ECE6095: CAD Algorithms

Optimization Techniques

Mohammad Tehranipoor

ECE Department

Optimization Techniques

- Objective:
  - A basic review of complexity
  - Review of basic algorithms
  - Some physical design examples
  - Study of algorithms that are useful in the VLSI Design Automation
  - Students must be familiar with algorithms to be able to understand and solve physical design problems.
  - Students can use optimization techniques in their research and projects.

Combinatorics

- One of the fastest growing areas of applied mathematics.
- It is concerned with the study of arrangements, patterns, designs, assignments, schedules, connections, and configurations.
- **It is everyday science!!**
  - Class scheduling
  - Traveling salesman
  - Pattern recognition
  - Airline
  - Tours
  - **VLSI** …

Optimization Problem

- A combinatorial optimization problem is defined as the set of all the instances of the problem, each instance being defined as a pair *(F,c)*.
  - *F* is any set, the domain of feasible points
  - *c* is the cost function

    $c: F \longrightarrow R^1$
  - *The problem is to find an* $f \in F$ *for which*

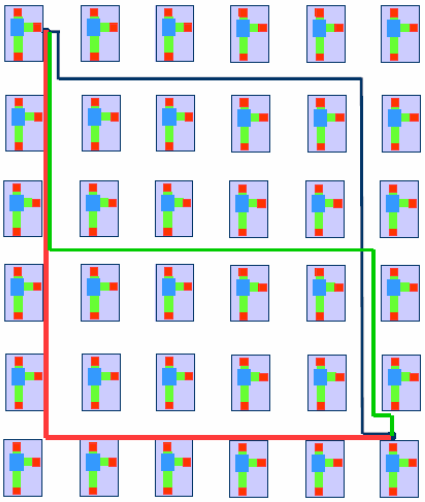
    $c(f) \leq c(y)$ *for all* $y \in F$

    f is called globally optimal solution.
- *Most VLSI problems belong to this class*

A VLSI Kind of Example

Routing problem:

How to optimally connect ≥ 2 nodes in the circuit layout?



Computational Complexity

- How fast will an algorithm run on a computer?
- How much memory will it need?
- Computational complexity tries to answer these questions without having to deal with specific hardware.
- Thus exact seconds and megabytes are insignificant.
- "**input size**" is the base parameter that defines complexity.

Analyzing Algorithms

- Analysis results in a viable candidate algorithm for solving a problem.
- Analysis is challenging and requires knowledge tools like combinatorics, some probability, mathematical skills etc.
- While we will not spend rest of the semester to learn these skills, it is important to know **basics for appreciating the algorithm design(s)**.

Basic Algorithm Analysis

- How can we calculate the running time of an algorithm?
- How can we compare two different algorithms?
- How do we know if an algorithm is '**optimal**'?
- Count the number of **basic operations** performed by the algorithm on the **worst-case input**
- Basic operation:
  - An assignment
  - A comparison between two variables
  - An arithmetic operation between two variables
- The worst-case input is the input assignment for which the most basic operations are performed.

Input Size and Run Time

- Number of items that are input to the problem define input size.
  - Example sorting, addition, multiplication, etc.
- The running time of an algorithm is the number of primitive operations or "steps" that are executed.
- The running time of an algorithm varies with the input and typically grows with the input size.

Analysis

- Best Case
- Worst Case
- Average Case
  - Usually we are more interested in worst case analysis as it also defines the **upper bound** on the running time of an algorithm.
  - Average case analysis is interesting but for a large class of problems average case is as bad as worst case.
  - Average case is difficult to determine
- We focus on the worst case running time
  - Easier to analyze
  - Crucial to some applications such as physical design problems

Big-O Notation

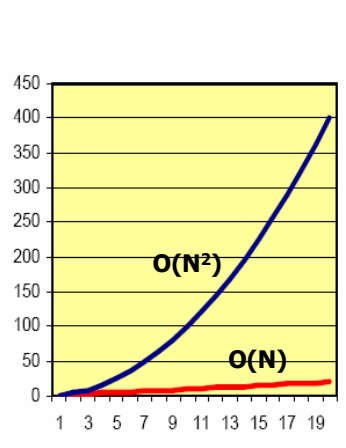
- **Big-O** is used for defining the upper bound on a function.
- We express complexity of the algorithms using **big-O notation**.
- For a problem of size N (Assume N is input):
  - A constant-time method is "order 1": O(1)
  - A linear-time method is "order N": O(N)
  - A quadratic-time method is "order N squared": $O(N^2)$
  - Cubic: $O(N^3)$
  - Logarithmic, O(log N)

How to Determine Complexities?

- Q: In general, how can we determine the running time of a piece of code?
- A: It depends on what kinds of statements are used.
- We don't need to count the exact number of steps in a program.
- **What matters is how that number grows with the size of the input to the program.**

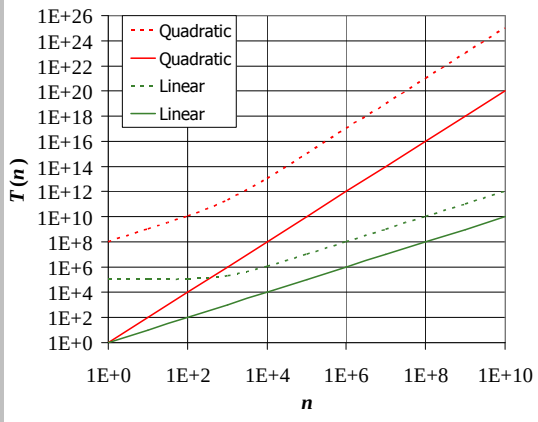
Rate of Growth of an Algorithm

- It is the rate of growth of running time of an algorithm that interests us the most, especially the CAD tool developers.
- Note that the O(N) algorithm takes more time than the $O(N^2)$ algorithm for small inputs.
- As N grows, the O(N) algorithm takes less and less time relative to the $O(N^2)$ algorithm. So for large N, **constant factors** don't matter, just the big-O bounds.



Constant Factors

- The growth rate is not affected by
  - constant factors or
  - lower-order terms
- Examples:
  - $10^2n + 10^5$ is a linear function: O(n)
  - $10^5n^2 + 10^8n$ is a quadratic function: $O(n^2)$
  - A n grows large, n^2 will dominate



Rate of Growth of an Algorithm

- Polynomial vs. Exponential
  - tractable vs. intractable
- Linear vs. Quadratic
  - execution realism in VLSI

Tractable and Intractable

- A problem that is solved in **deterministic polynomial** time is called **tractable**.
- A ***polynomial-time algorithm*** is defined to be an algorithm whose execution time is given by a polynomial on the size of the input.
- Problems that can't be solved by a polynomial-time algorithm are called ***intractable*** problems.
- NP complete problems are likely to be ***intractable***.

CAD Algorithms

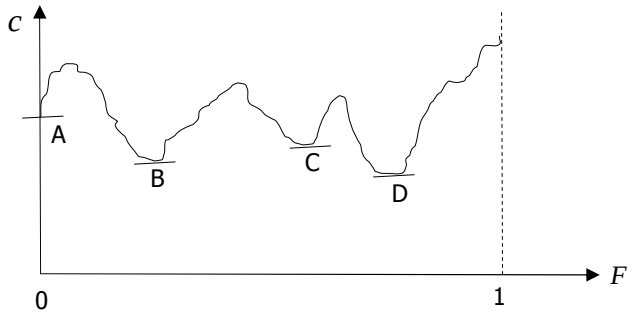
Local and Global Optima

Mohammad Tehranipoor

ECE Department

Local and Global Optima

- Finding globally optimal solution can be prohibitively difficult.



$c(f) \leq c(g)$ $f \in F$ and $g \in N(f)$, f is called locally optimal.

$N(f)$ is called Neighborhood function

Cont.

- A, B, C, and D are all locally optimal.
- Only D is globally optimal.

- *F* and *c* are given.
- Evaluating the entire search space when F is large is infeasible.
- The goal of every algorithm is to find the global optima without searching entire search space.
- Most of the algorithms get stuck in local optima.

Global Optimization

- The objective of global optimization is to find the globally best solution of models, in the (possible or known) presence of multiple local optima.

- The main question is how to find the global optima.