

**SECOND PART:  
WIRELESS NETWORKS  
2.B. SENSOR NETWORKS**

1

**THE CENTRALIZED  
DEPLOYMENT  
OF MOBILE SENSORS  
I.E.  
THE MINIMUM WEIGHT  
PERFECT MATCHING**



**ON BIPARTITE GRAPHS**

Prof. Tiziana Calamoneri  
Network Algorithms  
A.y. 2013/14

2



**THE PROBLEM**

3

**MOBILE SENSORS**

- Devices of small dimension and low cost (~150 \$)
- Monitoring Unit (sensing)
- Transmitter/receiver Unit
- Small battery
- Motion system



Mobile sensors are especially useful in critical environments (e.g. in presence of dispersion of pollutants, gas plumes, fires, ...)

4

## THE PROBLEM (1)

Given an Area of Interest (AoI) to cover:

We can assume that each sensor is able to monitor a disk centered at its position and having radius  $r$ =sensing radius.

The aim is to entirely cover the AoI (final equilibrium state).

5

## THE PROBLEM (2)

Coordination algorithm

Initial Config.

➔ Desired Config.

Can be:

- casual
- from a safe location

Can be:

- regular tassellation
- any configuration, provided that the AoI is covered

- At the same time, some parameters need to be optimized:
  - Traversed Distance
  - Number of starting/stopping
  - Communication costs
  - Computation costs

6

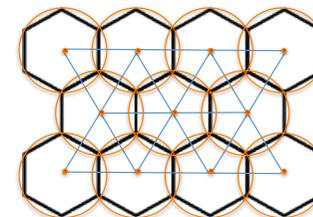
## THE PROBLEM (3)

- Traversed Distance:
  - It is the dominant cost
- Number of starting/stopping
  - start/stop moves 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

7

## THE PROBLEM (4)

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.



8

## THE PROBLEM (5)

In the centralized case:

- The whole coverage is guaranteed assigning to each sensor a position on the grid
- The total energy consumption should be minimized
- We model this problem with the classical **minimum weight perfect matching**
- **Obs.** This model works only for the centralized case, where the AoI and the initial position of each sensor are known.

9



## THE GRAPH MODEL

10

## THE GRAPH MODEL (1)

- Formal definition of the problem:
- Set of  $n$  mobile sensors  $S = \{S_1, S_2, \dots, S_n\}$
- Set of  $p$  locations on the AoI  $L = \{L_1, L_2, \dots, L_p\}$
- $n \geq p$  (in order 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.

11

## THE GRAPH MODEL (2)

- Define a weighted 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  $i = 1 \dots n$  and  $j = 1 \dots p$
  - 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

12



## THE PERFECT MATCHING ON BIPARTITE GRAPHS

13

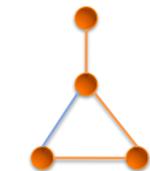
### MATCHING (1)

- **Def.** A **matching** is a set of edges  $M \subseteq E$  such that every node is adjacent to at least 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**
  - $|M| = n/2$ : each node is adjacent to exactly one edge in  $M$ .

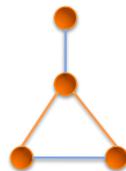
14

### MATCHING (2)

#### Example



Maximal matching

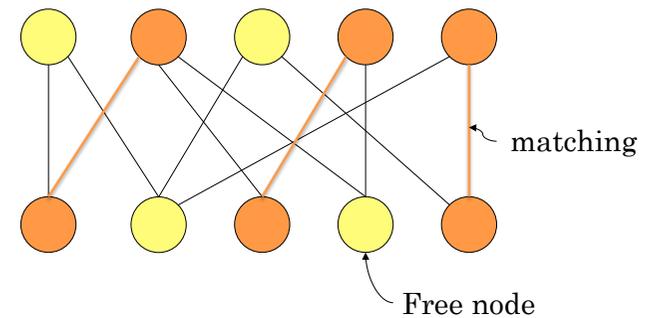


Maximum matching

15

### MATCHING (3)

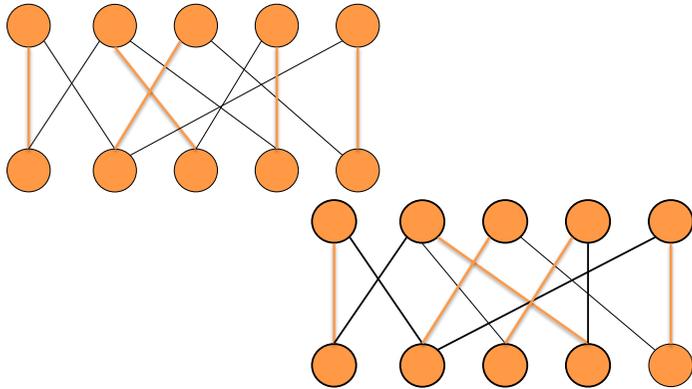
#### ○ Nomenclature



16

## MATCHING (4)

- **Note.** The maximum matching is not unique



17

## MATCHING (5)

Original problem: wedding problem

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



- **Maximum matching** aims at maximizing the number of couples

18

## MATCHING PROBLEMS

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

19

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (1)

- **TH. (P. Hall)** 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  $|S| \leq |adj(S)|$ .

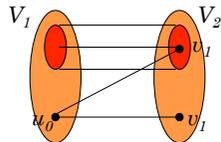
20

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (2)

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

- o **PROOF. 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  $|M| < |V_1|$ .
- o 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 cond., so there exists a  $v_1 \in V_2$  adjacent to  $u_0$ .

- a.  $v_1 \notin M$
- b.  $v_1 \in M$



21

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (3)

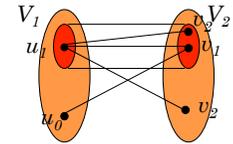
(proof of the Hall theorem - cntd)  $G$  bipartite with  $|V_1| \leq |V_2|$ ,  $G$  has a perfect matching iff  $\forall S \subseteq V_1, |S| \leq |adj(S)|$ .

- 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.  $v_2 \in M$



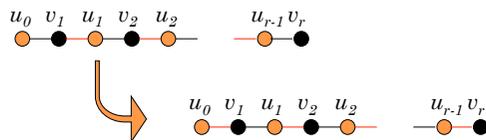
22

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (4)

(proof of the Hall theorem - cntd)  $G$  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  $u_0, u_1, \dots, u_{i-1}$ .

Analogously to the case  $r=2$ :



QED

23

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (5)

**COR.** If  $G$  is bipartite,  $k$ -regular, with  $|V_1| = |V_2|$ , then  $G$  has  $k$  disjoint perfect matchings.

**Proof.** Let  $S$  be a subset of  $V_1$ .

$adj(S)$  has at most  $k|S|$  nodes (if each node in  $adj(S)$  has degree 1 in the subgraph induced by  $S \cup adj(S)$ ).

$adj(S)$  has at least  $|S|$  nodes (if each node in  $adj(S)$  has degree  $k$  in the subgraph induced by  $S \cup adj(S)$ ).

In every case, the Hall condition is true and hence there is a perfect matching.

Remove it and get a new graph that is bipartite,  $(k-1)$ -regular and with  $|V_1| = |V_2|$ .

Repeat the reasoning.

QED

24

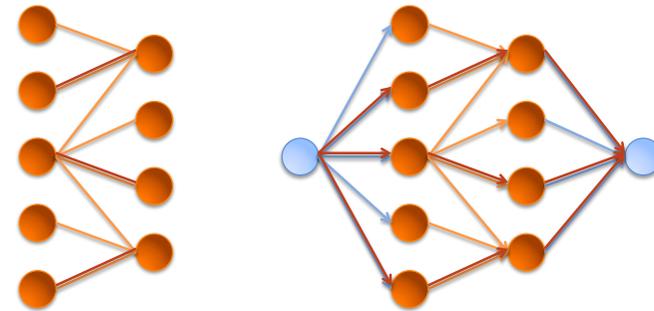
## MAXIMUM MATCHING IN BIPARTITE GRAPHS (6)

- The P. Hall Theorem does not provide an algorithmic method to construct a perfect matching.
- 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 a flow network  $G'=(V', E')$  as follows:
  - $V'=V \cup \{s\} \cup \{t\}$
  - $E'$ :
    - From the source  $s$  to all nodes in  $V_1$ :  $\{(s,u) \mid u \in V_1\} \cup$
    - All edges in  $E$ :  $\{(u,v) \mid u \in V_1, v \in V_2, e(u,v) \in E\} \cup$
    - From all nodes in  $V_2$  to the tale  $t$ :  $\{(v,t) \mid v \in V_2\}$
  - Capacity:  $c(u,v) = 1$ , for all  $(u,v) \in E'$

25

## 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.  $|M|=|f|$ .



26

## 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  $G'$ .*

27

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (9)

**Proof of the 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  $G$ .*

Let  $M$  be max and, by contradiction, let  $f$  be not max.

Then there exists  $f'$  s.t.  $|f'| > |f|$ .

For the integrality th.,  $f'$  has integer values and hence, for the Fact, a matching  $M'$  corresponds to it.

$|M'| = |f'| > |f| = |M|$ . So  $M$  is not max.

Analogously, we can assume  $f$  max and prove that  $M$  is max. **QED**

28

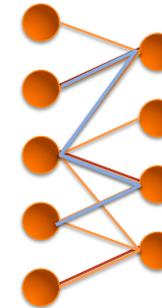
## MAXIMUM MATCHING IN BIPARTITE GRAPHS (10)

- The algorithm by Ford-Fulkerson for the max flow in a network runs in  $O(m|f|)$  time.
- 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  $O(nm)$  time.

29

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (11)

- **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  $E \setminus M$ .

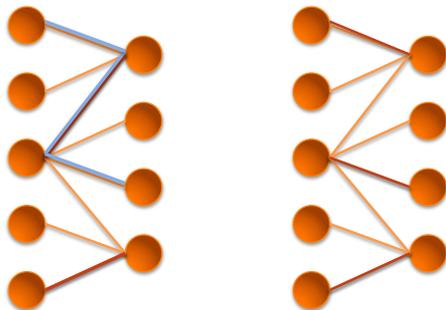


30

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (12)

- **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  $E \setminus M$ ,  $M$  has larger cardinality.



31

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (13)

- **Th. (Augmenting path)** [Berge 1975]  $M$  is a max matching iff there are no augmenting paths w.r.t.  $M$ .
- **Proof. ( $\Rightarrow$ )** 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  $M$ .

- ...

32

### MAXIMUM MATCHING IN BIPARTITE GRAPHS (14)

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

- (←) 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.

- ...

33

### MAXIMUM MATCHING IN BIPARTITE GRAPHS (15)

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

- $H$  has the following property:
  - For each  $v$  in  $H$ ,  $deg(v) \leq 2$ . (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

34

### MAXIMUM MATCHING IN BIPARTITE GRAPHS (16)

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

- 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,  $k > 1$



...

35

### MAXIMUM MATCHING IN BIPARTITE GRAPHS (17)

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

...

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  $M'$



36

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (18)

(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.



QED 37

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (19)

- 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.

38

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (20)

Idea of the algorithm:

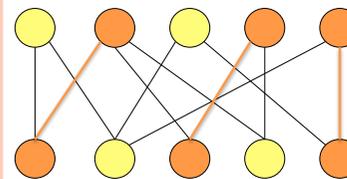
- Let  $M$  be an arbitrary matching (possibly empty)
- Find an augmenting path  $P$
- While there is an augmenting path:
  - Swap in  $P$  the role of the edges in and out of the matching
  - Find an augmenting path  $P$

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

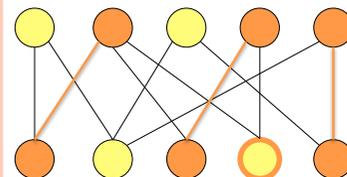
39

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (21)

- Let  $M$  be an arbitrary matching



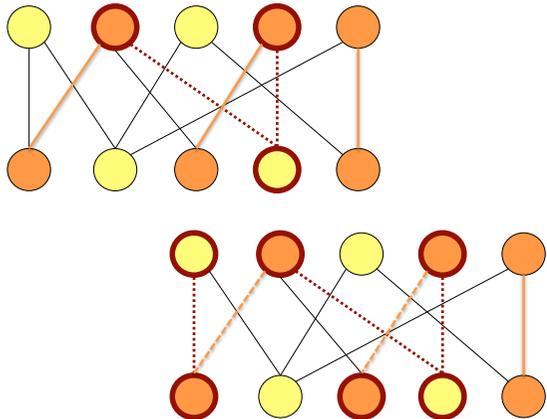
- Choose a free node...



- And consider all its adjacent nodes...

40

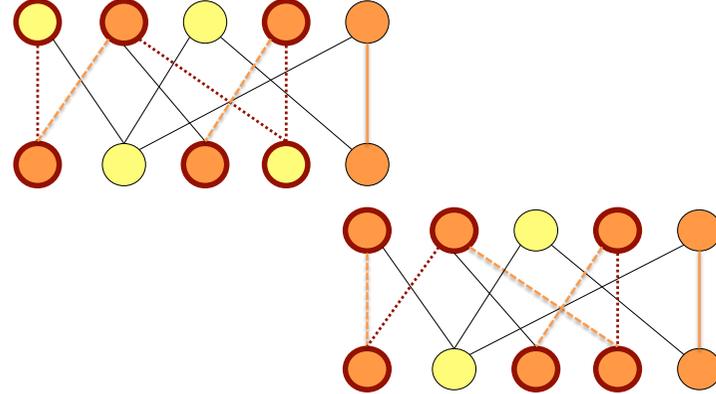
### MAXIMUM MATCHING IN BIPARTITE GRAPHS (22)



... continue until another free node is reached, i.e. an augmenting path has been found

41

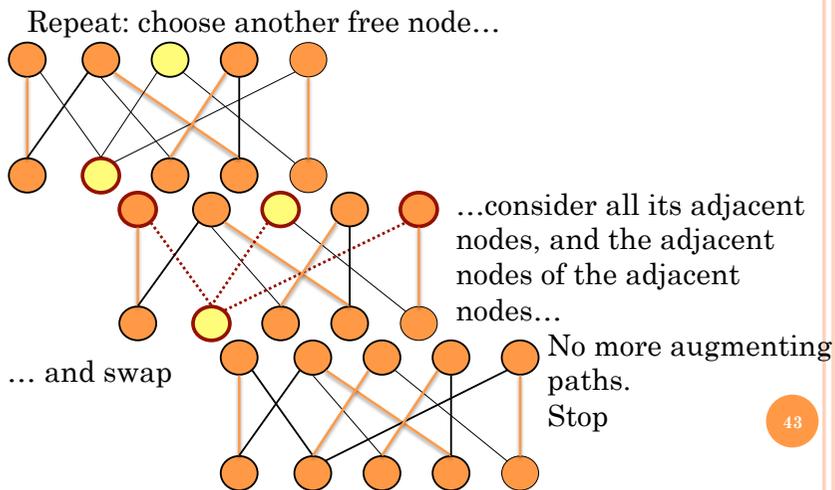
### MAXIMUM MATCHING IN BIPARTITE GRAPHS (23)



Swap the role of edges in and out of the matching

42

### MAXIMUM MATCHING IN BIPARTITE GRAPHS (24)



43

### MAXIMUM MATCHING IN BIPARTITE GRAPHS (25)

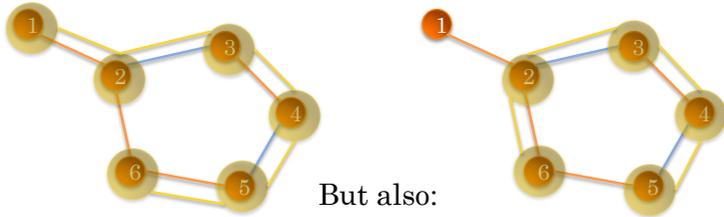
- o **Problem:** how to find an augmenting path w.r.t.  $M$ ?
- o **Idea:**
  - Choose a free node
  - Run a modified search as follows:
    - o Keep trace of the current layer
    - o If the layer is even, use an edge in  $M$
    - o If the layer is odd, use an edges in  $E-M$
    - o As soon as a free node has been encountered, a new augmenting path has been found

44

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (26)

Exemple:

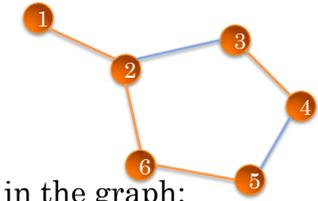
- Choose a free node
- Run a modified search as follows:
  - Keep trace of the current layer
  - If the layer is even, use an edge in  $M$
  - If the layer is odd, use edges in  $E-M$
  - As soon as a free node has been encountered, a new augmenting path has been found



But also:

45

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (27)



- **Problem:** presence of odd cycles in the graph:
  - in an odd cycle there is always a free node adjacent to two 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
- Graphs without odd cycles: bipartite graphs

46

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (28)

Algorithm `SearchAugmentingPathInBip` ( $G=(U \cup W, E), M$ )

- Choose a free node in  $U$
- Repeat
  - If the current node is in  $U$  follow an edge out of  $M$
  - Else follow an edge in  $M$
  - As soon as a free node in  $W$  has been reached, a new augmenting path has been detected

Complexity:  $O(n+m)$

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

max no. of iterations

Swapping of the edges on the aug. path

47

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (29)

- The **Hopcroft–Karp algorithm** (1973) finds a max matching in a bipartite graph in  $O(m\sqrt{n})$  time.
- 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.

48

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (30)

### Hopcroft–Karp Algorithm

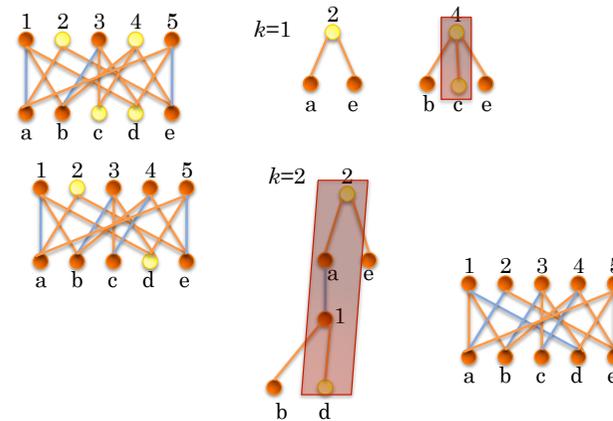
During the  $k$ -th step:

- 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  $k$ .
- All the detected free nodes in  $V_2$  at layer  $k$  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  $k$  aug. paths *node disjoint* using a **depth first search** from the nodes in  $F$  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.

51

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (31)

### Example: Hopcroft–Karp algorithm



50

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (31)

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

- Each step consists in a BFS and a DFS. Hence it runs in  $O(n+m)=O(m)$  time.
- The first  $\sqrt{n}$  steps take  $O(m \sqrt{n})$  time.
- Note. At each step, the length of the found aug. paths is larger and larger; indeed, during step  $k$ , ALL paths of length  $k$  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  $\sqrt{n}$  long.
- ...

51

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (32)

### 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.
- 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  $O(m)$  time, hence the time complexity is  $O(m \sqrt{n})$  in the worst case.

52

## MAXIMUM MATCHING IN BIPARTITE GRAPHS (33)

- 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 in average logarithmic length.
- As a consequence, the Hopcroft–Karp algorithm runs only  $O(\log n)$  steps and so it can be executed in  $O(m \log n)$  time.

53



## MINIMUM WEIGHT PERFECT MATCHING IN BIPARTITE GRAPHS

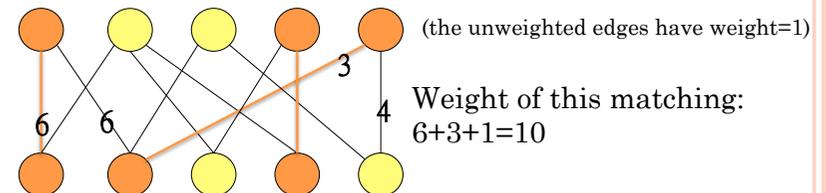
54

## WEIGHTED MATCHING (1)

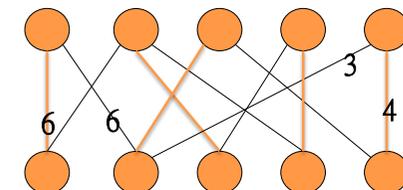
- Each edge has a cost
- The definition of weighted matching is the same as the simple matching (weight does not affect the definition)
- We look for a **minimum weight perfect matching**
- Note. This is equivalent to look for a maximum weight perfect matching, where the weights are all negative.

55

## WEIGHTED MATCHING (2)



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

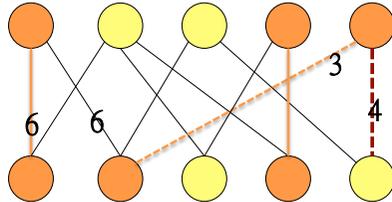


56

### WEIGHTED MATCHING (3)

Def. **augmenting path** (different w.r.t. the previous one!) It 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, aug. paths do not need to end in a free node.

57

### 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  $O(nm)$ .

58

### WEIGHTED MATCHING (5)

- It is possible to model the minimum weight perfect matching problem as an ILP problem (**Hungarian method**):
  - 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:

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

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

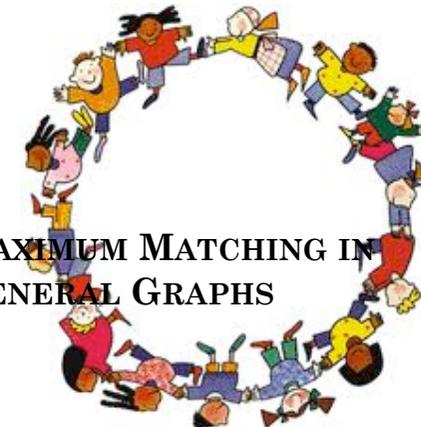
$$x_{ij} \geq 0, i \in A, j \in B$$

$$x_{ij} \text{ integer}, i \in A, j \in B$$

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

59

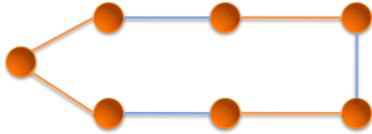
MAXIMUM MATCHING IN GENERAL GRAPHS



60

## BLOSSOMS (1)

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



- Such cycles are called blossoms

61

## BLOSSOMS (2)

- **Lemma (cycle contraction)**. Let  $M$  be a matching of  $G$  and let  $B$  be a blossom. Let  $B$  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  $B$ .

Two cases can hold:

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

62

## BLOSSOMS (3)

Proof of the Cycle contraction lemma – cntd.

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  $G$ . A contradiction.

...

63

## BLOSSOMS (4)

Proof of the Cycle contraction lemma – cntd.

- $M'$  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. **QED**

64

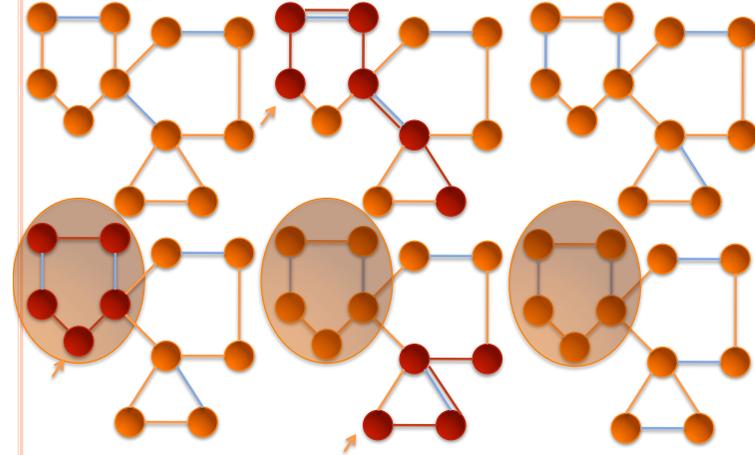
## MAX MATCHING IN GENERAL GRAPHS (1)

- 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, it is shrunk in a node and a new (reduced) graph is generated.
- Each aug. path found in this new graph can be easily “translated” into an aug. path in  $G$ .
- Thanks to the previous lemma, if  $M$  is max in the new graph, it is max even in  $G$ .

65

## MAX MATCHING IN GENERAL GRAPHS (2)

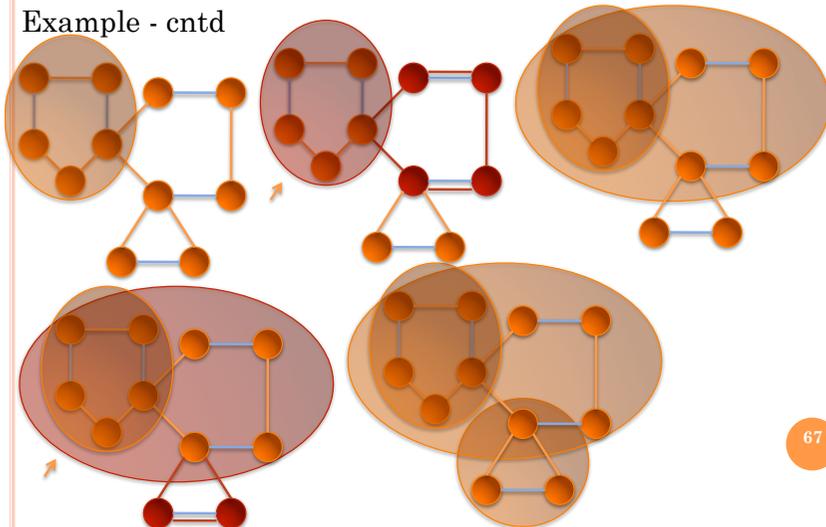
Example:



66

## MAX MATCHING IN GENERAL GRAPHS (3)

Example - cntd



67

## MAX MATCHING IN GENERAL GRAPHS (4)

Edmonds Algorithm [‘65]

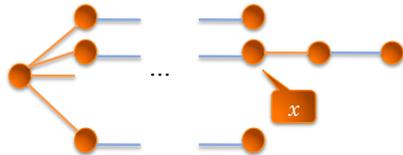
- $M$  matching for  $G$
- $L$  subset of the free nodes (if  $L$  empty  $\Rightarrow M$  max)
- $F$  forest s.t. each node of  $L$  is the root of a tree in  $F$
- Expand  $F$  by adding 
- Nodes that are at odd distance from a node of  $L$  have degree 2 (1 in  $M$  and 1 in  $E \setminus M$ ): we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

68

## MAX MATCHING IN GENERAL GRAPHS (5)

Edmonds algorithm – cntd

- 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  $M$ .

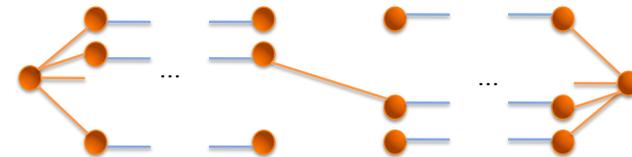


69

## MAX MATCHING IN GENERAL GRAPHS (6)

Edmonds algorithm – cntd

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

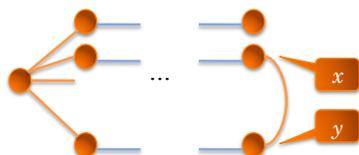


70

## MAX MATCHING IN GENERAL GRAPHS (7)

Edmonds algorithm – cntd

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  $\Rightarrow$  reduced graph  $G'$

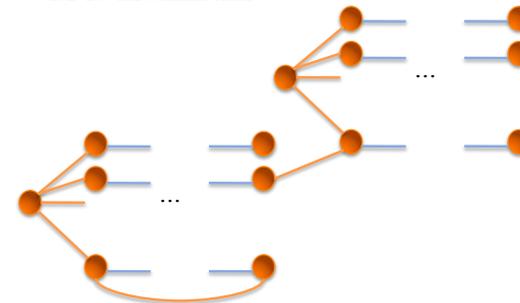


71

## MAX MATCHING IN GENERAL GRAPHS (8)

Edmonds algorithm – cntd

4. All the external nodes are adjacent to internal nodes:  
 $M$  is maximum.



72

## MAX MATCHING IN GENERAL GRAPHS (9)

**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]