


Notes, 12

ECE 606

Common mistakes; reconciling intractability

Important note: these are *mistakes*.

1. A problem is in **NP** means it is hard.
2. **NP** is Non-Polynomial time.
3. A problem is **NP**-hard implies no polynomial-time algorithm exists for it.
4. **NP** = **P** \cup **NP**-complete.
5. A **true** instance  problem that is in **NP** is associated with only one, or a few, certificates. *of a*
6. When we reduce a problem A to B and B is **NP**-complete, that implies that A is a hard problem.
7. If a problem A is **NP**-hard, and I am given an instance i of the problem A , then i is hard.
8. Addressing only the “only if” part of the “if and only if” of a Karp reduction.
9. Adopting a function that is not polynomial-time computable as a Karp-reduction.
10. **NP**-complete = **NP**-hard.
11. **NP** = **co-NP**.

So what if you encounter a problem which is **NP**-hard? Do you just throw your hands up with regards to devising an (efficient) algorithm?

Worse still, what if you're unable to (a) devise a polynomial-time algorithm, and, (b) unable to prove that the problem is hard, e.g., **NP**-hard?

- E.g., the problem may be **ISO**-complete or **RP**-complete.

Possible ways of dealing with this:

- First establish an upper-bound for hardness. E.g., is the problem in **NP**? Is it in **PSPACE**?

Then, reduce to a problem for which folks have created solvers. E.g., a SAT or ILP solver for **NP**, and a model checker for **PSPACE**.

- Efficient approximation: rich theory for problems that lie in **NP**, for example.
- Randomized algorithms.

E.g., even though primality checking is in **P**, the randomized Miller-Rabin primality test that exploits $\mathbf{P} \subseteq \mathbf{RP}$ is used widely.

- Via assumptions and “hacks.”

E.g., does your problem map to only certain classes of instances of SAT? If so, perhaps you can exploit that structure for “most” of the instances that arise for you in practice.

End of the course