Comparative Study of the Application of Evolutionary Computing Strategies to the Traveling Salesman Problem

Syed Yasin Dara

Naozumi Hiranuma

Evan Minter Albright

Overview

- What is the Travelling Salesman Problem?
- NP-Completeness, Expectations, Our Hypothesis
- Strategies:

  Heuristics

  Genetic Algorithms

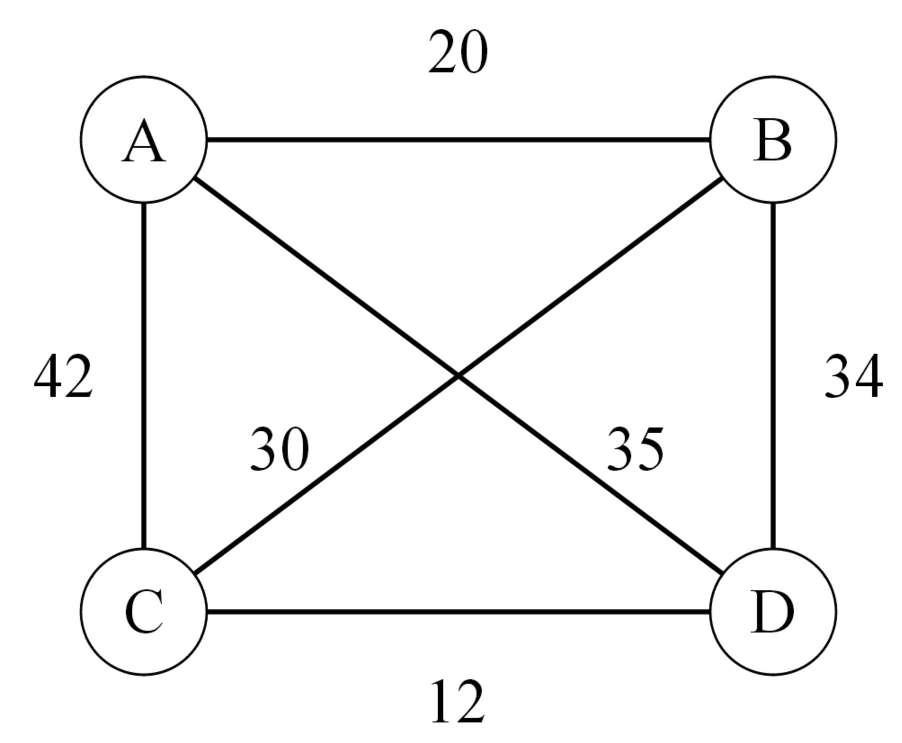
  Genetic Programming

  Swarm Intelligence

Algorithms

- Random Approach
- Nearest Neighbor Algorithms (Greedy)
- Insertion Heuristics
- Genetic Algorithms (GA)
- Genetic Algorithms + Niching Techniques
- Simulated Annealing
- Genetic Programming
- 2-opt Tour Improvement
- Ant-Colony Optimization (Swarm Intelligence)

The Travelling Salesman Problem



William Rowan Hamilton

We denote the Travelling Salesman Problem: the task to find, for finitely many points whose pairwise distances are known, the shortest route connecting the points.

Of course, this problem is solvable by finitely many trials.

Running Time

- The problem has been shown to be NP-hard, and the decision problem version ("*given the costs and a number x, decide whether there is a round-trip route cheaper than x*") is NP-complete.

- Running Time of Existing Solutions:

  Brute Force: $O(n!)$

  Held-Karp Algorithm (Dynamic Programming): $O(n^2 2^n)$

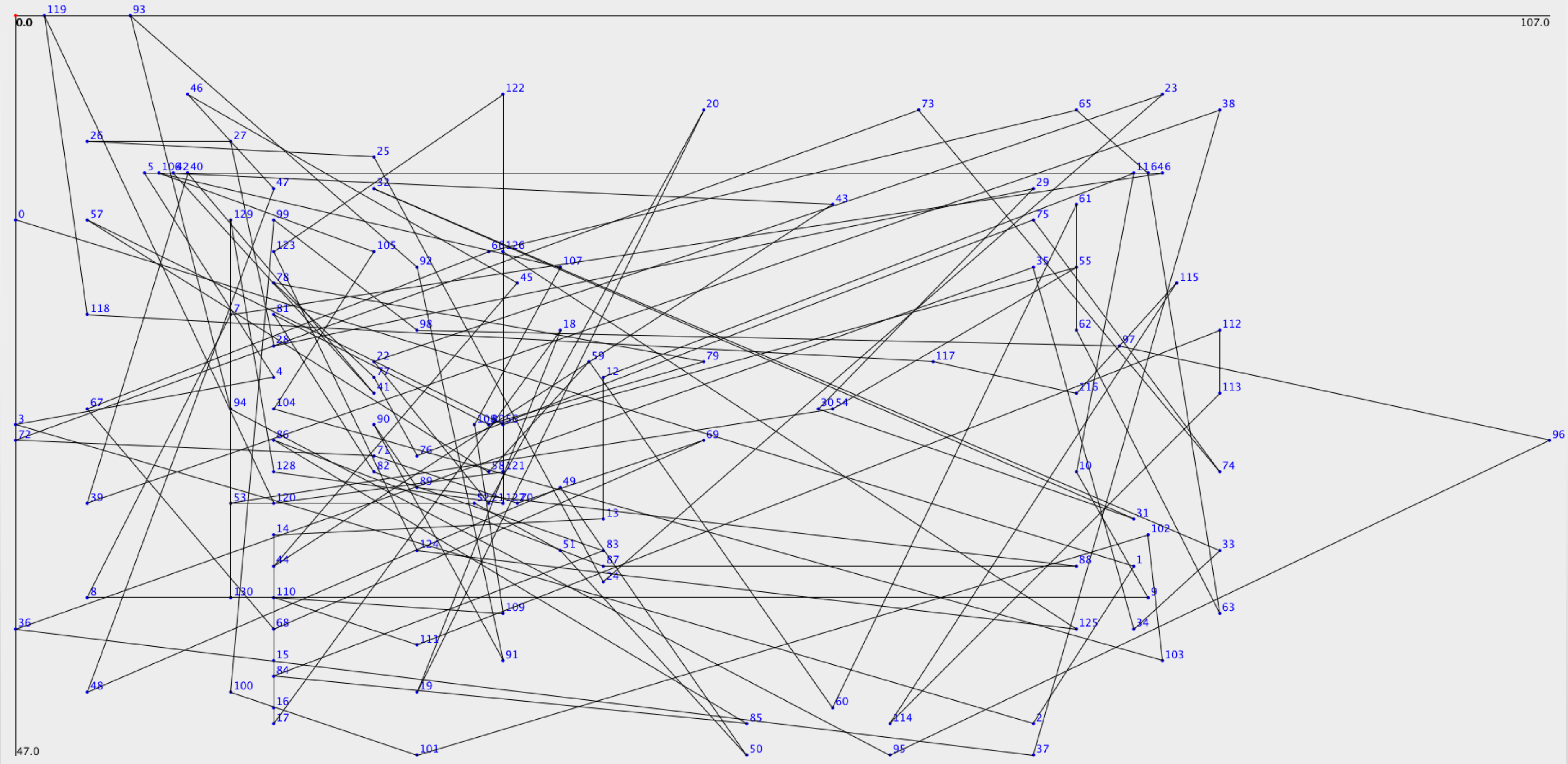
Hypothesis

- Evolutionary computing approaches outperform the known heuristics for approximating a solution.*
- We expect that a genetic algorithm with niching techniques will perform the best, as the fitness landscape for the TSP has many local optima.

* Limitations: Due to a lack of time and computing resources, our solutions will be qualitatively worse than solutions found by existing heuristic approaches.

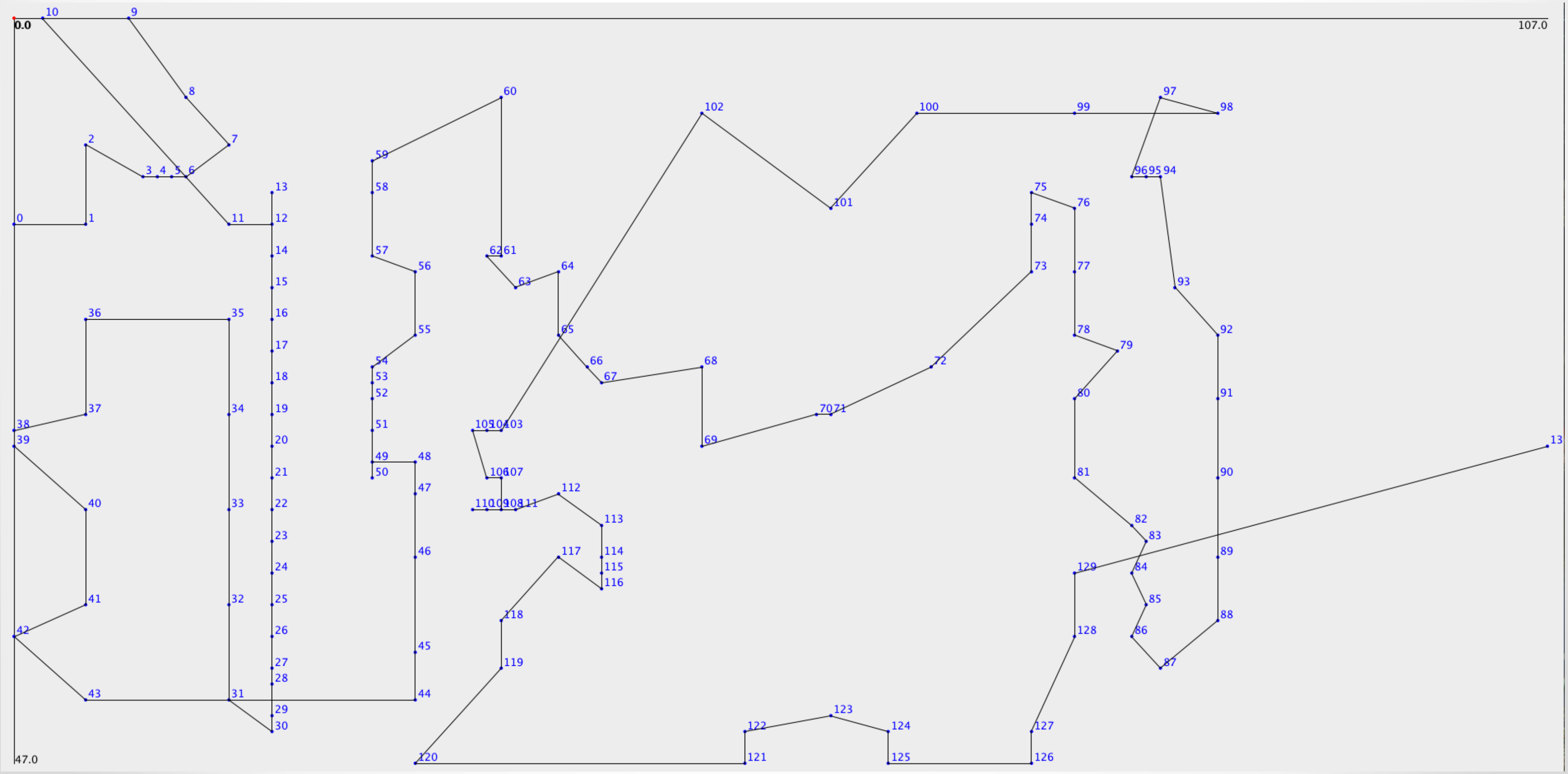
Random Search

Minimum Travel Distance: 3783.55



Nearest Neighbor

Minimum Travel Distance: 709.52



Insertion Heuristic

Begin with a sub-tour consisting of just the first city.

While there are unused cities:

{

Find what the smallest distance increase would be if any city was inserted anywhere in the sub-tour, and insert it.

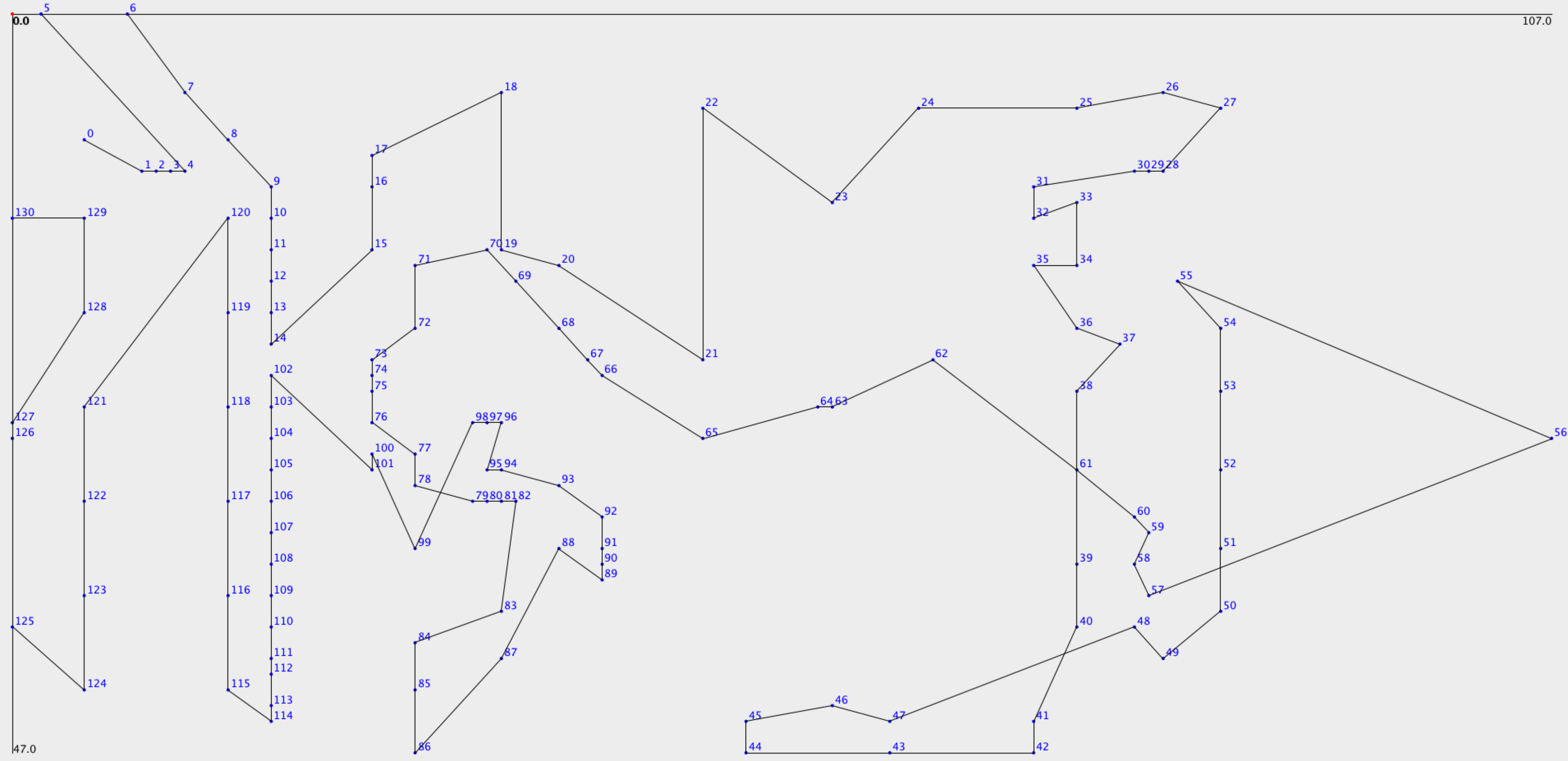
}

Return completed tour.

- The smallest distance finder is an $O(n^2)$ operation for the first pass, and decrements n each pass, making it slightly more efficient than just an $O(n^2)$ algorithm.

Insertion Heuristics

Minimum Travel Distance: 669.13



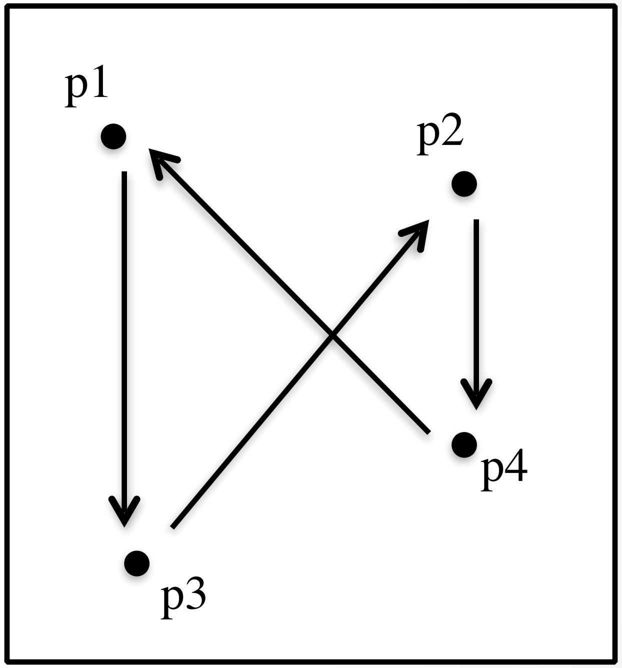
Evolutionary algorithms

- Simulated annealing
- Genetic algorithms
- Genetic algorithms w/ niching
- Genetic programming
- Swarm intelligence

Genome representation

- A list of points in the order of visits.
- Where you start traveling doesn't matter.

- E.g.)
  - [p1,p3,p2,p4]
    = [p3,p2,p4,p1]



Mutation

- Each city can only be visited once.
- Mutation 1
  - [p1,p2,p3,p4,p5] → [p1,p5,p3,p4,p2]
- Mutation 2
  - [p1,p2,p3,p4,p5] → [p1,p5,p2,p3,p4]
- Mutation 3
  - [p1,p2,p3,p4,p5] → [p1,p3,p4,p5,p2]

Crossover

- K-point, or uniform crossover does not work.

Parent 1: [1,8,2,6,3,4,5,7]   Parent 2: [1,4,5,2,6,8,7,3]

Child 1: [1,2,6,8,3,4,5,7]   Child 1: [1,2,3,4,6,8,7,5]

Simulated annealing

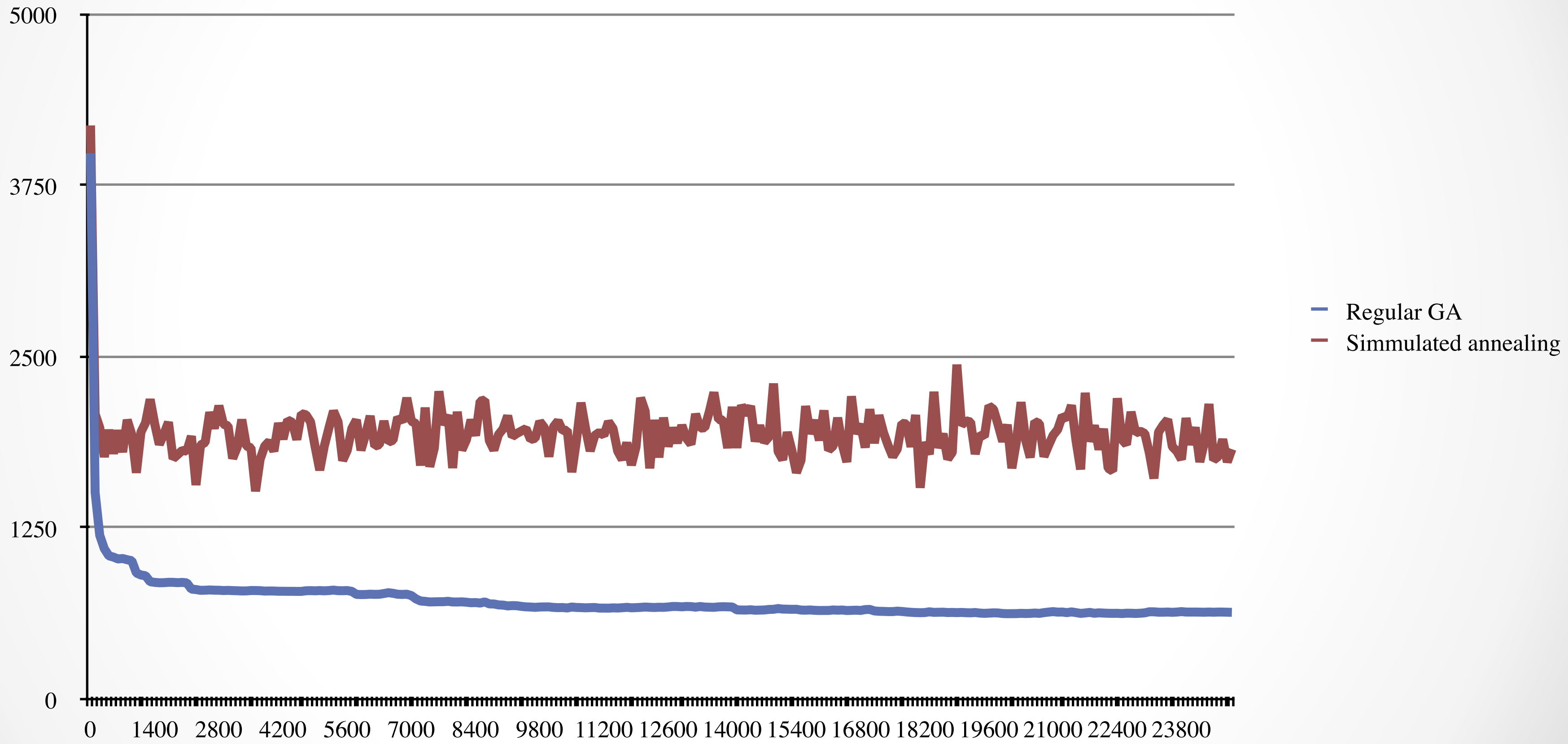
- 10,000,000 fitness evaluations.
- Average: 2075.8 (stderror 151.9)
  - Random: 3783.55
  - Nearest neighbor: 709.52
  - Insertion: 669.13

Genetic Algorithm

- 400 individuals x 25,000 generations

  = 10,000,000 fitness evaluations
- Average: 639.1 (stderror: 17.9)
  - Random: 3783.55
  - Nearest neighbor: 709.52
  - Insertion: 669.13
-

GA vs. Simulated annealing



Genetic Algorithm + Niching

- Hard to measure difference between 2 routes.
  - Hamming distance
    - O(N^2) x O(N) = O(N^3)
    - Creates a deceptive landscape
      - [1,2,3,4,5] and [1,3,4,5,2] are similar
  - Levenshtein distance (edit distance)
    - O(N^2) x O(N^2) = O(N^4)
    - Takes too much time

Genetic Programming

- An evolutionary algorithm-based methodology that evolves computer programs to perform a user-defined task.
- A set of instructions and a fitness function to measure how well a computer has performed a task.
- A specialization of genetic algorithms (GA) where each individual is a computer program.

Oltean, Mihai. "Evolving evolutionary algorithms using linear genetic programming." Evolutionary Computation 13.3 (2005): 387-410.

Genetic Programming

Standard GA Algorithm

S1. Randomly create the initial population $P(0)$
S2. for $t = 1$ to *Max Generations* do
S3. $P'(t) = \phi$;
S4. for $k = 1$ to $|P(t)|$ do
S5. $p1 = Select(P(t))$; // select an individual from the population
S6. $p2 = Select(P(t))$; // select the second individual
S7. *Crossover* ($p1$, $p2$, *offspr*); // crossover the parents p1 and p2
// an offspring *offspr* is obtained
S8. *Mutation* (*offspr*); // mutate the offspring *offspr*
S9. Add *offspr* to $P'(t)$; //move *offspr* in the new population
S10. endfor
S11. $P(t+1) = P'(t)$;
S12. endfor

- *Oltean, Mihai. "Evolving evolutionary algorithms using linear genetic programming." Evolutionary Computation 13.3 (2005): 387-410.*

Genetic Programming

```
void LGP Program(Chromosome Pop[8]) { // a population with of 8 individuals
        Randomly initialize the population();
        for (int k = 0; k < MaxGenerations; k++) {  // repeat for a number of generations
                Pop[0] = Mutate(Pop[5]);
                Pop[7] = Select(Pop[3], Pop[6]);
                Pop[4] = Mutate(Pop[2]);
                Pop[2] = Crossover(Pop[0], Pop[2]);
                Pop[6] = Mutate(Pop[1]);
                Pop[2] = Select(Pop[4], Pop[3]);
                Pop[1] = Mutate(Pop[6]);
                Pop[3] = Crossover(Pop[5], Pop[1]);
        }
}
```

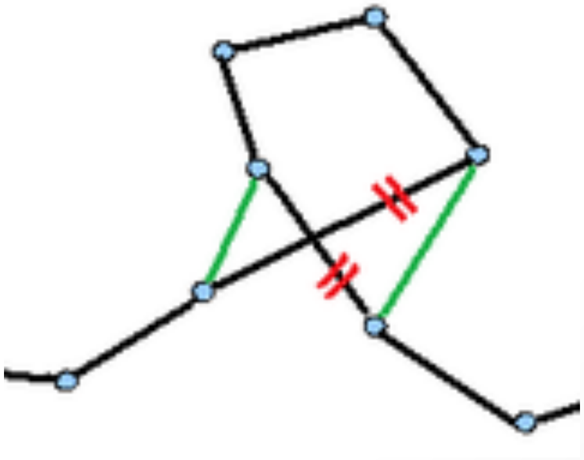
Oltean, Mihai. "Evolving evolutionary algorithms using linear genetic programming." Evolutionary Computation 13.3 (2005): 387-410.

2-opt

- Unrestricted 2 opt pseudocode
  - while improved:
    - for every pair of edges in a tour
      - swap the edges
      - If improved, continue with new tour
      - If no improvements, return current tour

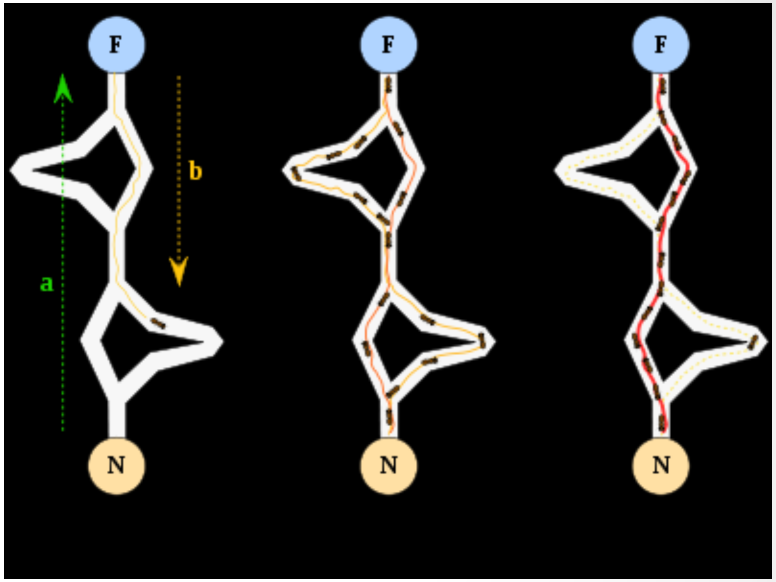
- Run time > $O(n^3)$
  - *~484 rounds*

- Good results!
  - *~604 distance*



Ant Colony Optimization

- Plain pseudocode
  - For generation
    - For all ants in population
      - Pick a path
      - Locally optimize the path
      - Lay down pheromones on path

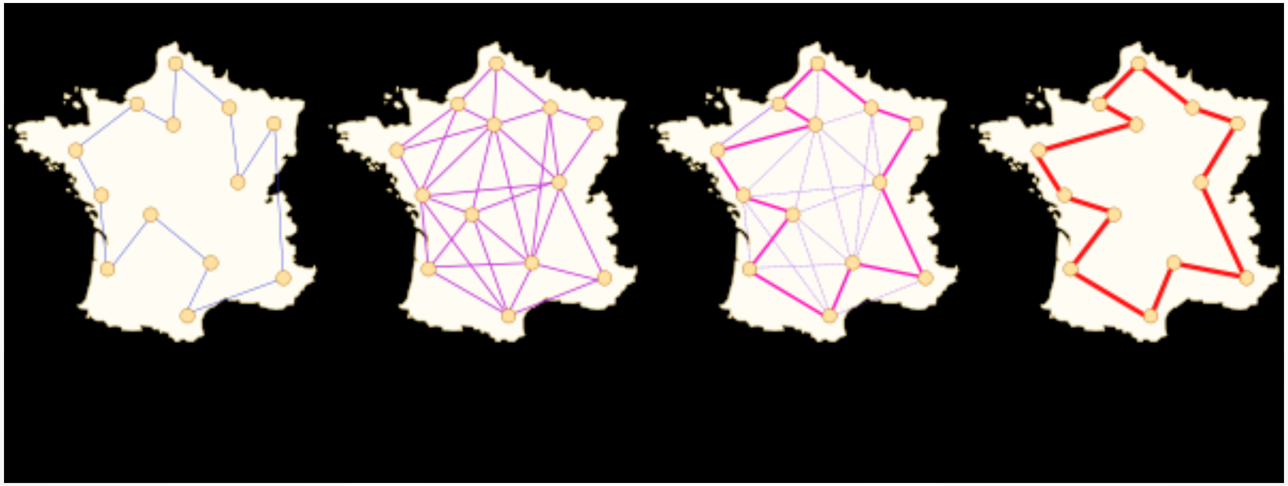


- Population – search strength (avoid local optima)

- Generation – honing in on optima

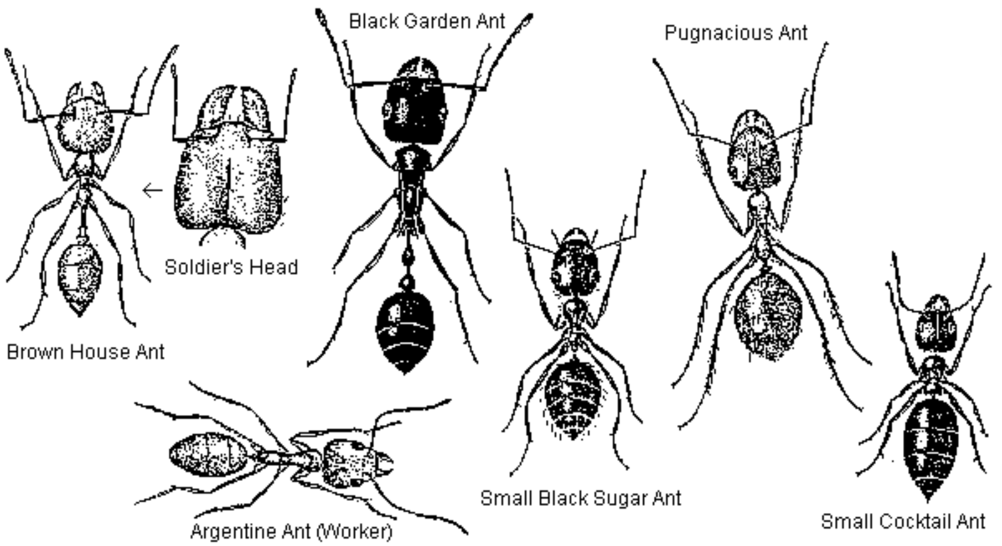
ACO Pheromones

- “Edge weights”
- Local Updates
- Global Updates
- Evaporation



ACO Types

- Vanilla
- Max Min
- Elite Ants
- Rank Based
- Continuous Orthogonal Ant Colony (??)



ACO Costs

- n^2 space for pheromone map
- Single Ant costs
  - n^2 for path picking
  - (horribly overstated) > n^3 for local optimization
  - n updating map
- * population size
- * generation

ACO Results

- Great tour ☺
  - *~591 distance*
- Monstrous costs ☹
  - (Opt wasn't restricted)