

Hard problems in computer science



Prof. Noah Snaveley

CS1114

<http://www.cs.cornell.edu/courses/cs1114/>



Cornell University
Computer Science

Administrivia

- Assignments:
 - A6 due Friday
- Prelim 3 next Thursday
 - Review in class Tuesday
- Final project proposals
 - Most of you now have feedback
 - Demo session on Tuesday, May 15, 1:30-3:30



Cornell University

Puzzle

- Does this program terminate?

```
i = 0;
while true
    i = i + 1;
end
```



Puzzle

- How about this one?

```
c = 2;
while true
    for a = 2:c
        for b = 2:c
            if a^3 + b^3 == c^3
                return;
            end
        end
    end
    c = c + 1;
end
```



Hard problems in computer science

- Many problems in computer science are “polynomial time” problems
 - We know of an algorithm that solves the problem exactly in $O(n^k)$ time, for some constant k
 - Examples?
- Many other problems have no known polynomial time algorithm
 - E.g., problems whose fastest known algorithm takes $O(2^n)$ time [exponential time]



Hard problems in computer science

- Other problems cannot be solved *at all* in general
 - Given a program written in Matlab, does that program ever terminate?
 - A version of the *halting problem*
 - Vaguely related to the puzzle a moment ago



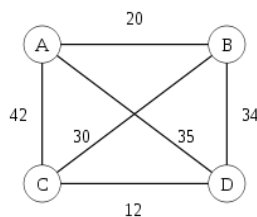
Exponential time algorithms

- Many other problems have no known polynomial time algorithm
 - E.g., algorithms whose best solution takes $O(2^n)$ time
- If it takes 1 second to solve such a problem with $n = 100$
- Then it takes 2 seconds to solve for $n = 101$
- And it takes 2^{50} seconds to solve for $n = 150$
 - About 3.6 million years

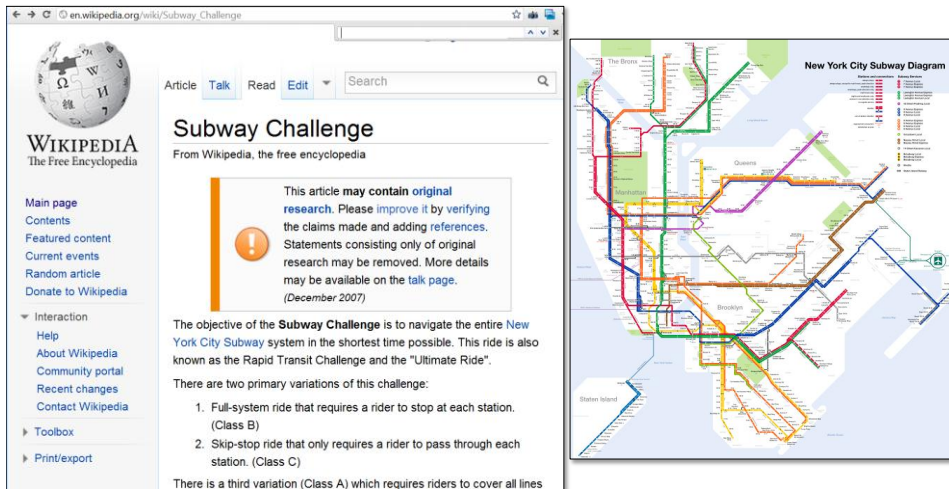


Travelling Salesman Problem (TSP)

- Classic hard problem in CS
- Problem statement:
 - Given a weighted, complete graph with n nodes
 - Compute a *tour* that starts and ends at the same nodes, and visits all other nodes
 - Find such a tour that has the lowest total cost
 - A tour is also just a permutation of the nodes



Subway challenge



The image shows a screenshot of a Wikipedia article titled "Subway Challenge" and a New York City Subway Diagram. The Wikipedia article is on the left, and the subway diagram is on the right. The article text includes:

Subway Challenge
From Wikipedia, the free encyclopedia

This article may contain **original research**. Please improve it by verifying the claims made and adding references. Statements consisting only of original research may be removed. More details may be available on the talk page. (December 2007)

The objective of the **Subway Challenge** is to navigate the entire New York City Subway system in the shortest time possible. This ride is also known as the Rapid Transit Challenge and the "Ultimate Ride".

There are two primary variations of this challenge:

1. Full-system ride that requires a rider to stop at each station. (Class B)
2. Skip-stop ride that only requires a rider to pass through each station. (Class C)

There is a third variation (Class A) which requires riders to cover all lines

The New York City Subway Diagram on the right shows the extensive network of lines and stations across the city, including labels for The Bronx, Queens, Brooklyn, and Staten Island.

Solving the TSP

- What kind of algorithm might solve the TSP?
- Can you come up with an example that breaks your algorithm?

Solving the TSP

- What if the graph is metric?
 - Means that nodes could be laid out on the plane, with weights corresponding to distance in the plane
- Easy to find a good approximation algorithm (on board)

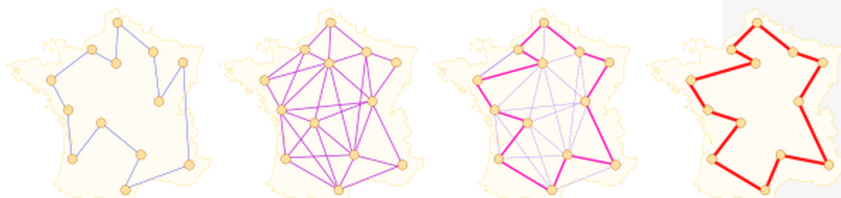


Ant colony optimization

[\[edit\]](#)

Artificial intelligence researcher [Marco Dorigo](#) described in 1997 a method of heuristically generating "good solutions" to the TSP using a [simulation of an ant colony](#) called ACS (Ant Colony System).^[19] It models behavior observed in real ants to find short paths between food sources and their nest, an [emergent](#) behaviour resulting from each ant's preference to follow [trail pheromones](#) deposited by other ants.

ACS sends out a large number of virtual ant agents to explore many possible routes on the map. Each ant probabilistically chooses the next city to visit based on a heuristic combining the distance to the city and the amount of virtual pheromone deposited on the edge to the city. The ants explore, depositing pheromone on each edge that they cross, until they have all completed a tour. At this point the ant which completed the shortest tour deposits virtual pheromone along its complete tour route (*global trail updating*). The amount of pheromone deposited is inversely proportional to the tour length: the shorter the tour, the more it deposits.



Other hard problems in CS

- Hamiltonian cycle
 - Does a given graph contain a Hamiltonian cycle?
 - (Very related to TSP)
- Graph coloring
 - Can a given graph be colored with k colors?



Other hard problems in CS

- Sudoku on $n^2 \times n^2$ boards of $n \times n$ blocks

| | | | | | | | |
|---|---|---|---|---|--|---|---|
| 8 | | | 4 | 6 | | | 7 |
| | | | | | | 4 | |
| | 1 | | | | | 6 | 5 |
| 5 | | 9 | | 3 | | 7 | 8 |
| | | | | 7 | | | |
| | 4 | 8 | | 2 | | 1 | 3 |
| | 5 | 2 | | | | | 9 |
| | | 1 | | | | | |
| 3 | | | 9 | 2 | | | 5 |

- General algorithm for solving Sudoku?

