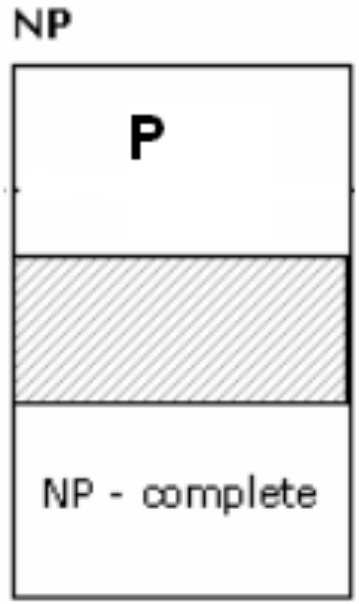
ALMA
ALGORITHMS

Fall 2016

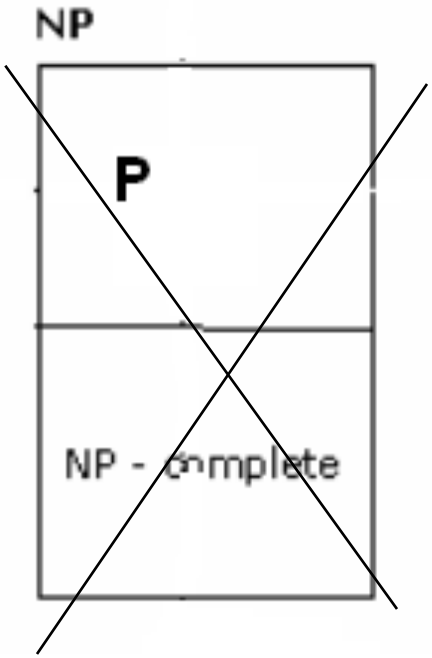
Ioannis Milis

Approximation Algorithms

P vs NP



The common belief



Not possible
(Ladner's theorem)

Nobody believes this

No O(poly) algorithm is known for any $\mathcal{NP}$-complete problem

P vs NP

Hard problems (**NP**-complete)	Easy problems (in **P**)
3SAT	2SAT, HORN SAT
TRAVELING SALESMAN PROBLEM	MINIMUM SPANNING TREE
LONGEST PATH	SHORTEST PATH
3D MATCHING	BIPARTITE MATCHING
KNAPSACK	UNARY KNAPSACK
INDEPENDENT SET	INDEPENDENT SET on trees
INTEGER LINEAR PROGRAMMING	LINEAR PROGRAMMING
RUDRATA PATH	EULER PATH
BALANCED CUT	MINIMUM CUT

No O(poly) algorithm is known for any $\mathcal{NP}$-complete problem

Coping with NP-complete problems

1. Small instances: an O(exp) algorithm may be satisfactory

2. Special cases: may be O(poly), e.g. 2-SAT

3. Exponential algorithms
   Pseudo-polynomial algorithms, Dynamic Programming,
   Backtracking, Branch-and-Bound

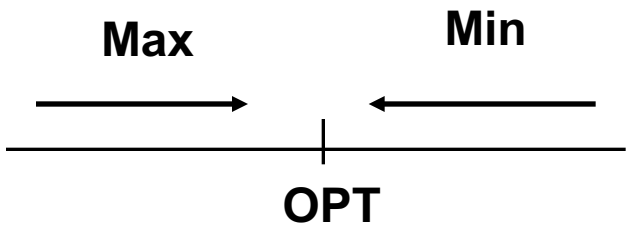
4. Approximation algorithms : O(poly) algorithms that produce solutions within a guaranteed factor away from the optimum solution

5. Randomized algorithms: O(poly) algorithms
   Monte Carlo: Deterministic complexity; whp a correct solution
   Las Vegas: Expected complexity; always a correct solution

5. Heuristic algorithms: any O(poly) approach without a formal guarantee of performance but valid in practical situations

Approximation algorithms

O(poly) algorithms obtaining a solution of cost C
within a factor ρ of the optimum cost OPT



Minimization problems: $C/OPT \leq \rho = 1 + \varepsilon, \ \varepsilon > 0$

Maximization problems: $C/OPT \leq \rho = 1 - \varepsilon, \ \varepsilon > 0$

$|OPT - C| / OPT \leq \varepsilon, \ \varepsilon > 0$

ε: relative error $(\varepsilon \times 100)$ %

Approximations: Good, better, best and more …

Non - constant approximation : $C/OPT \leq f(n)$

Constant (ρ-)approximation : $C/OPT \leq \rho$ (a constant, e.g. 3/2)

Polynomial Time Approximation Schemes (PTAS)

- $C/OPT \leq 1 + \varepsilon$, for any $\varepsilon > 0$
- $O(poly(|I|))$, $O(exp(1/\varepsilon))$, e.g. $O(n^{3/\varepsilon})$

Fully Polynomial Time Approximation Schemes (FPTAS)

- $C/OPT \leq 1 + \varepsilon$, for any $\varepsilon > 0$
- $O(poly(|I|))$, $O(poly(1/\varepsilon))$ e.g. $O((1/\varepsilon)^2 n^3)$

Additive approximation

- $C \leq OPT + f(n)$ or $C \leq OPT + k$ (a constant), e.g. $C \leq OPT + 1$!

Vertex Cover
Set Cover

(Weighted) Set Cover

WEIGHTED SET COVER (WSC)

I: a set U of n elements

a family F={S_1, S_2, …,S_m} of subsets of U,

a weight w(S_i) for each set S_i

Q: Find a minimum weight subset C $\subseteq$ F covering all elements of U,

i.e. $\bigcup_{S_i \in C} S_i = U$ and $W = \sum_{S_i \in C} w(S_i)$ is minimized

SET COVER (SC): $w(S_i) = 1, \forall S_i \in F,$ that is $W = \sum_{S_i \in C} w(S_i) = |C|$

Find a minimum size subset C $\subseteq$ F covering all elements of U,

i.e. $\bigcup_{S_i \in C} S_i = U$ and $|C|$ is minimized

(Weighted) Vertex Cover

WEIGHTED VERTEX COVER (WVC)

I: a graph G=(V, E),
a weight w(u) for each vertex u∈V

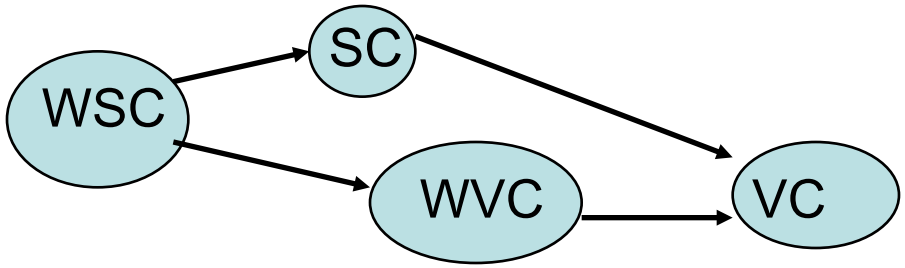
Q: Find a minimum weight subset C ⊆ V covering all edges of G
i.e. for every edge (u,v) ∈ E either u ∈ C or v ∈ C
and $W = \sum_{u \in C} w(u)$ is minimized

VERTEX COVER (VC): $w(u) = 1, \forall u \in V \Rightarrow W = \sum_{v \in C} w(u) = |C|$

Find a minimum size subset C ⊆ F covering all elements of U,
i.e. for every edge (u,v) ∈ E either u ∈ C or v ∈ C and |C| is minimized.

Set v.s. Vertex Cover

(WEIGHTED) VERTEX COVER	(WEIGHTED) SET COVER
I: (weighted) graph G=(V,E)	I: U=E |F| = |V| S_u ={(u,v) | (u,v) ∈ E }
Q: find C ⊆ V s.t. C covers E and C is of min size (cost)	Q: find C ⊆ F s.t. C covers U and C is of min size (cost)



Hence, all WSC, SC, and WVC problems are strongly NP-complete as generalizations of VC

Set v.s. Vertex Cover

f_u = frequency of an element $u \in U$ = # of S_i's element u belongs to

$f = \max_{u \in U} \{ f_u \}$ = frequency of the most frequent element

If f=2 (and $w(S_i) = 1$) then (W)SC reduces to (W)VC:
G=(V,E), V= F, E= { (u,v) | $S_u \cap S_v \neq 0$ }

There are approximation algorithms for WSC,
and hence, for SC, WVC and VC,
of ratios:

- O(log n) (n: the size of the universe U)
- f

(W) VERTEX COVER: f =2

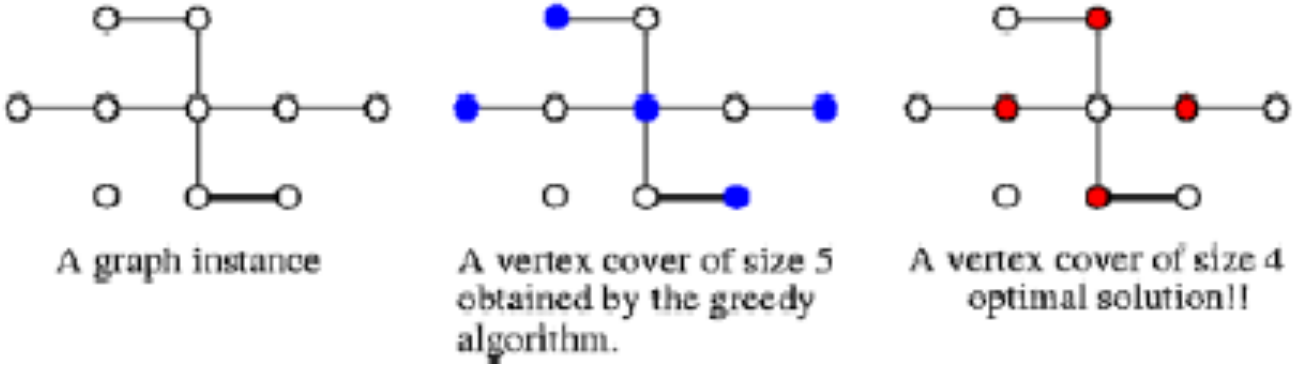
Vertex Cover (VC)

A natural greedy idea: at each step choose the node with the max degree

Greedy-best-node

C := ∅ ;
while E ≠ ∅ do
{ choose the vertex u ∈ V with the largest degree; (break ties arbitrarily)
delete u and its incident edges from G;
Add u to C }

A counterexample for Greedy-best-node



In fact, Greedy-best-node is an O(log n)-approximation algorithm

Vertex Cover (VC)

Greedy-any-node

C := ∅ ;

while E ≠ ∅ do

{ choose arbitrarily a vertex u ∈ V;

delete u and its incident edges from G;

Add u to C }

What is the approximation ratio this algorithm ?

Vertex Cover (VC)

Greedy-any-edge

C := ∅ ;

while E ≠ ∅ do

{ choose arbitrarily an edge (u,v) ∈ E ;

delete u and v and their incident edges from G;

Add u and v to C; }

This is a 2-approximation algorithm

Vertex Cover (VC)

Greedy-any-edge is a 2-approximation algorithm

Proof:

a) C is a VC

Greedy-any-edge T loops until E =∅,
therefore each edge is covered by C.

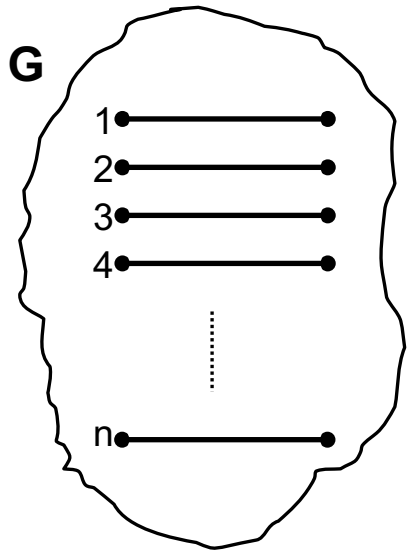
b) 2-approximation ratio

- Let A be the set of edges selected by Greedy-any-edge
- Each selected edge adds two vertices to C ⇨ $|C| = 2\,|A|$
- No two edges in A share a vertex (A is a maximal matching)
  (edges incident to the endpoints of a selected edge are removed)
- To cover the edges in A, each vertex cover
  (and the optimal one)
  must contain one of their endpoints ⇨ $|A| \leq OPT$
- Thus, $|C| = 2\,|A| \leq 2\ OPT$ ⇨ $|C| / OPT \leq 2$

Vertex Cover (VC)

The ratio 2 of Greedy-any-edge is tight

Counterexample:



C = 2n

OPT = n

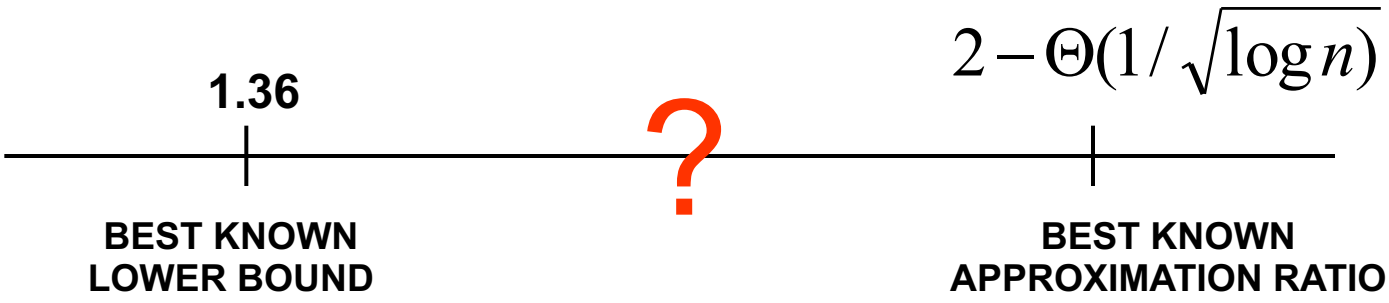
C/OPT = 2n/n = 2

Vertex Cover (VC)

Greedy-any-edge is almost the best known for VC

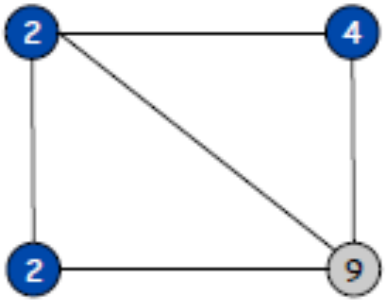
Is there a better approximation algorithm ?

We know a lower bound of 1.36 on the approximation factor for VC,
i.e.
Unless P=NP, VC can not be approximated with a ratio smaller than 1.36

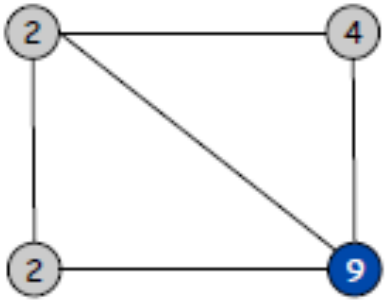


Weighted Vertex Cover (WVC)

Find a minimum weight subset $C \subseteq V$ covering all edges of G



weight = 2 + 2 + 4



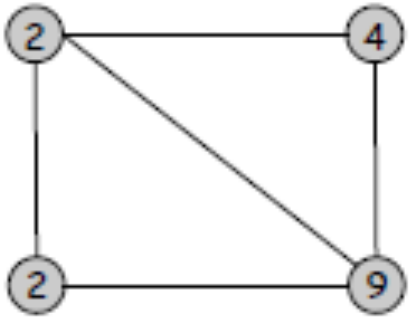
weight = 9

WVC – Pricing method

Pricing method. Each edge must be covered by some vertex i. Edge e pays price $p_e \geq 0$ to use vertex i.

Fairness. Edges incident to vertex i should pay $\leq w_i$ in total.

for each vertex i : $\sum_{e=(i,j)} p_e \leq w_i$



Claim. For any vertex cover S and any fair prices p_e: $\sum_e p_e \leq w(S)$.

Proof. ∎

$$\sum_{e \in E} p_e \leq \sum_{i \in S} \sum_{e=(i,j)} p_e \leq \sum_{i \in S} w_i = w(S).$$

each edge e covered by at least one node in S

sum fairness inequalities for each node in S

WVC – Pricing method

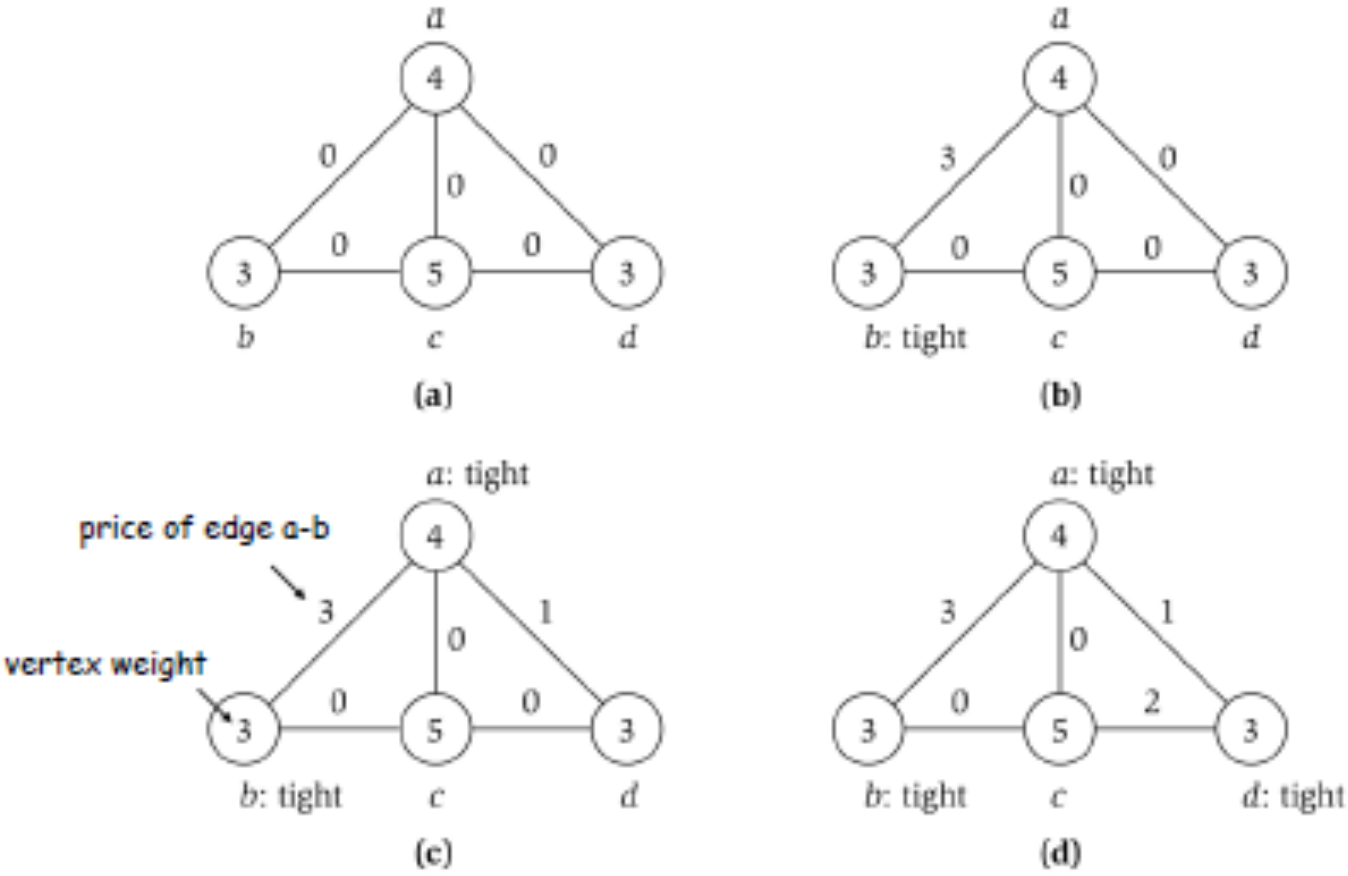
```
Weighted-Vertex-Cover-Approx(G, w) {
   foreach e in E
      p_e = 0

   while (∃ edge i-j such that neither i nor j are tight)
      select such an edge e
      increase p_e without violating fairness
   }

   S ← set of all tight nodes
   return S
}
```

$$\sum_{e=(i,j)} p_e = w_i$$

WVC – Pricing method



WVC – Pricing method

Theorem. Pricing method is a 2-approximation.

Pf.

- Algorithm terminates since at least one new node becomes tight after each iteration of while loop.
- Let S = set of all tight nodes upon termination of algorithm. S is a vertex cover: if some edge i-j is uncovered, then neither i nor j is tight. But then while loop would not terminate.
- Let S* be optimal vertex cover. We show $w(S) \le 2w(S^*)$.

$$w(S) = \sum_{i \in S} w_i = \sum_{i \in S} \sum_{e=(i,j)} p_e \le \sum_{i \in V} \sum_{e=(i,j)} p_e = 2 \sum_{e \in E} p_e \le 2w(S^*). \quad \blacksquare$$

- all nodes in S are tight
- $S \subseteq V$, prices ≥ 0
- each edge counted twice
- fairness lemma

Tightness ?

Set Cover

ΕΙΣΟΔΟΣ: Σύνολο U, n στοιχείων και m υποσύνολα $S_1,S_2,\ldots,S_m \subseteq U$

ΕΞΟΔΟΣ: Ο ελάχιστος αριθμός υποσυνόλων που καλύπτουν το U

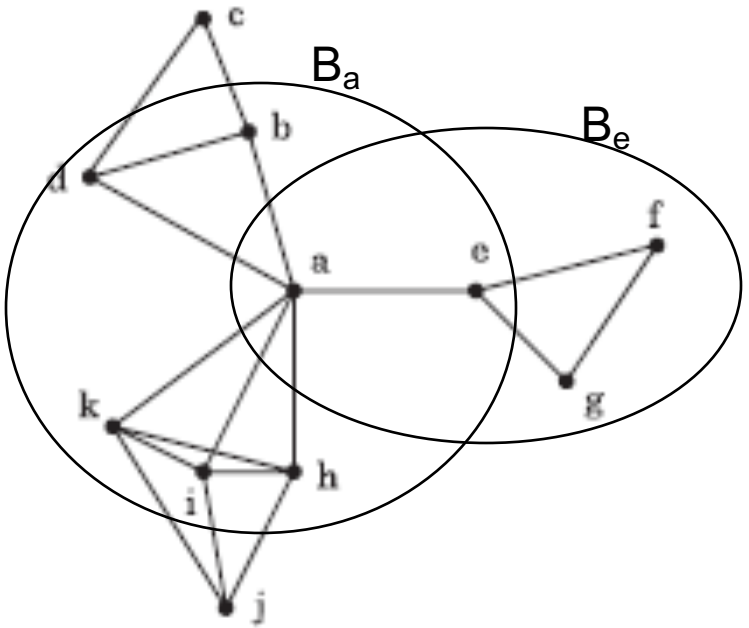
Παράδειγμα:

- U: σύνολο n πόλεων
- Τοποθέτηση σχολείων έτσι ώστε καμία πόλη να μην απέχει περισσότερα από 30 χιλ. από κάποιο σχολείο (να καλύπτονται όλες οι πόλεις)
- Υποσύνολα: Για κάθε πόλη i, S_i είναι το σύνολο των πόλεων που απέχουν μέχρι 30 χιλ. από αυτή
- Ποιος είναι ελάχιστος αριθμός σχολείων?

Set Cover

ΕΙΣΟΔΟΣ: Σύνολο U, n στοιχείων και m υποσύνολα $S_1,S_2,\ldots,S_m \subseteq U$

ΕΞΟΔΟΣ: Ο ελάχιστος αριθμός υποσυνόλων που καλύπτουν το U



Σύνολο U (n=11)

π.χ. πόλεις

Υποσύνολα $S_1,S_2,\ldots,S_m$

πόλεις που απέχουν ≤ 30 χιλ.

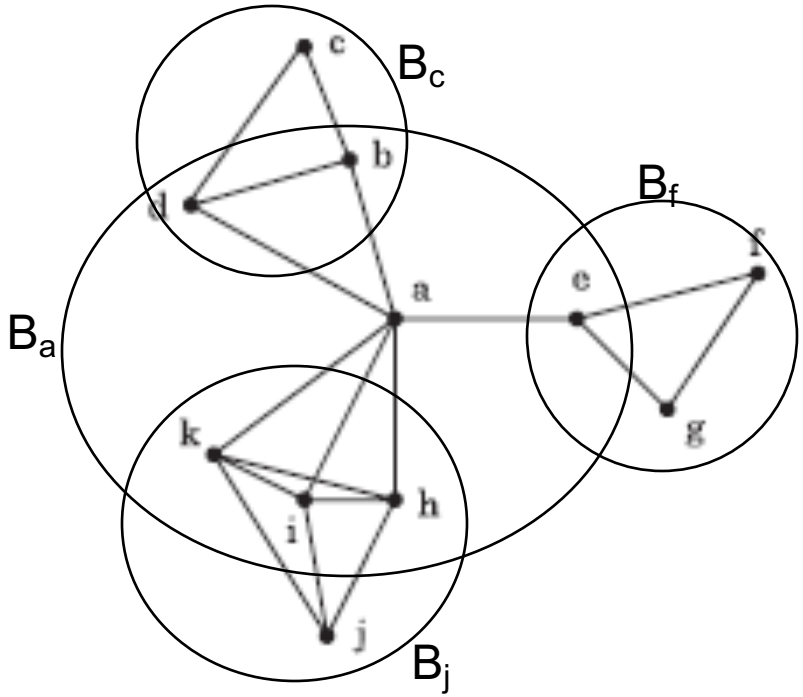
Greedy idea?

Set Cover

Greedy idea:

While υπάρχουν ακάλυπτες πόλεις:

Διάλεξε το υποσύνολο (πόλη) με το μεγαλύτερο πλήθος ακάλυπτων στοιχείων



1. S_a
2. S_f (or S_g)
3. S_c
4. S_j

C=4 (# συνόλων)

OPT=?

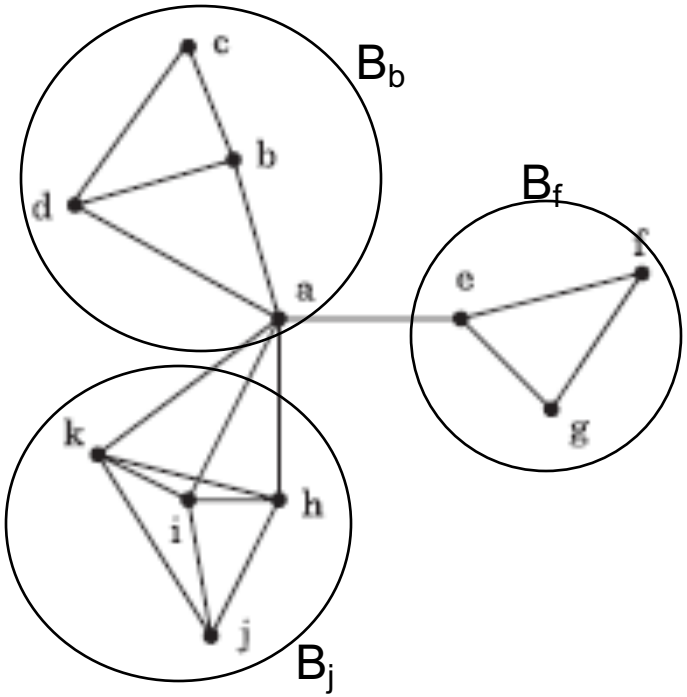
Is the greedy optimal?

Set Cover

Greedy idea:

While υπάρχουν ακάλυπτες πόλεις:

Διάλεξε το υποσύνολο (πόλη) με το μεγαλύτερο πλήθος ακάλυπτων στοιχείων



Greedy is NOT optimal

1. S_b
2. S_f
4. S_j

OPT=3

Set Cover

Greedy idea:

```
While υπάρχουν ακάλυπτες στοιχεία του U (πόλεις):
    Διάλεξε το υποσύνολο (πόλη) με το μεγαλύτερο πλήθος
    ακάλυπτων στοιχείων
```

This greedy is NOT optimal, but how good is ?

OPT: min # of subsets

n_t : the # of uncovered elements after the t-th iteration of greedy

n_0 = n (=|U|)

Key point: After any iteration t, there is subset covering at least n_t /OPT of uncovered elements n_t

Proof: the uncovered elements n_t are covered by the OPT subsets
hence, there is subset covering at least n_t /OPT of them
(otherwise, the OPT # of subsets doesn't cover all elements)

Set Cover

This greedy is NOT optimal, but how good is ?

Key point: After any iteration t, there is subset covering at least n_t /OPT of uncovered elements n_t

Thus, $$n_t \le n_{t-1} - \frac{n_{t-1}}{OPT} = n_{t-1}\left(1 - \frac{1}{OPT}\right)$$

and solving this recurrence with iterative substitutions we get

$$n_t \le n_{t-1}\left(1 - \frac{1}{OPT}\right) \le n_{t-2}\left(1 - \frac{1}{OPT}\right)^2 \le n_{t-3}\left(1 - \frac{1}{OPT}\right)^3 \le \ldots \le n_0\left(1 - \frac{1}{OPT}\right)^t$$

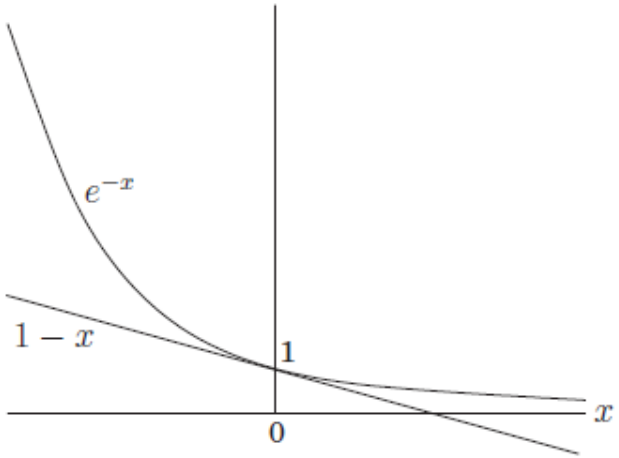
that is $$n_t \le n_0\left(1 - \frac{1}{OPT}\right)^t$$

Set Cover

This greedy is NOT optimal, but how good is ?

$$n_t \le n_0\left(1-\frac{1}{OPT}\right)^t = n_0(1-x)^t, \quad \text{where } x = \frac{1}{OPT}$$

but $(1-x) < e^{-x}, x \neq 0$



$$\text{Hence,} \quad n_t \le n_0\left(e^{-1/OPT}\right)^t = n_0 e^{-t/OPT}$$

$$\text{that is,} \quad n_t < n e^{-t/OPT}, \text{since } n_0 = n$$

Set Cover

This greedy is NOT optimal, but how good is ?

$$n_t < n e^{-t/OPT}$$

For $t = OPT \ln n$

$$\text{we get } n e^{-t/OPT} = n e^{-OPT \ln n / OPT} = n e^{-\ln n} = n \frac{1}{e^{\ln n}} = n \frac{1}{n} = 1,$$

that is $n_t < 1$, and hence all elements are covered

and the greedy returns a solution of $C = t$ subsets

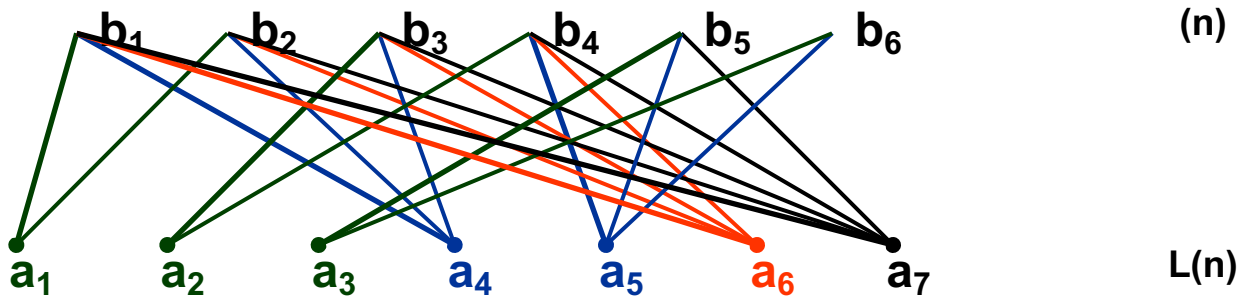
(a subset is selected in each one of the t iterations)

$$\text{Therefore, } \frac{t}{OPT} = \frac{C}{OPT} \leq \ln n$$

This greedy returns a solution AT MOST ln n times an optimal one

Vertex Cover (VC)

The O(logn) ratio of Greedy-best-node is tight



- Partition b-nodes into pairs, triples, quadtuples,…,(n-1)tuples
- Connect the nodes in each i-tuple above with a new a-node

$$L(n) = \sum_{j=2}^{n-1} \left\lfloor \frac{n}{j} \right\rfloor \leq \sum_{j=1}^{n} \frac{n}{j} \leq n \sum_{j=1}^{n} \frac{1}{j} = nO(\log n)$$

Vertex Cover (VC)

The O(logn) ratio of Greedy-best-node is tight

C = {a_7,a_6,a_5,a_4,a_4,a_2,a_1}
OPT = {b_1,b_2,b_3,b_4,b_5,b_6}

$$\frac{C}{OPT} = \frac{L(n)}{n} = O(\log n)$$

n	6	10	30	100	1000	…
C/OPT	2.17	2.6	3.67	4.8	7	…

Greedy-best-node is not a constant approximation algorithm

The O(logn) ratio of Greedy-best-set for SET COVER is also tight - why**?**

(Weighted) Set Cover

WEIGHTED SET COVER (WSC)

I: a set U of n elements

a family F={S_1, S_2, …,S_m} of subsets of U,

a weight w(S_i) for each set S_i

Q: Find a minimum weight subset C $\subseteq$ F covering all elements of U,

i.e. $\bigcup_{S_i \in C} S_i = U$ and $W = \sum_{S_i \in C} w(S_i)$ is minimized

SET COVER (SC): $w(S_i) = 1, \forall S_i \in F,$ that is $W = \sum_{S_i \in C} w(S_i) = |C|$

Find a minimum size subset C $\subseteq$ F covering all elements of U,

i.e. $\bigcup_{S_i \in C} S_i = U$ and $|C|$ is minimized

Weighted Set Cover (WSC)

Greedy-best-set

C := ∅ ;

while C ≠ U do

{ choose the best set S;

remove S from F;

C:= C U S ; }

C: elements covered before iteration i

S: Set chosen at iteration i

What means best set ?

S covers $|S\text{-}C|$ new elements

Covering those $|S\text{-}C|$ elements costs $w(S)$

Covering each element $x \in S - C$ costs $\dfrac{w(S)}{|S - C|}$

Best set: this with the smallest cost ratio $\dfrac{w(S)}{|S - C|} = p(x)$

Weighted Set Cover (WSC)

Greedy-best-set (cont.)

Let $x_1, x_2, \ldots x_k, \ldots x_n$ be the order in which U's elements are covered

$S_1, S_2, \ldots S_i, \ldots$ be the order in which F's sets are chosen

Set S_i covers element x_k

Claim: $p(x_k) \le \frac{OPT}{n-k+1}$

- $C = \bigcup_{j=1}^{i-1} S_j$: elements covered by iterations 1,2,…,i-1
- U-C: uncovered elements before iteration i
- |U-C| ≥n-k+1, since element x_k is covered in iteration i
- Those U-C elements are covered in the optimal solution with cost ≤ OPT
- There is an element x ∈ U-C of cost $\le \frac{OPT}{|U-C|} \le \frac{OPT}{n-k+1}$

Weighted Set Cover (WSC)

Greedy-best-set (cont.)

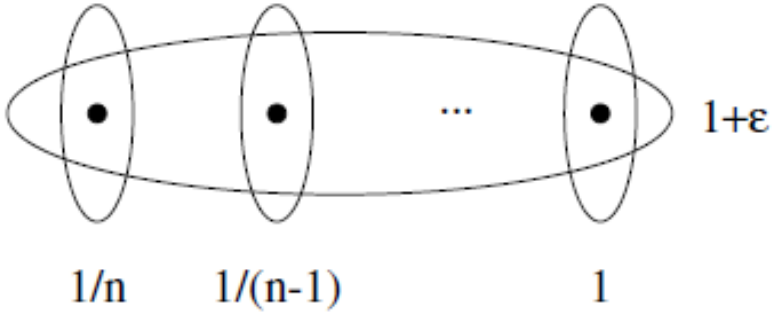
Claim: $p(x_k) \le \frac{OPT}{n-k+1}$

- There is a set $S_i, S_{i+1}, S_{i+2}, \ldots$ of cost ratio $\le \frac{OPT}{|U-C|} \le \frac{OPT}{n-k+1}$
- Among them the set S_i has the smallest cost ratio and covers x_k
- that is $p(x_k) \le \frac{OPT}{n-k+1}$ q.e.d.

$$W = \sum_{k=1}^{n} p(x_k) \le \sum_{k=1}^{n} \frac{OPT}{n-k+1} = OPT \sum_{i=1}^{n} \frac{1}{k} = OPT \cdot H_n = O(\log n) OPT$$

Weighted Set Cover (WSC)

Tightness



The greedy algorithm outputs the n singleton sets with total cost

$$W = \frac{1}{n} + \frac{1}{n-1} + \cdots + 1 = H_n$$

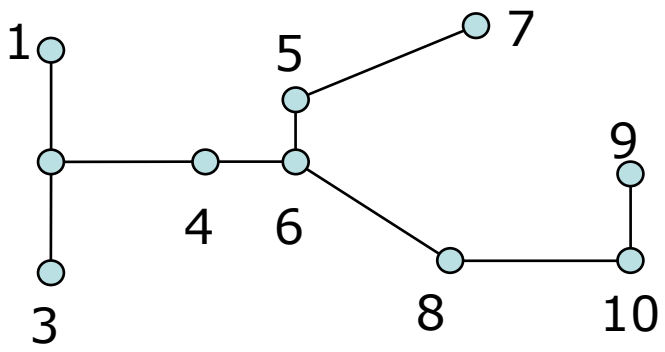
The optimal cover takes only the other set of cost $1+e$

Travelling Salesman Problem (TSP)

Δ-TSP: ρ=2

There is a 2-approximation algorithm for Δ-TSP

Find a minimum spanning tree, T, of G, of cost C(T)

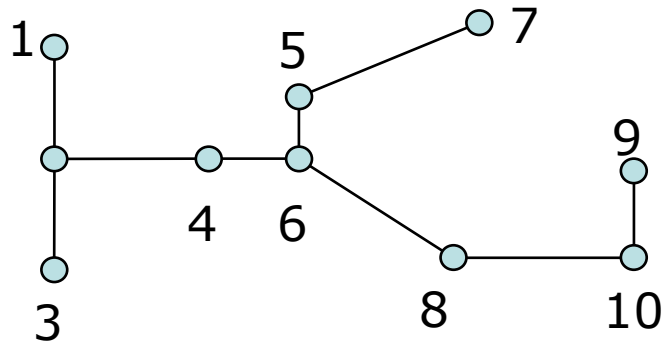


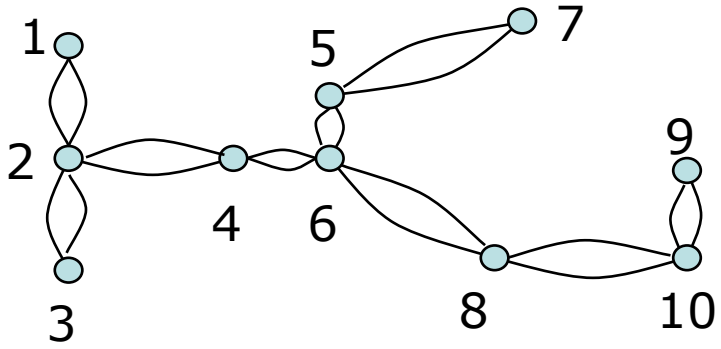
Let H* be an optimal tour of cost C(H*)

Let e be an edge of H* and T' be the rest of H* (this is a chain/tree)

$C(H^*) = w(e) + C(T') \geq w(e) + C(T) \geq C(T) \Rightarrow C(T) \leq C(H^*)$

Δ-TSP: ρ=2





Double the edges of T and let T'' be the obtained (multi)graph

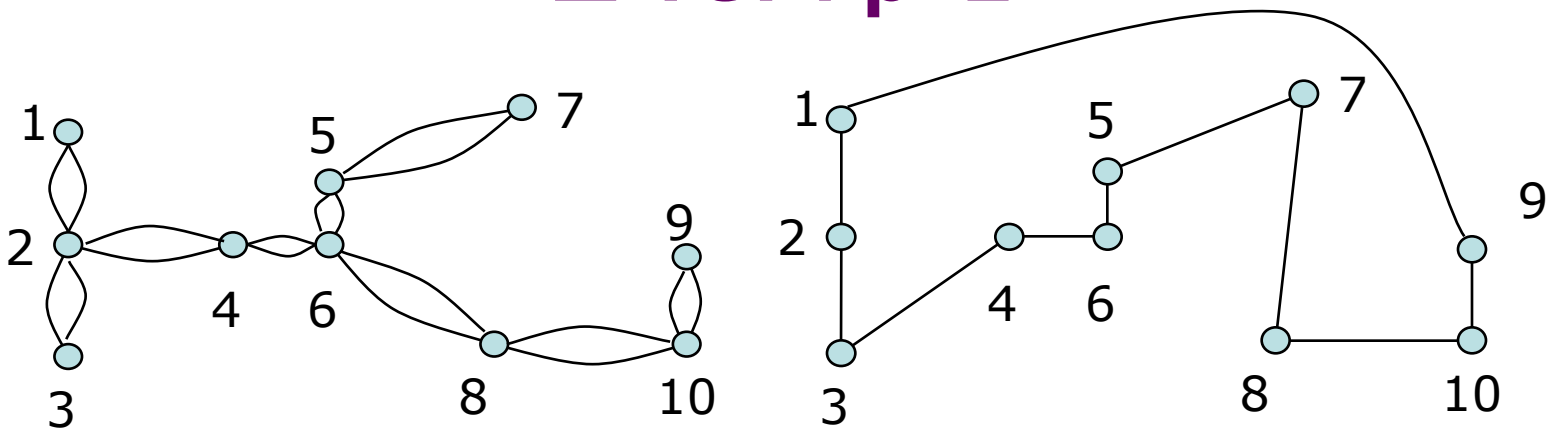
All vertices of T'' are of even degree

Find an Euler cycle, W, in T''

Euler cycle W: 1, 2, 3, 2, 4, 6, 5, 7, 5, 6, 8, 10, 9, 10, 8, 6, 4, 2, 1

W traverses each edge of T twice: $C(W) = 2\,C(T) \leq 2\,C(H^*)$

Δ-TSP: ρ=2



Find a tour H by shortcutting W:

H: 1, 2, 3, ~~2~~, 4, 6, 5, 7, ~~5~~, ~~6~~, 8, 10, 9, ~~10~~, ~~8~~, ~~6~~, ~~4~~, ~~2~~, 1

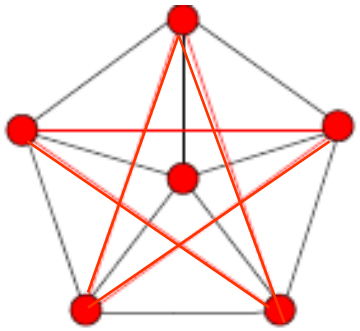
$C(H) \leq C(W)$, because of the triangle inequality

$C(H) \leq C(W) \leq 2\, C(H^*) \Rightarrow C(H) / C(H^*) \leq 2$

QUESTION: What is the complexity of this algorithm ?

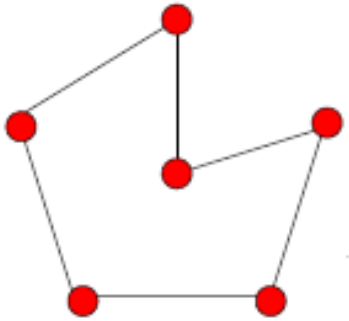
Δ-TSP: Tightness of ρ=2

Example



Complete graph K_n
Red edges: w =2
Other edges: w=1

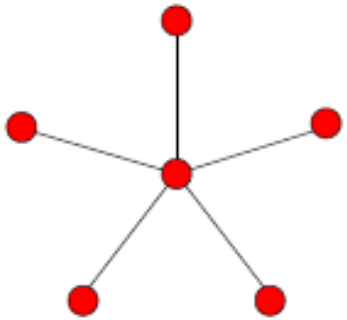
Optimal tour



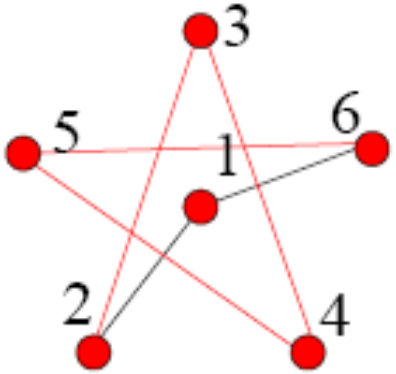
C(H*) = n

Δ-TSP: Tightness of ρ=2

Minimum MST



Solution



C(H) = (n-2)*2 + 2*1 = 2n-2

Hence, C(H) / C(H*) = (2n-2) / n→ 2

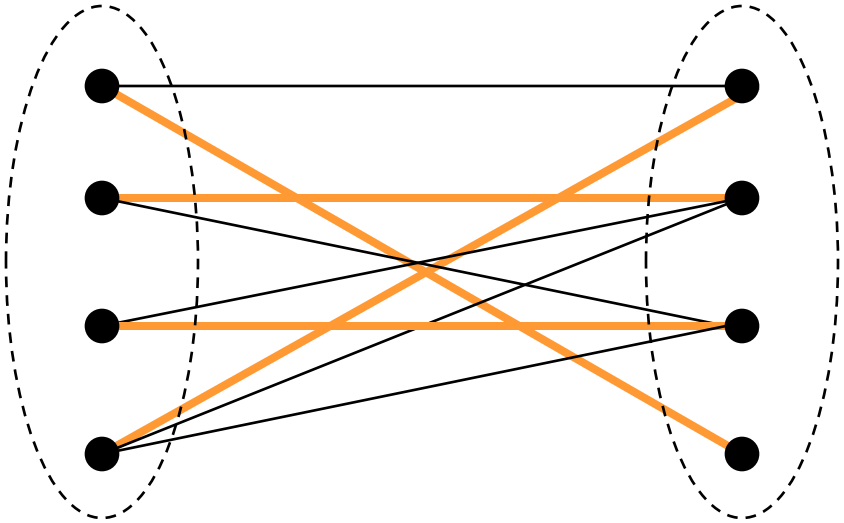
Matching problems

Matching in a graph G=(V,E):
A subset M $\subseteq$E of edges s.t. no two edges in M have a vertex in common.

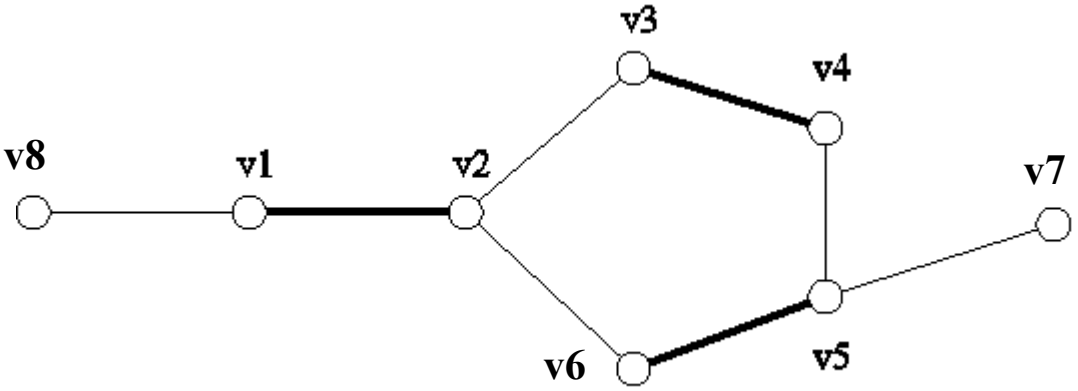
- Maximal: it is not a subset of another matching,
  i.e. it can not be extended
- Maximum: a matching of maximum cardinality, |M|,
  i.e. a maximum maximal matching
- Perfect: a matching of cardinality |M|=n/2
  (it is defined only for graphs with even # of vertices)

Matching problems

Examples



a matching in a bipartite graph



a matching in a general graph

Matching problems

Matching problems

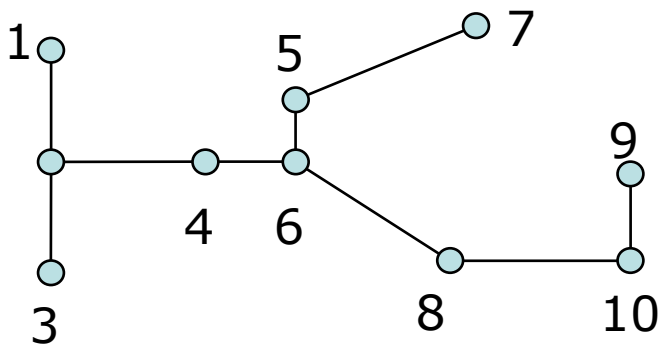
- Maximal matching: find a matching where no more edges can be added
- Maximum matching: find a matching with a maximum number of edges
- Perfect matching: find a matching where every vertex is matched
  (if |V| is even and if one exists)
- Maximum weight matching: given a weighted graph, find a matching with maximum possible total weight
- Minimum weight perfect matching: given a weighted graph, find a perfect matching with minimum cost

All matching problems are polynomial, even their weighted versions

Δ-TSP: ρ=1.5

There is a 1.5-approximation algorithm for Δ-TSP [Chistofides 1976]

Find a minimum spanning tree, T, of G, of cost C(T)

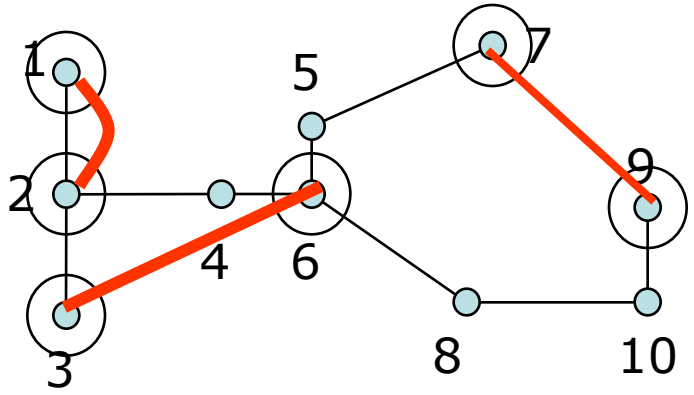


Let H* be an optimal tour of cost C(H*)

Let e be an edge of H* and T' be the rest of H* (this is a chain/tree)

$C(H^*) = w(e) + C(T') \geq w(e) + C(T) \geq C(T) \Rightarrow C(T) \leq C(H^*)$

Δ-TSP: ρ=1.5



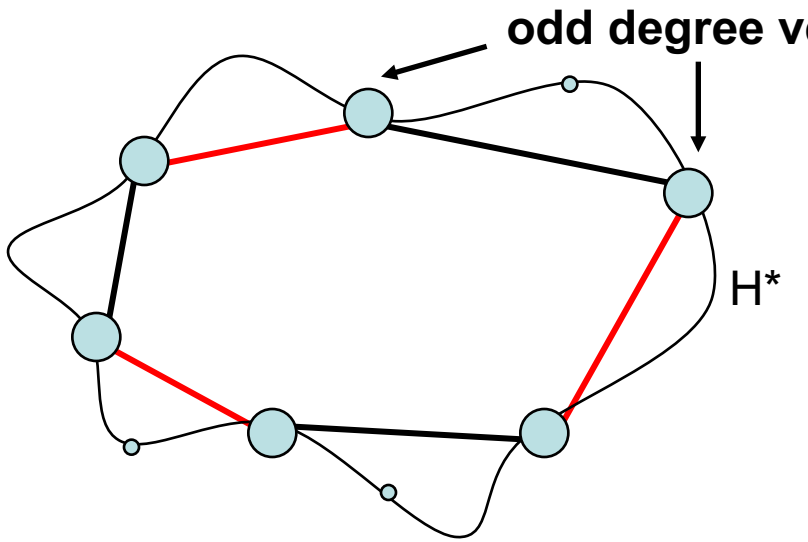
Find the set of vertices of T of odd degree (S)

S contains an even number of vertices – why ?

Consider the graph G_S induced by S

Find a minimum weight perfect matching, M, in G_S

Δ-TSP: ρ=1.5



Consider the odd degree vertices in S in the order that they appear in H*

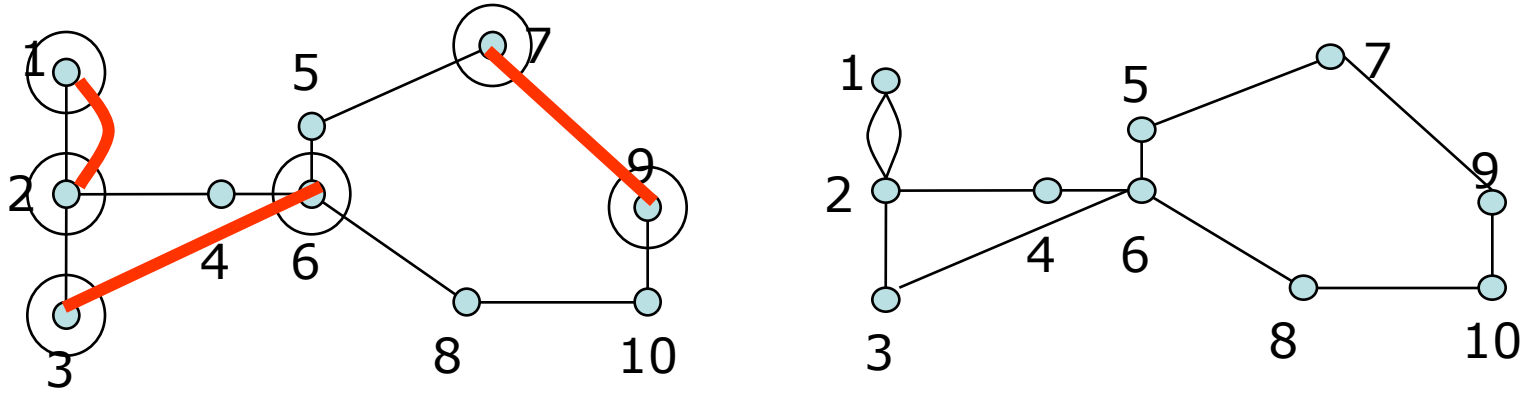
Consider the red, M_1, and the black, M_2, matchings between them

Then $C(H^*) \geq C(M_1) + C(M_2)$, by the triangle inequality

Also $C(H^*) \geq C(M_1) + C(M_2) \geq C(M) + C(M)$, since M is optimal

Hence, $C(M) \leq C(H^*) / 2$

Δ-TSP: ρ=1.5



Add the edges of M to T and let T'' be the obtained (multi)graph

All vertices of T'' are of even degree

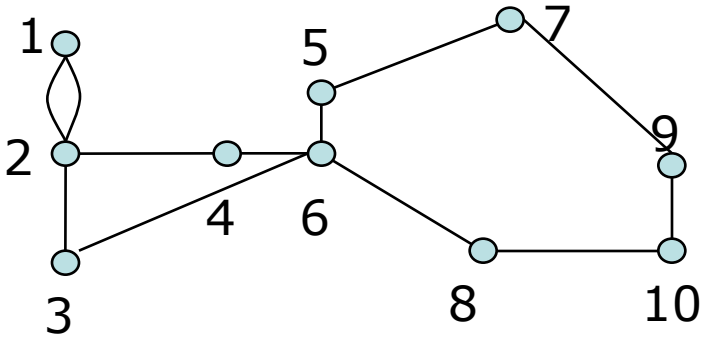
Find an Euler cycle, W, in T''

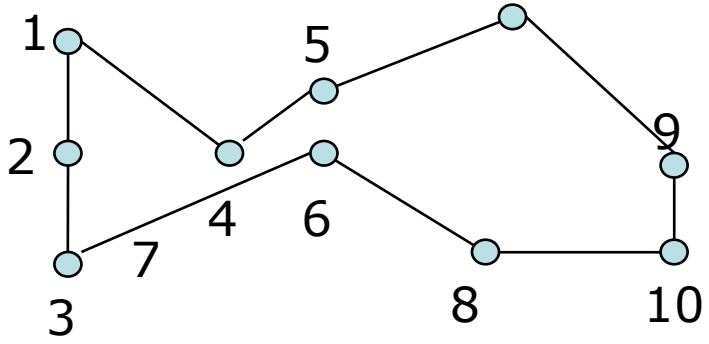
Euler cycle W: 1, 2, 3, 6, 8, 10, 9, 7, 5, 6, 4, 2, 1

W traverses each edge of T once:

$$C(W) = C(T) + C(M) \leq C(H^*) + C(H^*) / 2 = 1.5\, C(H^*)$$

Δ-TSP: ρ=1.5





Find a tour H by shortcutting W:

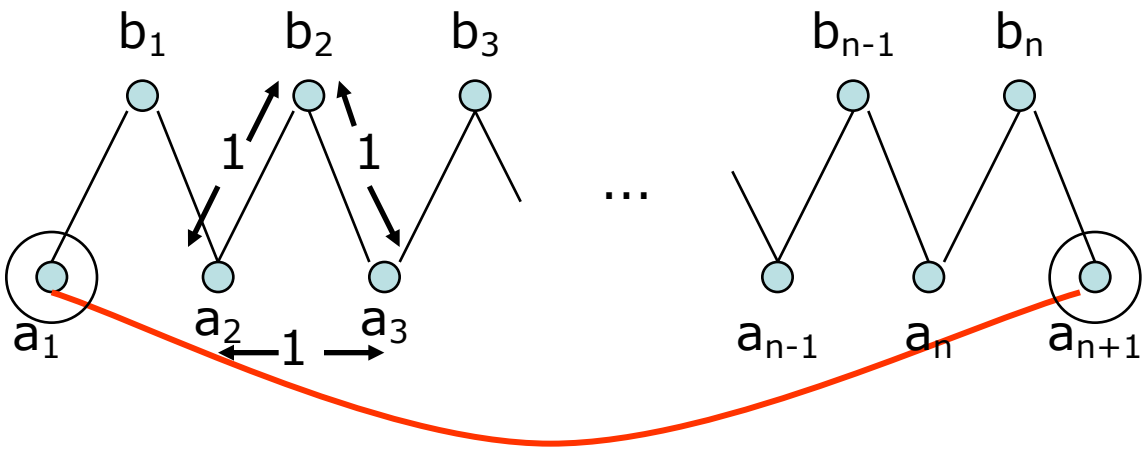
H: 1, 2, 3, 6, 8, 10, 9, 7, 5, ~~6~~, 4, ~~2~~, 1

$C(H) \leq C(W)$, by of the triangle inequality

$C(H) \leq C(W) \leq 1.5\ C(H^*) \Rightarrow C(H) / C(H^*) \leq 1.5$

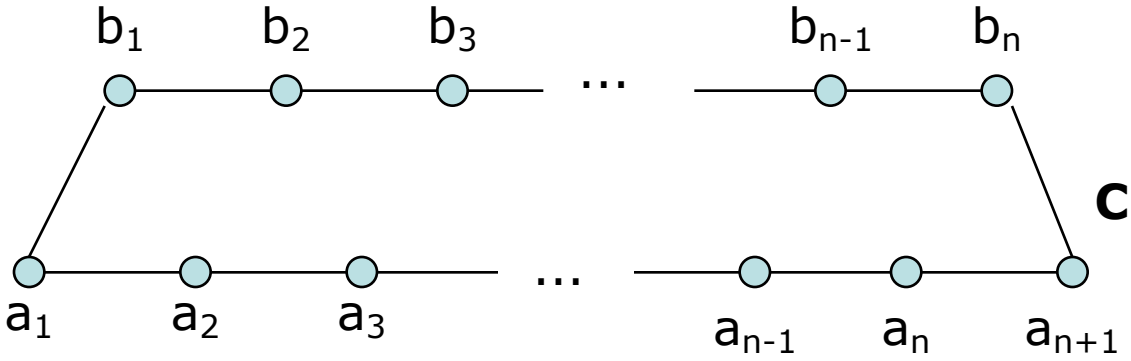
QUESTION: What is the complexity of this algorithm ?

Δ-TSP: Tightness of ρ=1.5



T, S, M, T'', H

C(H)=n+n+n=3n



H*

C(H)=n+(n-1) +2 =2n+1

C(H) / C(H*) → 3/2