Lecture 2: Notes on Complexity
CEE 509 / Com S 574

Prof. Christine Shoemaker
Prof. Bart Selman

Slide CS472–1

Why “bother” with heuristic methods?

Drawbacks:
- can be difficult to judge quality of solution
  (“local minima”)
- no guarantees on runtime

Why not design algorithms for specific applications, that are
  optimal or close to it, and fast?

Slide CS472–2
A: Because we (strongly) believe no such algorithms exist!
   Based on computational complexity theory from comp. sci.
   More specifically, most interesting applications
   are “NP-complete”.

No efficient algorithms known.
   Over 2000 problems so far shown the be NP-hard.

(Minor homework exercise: show P =/= NP. :-) )

---

**P vs. NP**

P stands for “in polynomial time.”

NP stands for “in non-deterministic polynomial time.”
   Will become clear later.

Let’s consider a basic problem from graph theory,
   about finding paths in a graph.
Problem A
Given a graph \( G \) on \( N \) nodes with \( E \) edges and

two nodes in the graph \( s \) and \( t \),

find the shortest path between \( s \) and \( t \).

How difficult / easy is this?

(I.e., how much time does it take to find the path?)

Slide CS472–5

It’s “easy”. That is, it can be done in polynomial time.

Dijkstra’s algorithm.

Most basic implementation: \( O(N^2) \) to find a shortest

path between a given node and every other node in the graph.

More efficient implementation: \( O(|E| + N)\log N) \).

It’s a clever algorithm. Constructs a table

incrementally computing shortest distances

between original node and all other nodes.

In each iteration of alg., it considers

a new node and whether going through that node reduces
the distance between any pair of nodes.

Slide CS472–6
Problem B
Given a graph $G$ on $N$ nodes with $E$ edges and two nodes in the graph $s$ and $t$,
find the **longest** path between $s$ and $t$.

How difficult / easy is this?

---

Surprise: No good algorithm known!
People have thought about it for over 40 years.
The “best” we can do is consider the set of **all possible paths** in the graph.

How?
One strategy: check all subsets $E'$ of edges.
For each check whether it’s a valid path and, if so, check its length and keep longest one. (Could also search for the shortest one this way...)
How long does this take?

$O(2^{|E|})$ or $O(2^{(N^2)})$.

E.g., $N = 10$ means up to $10^{30}$ operations

$N = 20$, gives $10^{120}$ operations. It’s a lot.

vs. at most $20^2 = 400$ operations for all shortest paths!!