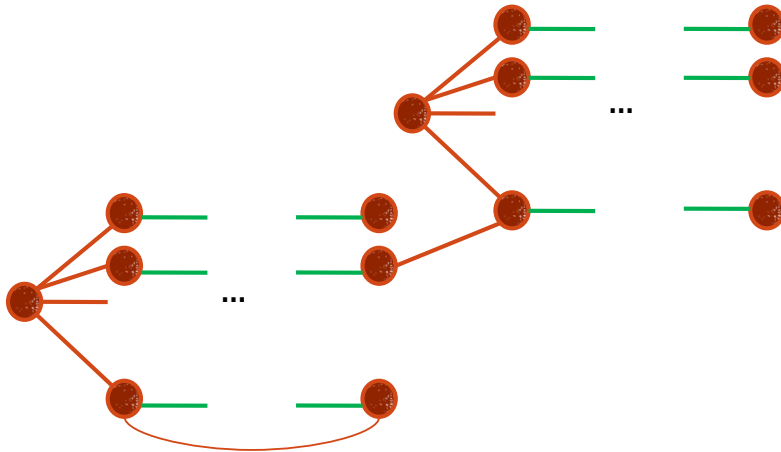# MAX MATCHING IN GENERAL GRAPHS (9)

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

    M is maximum.

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 ≤

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

num. of times a blossom is shrinked (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]

ANOTHER APPLICATION

SWITCH BUFFER (1)

Reminder:

- Interconnection topologies are constituted by layers of basic modules that are 2x2 cross-bar switches

- Any output can be reached by any input by properly setting some switches

- A single routing can be easily performed if the network is self-routing (e.g., Butterfly, Baseline, etc.)

SWITCH BUFFER (2)

- The log N-stage networks are not rearrangeable, i.e. not all routes can be done simultaneously

- Two packets may want to use the same link at the same time

- Solution: buffering (though buffers increase delay)

MULTISTAGE TOPOLOGIES WITH BUFFERS (1)

The multistage topologies are good to use, because they are:

- modular

- scalar

Nevertheless, the buffers at each node provoke:

- delays for going through the stages

- decreased throughput due to internal blocking

Solution: (input) buffers that are external to the topology

MULTISTAGE TOPOLOGIES WITH BUFFERS (2)

- Head of line (HOL) buffer: only the first packet can leave the buffer.

- Buffers are connected through a crossbar network to the inputs of the topology

- During each slot, the scheduler establishes the crossbar connections to transfer packets from the buffers to the inputs

Crossbar switch

Scheduler

X = connect
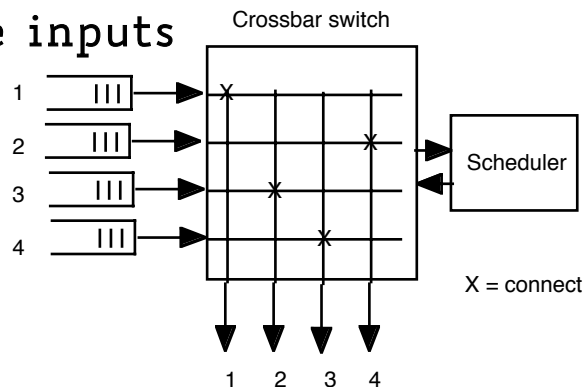
95

MULTISTAGE TOPOLOGIES WITH BUFFERS (3)

- When the packets at the head of two or more input queues are destined to the same input node, only one can be transferred, and the other is blocked

- This behavior limits throughput because some inputs (and consequently outputs) are kept idle during a slot even when they have other packets to send

- …

96

MULTISTAGE TOPOLOGIES WITH BUFFERS (4)

- If the inputs are allowed to transfer packets that are not at the head of their buffers, throughput can be improved

- Example:

input 1	1	2
input 2	3	2
input 3	4	3
input 4	4	2
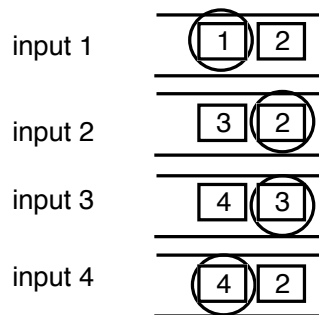
- How does the scheduler decide which input to transfer to the network?

MULTISTAGE TOPOLOGIES WITH BUFFERS (5)

Backlog matrix:

- rows: input buffers

- columns: outputs

- each entry (i,j) represents the number of packets in buffer i destined to output j

input \ output	1	2	3
1	3	3	0
2	2	0	0
3	0	0	2

MULTISTAGE TOPOLOGIES WITH BUFFERS (6)

- During each slot, the scheduler can transfer at most one packet from each buffer to each output

- The scheduler must choose at most one packet from each row and from each column of the backlog matrix

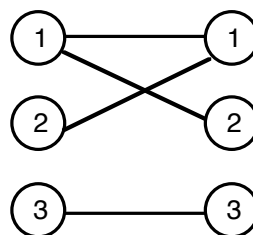- This can be done by solving a bipartite matching algorithm...

MULTISTAGE TOPOLOGIES WITH BUFFERS (7)

- The bipartite graph G=(V ∪ W, E) is built as follows:

- V: N nodes representing the buffers

- W: N nodes representing the outputs

- E: there is an edge from a buffer i to an output j iff there is a packet in the backlog matrix to be transferred from i to j.

- Example:

1	3	3	0
2	2	0	0
3	0	0	2

- Finding a maximum matching is equivalent to finding the largest set of packets that can be transferred simultaneously

MULTISTAGE TOPOLOGIES WITH BUFFERS (8)

- Finding a maximum matching during each time slot does not eliminate the effects of HOL blocking

- It is, indeed, necessary to look beyond a single slot when making scheduling decisions

- Solution: edge (i,j) is assigned a weight equal to the value of element (i,j) of the backlog matrix

- Theorem: A scheduler that chooses, during each time slot, the maximum weighted matching achieves full utilization.

- Proof and other details: see [McKeon et al. 1999]