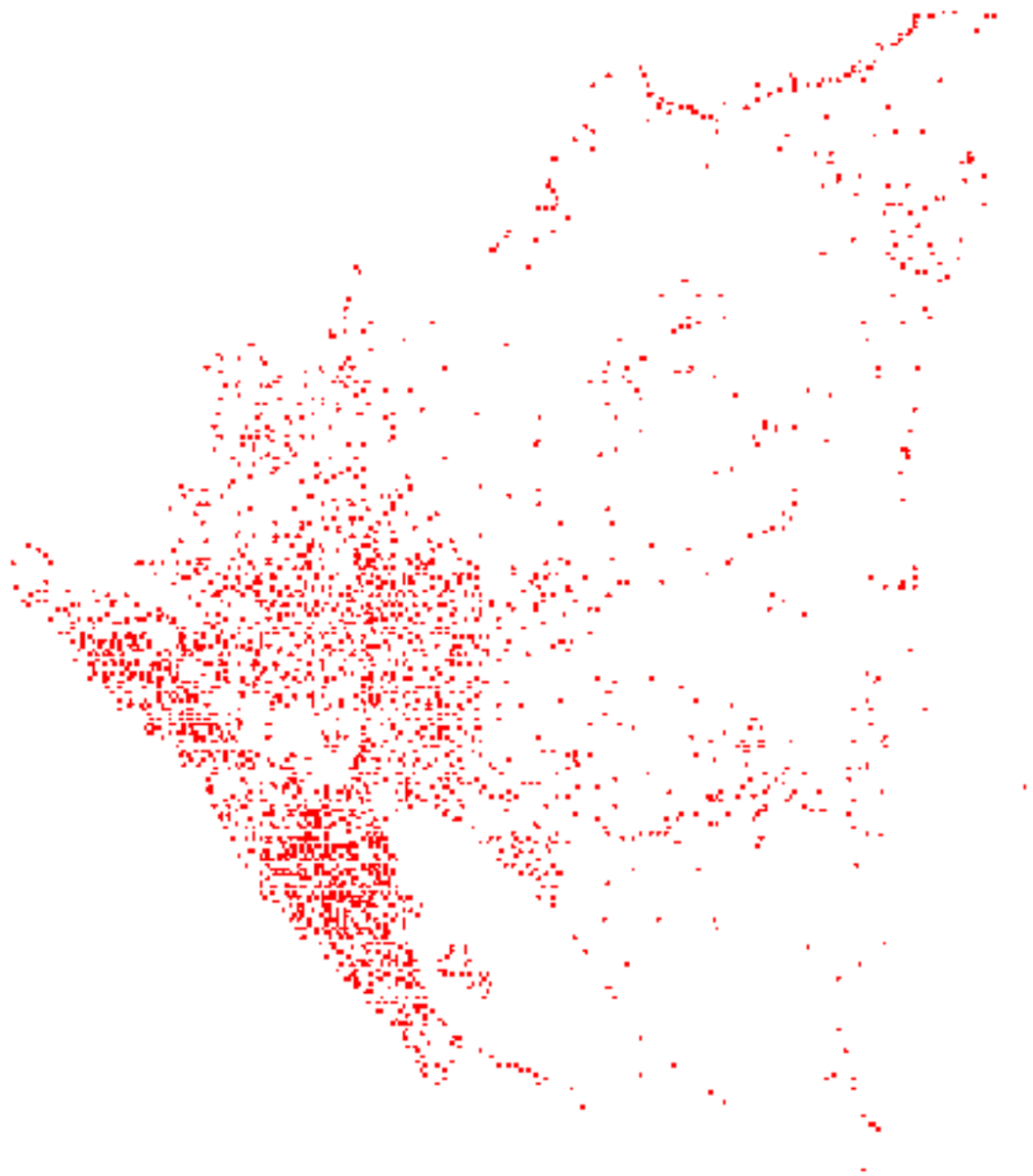
Hands-on #1: Travelling Salesman Problem

School of Computing, KAIST
Shin Yoo

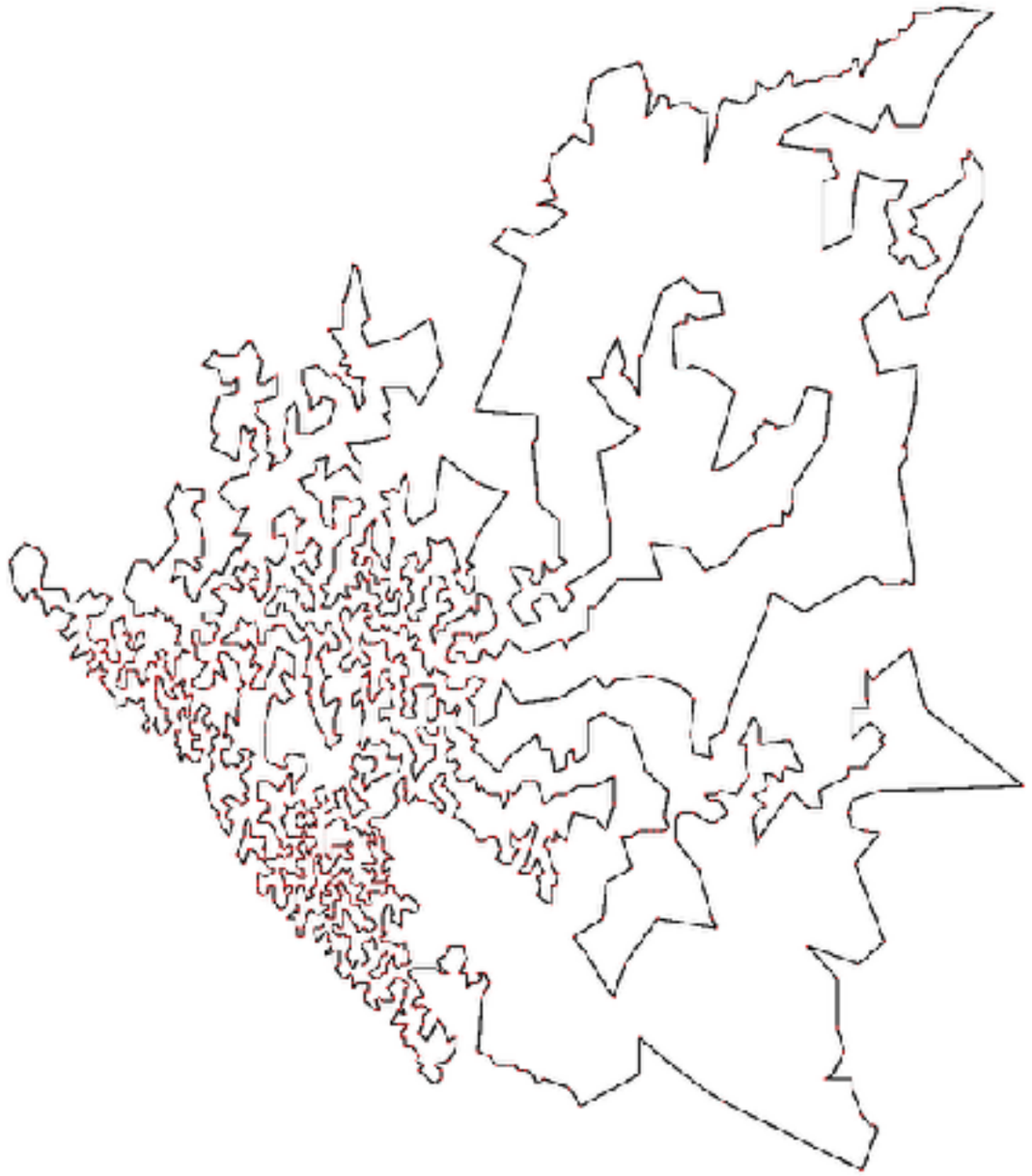
Travelling Salesman Problem

- Given **`N`** points in space (usually 2D surface)
- Find the shortest **tour** of all points.
- Search space: **`N!`**
- Computational complexity: NP-complete
- Brute Force: **`O(N!)`**



Travelling Salesman Problem

- Given **`N`** points in space (usually 2D surface)
- Find the shortest tour of all points.
- Search space: **`N!`**
- Computational complexity: NP-complete
- Brute Force: **`O(N!)`**



Exact Algorithms

- Early dynamic programming
  - Held-Karp algorithm: $O(n^2 2^n)$
- Linear Programming
  - 15,112 German cities: 22.6 CPU years on 500MHz Alpha, 2001
  - 33,810 points on a circuit board: 15.7 CPU years, 2005
  - 85,900 points: 136 CPU years

Heuristic Approach

- Many specific genetic operators have been designed.
- Use domain knowledge. For example:
  - Euclidean TSP observes triangular inequality.
- We're still introducing new algorithms: you can apply them as we go.

Heuristic Approach

- What is the suitable representation?
- What is the fitness function? (one is provided, actually)
- What is the operator to modify existing solutions?

TSP Hands-on

- The g rse page seso

- An ut e **bie**

- It contains coordinates of 127 beer gardens in Ausburg. Your goal is to find the shortest route that passes all the beer gardens!

TSP Hands-on

- Try your own solution for 1 hour. Remember:
  - Form a candidate solution(s)
  - Measure its(their) fitness
    - Modify the candidate solution(s) in the direction of better fitness
- There is a DEAP based GA template, if you want to try GA.
- **The person who produces the shortest tour after 1 hour will get a prize!**
- After an hour, we will go through a GA based solver together.

TSP using Genetic Algorithm

- What would be the GA formulation?
    - A population of permutations
    - A fitness function
    - A way to select individuals for reproduction (generic, not specific to TSP)
    - A way to cross-over permutations
    - A way to mutate permutations

Cross-over for Permutations

- Single Point Crossover does not work for obvious reasons
- [**1, 3, 2, 4, 5**] x [**4, 2, 5, 1, 3**] = [**1, 3, 5, 1, 3**] x [**4, 2, 2, 4, 5**] ??
- We need permutation-specific cross-over operators. This is a whole research area in optimisation:
  - A. Andreica and C. Chira. Best-order crossover for permutation-based evolutionary algorithms. Applied Intelligence, 42(4):751–776, Jun 2015.

Partially Mapped Crossover (PMX)

- First, randomly mark a subsequence in both parents: offsprings exchange the subsequences

- Offsprings try to inherit the remaining parts. However, this may not be possible, because the corresponding value appears in the copied subsequence.

- In this case, follow the partial-mapping between two subsequences until the conflict is resolved.

Partially Mapped Crossover (PMX)

Parents

(1 2 3 4 5 6 7 8)

(3 7 5 1 6 8 2 4)

(1 2 3 | 4 5 6 | 7 8)

(3 7 5 | 1 6 8 | 2 4)

Partial Mapping 1⇔4, 5⇔6, 6⇔8

Offsprings

(| 1 6 8 |)

(3 7 8 | 4 5 6 | 2 1)

(X X X | 1 6 8 | X X)

(X X X | 4 5 6 | X X)

Mutation

- Random pair swap
- 2-Opt
  - Copy seq[0] to seq[i-1] to the new sequence
  - Copy seq[i] to seq[i + k] to the new sequence in reverse order
  - Copy seq[i + k + 1] to end to the new sequence

2-Opt Mutation

- If there is a subsequence that cross itself over, 2-Opt can **untangle** it

- 3-Opt
  - delete 3 edges, which creates 3 sub tours
  - consider 7 different ways of recombining them