



# Table of contents

- 1 **Polynomial-time reductions for graph problems**
  - Hamiltonian problems
  - Traveling salesman problem
- 2 **Approaching NP-hard graph problems**
  - Metric TSP
  - Vertex coloring: Greedy-color algorithm
- 3 **Exercises for the 12th seminar (january 5 - 9 week)**
- 4 **Annex: Christofides Algorithm (proof of correctness)**

## Polynomial-time reductions - Hamiltonian problems

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

### Theorem 1

(Karp, 1972)  $SM \leq_P CH$ .

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

**Proof.** Let  $G = (V, E)$  and  $j \in \mathbb{N}$  an instance for the problem SM. We will build in polynomial time (w.r.t.  $n = |V|$ ) a graph  $H$  such that there is a stable set  $S$  in  $G$  with  $|S| \geq j$  if and only if  $H$  is Hamiltonian. Let  $k = n - j$ . Suppose that  $k > 0$  (to avoid trivial cases).

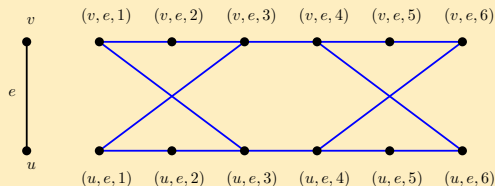
- (i) Let  $A = \{a_1, a_2, \dots, a_k\}$  be a set of  $k$  distinct vertices.
- (ii) For each edge  $e = uv \in E(G)$ , consider the graph  $G'_e = (V'_e, E'_e)$  with  $V'_e = \{(w, e, i) : w \in \{u, v\}, i = \overline{1, 6}\}$  and  $E'_e = \{(w, e, i)(w, e, i + 1) : w \in \{u, v\}, i = \overline{1, 5}\} \cup \{(u, e, 1)(v, e, 3), (u, e, 3)(v, e, 1), (u, e, 4)(v, e, 6), (u, e, 6)(v, e, 4)\}$ .

- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

## Polynomial-time reductions - Hamiltonian problems

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph  
Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -

### Proof cont'd.



The graph has the property that, if it is an induced subgraph of a Hamiltonian graph  $H$ , and no one of the vertices  $(w, e, i)$  with  $w \in \{u, v\}$  and  $i = \overline{2, 5}$  has another neighbor in  $H$ , then the only possibilities of traversing  $G'_e$  by a Hamiltonian cycle are:

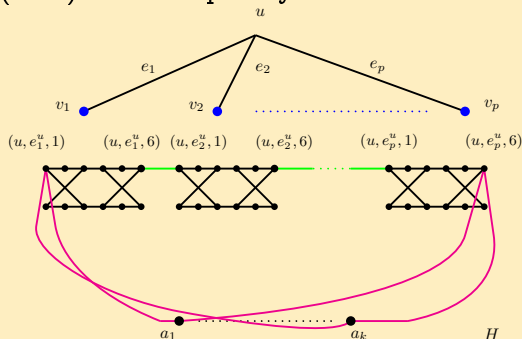
Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -  
Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*



# Polynomial-time reductions - Hamiltonian problems

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

**Proof cont'd.** The graph  $H$  has  $V(H) = A \cup \left( \bigcup_{e \in E} V'_e \right)$  and  $E(H) = \left( \bigcup_{e \in E} E'_e \right) \cup \left( \bigcup_{u \in V} (E''_u \cup E'''_u) \right)$ . Clearly, it can be constructed from  $G$  in polynomial (in  $n$ ) time complexity.



## Polynomial-time reductions - Hamiltonian problems

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

**Proof cont'd.** Now we show that there exists a stable set in  $G$  with at least  $j$  vertices if and only if  $H$  is Hamiltonian.

“ $\Leftarrow$ ” If  $H$  is Hamiltonian, then there exists a Hamiltonian cycle,  $C$ , in  $H$ . Since  $A$  is a stable set in  $H$ ,  $A$  decomposes the cycle  $C$  in exactly  $k$  internally disjoint paths:  $D_{a_{i_1} a_{i_2}}, D_{a_{i_2} a_{i_3}}, \dots, D_{a_{i_k} a_{i_1}}$ .

Let  $D_{a_{i_j} a_{i_{j+1}}}$  be such a path ( $j + 1 = 1 + (j \pmod k)$ ). By the construction of  $H$ , it follows that the first vertex after  $a_{i_j}$  on this path will be  $(v_{i_j}, e_1^{v_{i_j}}, 1)$  or  $(v_{i_j}, e_p^{v_{i_j}}, 6)$ , where  $p = d_G(v_{i_j})$ ,  $v_{i_j} \in V$ .

After that,  $D_{a_{i_j} a_{i_{j+1}}}$  will enter the component  $G'_{e_1}$  or  $G'_{e_p}$  that will be leaved also by a vertex corresponding to  $v_{i_j}$ . If the next vertex is not  $a_{i_{j+1}}$ , it will enter into the component corresponding to the next edge incident with  $v_{i_j}$ , that will be leaved also by a vertex corresponding to  $v_{i_j}$ .

- Graph Algorithms - C. Croitoru - Graph Algorithms - C. Croitoru - Graph Algorithms

## Polynomial-time reductions - Hamiltonian problems

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph  
Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -

**Proof cont'd.** It follows that each path  $D_{a_{i_j} a_{i_{j+1}}}$  corresponds to a unique vertex  $v_{i_j} \in V$ , such that the first and the last edge of  $D_{a_{i_j} a_{i_{j+1}}}$  are  $a_{i_j}(v_{i_j}, e_t^{v_{i_j}}, x)$ ,  $a_{i_{j+1}}(v_{i_j}, e_{t'}^{v_{i_j}}, x')$ , with  $t = 1$  and  $t' = d_G(v_{i_j})$ ,  $x = 1$ ,  $x' = 6$  or  $t = d_G(v_{i_j})$ ,  $t' = 1$ ,  $x = 6$ ,  $x' = 1$ .

It follows that the vertices  $v_{i_j}$  are distinct.

Let  $V^* = \{v_{i_1}, v_{i_2}, \dots, v_{i_k}\}$ . Since  $C$  is a Hamiltonian cycle in  $H$ , it follows that,  $\forall e \in E$ , there is a path  $D_{a_{i_j} a_{i_{j+1}}}$  traversing  $G'_e$ , hence there exists a vertex  $v \in V^*$  adjacent with  $e$ .

Therefore  $S = V \setminus V^*$  is a stable set in  $G$  and  $|S| = j$ .

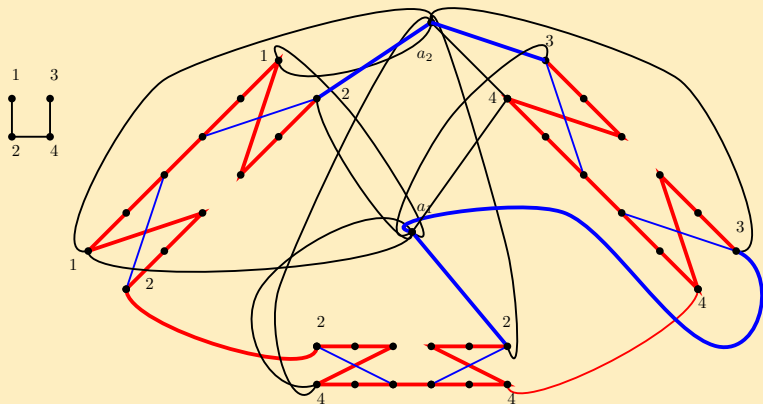
We proved that if  $H$  is Hamiltonian, then in  $G$  there is a stable set with  $j$  vertices, i. e., the answer to SM for the instance  $(G, j)$  is yes.

Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

# Polynomial-time reductions - Hamiltonian problems

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph  
Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -

## Proof cont'd.



Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

**Proof cont'd.** “  $\Rightarrow$  “ Suppose that the answer to the SM's question for the instance  $(G, j)$  is affirmative, hence there exists  $S_0$  a stable set in  $G$  with  $|S_0| \geq j$ . There exists  $S \subseteq S_0$  with  $|S| = j$ . Let  $V^* = V \setminus S = \{v_1, v_2, \dots, v_k\}$ .

Consider in  $H$  for each  $e = uv \in E$ :

- the two paths from case (c) in  $G'_e$  (depicted on one of the above slides) if  $u, v \in V^*$ .
- the path from case (b) in  $G'_e$  (depicted above) if  $u \in V^*$  and  $v \notin V^*$ .
- the path from case (a) in  $G'_e$  (depicted above) if  $u \notin V^*$  and  $v \in V^*$ .

To the union of all these paths we add the edges  $a_i(v_i, e_1^{v_i}, 1)$ ,  $(v_i, e_1^{v_i}, 6)(v_i, e_2^{v_i}, 1)$ ,  $\dots$ ,  $(v_i, e_{p-1}^{v_i}, 6)(v_i, e_p^{v_i}, 1)$ ,  $(v_i, e_p^{v_i}, 6)a_{i+1}$ , ( with  $p = d_G(v_i)$ ), for  $i = \overline{1, k}$ .

It's easy to check that we obtain, in this way, a Hamiltonian cycle in  $H$ .

□

## Polynomial-time reductions - Traveling salesman problem (TSP)

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph  
Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

TSP Given  $G = (V, E)$  a graph and  $d : E \rightarrow \mathbb{R}_+$  a non-negative weight function on its edges, find a Hamiltonian cycle,  $H_o$ , s.t. the sum of the weights of the edges of  $H_o$  is minimum (over all Hamiltonian cycles of  $G$ ).

Let the graph  $G$  be a network consisting of a set,  $V$ , of cities together with a set,  $E$ , of direct routes between cities and the weight function  $d$  giving, for each edge  $uv \in E$ ,  $d(uv)$  = the distance on the direct route between cities  $u$  and  $v$ . Fixing a starting city  $v_0$ , the Hamiltonian cycle  $H_o$  represents the shortest way of visiting all the cities exactly once (except  $v_0$ ) by a traveling salesman starting from  $v_0$  and returning to  $v_0$ .

This is probably, the most studied NP-hard optimization problem!

Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

## Polynomial-time reductions - Traveling salesman problem (TSP)

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

We consider the following equivalent formulation of this problem.

TSP Given  $n \in \mathbb{N}$  ( $n \geq 3$ ) and  $d : E(K_n) \rightarrow \mathbb{R}_+$ , find  $H_o$ , Hamiltonian cycle in  $K_n$ , with  $d(H_o)$  minimum over all Hamiltonian cycles of  $K_n$ , where

$$d(H_o) = \sum_{e \in E(H_o)} d(e).$$

If the graph  $G$ , on which it is required to solve TSP, is not the complete graph  $K_n$ , then we can introduce the missing edges with a very large weight,  $M \in \mathbb{R}_+$ , where  $M > |V| \cdot \max_{e \in E(G)} d(e)$ .

Note that we are limited here only to the symmetric TSP, a similar (asymmetric) problem can be considered for the case when  $G$  is a digraph.

In the study of the time complexity of this problem, we will take  $d(e) \in \mathbb{N}$ , for each edge  $e$ .

## Polynomial-time reductions - Traveling salesman problem (TSP)

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

The corresponding decision problem is

DTSP

Instance:  $n \in \mathbb{N}$  ( $n \geq 3$ ),  $d : E(K_n) \rightarrow \mathbb{N}$  and  $B \in \mathbb{N}$ .

Question: Is there  $H_o$ , Hamiltonian cycle in  $K_n$ , s. t.  $d(H_o) \leq B$ ?

Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -

### Theorem 2

$CH \leq_P DTSP$ .

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru -

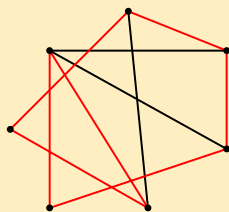
**Proof.** Let  $G = (V, E)$  ( $|V| = n$ ) be an instance of the problem CH. We construct in polynomial time an instance of the problem DTSP,  $d : E(K_n) \rightarrow \mathbb{N}$  and  $B \in \mathbb{N}$ , such that there exists a Hamiltonian cycle in  $K_n$  of total weight not greater than  $B$  if and only if  $G$  is a Hamiltonian graph.

## Polynomial-time reductions - Traveling salesman problem (TSP)

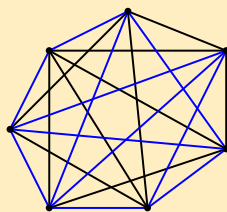
Let

$$d(vw) = \begin{cases} 1, & \text{if } vw \in E(G) \\ 2, & \text{if } vw \in E(\overline{G}) \end{cases} \quad \text{and } B = n.$$

Then, there exists in  $K_n$  a Hamiltonian cycle of weight  $\leq n$  if and only if there is a Hamiltonian cycle  $C$  in  $K_n$  such that  $\forall e \in E(C), d(e) = 1$ , that is, if and only if  $G$  has a Hamiltonian cycle:



$G$



weight 2  
weight 1

It follows that TSP is NP-hard problem. □

## Polynomial-time reductions - Traveling salesman problem (TSP)

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

A possible approach is to consider an approximation algorithm  $\mathcal{A}$ , which builds, in polynomial time complexity, for each instance of TSP, a Hamiltonian cycle  $H_{\mathcal{A}}$  of  $K_n$ , approximating the optimal solution  $H_o$ .

The quality of the approximation can be expressed using the following ratios:

$$R_{\mathcal{A}}(n) = \sup_{d: E(K_n) \rightarrow \mathbb{R}_+, d(H_o) \neq 0} \frac{d(H_{\mathcal{A}})}{d(H_o)}$$
$$R_{\mathcal{A}} = \sup_{n \geq 3} R_{\mathcal{A}}(n).$$

Obviously, the approximation algorithm  $\mathcal{A}$  is useful only if  $R_{\mathcal{A}}$  is finite. Unfortunately, if the weight function  $d$  is arbitrary, finding an algorithm  $\mathcal{A}$  such that  $R_{\mathcal{A}}$  is finite is as hard as solving exactly TSP. More precisely, we have the following result:

- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

## Theorem 3

If there is a polynomial time approximation algorithm  $A$  for TSP s.t.  $R_A < \infty$ , then the problem CH can be solved in polynomial time.

\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

**Proof.** Let  $\mathcal{A}$  be a polynomial time approximation algorithm with  $R_{\mathcal{A}} < \infty$ : there exists  $k \in \mathbb{N}^*$  such that  $R_{\mathcal{A}} \leq k$ .

Let  $G = (V, E)$  be an arbitrary graph, an instance of CH. If  $n = |V|$ , then we consider  $d : E(K_n) \rightarrow \mathbb{N}$  defined by

$$d(uv) = \begin{cases} 1, & \text{if } uv \in E(G) \\ kn, & \text{if } uv \notin E(G) \end{cases} .$$

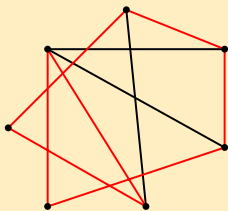
Obviously,  $G$  is Hamiltonian if and only if  $H_o$ , the optimal solution of this instance of TSP, satisfies  $d(H_o) = n$ .

## Polynomial-time reductions - Traveling salesman problem (TSP)

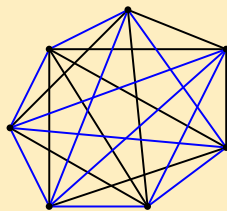
Call  $\mathcal{A}$  to approximately solve this instance of TSP.

- If  $d(H_{\mathcal{A}}) \leq kn$ , then  $d(H_{\mathcal{A}}) = n$  and  $H_{\mathcal{A}}$  is optimal.
- If  $d(H_{\mathcal{A}}) > kn$ , then  $d(H_o) > n$ . Indeed, assuming that  $d(H_o) = n$ , we have  $\frac{d(H_{\mathcal{A}})}{d(H_o)} \leq k$ , therefore  $d(H_{\mathcal{A}}) \leq kd(H_o) = kn$ , contradiction.

It follows that  $G$  is Hamiltonian if and only if  $d(H_{\mathcal{A}}) \leq kn$ , and, since  $\mathcal{A}$  runs in polynomial time, it follows that CH can be solved in polynomial time.  $\square$



$G$



weight  $7k$   
weight 1



## Approaching NP-hard graph problems - Metric TSP

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

### Theorem 4

If in TSP the weight function  $d$  satisfies

$$\forall u, v, w \in V(K_n) \text{ distinct, } d(uv) \leq d(uw) + d(wv),$$

then there is a polynomial time approximation algorithm  $\mathcal{A}$  with  $R_{\mathcal{A}} = 2$ .

\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

The idea is to compute a minimum cost spanning tree of  $K_n$  and then create a hamiltonian cycle based on this tree:

compute  $T^0$  a MST of  $(K_n, d)$ ; // using Prim's algorithm.

$dfs(T^0, v_1)$  and let  $v_1, v_2, \dots, v_n$  be the  $dfs$  order; //  $v_1$  could be any vertex;

return the cycle  $\{v_1, v_2, \dots, v_n, v_1\}$ ;

- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

## Proof.

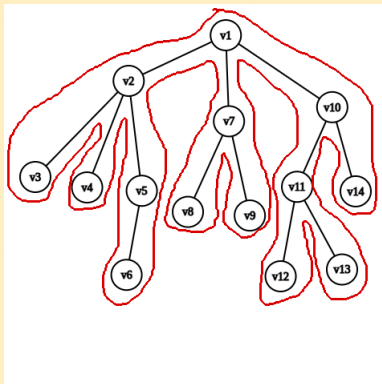
- Let  $H_o$  be a minimum cost hamiltonian cycle in  $(K_n, d)$ ; by removing an edge  $e$  from  $H_o$  one gets a spanning tree  $T = H_o - e$ , hence  $d(T) = d(H_o - e) \leq d(H_o)$ .
- Consider the walk,  $P$ , of  $T^0$  which traverse each edge exactly twice:  $d(P) = 2d(T^0)$ .
- By keeping only the first occurrence of each vertex, except  $v_1$  for which we keep the last occurrence also, we get the hamiltonian cycle  $H$ .
- On the other hand,

$$d(H) \stackrel{(*)}{\leq} d(P) = 2d(T^0) \leq 2d(T) \leq 2d(H_o).$$

- $(*)$  comes from the triangle inequality.

# Approaching NP-hard graph problems - Metric TSP

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph



C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

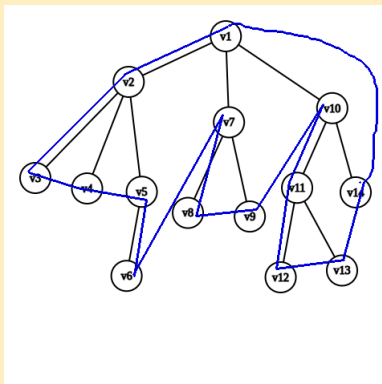
The walk  $P$ :

$v_1, v_2, v_3, v_2, v_4, v_2, v_5, v_6, v_5, v_2, v_1, v_7, v_8, v_7, v_9, v_7, v_1, v_{10}, v_{11}, v_{12}, v_{11}, v_{13}, v_{11}, v_{10}, v_{14}, v_{10}, v_1$

- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

# Approaching NP-hard graph problems - Metric TSP

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph



C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

The hamiltonian cycle  $H$ :

~~$v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_1$~~

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms



## Approaching NP-hard graph problems - Greedy-color algorithm

Let  $G = (V, E)$  be a graph,  $V = \{1, 2, \dots, n\}$  and let  $\pi$  be a permutation of  $V$ . We construct a vertex-coloring  $c : V \rightarrow \{1, \dots, \chi(G, \pi)\}$ .

```
 $c(\pi_1) \leftarrow 1; \chi(G, \pi) \leftarrow 1; S_1 \leftarrow \{\pi_1\};$   
for ( $i = \overline{2, n}$ ) do  
   $j \leftarrow 0;$   
  repeat  
     $j ++; v \leftarrow$  first vertex (in  $\pi$ ), from  $S_j$  s. t.  $\pi_i v \in E(G);$   
    if ( $\exists v$ ) then  
       $first(\pi_i, j) \leftarrow v;$   
    else  
       $first(\pi_i, j) \leftarrow 0; c(\pi_i) \leftarrow j; S_j \leftarrow S_j \cup \{\pi_i\};$   
    end if  
  until ( $first(\pi_i, j) = 0$  or  $j = \chi(G, \pi)$ )  
  if ( $first(\pi_i, j) \neq 0$ ) then  
     $c(\pi_i) \leftarrow j + 1; S_{j+1} \leftarrow \{\pi_i\}; \chi(G, \pi) \leftarrow j + 1;$   
  end if  
end for
```

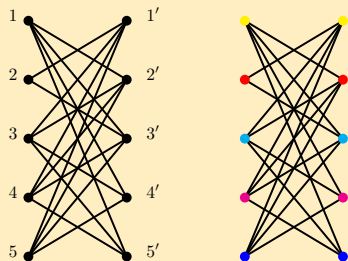
## Approaching NP-hard graph problems - Greedy-color algorithm

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

Note that the number of colors used by the greedy algorithm is not greater than  $1 + \Delta(G)$ . It follows that  $\chi(G) \leq 1 + \Delta(G)$ .

$\chi(G, \pi)$  the number of colors returned by the Greedy-color algorithm, can be arbitrarily larger than  $\chi(G)$ . For example, let  $G$  be the graph obtained from the complete bipartite graph  $K_{n,n}$ , with vertex set  $\{1, 2, \dots, n\} \cup \{1', 2', \dots, n'\}$ , by removing the edges  $11', 22', \dots, nn'$ .



Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

## Approaching NP-hard graph problems - Greedy-color algorithm

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

If  $\pi = (1, 1', 2, 2', \dots, n, n')$ , then the Greedy-color algorithm returns  $c(1) = c(1')$ ,  $c(2) = c(2') = 2$ ,  $c(n) = c(n') = n$ . Therefore,  $\chi(G, \pi) = n$ , while  $\chi(G) = 2$ .

On the other hand, for any graph  $G$ , there is a permutation  $\pi$  of its vertices such that  $\chi(G, \pi) = \chi(G)$ .

Indeed, let  $S_1, S_2, \dots, S_{\chi(G)}$  be the coloring classes of an optimal coloring, such that each  $S_i$  is a maximal (w. r. t. inclusion) stable

set in  $G - \bigcup_{j=1}^{i-1} S_j$ . Let  $\pi$  be a permutation which induces an ordering s.t. vertices occurs in the non-decreasing order of their colors. Then,  $\chi(G, \pi) = \chi(G)$ .

A sufficient condition for the correctness of the Greedy-color algorithm:

Graph Algorithms - C. Croitoru - Graph Algorithms - C. Croitoru - Graph Algorithms - C. Croitoru  
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*





**Exercise 1.** The regions (including the infinite region) formed by  $n$  circles in the plane can be colored with two colors such that any two regions that share a common boundary arc should be colored differently.

**Exercise 2.** Let  $G = (V, E)$  be a graph having no disjoint odd cycles. Prove that  $G$  is 5-colorable.

**Exercise 3.** For a given graph  $H$  we define the average degree as  $ad(H) = \frac{2|E(H)|}{|V(H)|}$ . For a graph  $G$ , the maximum average degree is

$$mad(G) = \max \{ ad(H) : H \text{ induced subgraph of } G \}$$

Let  $k \geq 3$  an integer. Prove that, if a graph  $G$  has its chromatic number strictly greater than  $k$  and its maximum average degree at most  $k$ , then  $G$  contains an induced  $k$ -regular subgraph.

## Exercise 4.

- (a) Prove that any graph can be vertex-colored with  $\Delta(G) + 1$  colors.
- (b) Consider the following recursive algorithm for coloring the vertices of a given 3-colorable graph with  $n$  vertices:

```

color( $G$ ) {
  if ( $\Delta(G) \leq \sqrt{n}$ ) then
    color all the vertices of  $G$  with  $\Delta(G) + 1$  new colors;
  else
    let  $x_0 \in V(G)$  s. t.  $d_G(x_0) = \Delta(G)$ ;
    color  $x_0$  with a new color;
    color the vertices of  $[N_G(x_0)]_G$  with two new colors;
     $G \leftarrow G - (\{x_0\} \cup N_G(x_0))$ ;
    color( $G$ );
  end if }

```

Prove that the above algorithm uses  $\mathcal{O}(\sqrt{n})$  colors.

**Exercise 5.** Consider the following problem:

Set Cover

Instance: A non empty set  $X$ , a family of subsets of  $X$ :  $\mathcal{F} = \{X_1, \dots, X_p\}$  and  $k \in \mathbb{N}^*$ .

Question:  $X$  can be covered with at most  $k$  subsets of  $\mathcal{F}$ ?

Let us consider the following heuristic (a greedy type algorithm) for solving this problem

$\mathcal{F}' \leftarrow \emptyset$ ;

while ( $\exists x \in X$  uncovered by  $\mathcal{F}'$ ) do

    let  $X_i$  a subset in  $\mathcal{F}$  with the largest number of uncovered elements;

$\mathcal{F}' \leftarrow \mathcal{F}' \cup \{X_i\}$ ;

end while

return  $\mathcal{F}'$ ;

Prove that if  $m$  is the minimum cardinality of an optimal subfamily of  $\mathcal{F}$ , then the above algorithm has gives a subfamily of at most  $m \ln n$  subsets ( $n = |X|$ ).

**Exercise 6.** Let  $G = (V, E)$  be a graph with  $n$  vertices and  $m$  edges. An ordering  $\{x_1, x_2, \dots, x_n\}$  of the vertices of  $G$  is called  $k$ -bounded if in the digraph  $\vec{G}$ , obtained from  $G$  by replacing the edge  $x_i x_j$  with an arc  $x_{\min\{i,j\}} \rightarrow x_{\max\{i,j\}}$ , we have  $d_{\vec{G}}^+(x) \leq k, \forall x \in V$ .

- (a) Devise an algorithm to test in  $\mathcal{O}(n + m)$  time complexity if  $G$ , has a  $k$ -bounded order, where  $k \in \mathbb{N}$ .
- (b) Use the above algorithm in order to determine in  $\mathcal{O}((n + m) \log n)$  time the number

$$o(G) = \min\{k \in \mathbb{N} : G \text{ has a } k\text{-bounded order.}\}$$

- (c) Prove that any graph  $G$  has a vertex coloring which uses  $o(G) + 1$  colors.



**Exercise 8.** Let  $G = (V, E)$  be a graph with  $V = \{1, 2, \dots, n\}$  and  $\omega(G) = 2$ . We define a new graph  $M(G)$  by considering the disjoint union of  $G$  and  $K_{1,n}$  (whose bipartition is  $(\{0\}, \{1', 2', \dots, n'\})$ ) and adding all the edges  $\{i'j, ij' : ij \in E(G)\}$ .

- Prove that  $\omega(M(G)) = 2$  and  $\chi(M(G)) = \chi(G) + 1$ .
- Show that, for every  $p \in \mathbb{N}^*$ , there exists a  $K_3$ -free graph having chromatic number  $p$ .

**Exercise 9.** Let  $\mathcal{S}$  be a society formed with  $n$  individuals. Each person,  $i \in \mathcal{S}$ , knows a subset  $c(i) \subseteq \mathcal{S} \setminus \{i\}$  of other persons. Two different persons  $i, i' \in \mathcal{S}$  cannot be part of the same jury if one knows the other (we can have single member juries).

Prove that if each person knows at most  $k$  other persons ( $|c(i)| \leq k$ ,  $\forall i \in \mathcal{S}$ ), then there exists a family of at most  $(2k + 1)$  disjoint juries that cover together the entire society.

## Theorem 6

(Christofides, 1976) If in TSP the weight function  $d$  satisfies

$$\forall u, v, w \in V(K_n) \text{ distinct, } d(uv) \leq d(uw) + d(wv),$$

then there is a polynomial time approximation algorithm  $\mathcal{A}$  with  $R_{\mathcal{A}} = 3/2$ .

**Proof.** Let  $\mathcal{A}$  be the following algorithm:

- Find  $T^0$  the edge set of MST in  $K_n$  (the cost of each edge  $e$  is  $d(e)$ ) (this takes polynomial time using any MST algorithms).
- Find  $M^0$  a minimum weight perfect matching in the subgraph induced in  $K_n$  by the set of vertices of odd degrees of  $T^0$  (this takes polynomial time using any maximum matching algorithm).

## Proof cont'd.

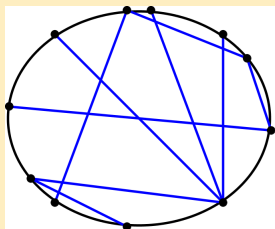
- In the multi-graph obtained from  $\langle T^0 \cup M^0 \rangle_{K_n}$ , by duplicating the edges from  $T^0 \cap M^0$  (it is connected and all vertices have even degree) find a closed Euler trail,  $(v_{i_1}, v_{i_2}, \dots, v_{i_n})$ . Eliminate all the multiple occurrences of the internal vertices to obtain a Hamiltonian cycle  $H_A$  in  $K_n$  with the edge set  $H_A = \{v_{j_1} v_{j_2}, v_{j_2} v_{j_3}, \dots, v_{j_n} v_{j_1}\}$  (both constructions need  $\mathcal{O}(n^2)$  time, the closed Euler trail can be found with Hierholzer's algorithm).

$H_A$  is an approximative solution of TSP given by Christofides. Let  $m = \lfloor n/2 \rfloor$  and  $H_o$  be the optimal solution. We prove (after Cornuejols & Nemhauser) that

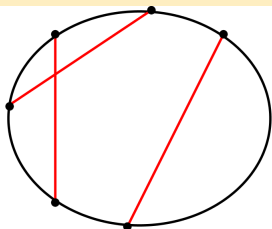
$$\forall n \geq 3, d(H_A) \leq \frac{3m - 1}{2m} d(H_o).$$

# Approaching NP-hard graph problems - Christofides Algorithm

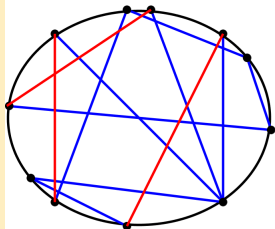
C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms



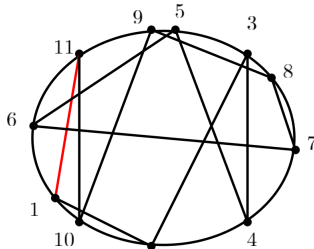
$T^0$



$M^0$



Eulerian graph



$H_A$

## Approaching NP-hard graph problems - Christofides Algorithm

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms  
\* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph

**Proof cont'd.** Let  $H_o = \{v_1 v_2, v_2 v_3, \dots, v_n v_1\}$  (if necessary, we can re-name the nodes).

Let  $W = \{v_{i_1}, v_{i_2}, \dots, v_{i_{2k}}\}$  the set of odd degree vertices in  $\langle T^0 \rangle_{K_n}$ ,  $i_1 < i_2 < \dots < i_{2k}$ . Let  $H = \{v_{i_1} v_{i_2}, v_{i_2} v_{i_3}, \dots, v_{i_{2k-1}} v_{i_{2k}}, v_{i_{2k}} v_{i_1}\}$  be the cycle generated by  $W$  in  $K_n$ . Applying repeatedly the triangle inequality, we obtain  $d(H) \leq d(H_o)$ , (the weight of each chord,  $d(v_{i_j} v_{i_{j+1}})$  is upper bounded by the sum of the weights of the edges on  $H_o$  joining the extremities of the chord  $v_{i_j} v_{i_{j+1}}$ ).

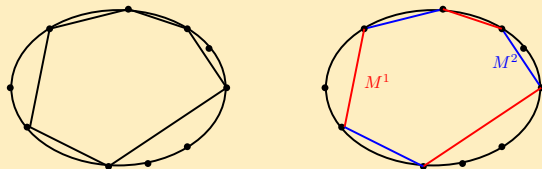
Since  $H$  is an even cycle, it is the union of two perfect matchings in  $[W]_{K_n}$ ,  $M^1 \cup M^2$ . Suppose that  $d(M^1) \leq d(M^2)$ .

By the choosing of  $M^0$ , we have  $d(M^0) \leq d(M^1) \leq (1/2)[d(M^1) + d(M^2)] = (1/2)d(H) \leq (1/2)d(H_o)$ . Let  $\alpha \in \mathbb{R}_+$  s. t.  $d(M^0) = \alpha d(H_o)$ . Obviously,  $0 < \alpha \leq 1/2$ .

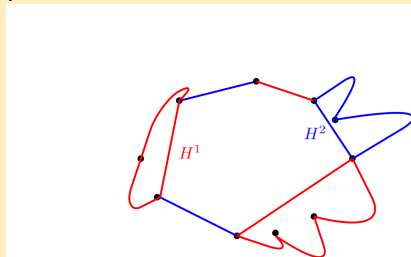
- Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \*

# Approaching NP-hard graph problems - Christofides Algorithm

Proof cont'd.



Decompose  $H_0$  into  $H^1 \cup H^2$  by taking into  $H^i$  the edges of  $H_0$  connecting the extremities of each chord in  $M^i$ : ( $v_{i_j} v_{i_{j+1}} \in M^i \Rightarrow v_{i_j} v_{i_{j+1}}, \dots, v_{i_{j+1}-1} v_{i_{j+1}} \in H^i$ ).



# Approaching NP-hard graph problems - Christofides Algorithm

C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms \* C. Croitoru - Graph Algorithms

**Proof cont'd.** By the triangle inequality,  $d(H^i) \geq d(M^i)$ ,  $i = 1, 2$ .

At least one of  $H^1$  or  $H^2$  has at most  $m = \lfloor n/2 \rfloor$  edges. Suppose that  $H^1$  has this property. Since  $d(H^1) \geq d(M^1) \geq d(M^0) = \alpha d(H_o)$ , it follows that there exists  $e \in H^1$  such that  $d(e) \geq (\alpha/m)d(H_o)$ .

Let  $T$  be the spanning tree obtained from  $H_o$  by deleting an edge of maximum weight. We have  $d(T) = d(H_o) - \max_{e \in E(H_o)} d(e) \leq d(H_o) - (\alpha/m)d(H_o)$ .

Since  $T^0$  is MST in  $K_n$  it follows that  $d(T^0) \leq d(H_o)(1 - \alpha/m)$ .

Using the triangle inequality, we have

$$\begin{aligned} d(H_{\mathcal{A}}) &\leq d(T^0) + d(M^0) \leq d(H_o) \left(1 - \frac{\alpha}{m}\right) + \alpha d(H_o) = \\ &= \left(1 + \frac{\alpha(m-1)}{m}\right) d(H_o) \stackrel{\alpha \leq 1/2}{\leq} \frac{3m-1}{2m} d(H_o), \forall n \geq 3. \quad \square \end{aligned}$$