Heuristic algorithms

Solving hard problems

- Many important problems are NP-complete
  - There most likely doesn't exist a polynomial-time algorithm that is guaranteed to give the correct solution
- Monte-Carlo algorithms can sometimes be used to produce the right answer with high probability
- In other cases, all we can do is give an algorithm that (maybe) gives a (pretty) good answer
  - Better to have an imperfect solution than no solution at all!
  - These are called heuristic algorithm

Heuristic algorithms

- These algorithms come with no guarantee of producing the correct answer, but in practice tend to work well.
- Often used for difficult *optimization* problems
  - These are problems where we seek the "best" solution among a set of possible solutions
    - Shortest path between two vertices in a graph
    - Shortest Traveling Salesperson tour
    - Maximum value you can put in your backpack
  - Heuristics will give solution that might not be optimal, but that will hopefully be close
- You can imagine all kinds of heuristics, some better than others. Use your imagination!

Traveling Salesperson Problem

- Given:
  - A set of n cities to be visited
  - The distance matrix D, where D(i,j) is the distance between cities i and j
- Find:
  - A way to visit each city exactly once and to return to your starting point, to minimize the total distance traveled
- Example: Solution: ADCBE = 23

	A	B	C	D	E
A	-	8	3	4	5
B	8	-	7	8	5
C	3	7	-	2	6
D	4	8	2	-	7
E	5	5	6	7	-

Heuristics for TSP

- TSP (in its decision problem version) is NP-Complete
  - No known algorithm can give the optimal solution in polynomial time
  - Still, getting some suboptimal solution is better than getting no solution at all
  - For 50 cities, there are $49! = 6 * 10^{62}$ possible solutions
- People have come up with all kinds of approaches to get "good solutions"… We'll look at a few

TSP heuristics - Greedy algorithm

- Greedy algorithm:
  - Start at some randomly chosen city
  - Always move to the closest unvisited city
- Example: A C D E B -> 3 + 2 + 7 + 5 + 8 = 25

	A	B	C	D	E
A	-	8	3	4	5
B	8	-	7	8	5
C	3	7	-	2	6
D	4	8	2	-	7
E	5	5	6	7	-

TSP heuristics - Greedy algorithm II

- Other greedy algorithm:
  – Repeatedly pick the pair of cities that are the closest, as long at they don't close a cycle (except for the last one)
- Example: B E A C D, score = 5+5+3+2+8 = 23

	A	B	C	D	E
A	-	8	3	4	5
B	8	-	7	8	5
C	3	7	-	2	6
D	4	8	2	-	7
E	5	5	6	7	-

Fastest descent heuristics

Start with some random solution S
Repeat
  Consider a set of solutions $T_1 \ldots T_k$ in the "neighborhood" of S
  Replace S by the solution T_i with the best score
Until no improvement is possible

What does "neighborhood" mean? Many possibilities…
$T_1 \ldots T_k$ could be the solutions obtained from S by
- Changing the position of one city in S, or
- Exchanging the position of two cities in S, or
- Reverse the order in which a set of consecutive cities are visited

Fastest descent: Example

	A	B	C	D	E
A	-	8	3	4	5
B	8	-	7	8	5
C	3	7	-	2	6
D	4	8	2	-	7
E	5	5	6	7	-

We consider the method variations are obtained by reversing the order of a consecutive set of cities

Start with random solution:
ABDEC: 32
All possible "reversals":
BADEC: 32
ADBEC: 26
ABEDC: 25
ABDCE: 33
DBAEC: 29
AEDBC: 30
ABCED: 32

ABEDC: 25

BAEDC: …
AEBDC: 23
ABDEC: …
ABECD: …
EBADC: …
ADEBC: …
ABCDE: …

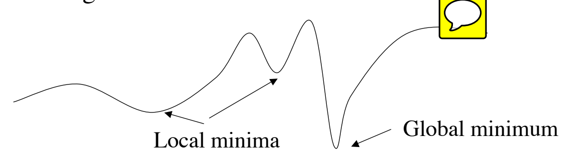
AEBDC: 23

EABDC
ABEDC
AEDBC
AEBCD
BEADC
ADBEC
AECDB

No improvement possible

Fastest descent - Trade-offs

- The larger the neighborhood, the higher the chance of converging to a good solution, but the more time it takes to evaluate each neighbor
- Fastest descent heuristics will often get stuck in a local optima solution:
  – Solution that is not optimal but that doesn't have a neighbor that is better than itself



Avoiding local minima Randomization

- Many many approaches have been proposed to avoid falling into local minima

Randomized fastest descent:
  Similar to fastest descent, but instead of choosing the "best" neighbor, add some randomization to it, so that
  - Good neighbors have higher probability of being chosen than bad ones
  - Even bad neighbors have a small chance of being selected

Randomization allows to escape from local minima, but slows down convergence

Avoiding local minima Genetic algorithm

Idea: Mimic species evolution in biology

Start with a random "population" of random solutions $S_1 \ldots S_n$
Repeat for many generations:
  For k = 1…n
    1) Randomly pick two parent solutions S_i and S_j, with probability that depends on their score
    2) Create a hybrid offspring T_k from S_i and S_j, which inherits some of the properties of the parents
    3) Insert a few random "mutations" in T_k
  Replace solutions $S_1 \ldots S_n$ by solutions $T_1 \ldots T_k$