Summary so far

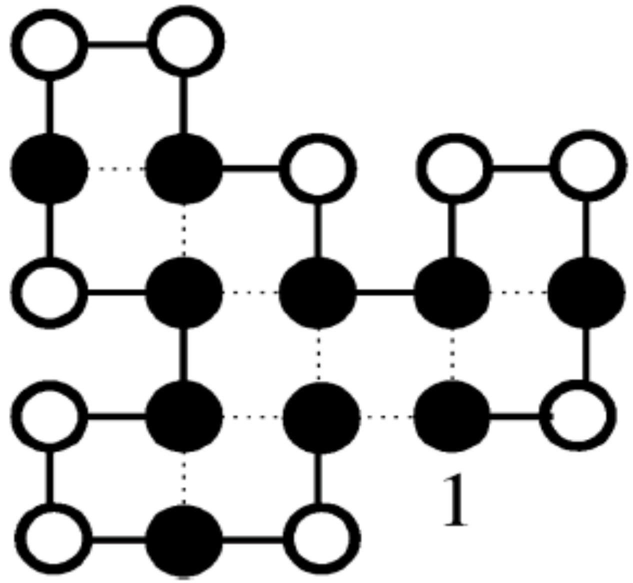
- We know everything about the complexity classes of decision problems.
- A “simple” interesting model, indeed.
- What if “my problem” is NOT a decision problem?
- How do I prove (un)tractability of other type of problems?

NP-hard problems: the idea.

- There are problems to which problems in NPC can be reduced, but that are not NP-complete because it is not possible to prove that they belong to NP.
- These problems are NP-hard:
  - They are at least as difficult as NP-complete problems.
  - They are not in NP.

Protein folding

- Lattice model assumes amino acids are of two types: hydrophobic, which are black, and hydrophilic, which are white
- They can take on discrete positions only
- The energy value of a fold is determined by the number of non-adjacent hydrophobic residues



Protein folding

- Finding the optimal fold in the 2D lattice is NP-hard.
- There are at least an exponential number of possible folds (as demonstrated by the staircase folds).

Optimization problems

- An optimization problem is $(I_P, Sol_P, f_P, \{max/min\})$ where:
  - I_P is the set of instances of P.
  - Sol_P is the set of admissible solutions.
  - f_P is a measure of the goodness of a solution.
  - {max/min} tells whether it is desiderable to maximize or minimize f_P.
- The set of optimal solutions of an instance $i \in I_P$ is the subset of Sol_P that maximizes/minimizes f_P.
- You will meet more optimization problems that decision problems...

An example: minimun vertex cover

- minumum vertex cover:
  - INPUT: A graph G=(V,E)
  - OUTPUT: A subset of nodes $U \subseteq V$ of minimum size such that for each $(i,j) \in E$, either $i \in U$, or $j \in U$.
    - The set I_P is the set of all possible graphs.
    - The set Sol_P is the subsets U of V such that for each $(i,j) \in E$, either $i \in U$, or $j \in U$; i.e. It is a *vertex cover* of G.
    - The function f_P is $|U|$, and it has to be minimized.

Complexity Theory of Optimization Problems

- In bioinformatics they are more frequent than decision problems:
  - Parsimony in phylogeny.
  - Consensus models.
  - Sequences alignments.
  - Genomic distances.
  - Protein folding.
  - Fragment Assembly.
  - ...
- There is a whole theory somehow parallel and related to that of decision problems.

The class NPO

- An optimization problem $P = (I_P, Sol_P, f_P, \{max/min\})$ belongs to the class NPO if and only if:

  - The set I_P is recognizable in polynomial time.
  - All solutions have polynomial size and can be verified in polynomial time.
  - The function f_P can be computed in polynomial time.

Minimum Vertex Cover ∈ NPO

- Minimum Vertex Cover is such that:
  - The set of the instances is that of undirected graphs, recognizable in polynomial time.
  - Any solutions is a subset U of V, hence of polynomial size; whether it is a vertex cover can be verified in polynomial time by checking for all edges if each one of them involves a node in U.
  - The cost function is the size of U, that can be computed in polynomial time.

The class PO

- An optimization problem belongs to PO if it is in NPO and there exists a polynomial time algorithm that, for any instance $i \in I_P$, returns an optimal solution together with its value.
- In order to prove that a problem is in PO, one has to:
  - Show that the problem is in NPO.
  - Give a polynomial algorithm that finds always the optimal solution.

The class NP-hard

- An optimization problem Popt is NP-hard if, for any decision problem Pd $\in$ NP, Pd $\leq_p$ Popt.
- In other words, if, assuming that we have a polynomial solution for Pd, then we can have a polynomial solution for Popt as well.
- As with NP-completeness, rather than reducing from all problems in NP, it is enough to reduce from one NP-complete problem.

Tractable Optimization Problems

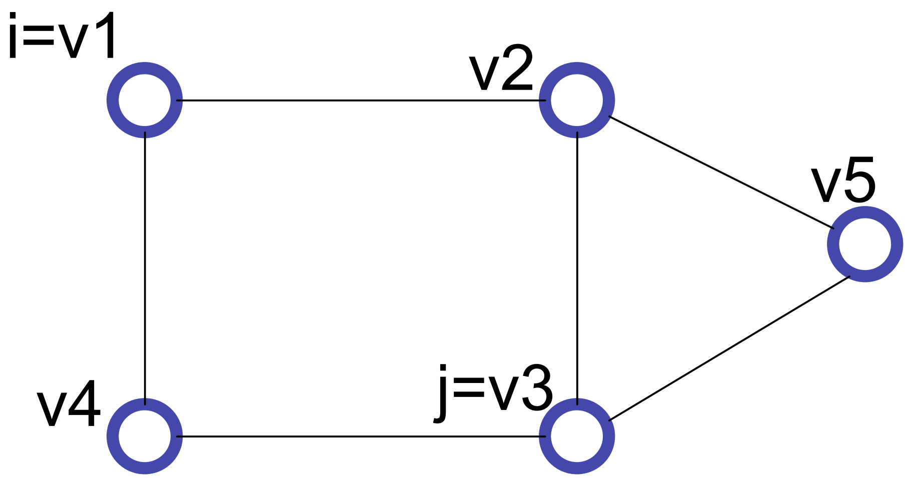
- Shortest Path:
  - INPUT: A graph G=(V,E), two nodes $i,j \in V$.
  - OUTPUT: The shortest path in G from i to j.

- Shortest Path $\in$ PO!

Shortest Path ∈ NPO

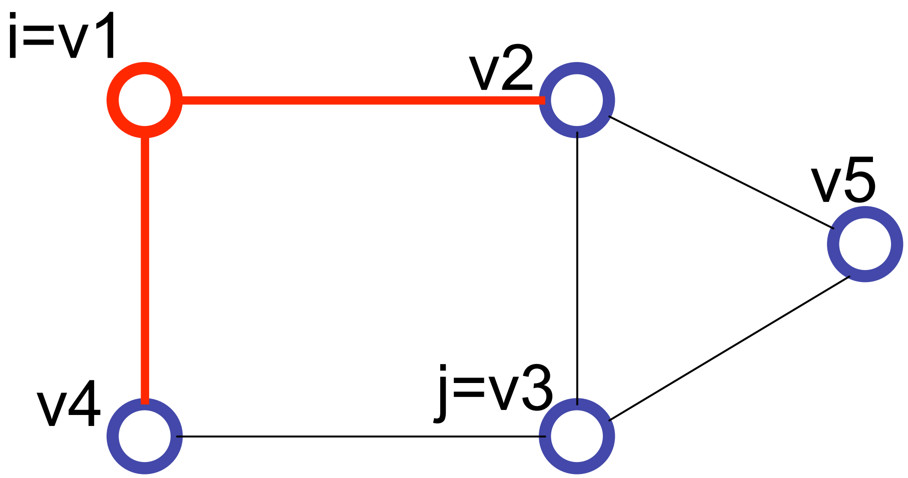
- Shortest Path:
  - INPUT: A graph G=(V,E), two nodes $i,j \in V$.
  - OUTPUT: The shortest path in G from i to j.
- We first need to show that Shortest Path ∈ NPO
  - The set of instance (graphs) is recognizable in polynomial time.
  - A solution is a set of nodes: polynomial size and verification.
  - Cost computable in polynomial time.

Polynomial solution of SP



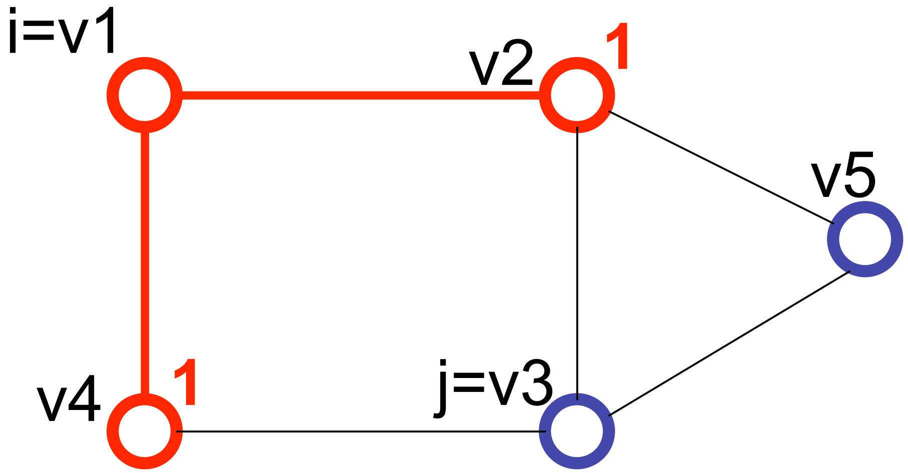
1. BFS starting from i;
2. First time you reach j it's done;

Polynomial solution of SP



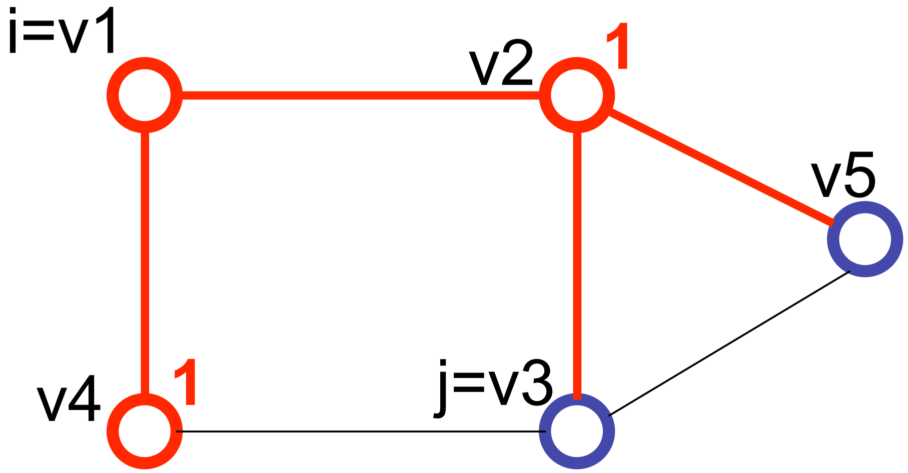
1. BFS starting from i;
2. First time you reach j it’s done;

Polynomial solution of SP



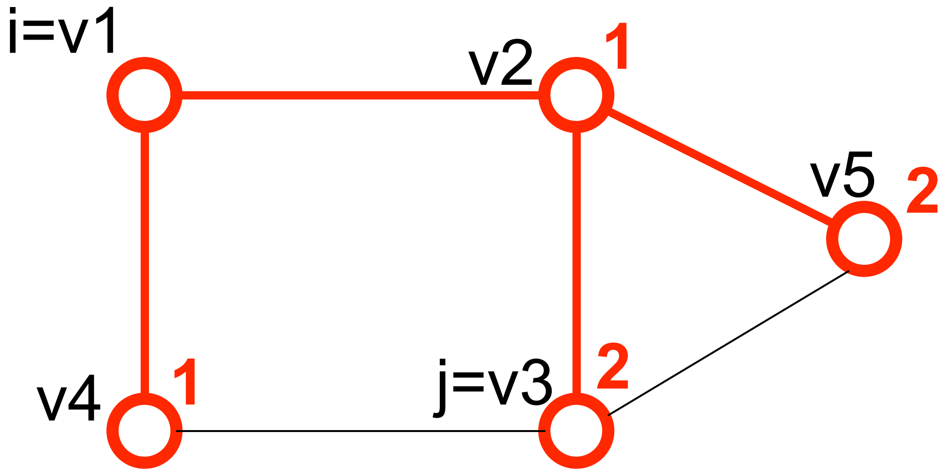
1. BFS starting from i;
2. First time you reach j it's done;

Polynomial solution of SP



1. BFS starting from i;
2. First time you reach j it's done;

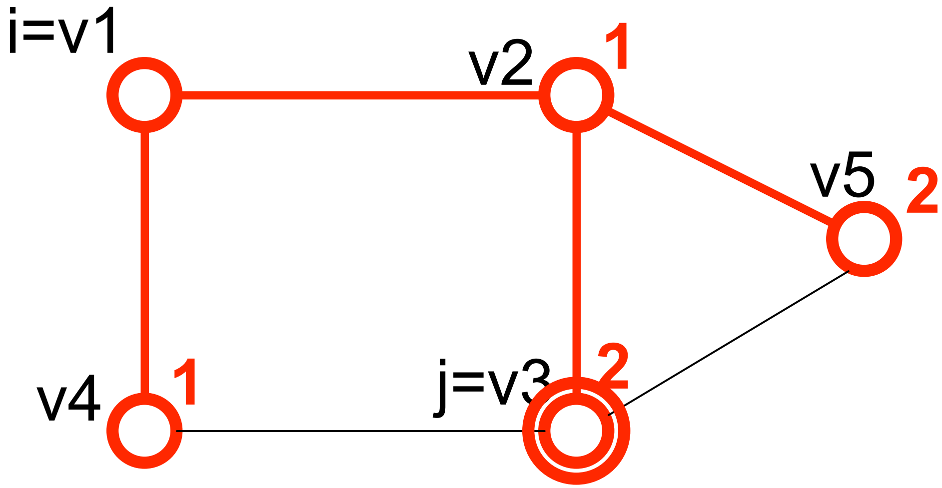
Polynomial solution of SP



1. BFS starting from i;
2. First time you reach j it’s done;

I don’t visit twice the same node

Polynomial solution of SP

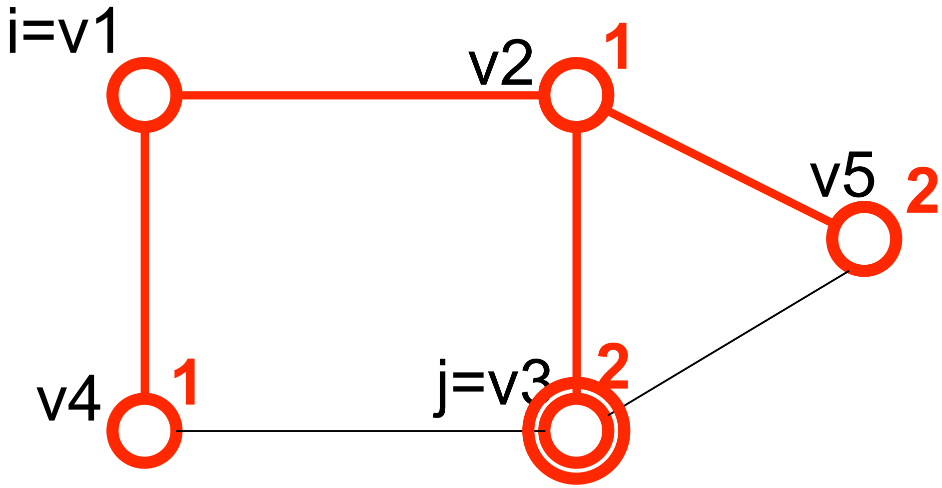


1. BFS starting from i;
2. First time you reach j it's done;

SP(i,j)=2

I don't visit twice the same node

Polynomial solution of SP



1. BFS starting from i;
2. First time you reach j it's done;

SP(i,j)=2

I don't visit twice the same node

⇓

In O(n+m) steps I am done

PO, NPO, NP-hard

- Clearly PO $\subseteq$ NPO.
- Is PO $\neq$ NPO?
- Do NP-hard problems belong to NPO \ PO?
- What are the relations with P, NP, NPC?

Decision problem associated to an optimization problem

- TSP:
  - INPUT: a graph with n nodes (cities) and weighted edges (distances).
  - OUTPUT: the cost of the path visiting all nodes having the minimum total weight (the fastest tour of all cities).
- TSPd:
  - INPUT: a graph with n nodes and weighted edges, and an integer k.
  - OUTPUT: is there a path visiting all nodes and having total weight at most k?
- TSPd is the *decision problem associated to the optmization problem* TSP.

Decision problem associated to an optimization problem

- Optimization problem:
  - INPUT: Instance x, set of admissible solutions, cost function f, {min/max}.
  - OUTPUT: A solution of x that {min/max}imizes f.
- Associated decision problem:
  - INPUT: As above plus an integer k.
  - OUTPUT: is there a solution of cost at most/least k?
- For any optimization problem in NPO, the corresponding decision problem is in NP.

P, NP and PO, NPO

- For any problem Popt in NPO, if the associated decision problem Pd is NP-complete, then Popt is NP-hard.

- See for example how we proved that TSP is NP-hard using that TSPd is NP-complete.

- If P $\neq$ NP, then PO $\neq$ NPO.