Exhaustive search(brute force)

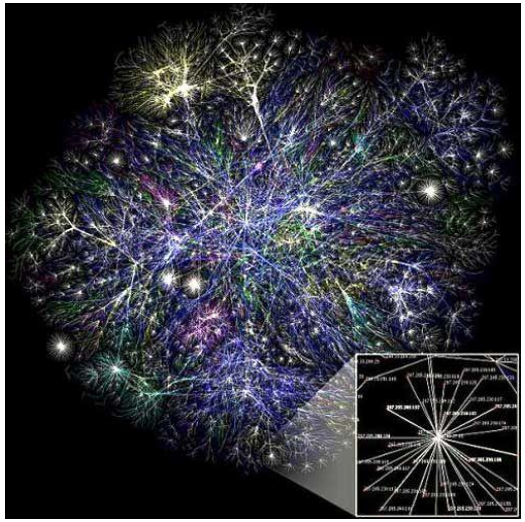
Yulia Burkatovskaya

Outline

- Optimization in graph theory.
- Generation of subsets. Shortest cover problem.
- Generation of permutations. Travelling salesman problem.
- Generation of combinations. Facility location problem.

Optimization in graph theory

- Discrete structure G(V,E) (V- vertices, E-edges)
- $f(X) \rightarrow \min_X \left(\max_X (X \subset V, X \subset E) \right)$.
- No necessary and sufficient conditions for the minimum (maximum).
- Exhaustive search to find X which provides the optimum.



Generation of subsets

- **Boolean vector** is a vector with the components equal to 0 or 1.
- Boolean vector represents a subset:
  - Let $M = \{m_1, \dots, m_k\}$ be a set;
  - Let $A \subseteq M$;
  - Vector $\alpha = a_1 \dots a_k$ represents subset A, $a_j = \begin{cases} 1, & m_j \in A; \\ 0 & m_j \notin A. \end{cases}$
  - $M = \{1,2,3,4,5,6,7,8,9\}$;
  - $A = \{2,3,5,7\}$
  - $\alpha = 011010100$
  - 2^k subsets

Generation of subsets

- Generation in the natural order (ascending numbers)
  - 000000
  - 000001
  - 000010
  - …
  - 111110
  - 111111

- Add 1 to the previous number

Generation of subsets

- Pseudocode
  - `while (B[0]!=0)`
    - `{`
      - `i=k;`
      - `while (B[i]==1)`
        - `{`
        - `B[i]:=0; //dot-and-carry-one`
        - `i=i-1;`
        - `}`
      - `B[i]:=1; //add 1`
      - `For (i=1; i<=N; i++) //print a subset`
        - `if B[i]=1 then cout << i;`
    - `cout << endl;`
    - `}`

Generation of subsets

- Generation in the Grey code (two successive values differ in only one bit)
  - 000000
  - 000001
  - 000011
  - 000010
  - 000110
  - 000111
  - 000101
  - 000100
  - …
- Convenient to calculate an objective function (sum, product…)

Generation of subsets

```
unsigned int BinaryToGray(unsigned int num)
{
return num ^ (num >> 1);
}

unsigned int GrayToBinary(unsigned int num)
{
unsigned int mask = num >> 1;
while (mask != 0)
  {
  num = num ^ mask;
  mask = mask >> 1;
   }
return num;
}
```

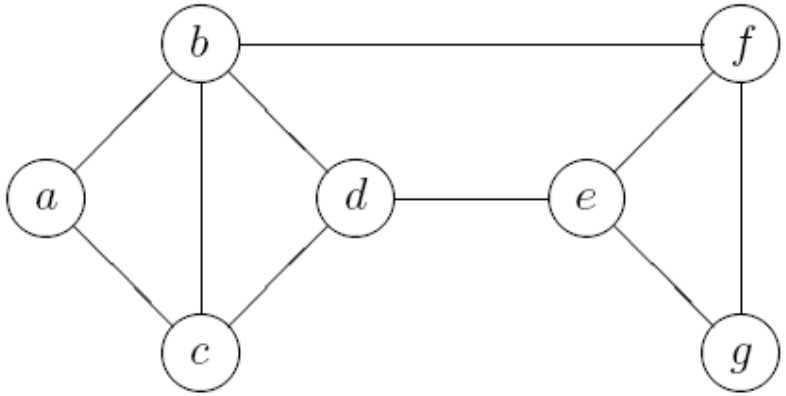
Covering sets

A **vertex covering set (vertex cover)** is a set of vertices of G covering all edges of G.

Example.

{a,b,d,e,f} – a vertex covering set.

{ab,ac,de,fg} – an edge covering set.



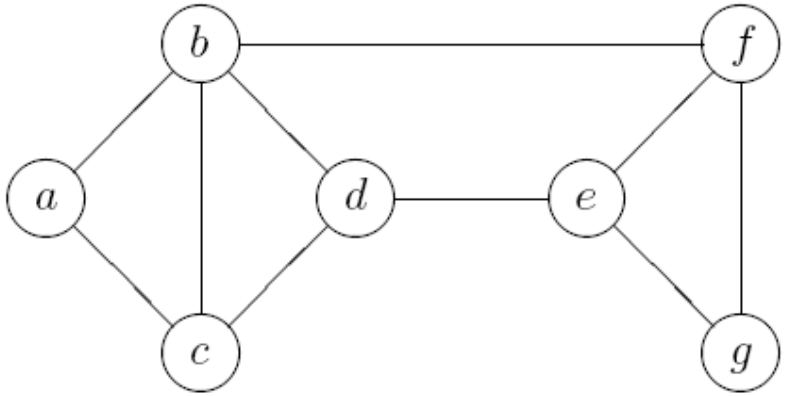
Covering sets

A cover is called **shortest** when it contains the smallest possible number of vertices (edges).

Example.

{a,b,c,d,e,f} is not a shortest vertex cover

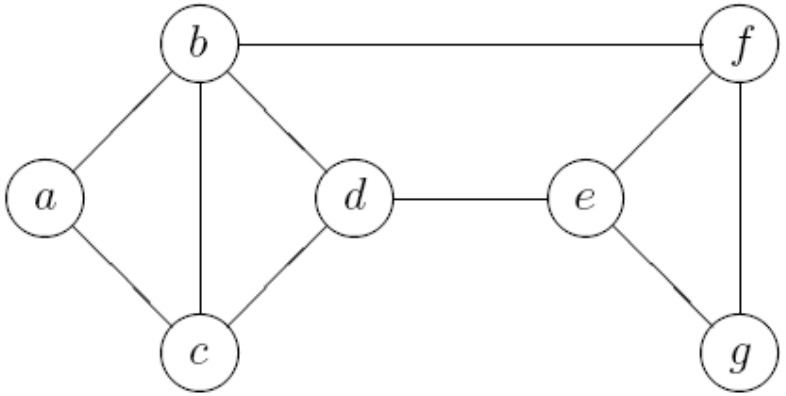
{b,c,e,g} is a shortest vertex cover.



Covering sets

The **vertex cover number** α_0 of a graph G is the size of a shortest vertex cover in a graph, i.e., the minimum number of vertices covering all edges.

Example. α_0 =4, {b,c,e,f} – shortest vertex cover.



Covering sets

To find the shortest cover:

- Generate all subsets
- Check if they are covers
- Find a minimum-cardinality cover

Generation of permutations

- **Permutation** is the act of **arranging** the members of a set into a sequence or order, or, if the set is already ordered, rearranging its elements.
- 123456789
- 123456798
- 123456879
- 123456798
- …
- 512398764
- 512436789
- …
- 987654321
- **Natural order, n! permutations**

Generation of permutations

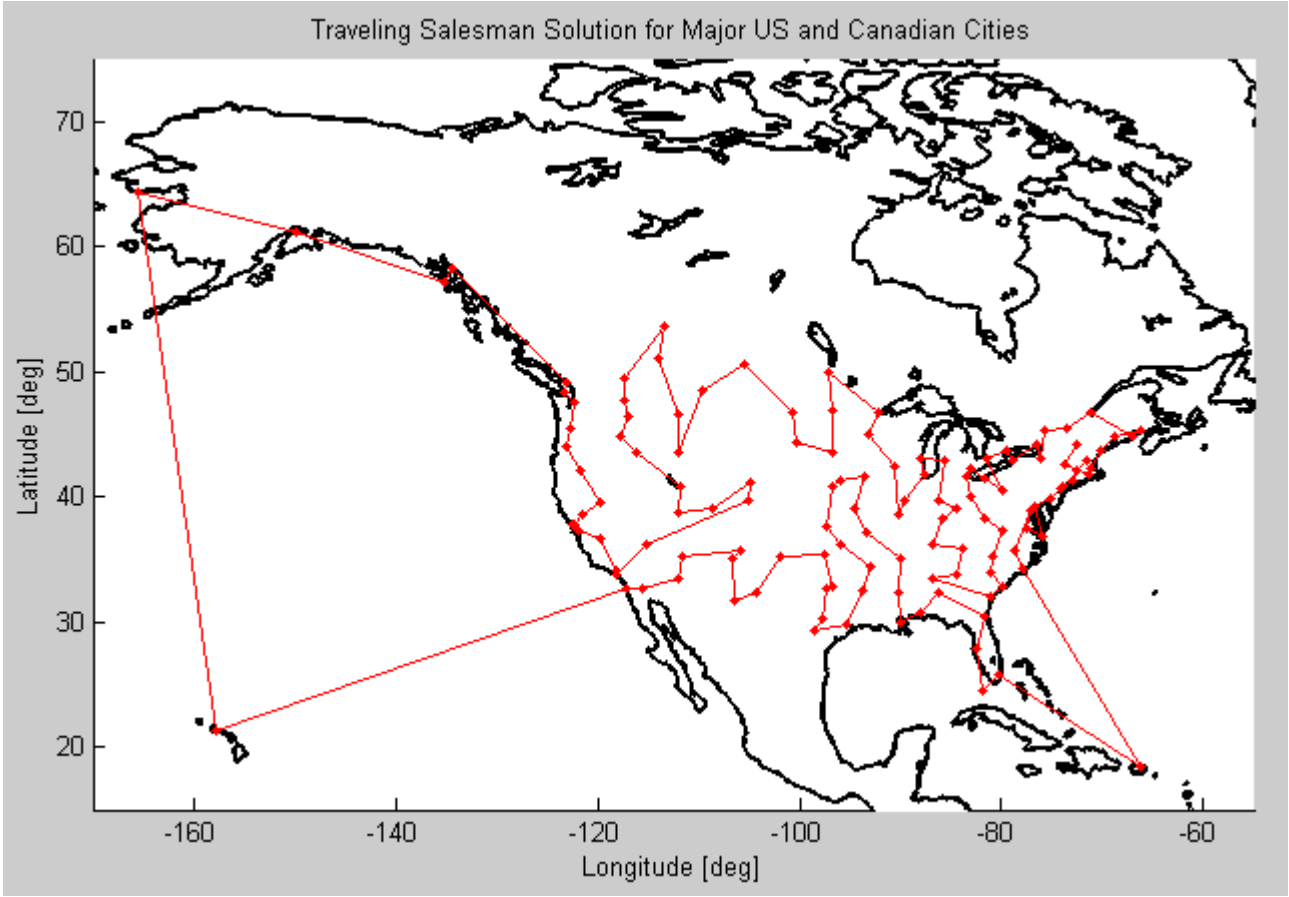
- **Narajana algorithm** (XIV century)
- Find maximum j: $a[j] < a[j+1]$ ($a[j+1], \dots, a[n]$ are in descending order);
  - 312**5**98764
- Find maximum $l > j$: $a[l] > a[j]$;
  - 312**5**987**6**4
- Swap $a[j]$ and $a[l]$ (after that $a[j+1], \dots, a[n]$ are still in descending order);
  - 312**6**987**5**4
- Rearrange $a[j+1], \dots, a[n]$ in reverse order, swapping the elements (after that $a[j+1], \dots, a[n]$ are in ascending order).
  - 3126**45789**

Traveling Salesman Problem

The **travelling salesman problem** (**TSP**) asks the following question:

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?

Traveling Salesman Problem



Traveling Salesman Problem

Consider graph G(V,E) in which each vertex represents a city and each edge represents a road connecting two cities, and $d(x,y)$ is the weight of the edge (x,y) standing by the distance between cities x and y.

Problem: finding a cycle in G which visits each vertex once in minimum total distance.

This problem is **NP-hard**.

Solution: generate n! **permutations**

Generation of combinations

- A **combination** is a selection of items from a collection, such that (unlike permutations) the order of selection does not matter.
- $\binom{k}{n} = \frac{n(n-1)\ldots(n-k+1)}{k(k-1)\ldots 1} = \frac{n!}{k!(n-k)!}$
- $\binom{2}{5} = 10$
- `12`
- `13`
- …
- `24`
- `25`
- …
- `45`

Generation of combinations

- A Boolean vector represents a combination
- 11000
- 10100
- 10010
- 10001
- 01100
- 01010
- 01001
- 00110
- 00101
- 00011

Generation of combinations

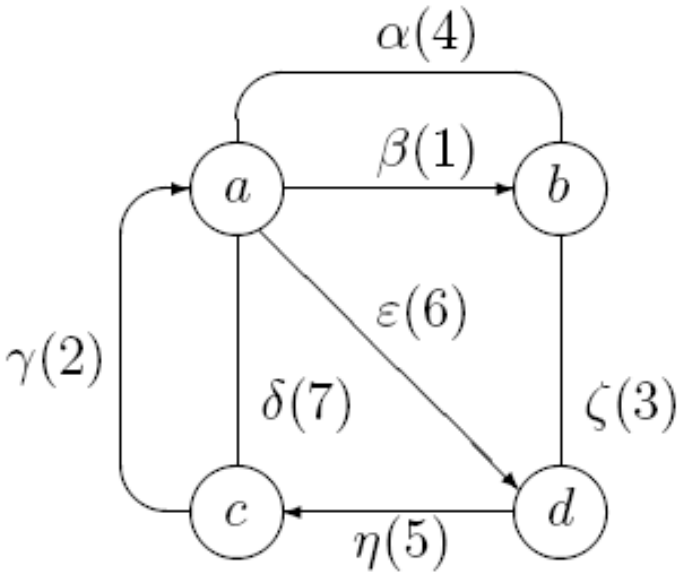
- **Pseudocode**
- `bool next_combination (vector<int> & a, int n)`
- `{`
  - `int k = (int)a.size();`
  - `for (int i=k-1; i>=0; --i)`
    - `if (a[i] < n-k+i+1)`
    - `{`
    - `++a[i];`
    - `for (int j=i+1; j<k; ++j)`
      - `a[j] = a[j-1]+1;`
    - `return true;`
    - `}`
- `return false;`
- `}`

Vertex-vertex distance

The **vertex-vertex distance** between vertices i and j (notation $d(i,j)$) is the weight of the shortest path $\langle i,j \rangle$.

It can be found by the Floyd–Warshall algorithm.

Example.



	a	b	c	d
a	0	1	7	4
b	4	0	8	3
c	2	3	0	6
d	7	3	5	0

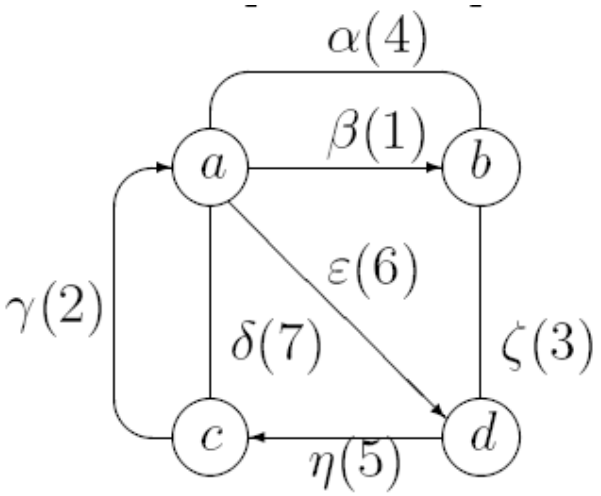
.

Center

A **center** of graph G is any vertex v of graph G such that

$$\mathrm{MVV}(v) = \min_j \mathrm{MVV}(j).$$

Example. Vertex *c* is the center.



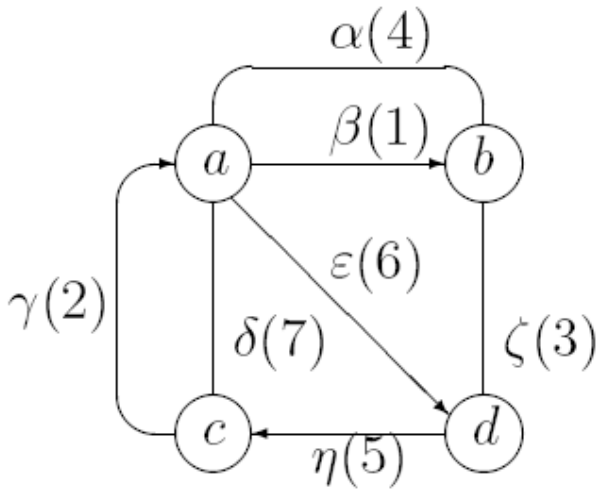
	a	b	c	d	$\mathrm{MVV}(v)$	
a	0	1	7	4	7	
b	4	0	8	3	8	
c	2	3	0	6	6	min
d	7	3	5	0	7	

Median

A **median** of graph G is any vertex v of graph G such that

$$\mathrm{TVV}(v) = \min_j \mathrm{TVV}(j).$$

Example. Vertex *c* is the median.



	a	b	c	d	$\mathrm{TVV}(v)$	
a	0	1	7	4	12	
b	4	0	8	3	15	
c	2	3	0	6	11	min
d	7	3	5	0	15	

Multicentres and multimedians

Let X_r be a subset of points of graph $G(V,E)$ containing r points.

Set-vertex distance $d(X_r,j)$ is the minimum distance between any one of the points in set X_r and vertex j; i.e.

$$d(X_r, j) = \min_{i \in X_r} d(i, j).$$

Absolute p-centre

Problems:

- (a) Find the optimal location anywhere on the graph of a given number (say p) of centres so that the distance required to reach the most remote vertex from its nearest centre is a minimum.
- (b) For a given "critical" distance, find the smallest number (and location) of centres so that all the vertices of the graph lie within this critical distance from at least one of the centres.

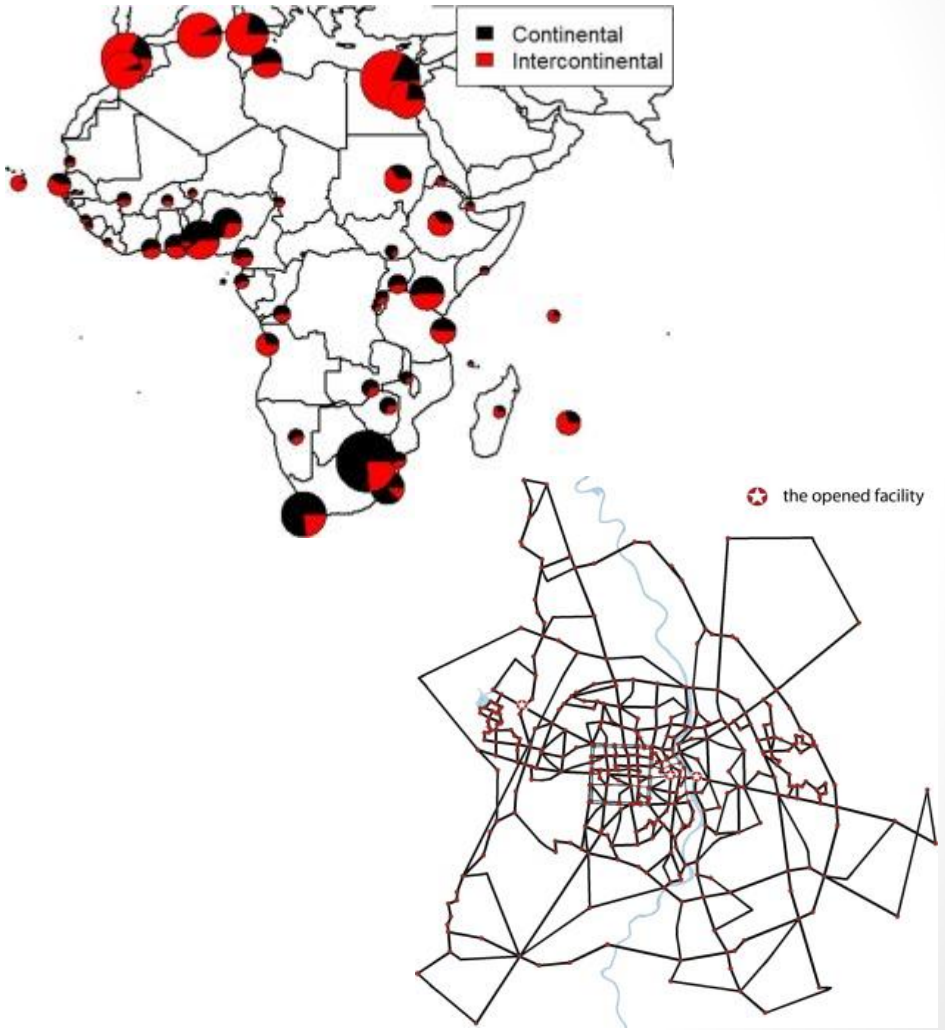
P-median

Problems:

- (a) Find the optimal location anywhere on the graph of a given number (say p) of medians so that the total distance required to reach all the vertices from its nearest median is a minimum.
- (b) For a given "critical" distance, find the smallest number (and location) of medians so that the total distance required to reach all the vertices from its nearest median lie within this critical distance.

Location problem

- Which ambulances or patrol cars can respond quickest to an incident?
- What market areas does a business cover?
- Where can a business open a store to maximize market share?



Location problem

- **Solution:** generate all combination of vertices, choose the best one
- **NPhard**!