P and NP

Inge Li Gørtz

Thank you to Kevin Wayne, Philip Bille and Paul Fischer for inspiration to slides

Overview

- Problem classification
  - Tractable
  - Intractable
- Reductions
  - Tools for classifying problems according to relative hardness

Warm Up: Super Hard Problems

- Undecidable. No algorithm possible.
- Example. Halt (P, x) = true iff and only if P halts on input x.
- Claim. There is no general algorithm to solve Halt(P, x)
- Proof (by contradiction)
  - Suppose algorithm for Halt(P, x) exists.
  - Consider algorithm A(P) which loops infinitely if Halt(P,P) and otherwise halts.
  - Since Halt(P,x) exists for all algorithms P we can use it on A(A) and the following happens:
    - If Halt(A,A) then we loop infinitely.
    - Else (not Halt(A,A)) we halt.

Problem Classification

- Q. Which problems will we be able to solve in practice?
- A. Those with polynomial-time algorithms. (working definition) [von Neumann 1953, Godel 1956, Cobham 1964, Edmonds 1965, Rabin 1966]

Yes	No
Shortest path	Longest path
Min cut	Max cut
Soccer championship (2-point rule)	Soccer championship (3-point rule)
Primality testing	Factoring

Problem Classification

- Ideally, classify problems according to those that can be solved in polynomial-time and those that cannot.

- Provably requires exponential-time.
  - Given a board position in an n-by-n generalization of chess, can black guarantee a win?
- Provably undecidable.
  - Given a program and input there is no algorithm to decide if program halts.

- Frustrating news. Huge number of fundamental problems have defied classification for decades.

P and NP

- A problem (problem type) is the general, abstract term:
  - Examples: Shortest Path, Maximum Flow, Closest Pair, Euler Cycle, Hamilton Cycle.
- A problem instance is the concrete realization of a problem.
  - Example: Maximum flow. The instance consist of flow network.
  - Example: Closest Pair. The instance is a set of points

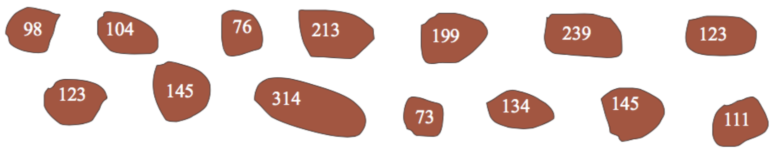
The class P

- P ~ problems solvable in deterministic polynomial time.
  - Given a problem type T, there is a deterministic algorithm *A* which for every problem instance *I* ∊ T solves *I* in a time that is polynomial in |*I*|, the size of *I*.
  - I.e., the running time of *A* is $O(|I|^k)$ for all *I* ∊ T,where k is constant independent of the instance T.

- Examples.
  - Closest pair: There is an algorithm A such that for every set S of points, A finds a closest pair in time $O(|S|^2)$.
  - Maximum flow: There is an algorithm A such that for any network, A finds a closest pair in time $O(|V|^3)$, where V is the set of vertices.

Hard problems: Example

- Problem [POTATO SOUP]. A recipe calls for B grams of potatoes. You have a K kilo bag with n potatoes. Can one select some of them such that their weight is exactly B grams?



- Best known solution: create all 2^n subsets and check each one.

Hard problems

- Many problems share the above features
  - Can be solved in time $2^{|T|}$ (by trying all possibilities.)
  - Given a potential solution, it can be checked in time $O(|T|^k)$, whether it is a solution or not.
- These problems are called polynomially checkable.
- A solution can be guessed, and then verified in polynomial time.

The class NP

- A problem T is in the class NP (Non-deterministic Polynomial time) if
  - There is randomized algorithm A and a constant k such that
  - given an instance $T \in$ T , algorithm A guesses and checks a solution for T in time $O(|T|^k)$ and
  - there is a positive chance that A guesses a correct solution (if T has a solution).
- For the running time, only the correct guessing is counted.

Optimization vs decision problems

- Consider decision problems (yes-no-problems).
- Example.
  - [POTATO SOUP]. A recipe calls for B grams of potatoes. You have a K kilo bag with n potatoes. Can one select some of them such that their weight is exactly B grams?
- Optimization vs decision problem
  - [OPTIMIZATION LONGEST PATH] Given a graph G and integer k. What is the length of the longest simple path?
  - [DECISION LONGEST PATH] Given a graph G and integer k. Is a there a simple path of length $\geq$ k?
- Exercise. Show that OPT can be solved in polynomial time if and only if DEC can be solved in polynomial time.

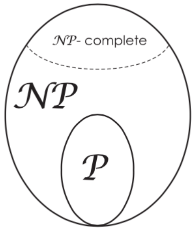
The class NP

- Definition. A yes-no-problem is in NP if there is a polynomial p and a randomized $p(|I|)$-bounded algorithm A such that for every instance *I* the following holds:
  - True answer for *I* is YES then $P[A(I) = YES] > 0$
  - True answer for *I* is NO then $P[A(I) = YES] = 0$

where P[Z] denotes the probability of event Z over the algorithms randomization.

P vs NP

- P solvable in deterministic polynomial time.
- NP solvable in non-deterministic (with guessing) polynomial time. Only the time for the right guess is counted.
- P⊆NP (every problem T which is in P is also in NP).
- It is not known (but strongly believed) whether the inclusion is proper, that is whether there is a problem in NP which is not in P.
- There is subclass of NP which contains the hardest problems, NP-complete problems.



Examples of NP-complete problems

- Preparing potato soup
- Packing your suitcase
- Satisfiability of clauses
- Partition, Subset-sum
- Hamilton Cycle, Travelling Salesman
- Bin Packing, Knapsack
- Clique, Independent Set
- Vertex Cover

NP-complete problems

- [SOCCER CHAMPIONSHIP 3-POINT RULE] In a football league n teams compete for the championship. The leagues uses the 3-point rule, i.e., the points of match are distributed as 3:0, 1:1, or 0:3.
  - **Input.** The table with the points of every team at some point in the season, a list of the matches to be played in that season and the name of some team.
  - **Output.**
    - YES if the named team still can become champion
    - NO otherwise.

NP-complete problems

- [SATISFIABILITY]
  - **Input:** A set of clauses C = {c1, . . . , ck} over n boolean variables x1,…,xn.
  - **Output:**
    - YES if there is a satisfying assignment, i.e., if there is an assignment a: $\{x_1,\ldots,x_n\} \rightarrow \{0,1\}$ such that every clause is satisfied,
    - NO otherwise.

$$(\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\overline{x_1} \vee \overline{x_3} \vee \overline{x_4})$$

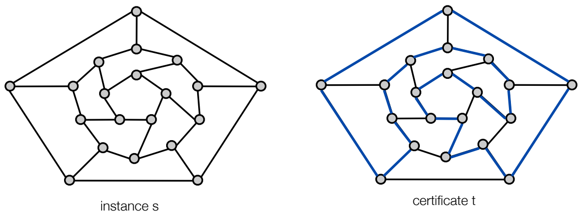
instance s

$$x_1 = 1,\ x_2 = 1,\ x_3 = 0,\ x_4 = 1$$

proposed solution/certificate t

NP-complete problems

- [HAMILTONIAN CYCLE].
  - **Input:** Undirected graph G
  - **Output:**
    - YES if there exists a simple cycle that visits every node
    - NO otherwise



NP-completeness

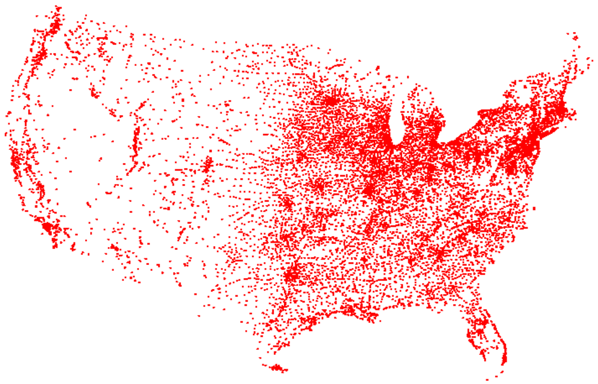
- Intuition:
  - The NP-Complete problem are the "hardest" problems within NP.
  - All NP-Complete problems poly-time reduce to each other.

Polynomial-time reduction

- Reduction. Problem X polynomial reduces to problem Y if arbitrary instances of problem X can be solved using:
  - Polynomial number of standard computational steps, plus
  - Polynomial number of calls to oracle that solves problem Y.

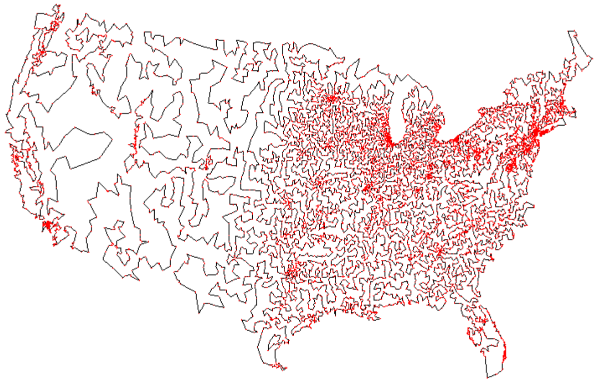
- Notation. $X \leq_P Y$.
- We pay for time to write down instances sent to black box ⇒ instances of Y must be of polynomial size.

- Traveling Salesperson Problem TSP: Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?



All 13,509 cities in US with a population of at least 500
Reference: http://www.tsp.gatech.edu

- Traveling Salesperson Problem TSP: Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?



Optimal TSP tour
Reference: http://www.tsp.gatech.edu

Reduction example

- [HAMILTONIAN CYCLE]. Given a undirected graph G=(V,E), does there exists a simple cycle that visits every node?
- [TRAVELLING SALESMAN (TSP)] Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length ≤ D?

- Show Hamiltonian Cycle $\leq_P$ TSP:
  - For every instance of Hamiltonian Cycle create an instance of TSP such that the TSP instance has tour of length ≤ n if and only if G has a Hamiltonian cycle.

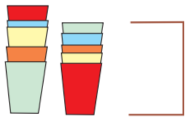
- Reduction.
  - Given instance G=(V,E) of Hamiltonian Cycle, create n cities with distance function

$$d(u, v) = \begin{cases} 1 & \text{if } (u, v) \in E \\ 2 & \text{if } (u, v) \notin E \end{cases}$$

  - TSP instance has tour of length ≤ n if and only if G has a Hamiltonian cycle.

Reduction example

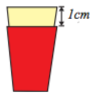
- [GLASSES IN A CUPBOARD]. You have n glasses of equal height. If glass g_j is put into glass g_i let d_{ij} be the amount of g_j above the rim of g_i. You want to stack them into a single stack, so they fit into a cupboard of height h; is that possible?



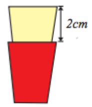
- Glasses in a Cupboard in NP: Proposed solution can be verified in polynomial time.
- NP-completeness:
  - Reduction from Directed Hamiltonian Path (DHP).
  - Directed Hamiltonian Path: Given a directed graph G, is there a directed simple path visiting all vertices.
  - DHP is NP-complete
  - Reduction: For every instance (graph) of DHP make a set of glasses and a cupboard, such that the glasses can be stacked into the cupboard if and only if the graph has a Hamiltonian path.

Reduction example

- Let G = (E, V) a directed graph.
  - Make one glass for every node i ∈ V.
  - If (i, j) ∈ E ensure:

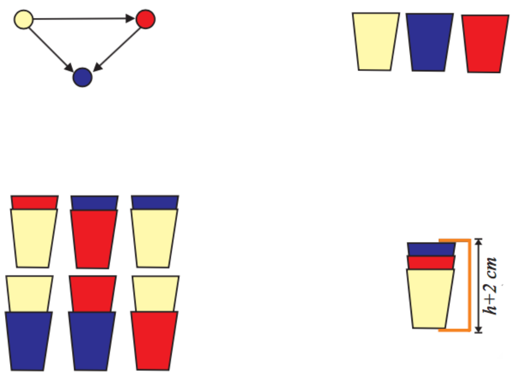


  - If (i, j) ∉ E ensure:

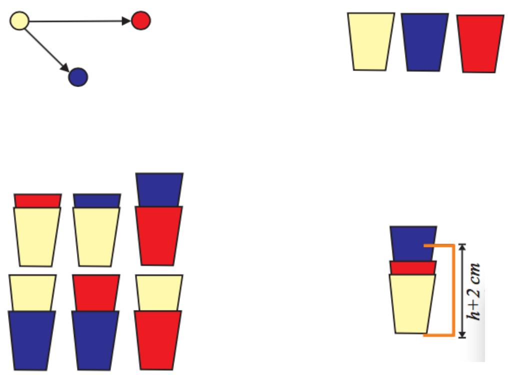


  - Glass i is red, glass j is yellow.
  - Height of the cupboard is |V| – 1 + height of glass

Reduction Example

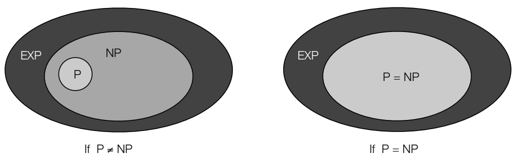


Reduction Example



The Main Question: P Versus NP

- Does P = NP? [Cook 1971, Edmonds, Levin, Yablonski, Gödel]
  - Is the decision problem as easy as the certification problem?
  - Clay $1 million prize.



- Consensus opinion on P = NP? Probably no.

The Simpsons: P = NP?

The P versus NP page

- Lately several papers/notes claiming to resolve P versus NP problem have appeared (especially after $ 1 million reward in 2000)
- Gerhard Woeginger maintains a webpage collecting the contributions: **www.win.tue.nl/~gwoegi/P-versus-NP.htm**

- Scott Aaronson on why you should believe P not equal to NP : http://www.scottaaronson.com/blog/?p=1720