Algorithms and Complexity

NP-completeness

Julián Mestre

School of Information Technologies
The University of Sydney

THE UNIVERSITY OF SYDNEY

Reductions

We have seen a number of reductions in the last few lectures:

- Maximum matching → Maximum flow
- Minimum cut → Maximum flow
- Open-pit mining → Minimum cut
- Maximum number of disjoint paths → Maximum flow

In all these cases we reduced X to Y, where

- X = new problem
- Y = problem we already knew how to solve

Reducing X to Y is, in a sense, equivalent to saying

If "Y is easy" then "X is easy"

Reductions are double-edged swords

Reducing X to Y also gives us the following statement:

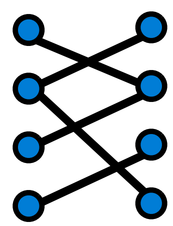
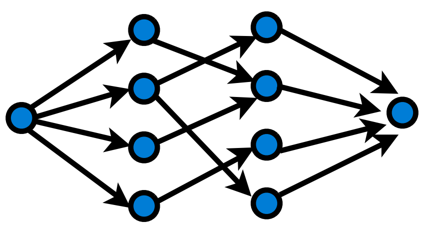
If "X is hard" then "Y is hard"

Our proof techniques do not allow to show unequivocally that a certain problem is "hard", but certain problems are widely believed to be "hard". Reductions allow us to transfer this belief.

Reducing X to Y gives us the following statement:

If "we believe that X is hard" then "we believe that Y is hard"

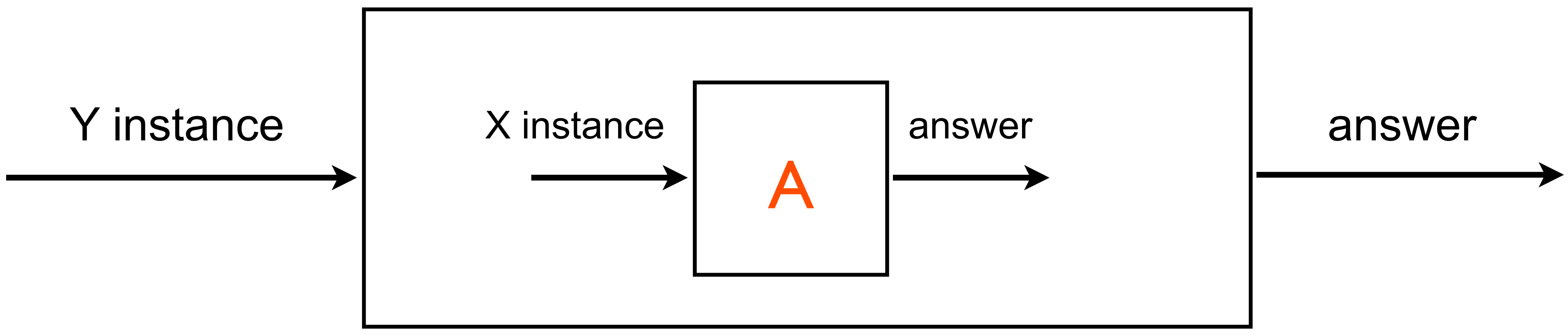
Reduction in action

Problem	Perfect Matching		Maximum flow
Instance			
Question	Does the graph have a PM?		Is there a flow with value n/2?
Proof	Yes instance	⟶	Yes instance
	No instance	⟶	No instance

Polynomial-time reduction

Def. : Let X and Y be two computational problems and A be a black box routine for solving X. Suppose Y can be solved in a polynomial number of computational steps plus a polynomial number of calls to A. Then we say that Y is polynomial-time reducible to X, and we write

$$Y \leq_P X$$



Properties of poly-time reductions

Suppose $Y \leq_P X$. If X can be solved in polynomial time, then Y can be solved in polynomial time too

Suppose $Y \leq_P X$. If Y cannot be solved in polynomial time, then X cannot be solved in polynomial time either

Reductions are transitive: If $Z \leq_P Y$ and $Y \leq_P X$ then $Z \leq_P X$

Example of a reduction

Let $G=(V,E)$ be an undirected graph. We say S subset of V is

- a vertex cover if every (u,v) in E has an endpoint in S (u in S or v in S)
- an independent set if every (u,v) in E has at most one an endpoint in S (u not in S or v not in S)

The vertex cover problem is the following:

- Input: graph G and a number k
- Question: Does G have a vertex cover of size at most k?

The independent set problem is the following:

- Input: graph G and a number t
- Question: Does G have an independent set of size at least t?

Satisfiability

Def.: A Boolean formula ϕ is defined on variables $x_1, x_2, \ldots$ in $\{0,1\}$

$$\phi = ((x_1 \wedge x_2) \vee (\neg x_3 \wedge x_2)) \wedge (\neg x_2 \vee x_1)$$

Def.: A truth assignment is $v: X \to \{F, T\}$, is said to satisfy ϕ if the formula evaluates to T

Def.: A conjunctive normal form (CNF) formula is the conjunction of disjunctions of literals (a variable or its negation)

$$\phi = (x_1 \vee x_2) \wedge (\neg x_3 \vee x_2 \vee x_1) \wedge (\neg x_2 \vee x_1)$$

Def.: A k-CNF formula is CNF and has k literals per disjunction

k-SAT problem

Input:

- A k-CNF formula ϕ

Question:

- Is there a satisfying assignment for ϕ

What do we know about this problem?

- 2-SAT can be solved in polynomial time
- 3-SAT is thought to be a "hard" problem

Today we will see that 3-SAT $\leq_P$ IS. If we believe that 3-SAT is hard, this indicates that IS is a "hard" problem as well.

Decision Problems

For technical reasons, we will deal with problems that involve answering a yes/no question. These are called *decision* problems

A decision problem is simply a partition of instances into "yes instances" and "no instances"

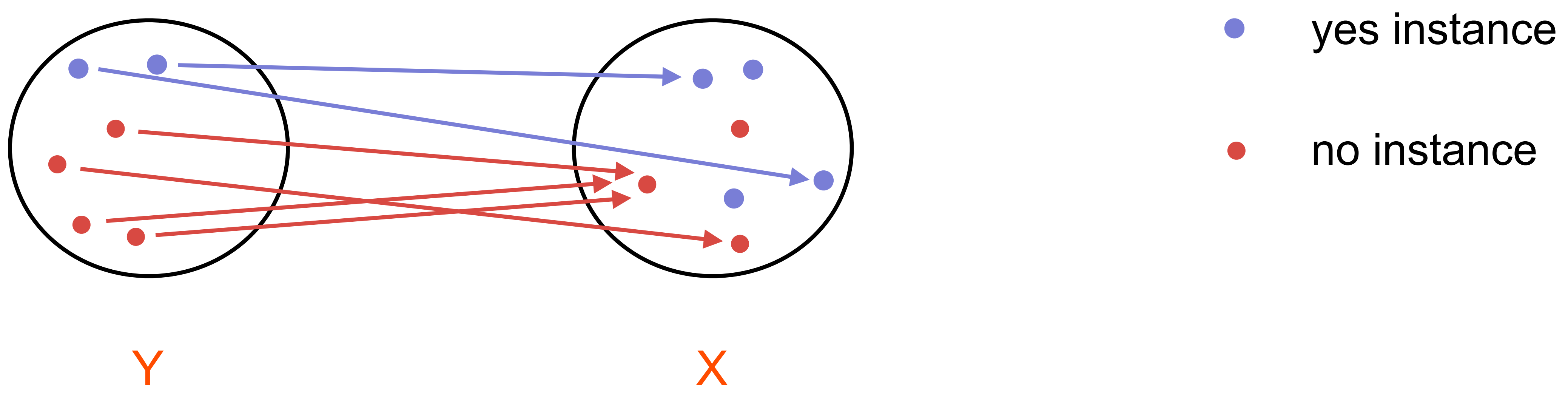
Examples:

- { (G,k) : G has a vertex cover of size k }
- { (G, c, s, t, T) : (G,c) has an s-t flow of value T }
- { board : there is a domino tiling of board }
- { (n, t) : t is the number of binary search trees on n keys }

Reduction template

When trying to show $Y \leq_P X$ we define a mapping from instances of problem Y to instances of problem X such that:

- the mapping can be computed in polynomial time
- "yes instances" of problem Y map to "yes instances" of problem X
- "no instances" of problem Y map to "no instances" of problem X



Complexity classes (informal defs)

Problems are classified according to their complexity.

<u>*Def.*</u>: P are those problems that admit a polynomial time algorithm

<u>*Def.*</u>: NP are those problems that admit a polynomial time algorithm for *verifying* "yes instances"

<u>*Obs.*</u>: All problems in P belong to NP as well.

The most central problem in computer science is to determine whether $P = NP$ or whether $P \subset NP$

Polynomial-time algorithms

<u>*Def.*</u>: Let X be a computational problem. An algorithm A *efficiently solves* X if

- A is polynomial time algorithm that takes one input s (an instance of X)
- for every s of X we have: s is "yes instance" if and only if A(s) returns "yes"

<u>*Def.*</u>: P are those problems that admit a polynomial time algorithm

Example:

- X = "Given an undirected bipartite graph, does it have a perfected matching?
- X in P by the reduction to max flow + Ford-Fulkerson

Polynomial-time verifier

Def.: Let X be a computational problem. An algorithm A is an *efficient verifier* for X if

- A is a polynomial time algorithm that takes two inputs s (an instance of X) and t (a certificate)
- There is a polynomial function p such that for every instance s of X we have, s is a "yes instance" if and only if there is a certificate t such that $|t| < p(|s|)$ and A(s,t) returns "yes"

Def.: NP are those problems that admit a polynomial time verifier

Example:

- X = "Given a general undirected graph, does it have a perfect matching?
- X in NP because given the matching the property is trivial to check!

NP-complete problems

Def.: A problem X is NP-hard if every problem in NP is polynomial-time reducible to X; that is, if $Y \leq_P X$ for all Y in NP.

Def.: A problem X is NP-complete if it belongs to NP and is NP-hard; that is, if X in NP and $Y \leq_P X$ for all Y in NP.

Cook and Levin independently showed that such problems exist

Thm.

3-SAT is NP-complete

Independent Set

Consider the independent set problem:

- it belongs to NP because it is easy to verify a “yes instance”
- we know that 3-SAT $\leq_P$ IS and that $\leq_P$ is transitive therefore IS is NP-hard

Thm. Independent set is NP-complete

Graph k-coloring

Let $G=(V, E)$ be an undirected graph

A k-coloring of G is a function $\varphi : V \rightarrow \{1,\dots, k\}$

A coloring φ is feasible is there are no monochromatic edges

Input:

- An undirected graph G

Question:

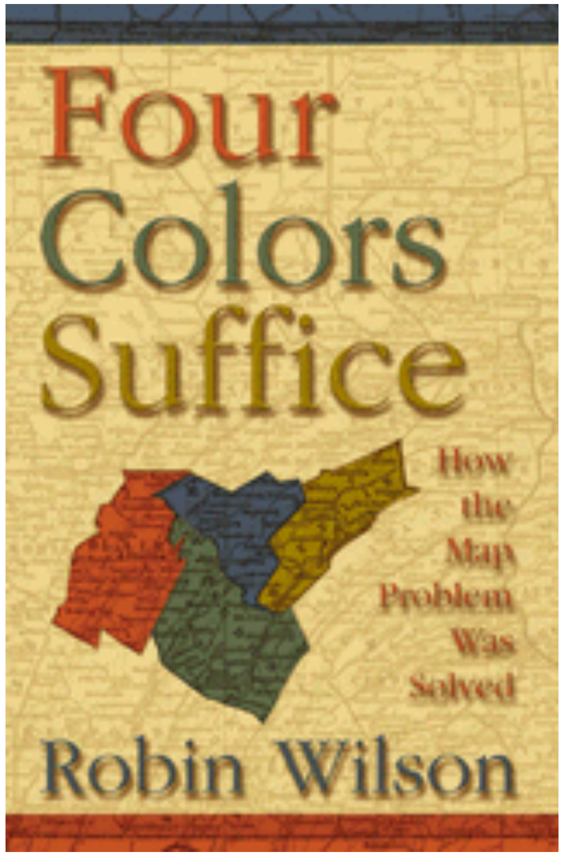
- Does G admit a feasible k-coloring?

Graph coloring

Originally studied as a subproblem in map drawing, graph coloring has many applications.

What do we know about this problem?

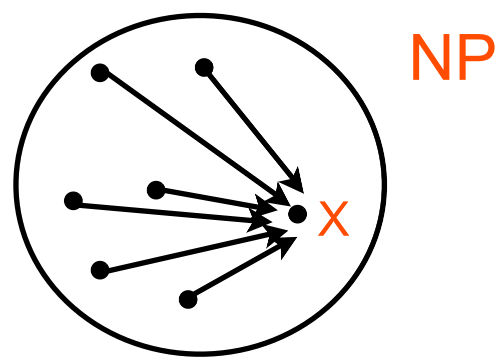
- graph 2-coloring is "easy"
- graph 3-coloring is NP-hard even for planar graphs
- every planar graph admits a feasible 4-coloring



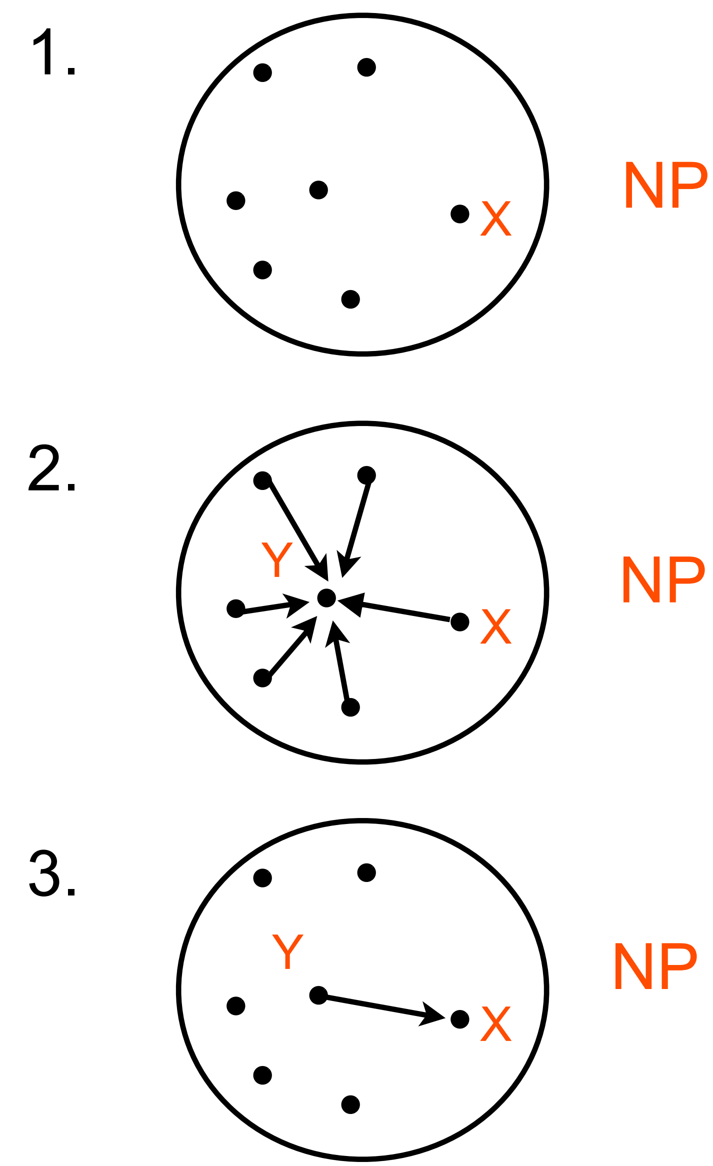
Today we'll show that 3-coloring is NP-hard in general graphs

Template of NP-completeness proof

Suppose we want to show that X is NP-complete.



what we want



Then we need to:

1. First, argue that X belongs in NP
2. Pick a known NP-complete problem Y
3. Finally, argue that Y $\leq_P$ X