SECOND PART: WIRELESS NETWORKS 2.C. MOBILE SENSOR NETWORKS

MOBILE SENSOR NETWORKS (1)

- Wireless sensor networks are large-scale wireless multi-hop networks where nodes have limited resources such as energy, bandwidth, storage, and processing power.
- They are dedicated to provide low-level surveillance and data gathering services in an area of interest (AoI) for various applications.
- So, they must fully cover the AoI without internal sensing holes.
- It is critical to position sensors inside the AoI, due to operational factors such as human inaccessibility to the AoI and tight deployment budget.

Sensor mobility

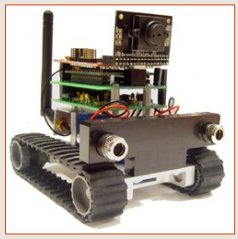
MOBILE SENSOR NETWORKS (2)

There are two different ways to place sensors by exploiting mobility:

1. Use robots, that carry static sensors as payload and move around in the AoI. While traveling, they deploy sensors at proper positions (e.g., vertices of certain geographic graph): carrier-based method.
2. Sensors autonomously and intelligently change their geographic location, adjusting the overall distribution to a desired one: self-deployment method.

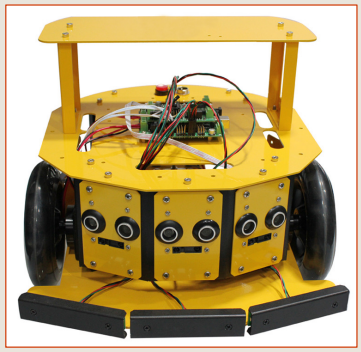
MOBILE SENSORS (1)

Devices of tiny dimension and low cost able to produce measurable response to a change in a physical condition.



They are equipped with:

- detecting/monitoring Unit (sensing)
- communication Unit
- computing Unit
- small battery
- motion system



MOBILE SENSORS (2)

Mobile sensors collaborate to form an ad-hoc network and are especially useful in critical environments (e.g., in presence of dispersion of pollutants, gas plumes, fires, …)

It is not restrictive to assume each sensor able to monitor a disk centered at its position and having radius r_s = sensing range and to communicate with the sensors that are inside a disk centered at its position and having radius r_c = communication range.

MOBILE SENSORS (3)

The sensing and the communication units are independent components; therefore, the communication and the sensing ranges are not directly associated from a hardware point of view. However, they are integrated because connectivity and coverage must both be guaranteed: it can be proved that the protocols working in the assumption that r_c is at least twice r_s only need to guarantee coverage and will satisfy the connectivity constraint as well.

COVERAGE ISSUES (1)

Coverage is one of the most important issues for environmental monitoring in wireless sensor networks.

In general, coverage is defined as the measurement of the quality of surveillance of sensing function.

COVERAGE ISSUES (2)

Information from a target field is collected by deploying sensor nodes in different locations.

After deployment, sensor nodes form a network through which the collected data are propagated to a sink.

The quality of the collected information depends on how well the AoI is covered by the set of sensor nodes.

COVERAGE ISSUES (3)

Depending on how an area should be covered, coverage problems are broadly categorized in three types:

1. point coverage: a set of discrete points is continuously monitored
   Example: in building monitoring, every access is controlled
2. area coverage: all points within a bounded region are continuously monitored
   Example: in forest monitoring, every location of the forest must be covered by at least one sensor node in order to immediately detect any unusual activities like forest fire, activities of poachers, etc.
3. …

COVERAGE ISSUES (4)

2. …
3. barrier coverage: a specified path or the boundary of a region is continuously monitored by sensor nodes.
   Example: in forest monitoring, covering the boundary allows controlling and elimination of the poaching activities, and illegal entry through the boundary

A continuous monitoring with static sensors is required for all the aforementioned types of coverage problems.

COVERAGE ISSUES (5)

There are typical applications where only periodic patrol inspections are sufficient for a certain set of points of interest instead of continuous monitoring like in traditional coverage: sweep coverage.

COVERAGE ISSUES (6)

Sweep coverage concept in WSNs is introduced in the literature in [Cheng et al.'08]:

Def. A point of interest (PoI) is said to be t-sweep covered if and only if at least one mobile sensor visits the point within every t time periods, where t is called sweep period of the point.

Objective of the sweep coverage problem is to find minimum number of mobile sensors to guarantee sweep coverage for the set of PoIs.

COVERAGE ISSUES (7)

Instead of using only mobile sensors, the use of both static and mobile sensors can be more effective in terms of total number of sensor used [Gorain & Mandal '14].

It is possible to introduce an energy efficient sweep coverage problem whose objective is to guarantee sweep coverage by a combined set of mobile and static sensors such that total energy consumption is minimized [Gorain & Mandal '15].

☞ some possible students' lessons in this context

LOCALIZATION TECHNIQUES (1)

- The location information of the sensors is essential as deployment protocols often depend on the position of sensors.
- for outdoor applications, GPS is the most popular solution.
- …

LOCALIZATION TECHNIQUES (2)

- …
- for indoor centralized applications, a grid-based approach is often used when global position information is needed for deployment: grids are used as landmarks where sensors are placed; sometimes, the deployed sensors are used themselves as landmarks.
- for the indoor distributed approach, techniques based on the received signal strength, time difference of arrival of two different signals, angle arrival, etc. can be also used.

THE CENTRALIZED DEPLOYMENT OF MOBILE SENSORS I.E. THE MINIMUM WEIGHT PERFECT MATCHING ON BIPARTITE GRAPHS

Prof. Tiziana Calamoneri
Network Algorithms
A.y. 2025/26

THE DEPLOYMENT PROBLEM (1)

In the context of area coverage:

Deployment problem – two approaches:

1. Given an AoI to cover, the aim is to entireley cover the AoI minimizing the number of used sensors.
2. Given a set of mobile sensors, the aim is to maximize the covered area.

Coverage multiplicity: either single or multiple, according to the specific application requirements

THE DEPLOYMENT PROBLEM (2)

Coordination algorithm

Initial Config. → Desired Config.

Can be:
- random
- from a safe location

Can be:
- a regular tassellation
- any configuration, provided that the AoI is covered

- At the same time, some parameters need to be optimized:

THE DEPLOYMENT PROBLEM (3)

- Traversed Distance: it is the dominant cost
- Number of starts/stops: they are more expensive than a continuous movement
- Communication cost: it depends on the number of exchanged messages and on the packet dimension at each transmission
- Computation cost: Usually negligible, unless processors are extremely sophisticated

THE DEPLOYMENT PROBLEM (4)

Random deployment is the easiest way to place sensors. When the target region is subject to severe changes in condition or no a priori knowledge is available, it achieves a relatively satisfactory coverage.
It is also practical in military application, where WSNs are initially established by dropping or throwing.

THE DEPLOYMENT PROBLEM (5)

Nevertheless, random deployment may not provide a uniform distribution, which is desirable for a longer system lifetime of the AoI.

So the random deployment may serve as an initial phase of another deployment strategy.

THE DEPLOYMENT PROBLEM (6)

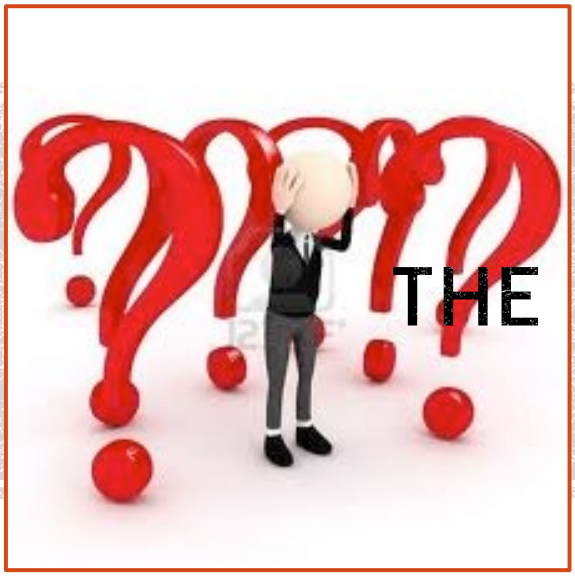
Self-deployment:

- either centralized (or global)
- or distributed (or local).

The global approach relies on global information which is usually not scalable.

Local algorithms are based on iterative nearest neighbor exchange.

We will discuss both the approaches.



THE PROBLEM

THE CENTRALIZED DEPLOYMENT PROBLEM (1)

If no information on the AoI is given, incremental deployment:

- It is a one-at-a-time approach: each node makes use of information gathered by the previously deployed nodes to determine its ideal deployment location, which is calculated at a powerful base station.
- …

THE CENTRALIZED DEPLOYMENT PROBLEM (2)

incremental deployment (contd)

- …
- Each deployed node is responsible for communicating its local information back to the base station for being used in the next iteration. Hence each node has to maintain bidirectional communication with the sink.
- No localization technique is needed: the location of the next node can be found by using the nodes themselves as landmarks.

THE CENTRALIZED DEPLOYMENT PROBLEM (3)

incremental deployment (contd)

PROs:

- certainty of optimal location in each step
- since the sensors are fixed once they are deployed, little energy is consumed

CONs:

- the deployment time is very long, which can significantly increase the network initialization time
- computationally expensive: a lot of work for the computation of a new location

THE CENTRALIZED DEPLOYMENT PROBLEM (4)

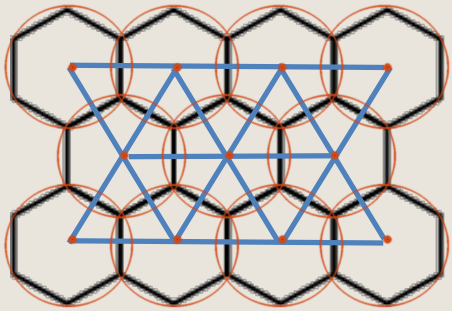
If information on the AoI is known:

The whole coverage is guaranteed assigning to each sensor a position on a grid covering the AoI

- The total energy consumption should be minimized

THE CENTRALIZED DEPLOYMENT PROBLEM (5)

It is well known that an optimal coverage using equally sized circles is the one positioning the centers on the vertices of a triangular grid opportunely sized.



- We model this problem with the classical minimum weight perfect matching in bipartite graphs.

THE GRAPH MODEL

29

THE GRAPH MODEL (1)

- Set of n mobile sensors $S=\{S_1, S_2, \ldots, S_n\}$
- Set of p locations on the AoI $L=\{L_1, L_2, \ldots, L_p\}$ (e.g., in correspondence of a hexagonal grid)
- $n \geq p$ (to guarantee the complete coverage)
- For each S_i, determine the location L_j that S_i will have to reach, so to minimize the total consumed energy.

THE GRAPH MODEL (2)

Define a weighted complete bipartite graph $G=(S \cup L, E, w)$ as follows:

- One node for each sensor S_i
- One node for each location L_j
- An edge between S_i and L_j for each i=1…n and j=1…p
- For each edge e_{ij} , $w(e_{ij})$ is proportional to the energy consumed by S_i to reach location L_j
- The aim is to choose a matching between sensors and locations so that the total consumed energy is minimized.

THE PERFECT MATCHING ON BIPARTITE GRAPHS

MATCHING (1)

Given G=(V, E):

- **Def.** A matching is a set of edges $M \subseteq E$ such that every node is adjacent to at most one edge in M.
- Maximal matching

  There exists no $e \notin M$ such that $M \cup \{e\}$ is a matching
- Maximum matching

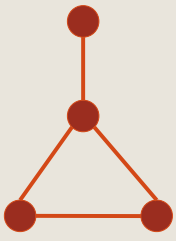
  Matching M such that |M| is maximum
- Perfect matching

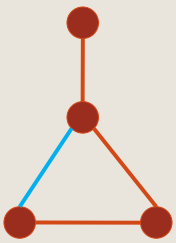
  Assuming n even,|M| = n/2: each node is adjacent to exactly one edge in M.

  If G is bipartite and $V=V_1 \cup V_2$, $|M|=\min\{|V_1|, |V_2|\}$.

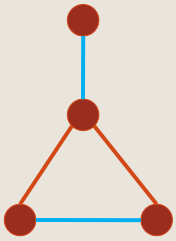
MATCHING (2)

Example:





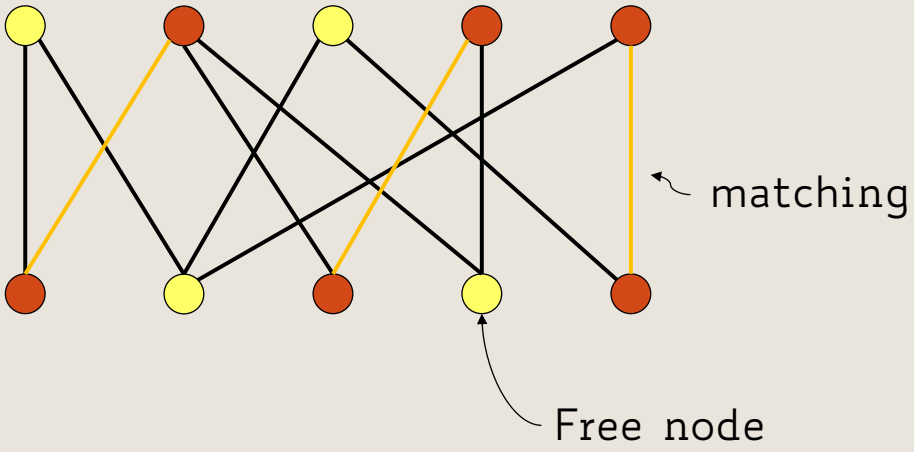
Maximal matching



Maximum matching

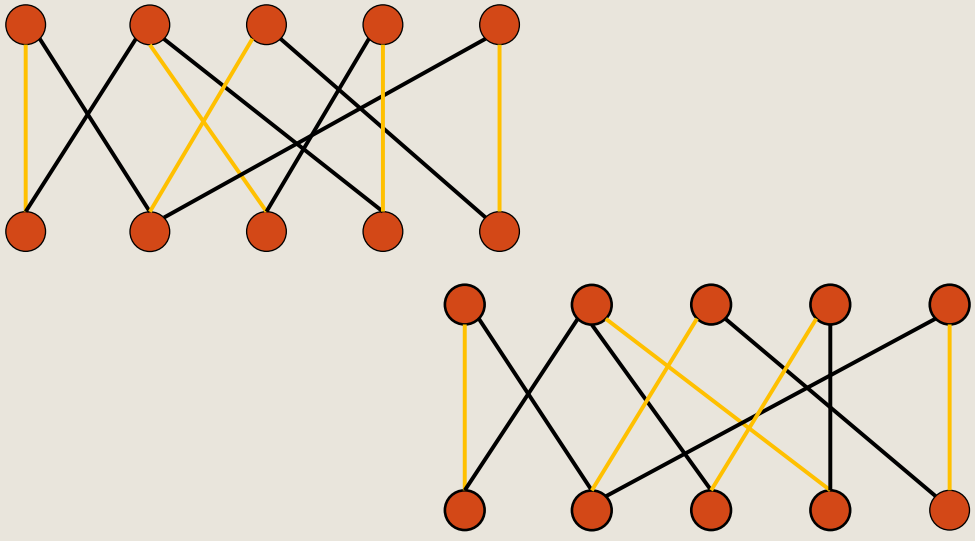
MATCHING (3)

- Nomenclature



MATCHING (4)

- Note. The maximum matching is not unique



MATCHING (5)

Original problem: **wedding problem**

- the nodes of a set are men
- the nodes of the other set are wemen
- An edge connects a man and a woman who like each other

- Maximum matching aims at maximizing the number of couples.

MATCHING PROBLEMS

- Given a graph G, to find a:
  - Maximal matching is easy (greedy)
  - Maximum matching is
    - polynomial; not trivial.
    - easier in the case of bipartite graphs
  - Perfect matching
    - it is a special case of the maximum matching
    - for it, some theorems can help

MAXIMUM MATCHING IN BIPARTITE GRAPHS (1)

TH. (P. Hall's marriage theorem) Given a bipartite graph G with $|V_1| \leq |V_2|$, G has a perfect matching iff for each set S of k nodes in V_1 there are at least k nodes in V_2 adjacent to some node in S.

In symbols, $\forall\ S \subseteq V_1$, $|S| \leq |adj(S)|$.

PROOF.

Necessary condition: If G has a perfect matching M and S is any subset of V_1, each node in S is matched through M with a different node in adj(S). Hence $|S| \leq |adj(S)|$.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (2)

(proof of the Hall theorem - contd) G bipartite with $|V_1| \leq |V_2|$, G has a perfect matching iff $\forall\ S \subseteq V_1$, $|S| \leq |adj(S)|$.

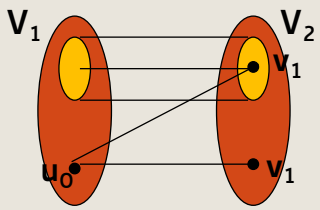
sufficient condition: We have to prove that if the Hall condition is true then there is a perfect matching. By contradiction, assume that M is a maximum matching but $|M| < |V_1|$.

By hypothesis, $|M| < |V_1| \Rightarrow \exists\ u_0 \in V_1$ s.t. $u_0 \notin M$.

Let $S=\{u_0\}$; it holds $1=|S| \leq |adj(S)|$ from the Hall condition, so there exists a $v_1 \in V_2$ adjacent to u_0.

a. $v_1 \notin M$

b. $v_1 \in M$



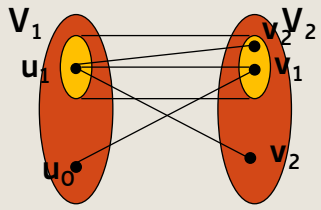
MAXIMUM MATCHING IN BIPARTITE GRAPHS (3)

(proof of the Hall theorem - contd) G bipartite with $|V_1| \leq |V_2|$, G has a perfect matching iff $\forall\ S \subseteq V_1$, $|S| \leq |adj(S)|$.

a. If $v_1 \notin M$ OK
b. Consider the node matched with v_1 through M, call it u_1.

$S=\{u_0,u_1\}$ and $2= |S| \leq |adj(S)|$.
There exists another node v_2,
Different from v_1, and adjacent either to u_0 or to u_1.

a. $v_2 \notin M$
b. $v_2 \in M$

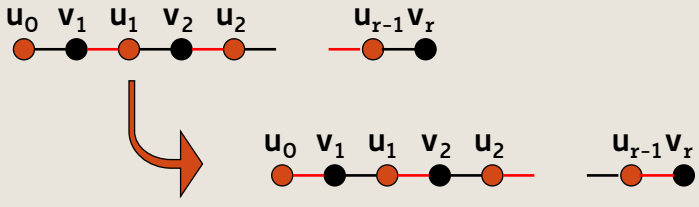


MAXIMUM MATCHING IN BIPARTITE GRAPHS (4)

(proof of the Hall theorem - contd) G bipartite with $|V_1| \leq |V_2|$, G has a perfect matching iff $\forall\ S \subseteq V_1$, $|S| \leq |adj(S)|$.

Continue in this way. As G is finite, we will eventually reach a node v_r that is free w.r.t. M. Each v_i is adjacent to at least one among $u_0,u_1,\ldots,u_{i-1}$.

Analogously to the case r=2:



MAXIMUM MATCHING IN BIPARTITE GRAPHS (5)

- The P. Hall Theorem does not provide an algorithmic method to construct a perfect matching (unless all subsets in V_1 are enumerated – exponential time issue).
- The perfect matching problem in a bipartite graph is equivalent to the maximum flow problem in a network:

  Given $G=(V=V_1 \cup V_2, E)$, construct the associated flow network $G'=(V', E')$ as follows:

  …

MAXIMUM MATCHING IN BIPARTITE GRAPHS (6)

V': $V \cup \{s\} \cup \{t\}$

E': From the source s to all nodes in V_1 :

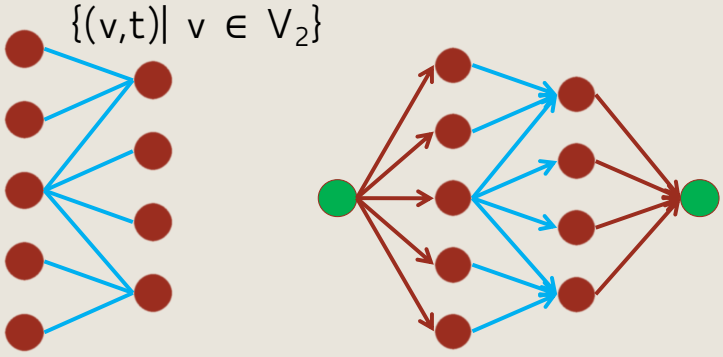
$\{(s,u) | u \in V_1\} \cup$

All edges in E:

$\{(u,v) | u \in V_1, v \in V_2, e\ (u,v) \in E\} \cup$

From all nodes in V_2 to the tale t:

$\{(v,t) | v \in V_2\}$

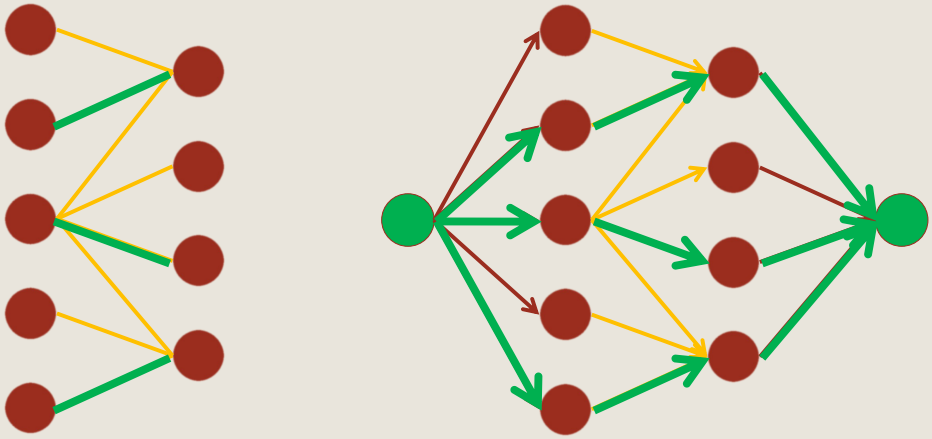


Capacity: $c(u,v) = 1$, for all $(u,v) \in E'$

MAXIMUM MATCHING IN BIPARTITE GRAPHS (7)

Fact: Let M be a matching in a bipartite graph G. There exists a flow f in the network G' s.t. |M|=|f|.

Vice-versa, if f is a flow of G', there exists a matching M in G s.t. |M|=|f|.



MAXIMUM MATCHING IN BIPARTITE GRAPHS (8)

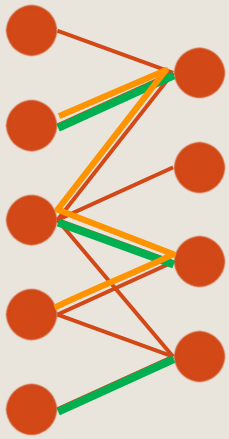
- **Th.**: (integrality) If the capacity c assumes only integer values, the max flow f is such that |f| is integer. Moreover, for all nodes u and v, f(u,v) is integer.
- **Corol.**: The cardinality of a max matching M in a bipartite graph G is equal to the value of the max flow f in the associated network G'.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (9)

- The algorithm by Ford-Fulkerson for the max flow in a network runs in O(m|f|) time.
- In our special setting, the max flow of G' has cardinality upper bounded by $\min\{|V_1|, |V_2|\}$.
- Hence, the complexity of an algorithm for the max matching exploiting the max flow runs in O(n m) time.

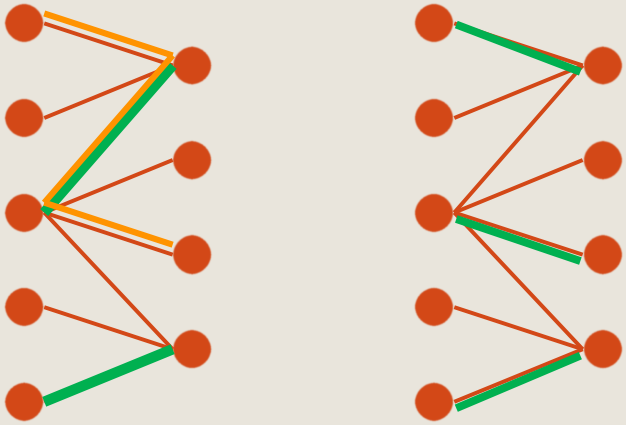
MAXIMUM MATCHING IN BIPARTITE GRAPHS (10)

- **Def.** Given a matching M in a graph G, an alternating path w.r.t. M is the path alternating edges of M and edges in E\M.



MAXIMUM MATCHING IN BIPARTITE GRAPHS (11)

- **Def.** Given a matching M in a graph G, an augmenting path w.r.t. M is an alternating path starting and finishing in two free nodes w.r.t. M.



Swapping the role of the edges in M and in E\M,M has larger cardinality.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (12)

- **Th.** (Augmenting path) [Berge 1975] M is a max matching iff there are no augmenting paths w.r.t. M.
- Proof.
- (➔) If M max, then there are no augmenting paths.

  Negating, if there are some augmenting paths, then M is not max. This is obvious because we can swap the role of the edges in the augmenting path and increase the cardinality of M.
- …

MAXIMUM MATCHING IN BIPARTITE GRAPHS (13)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

- (🡐) If there are no augmenting paths, then M is max.

  By contradiction, M is not max. Let M' s.t. |M'|>|M|.

  Consider graph H induced by M and M'. Edges that are both in M and in M' are put twice. So, H is a multigraph.

- …

MAXIMUM MATCHING IN BIPARTITE GRAPHS (14)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

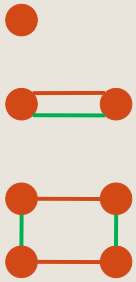
- H has the following property:
  - For each v in H, deg(v)≤2. (indeed, each node has at most one edge from M and one edge from M')
- So, each connected component of H is either a cycle or a path.
  - Cycles necessarily have even length; otherwise, a node would be incident to two edges of the same matching (M or M'); this is absurd by the definition of matching.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (15)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

- More in detail, the connected components of H can be classified into 6 kinds:

1. An isolated node
2. a 2-cycle
3. a 2k-cycle, k>1



…

MAXIMUM MATCHING IN BIPARTITE GRAPHS (16)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

…

4. a 2k-path

5. a (2k+1)-path whose extremes are incident to M

6. a (2k+1)-path whose extremes are incident to M'

MAXIMUM MATCHING IN BIPARTITE GRAPHS (17)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

- Reminder: |M|<|M'| by hp.
- Among all the components just defined, only 5 and 6 have a different number of edges from M and from M'; only 6 has more edges from M' than from M.
- So, there is at least one component of kind 6

- This comp. is an augmenting path w.r.t. M: contradiction. ■

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.

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 dipends on the complexity of finding an augmenting path.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (20)

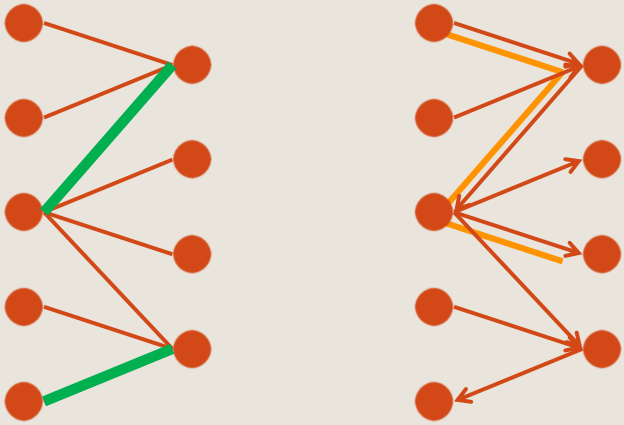
Question: how to decide the existence of an augmenting path and how to find one, if one exists?

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.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (21)



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.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (22)

- 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.

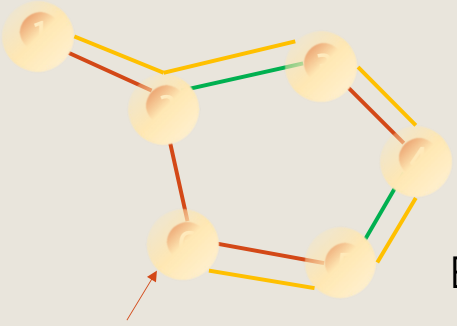
Complexity: O(n+m)

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

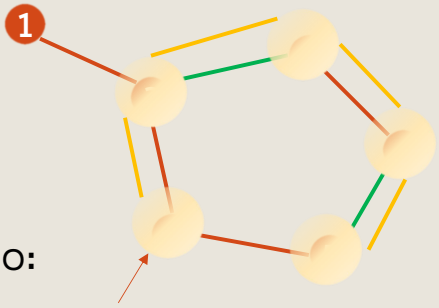
MAXIMUM MATCHING IN BIPARTITE GRAPHS (23)

(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\M
  - As soon as a free node has been encountered, a new augmenting path has been found

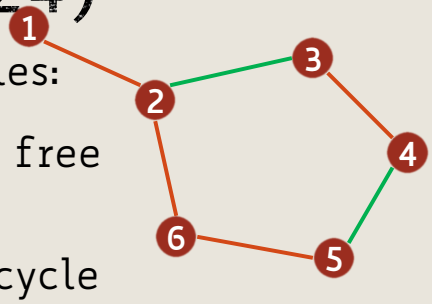


But also:



MAXIMUM MATCHING IN BIPARTITE GRAPHS (24)

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…

MAXIMUM MATCHING IN BIPARTITE GRAPHS (25)

- 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.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (26)

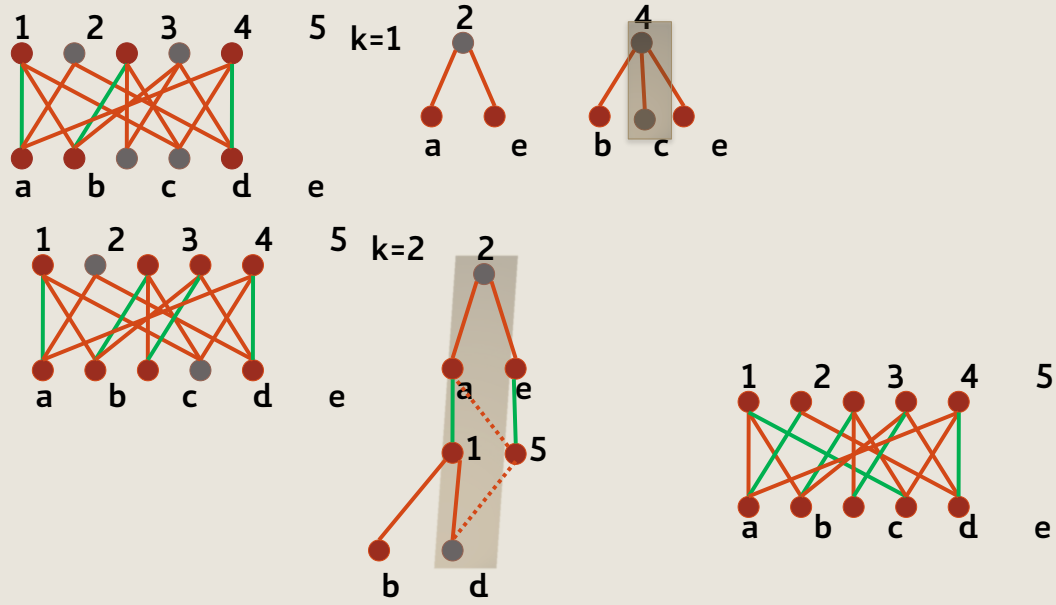
Hopcroft-Karp Algorithm

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.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (27)

Example: Hopcroft-Karp algorithm



MAXIMUM MATCHING IN BIPARTITE GRAPHS (28)

Analysis of the Hopcroft-Karp algorithm (sketch)

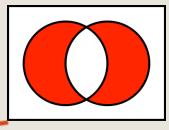
Each step consists in a BFS and a DFS. Hence it runs in $\Theta(n+m) = \Theta(m)$ time.

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.
- ...

MAXIMUM MATCHING IN BIPARTITE GRAPHS (29)

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.
- …

MAXIMUM MATCHING IN BIPARTITE GRAPHS (30)

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

- …
- Each step of the algorithm augments the dimension of M by one, so at most $\sqrt{n}$ furhter 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.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (31)

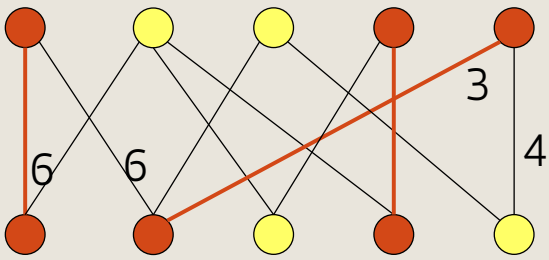
- In many cases, this complexity can be improved.
- For example, in the case of random sparse bipartite graphs, it has been proved [Bast et al.'06] that the augmenting paths have an average logarithmic length.
- As a consequence, the Hopcroft–Karp algorithm runs only $O(\log n)$ steps so it can be executed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

MINIMUM WEIGHT PERFECT MATCHING IN BIPARTITE GRAPHS

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.

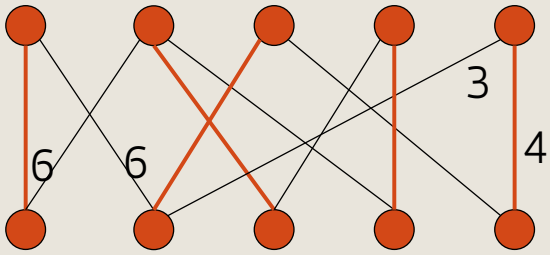
WEIGHTED MATCHING (2)



(the unweighted edges have weight=1)

Weight of this matching:
6+3+1=10

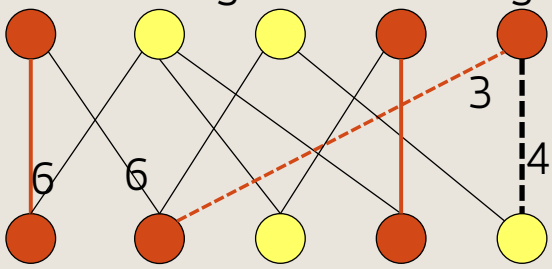
Max weight matching:
6+4+1+1+1=13



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 start and end with edges outside M.

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 of max weight).

- Time complexity: at least O(n m).

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:

$$\text{minimize} \sum_{i,j} c_{ij}x_{ij} \quad \text{subject to:}$$

$$\sum_{j} x_{ij} = 1, i \in V_1$$

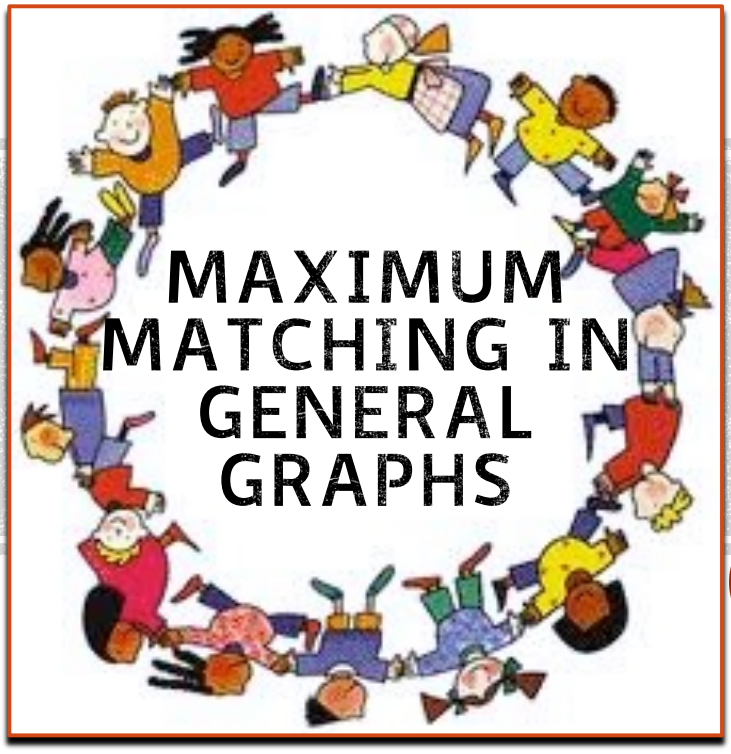
$$\sum_{i} x_{ij} = 1, j \in V_2$$

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

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

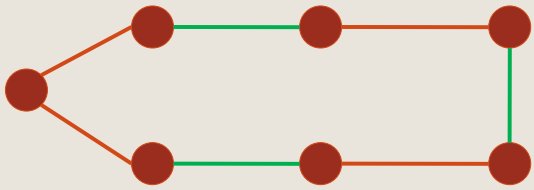
Complexity: $O(n^3)$.

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



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.

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 ⟹ 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. 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.

…

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. ■

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].

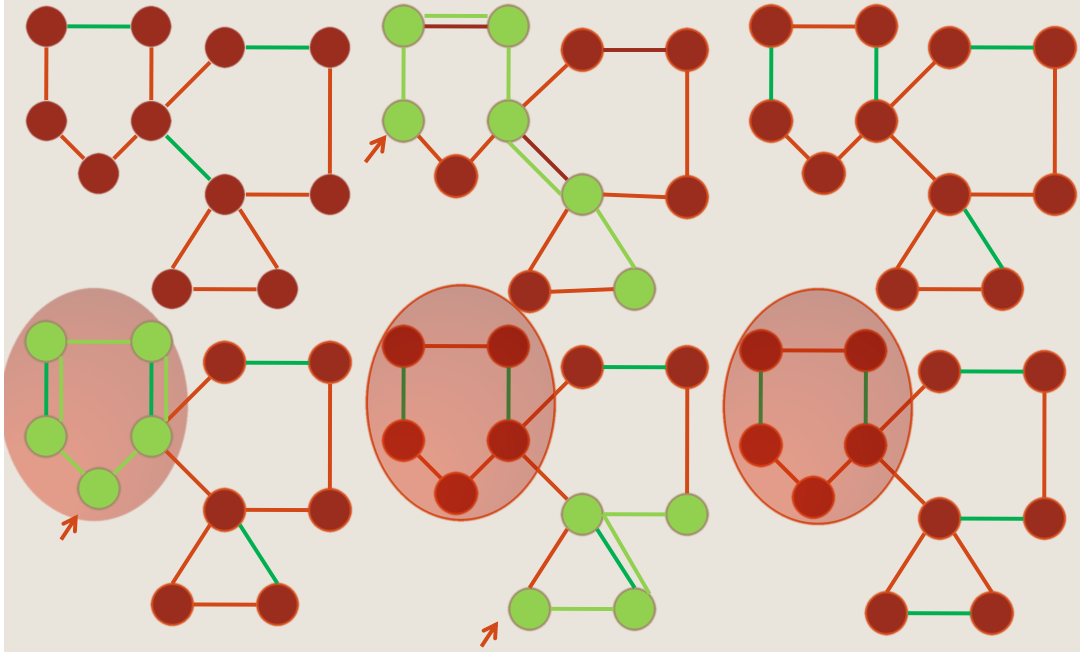
The time complexity depends on how blossoms are handled. Varying with the used data structures, it can be either $O(n^3)$ or $O(m\ n^2)$.

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

[Micali & Vazirani '80]

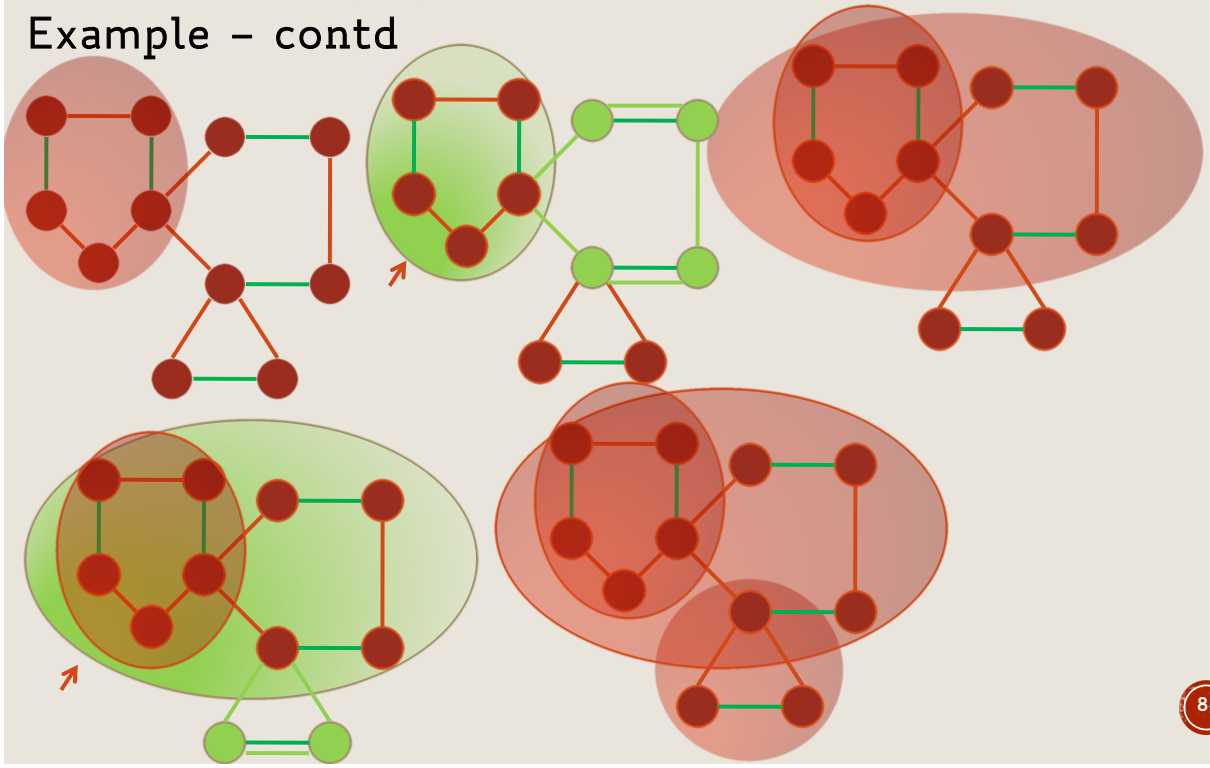
MAX MATCHING IN GENERAL GRAPHS (3)

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)

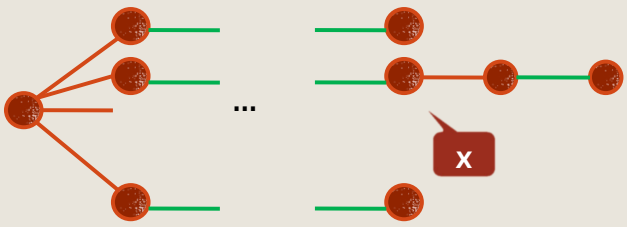
Edmonds algorithm – contd

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.

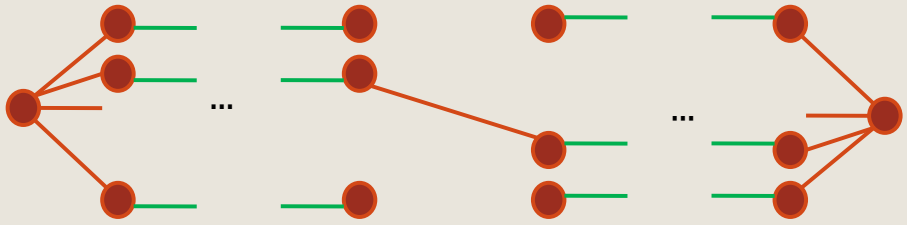


MAX MATCHING IN GENERAL GRAPHS (7)

Edmonds algorithm – contd

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

   → augmenting path

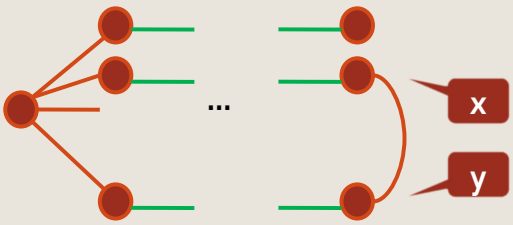


MAX MATCHING IN GENERAL GRAPHS (8)

Edmonds algorithm – contd

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'

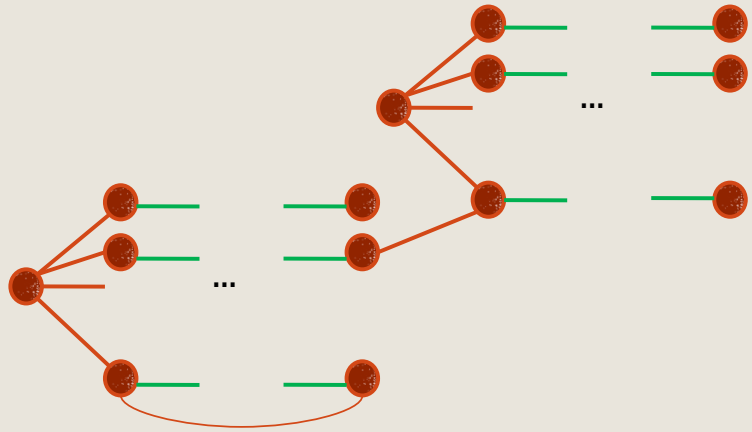


MAX MATCHING IN GENERAL GRAPHS (9)

Edmonds algorithm – contd

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 are:

- modular
- scalar

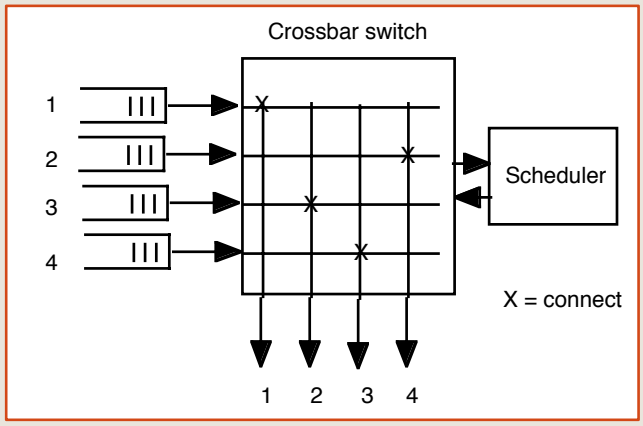
Nevertheless, the buffers at each node provoke:

- 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

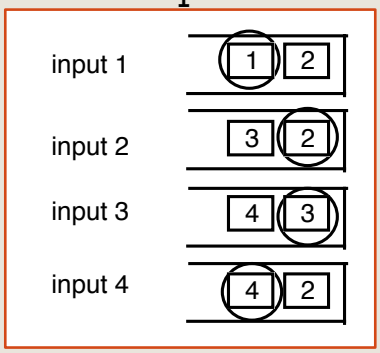


MULTISTAGE TOPOLOGIES WITH BUFFERS (3)

- 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
- …

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:



- 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

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

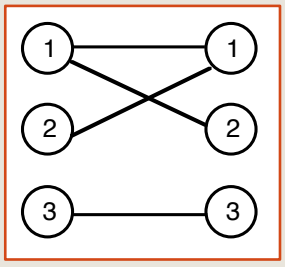
MULTISTAGE TOPOLOGIES WITH BUFFERS (6)

- 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	2	3
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]