CAD Algorithms

The Classes P and NP

Mohammad Tehranipoor

ECE Department

P and NP

- P and NP are two families of problems.
- P is a class which contains all of the problems we solve using computers.
- If we think about the problems we actually present to the computer we note that not too many computations require more than $O(n^3)$ or $O(n^4)$ time assuming that n is very large.
- Most of the important algorithms we compute are somewhere in the O(log n) to $O(n^3)$ range.

Class P

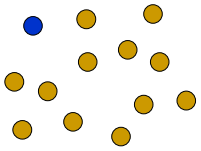
- Practical computation resides within ***polynomial time*** ($O(N^c)$, $c > 1$) bounds.
- **Definition:**
  - The class of ***polynomially solvable problems***, **P** contains all sets in which membership may be decided by an algorithm whose running time is **bounded** by a polynomial.
- **Advantage:**
  - Its use allows us to not worry about our machine model since all reasonable models of computation (including programs and Turing machines) have time complexities, which are polynomially related.

Class NP

- NP is a class of problems that we would love to solve but are **unable** to do so **exactly**.
- Example:
  - Consider final examination scheduling.
  - A school has *n* courses and five days in which to schedule examinations.
  - An optimal schedule would be one where no student has to take two examinations on the same day.
  - There are $O(5^n)$ possible different schedules.
  - 200,000,000,000,000,000,000 years! (regardless of the type of machine used)
  - *Obviously this will not be done between registration and the end of the semester!*

Class NP

- Another Example:
  - Find minimal length tour of *n* cities where we begin and end at the same place.
  - This is called *closed tour problem*.
  - ***N!*** different tours are possible.



  - If we have a tour (a solution), we can easily check to see how long it is.
  - Thus, if we want a tour of less than some fixed length, we can quickly check candidates to see if they qualify.
  - Also, previous tours can help find a new one that could potentially provide better result
    - Genetic algorithm

NP

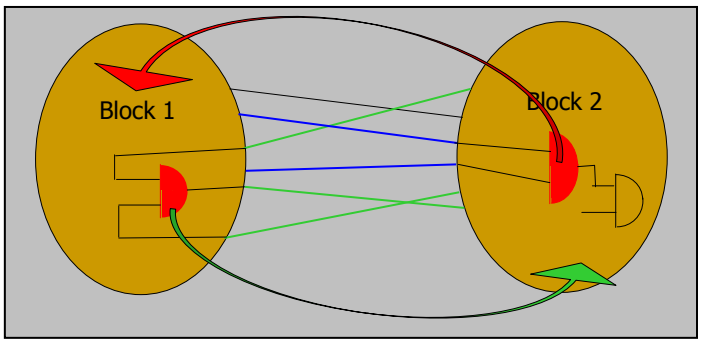
- There are many tasks for which we may apply fast (polynomial) algorithms. There are also some problems that cannot be solved algorithmically.
- One example of a class of problems which cannot be solved in the "traditional" way, are NP problems.
- There are many important problems in which it is very difficult to find a solution, but once we have it, it is easy to check the solution.
  - This is key in developing many algorithms in solving very complex problems.
- If we can determine the worth of an answer, then maybe we can investigate ***promising solutions*** and ***keep the best one***.
- This fact led to **NP-complete problems**.

NP Problems

- **Definition:**
  - The class of ***nondeterministic polynomially*** acceptable problems, **NP**, contains all sets in which membership **can** be verified in polynomial time.
  - It means that it is possible to "guess" the solution (by some nondeterministic algorithm) and then check it.
  - Nondeterministic is just a fancy way of talking about guessing a solution.
  - A problem is in NP if you can quickly (in polynomial time) test whether a solution is correct (without worrying about how hard it might be to find the solution).
- *NP does not stand for "**non-polynomial**"*. There are many complexity classes that are much harder than NP.

VLSI Kind of Example

- **Partitioning Problem:**
  - A large circuit is divided into smaller subcircuits (blocks)
  - Objective: Minimize cutsize



Cutsize = 6

NP Problems

- As of today, there are no faster algorithm to provide "exact" answers to NP-problems.
- The discovery of such algorithms remains a big task for researchers (maybe you!).
- Today many people think that such algorithms do not exist and so they are looking for alternative methods.
- An example of an alternative method is the **genetic algorithm**.

NP Examples

- Many graph problems used in CAD algorithms for computer chip design fit in class of NP.
- Other examples:
  - **Scheduling problems**
  - **Steiner tree**
  - **Satisfiability Problem**
  - **Closed Tour**
    - *Given n cities and an integer k, is there a tour, of length less than k, of the cities which begins and ends at the same city?*

NP Examples

- **Examination Scheduling**
  - *Given a list of courses, a list of conflicts between them, and an integer k; is there an exam schedule consisting of k dates such that there are no conflicts between courses which have examinations on the same date?*
- **Vertex Cover**
  - *Given a graph and an integer k, is there a collection of k vertices such that each edge is connected to one of the vertices in the collection?*
- **Chromatic Number (Color)**
  - *Given a graph and an integer k, is there a way to color the vertices with k colors such that adjacent vertices are colored differently?*

Solutions

- **Use a heuristic.** If you can't quickly solve the problem with a good worst case time, maybe you can come up with a method for solving a reasonable fraction of the common cases.
- **Solve the problem approximately instead of exactly.** A lot of the time it is possible to come up with a provably fast algorithm, that doesn't solve the problem exactly but comes up with a solution you can prove is close to right.

Solutions

- **Use an exponential time solution anyway.** If you really have to solve the problem exactly, you can settle down to writing an exponential time algorithm and stop worrying about finding a better solution.