CS 170 DISCUSSION 12

SEARCH PROBLEMS AND INTRACTABILITY

Raymond Chan
raychan3.github.io/cs170/fa17.html
UC Berkeley Fall 17

PROBLEMS

- Decision problem: Given an input, determine if there **exists** some solution of size at most b that would satisfy some constraints. (True or False)
- Search problem: Given an input, **find** a solution of size at most b that satisfy some constraints. (outputs instance of the problem)
- Optimization problem: Given an input, optimize over some objective and find the solution satisfying some constraints. (find "best" instance)

NP SEARCH PROBLEMS

- Instances are some confirmation of a problem.
- Ex. Traveling salesman instances are a configuration of graph with edge weights.
- Search Problems when talking about NP-Completeness (more later):
  - Given an instance I and a proposed solution S, we can check whether S is a solution to I in polynomial time with respect to the size of I.

NP SEARCH PROBLEMS

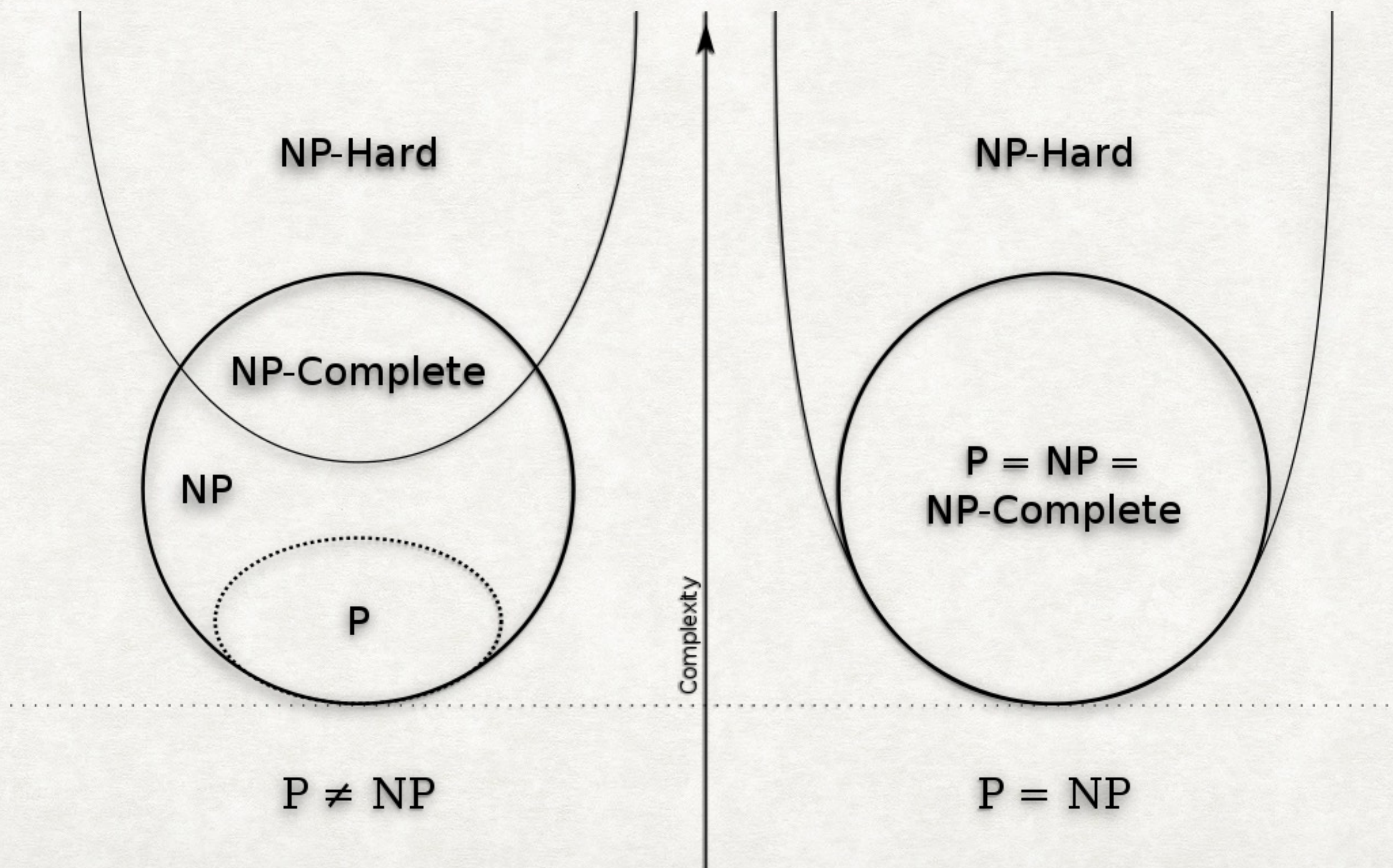
- Optimizations are also search problems.
- Given you have some "optimal" solution, we can tell whether it is a valid solution to the instance via the "solution verifier" algorithm.
- Use binary search to ensure that this is the optimal solution.

P VS NP

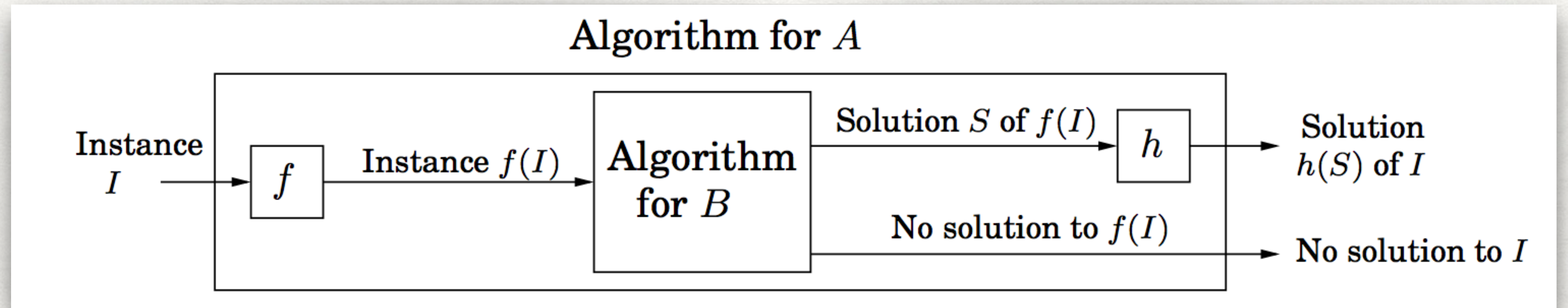
- **P**: Contains the set of all problems that can be **solved** (and thus verified) in **polynomial** time.
- **NP**: Set of all problems that can be **verified** in polynomial time.
- **NP-complete**: A problem to which all others problems in NP reduce.
  - Remember reductions A —> B
  - B is NP-complete if A —> B for all A in NP.

P VS NP

- **NP-Hard:** at least as hard as the hardest problem in NP.
- If P = NP, then we should be able to solve all these hard problems such as traveling salesman in polynomial time.



REDUCTIONS



- Preprocess instance of A to be an instance of B.
- Solve B.
- If solution exists, postprocess solution of problem B instance to be solution of original problem A instance.
- To proof valid reduction, need to proof the following:
  - If there is a solution to f(I), then there is a solution I.
  - If there is no solution to f(I), then there is no solution to I.

REDUCTIONS

- To prove that a problem B is in a complexity class, need to reduce a problem A in that desired complexity class to B.
- A —> B
- B is at least as hard as A.
- This does not say anything about A.
    - If I can solve A with best runtime exponential and I can solve A by solving B, then I must have to solve B in exponential time. If I can solve B in polynomial time, then either A has a better runtime or it is not a valid reduction.

PROVING NP-COMPLETENESS

- Show problem A is NP-Complete by:
  - Prove that it is in NP.
    - Show that there is a polynomial time verifier.
  - Prove that it is NP-Hard
    - Reduce a NP-Hard problem to the A.
    - Not all NP-Hard problems are NP-Complete, but all NP-Complete problems are NP-Hard
    - The reduction should take polynomial time.

CERTAIN PROBLEMS

- Not all NP-Hard problems reduce NP-Complete.
  - NP-Complete is a subset of NP-Hard.
    - Halting problem (whether program will finishing running or continue to run forever) is NP-Hard but not NP-Complete.
- Factoring, finding all prime factors of a given integer, is in NP but not NP-Complete.
- NP stands for Nondeterministic Polynomial time.
  - Can be solved in polynomial time **using** non-deterministic Turing Machine