

Originality Statement: I confirm that all work submitted as part of this document is my own. I used sources such as GeeksForGeeks, Python documentation, and ChatGPT for things like zip(), sort(), plotting and other Python nitpicky things

Algorithms Used

Simulated Annealing

Start with a random tour and initial temp of 1

Start loop

Swap two cities randomly

Compute the cost of the old tour and the new tour

If new tour is better, accept it

Else accept new tour with probability based on exponential decay function which I defined as

```
def exp_decay(time, init_temp=100, min_temp=0.001, exp_const=0.005):
```

```
    Return max(init_temp * math.exp(-exp_const * time), min_temp)
```

Repeat for 1000 iterations

Evolutionary Algorithm

Generate k initial tours, where k is 20

Start Loop

Generate k successor tours with mutation (defined as swapping two random cities)

Pick the k-best tours from the initial and successor tours by least cost is better

The k-best are now the initial tours

Repeat for 1000 iterations

Population-based Search

Generate k initial tours, where k is 20

Start Loop

Generate k successor tours with mutation (defined as swapping two random cities)

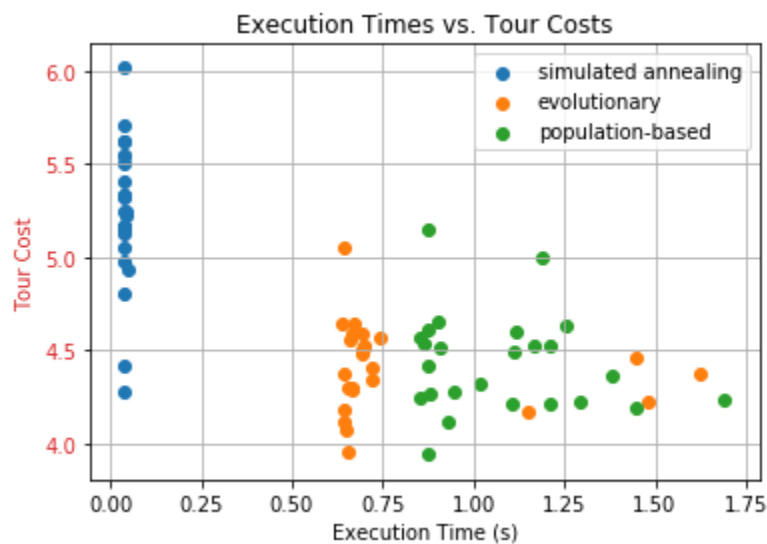
Pick the k -best (beam width) tours from the initial and successor tours by least cost is better

The k -best are now the initial tours

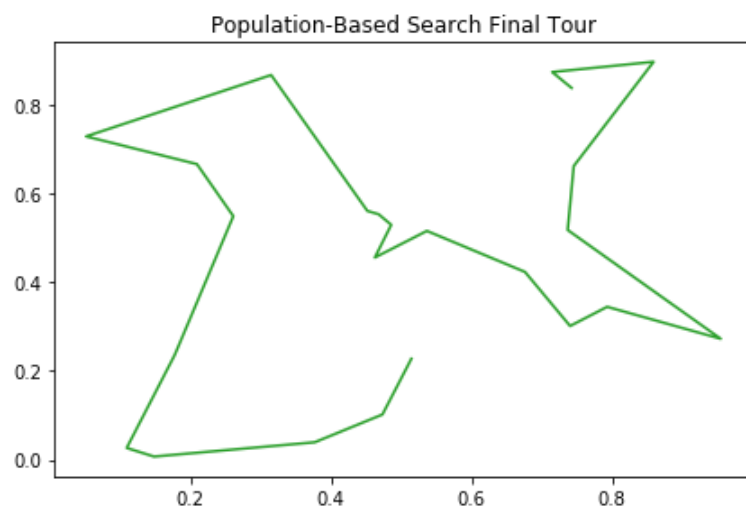
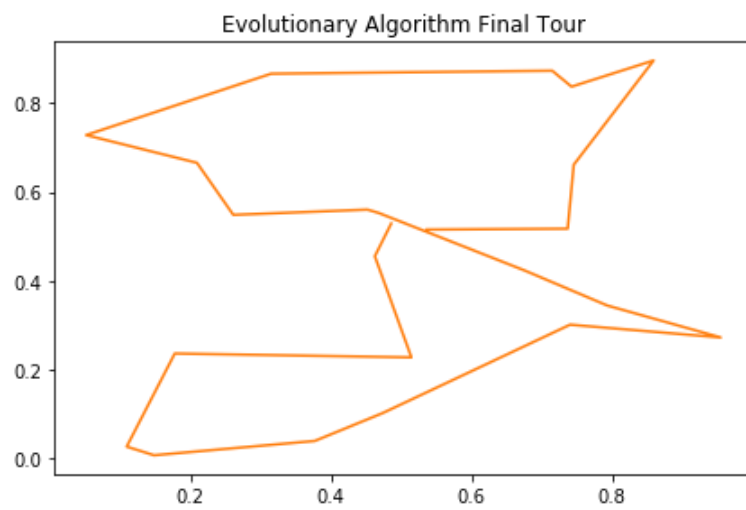
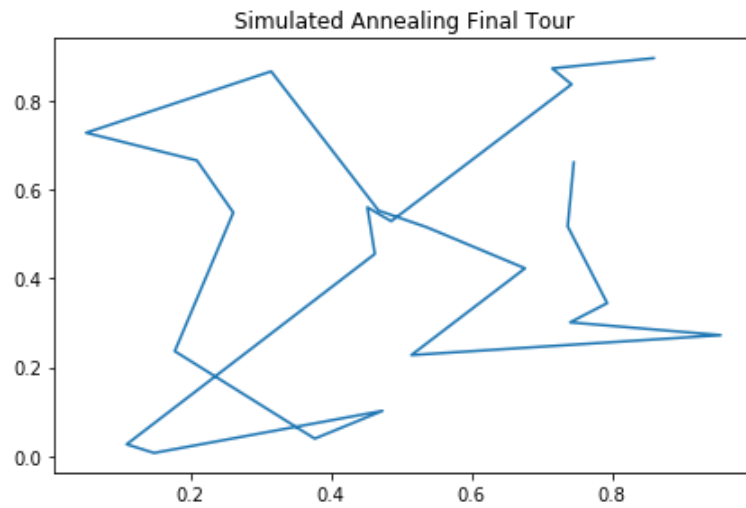
Repeat for 1000 iterations

Experimental Methodology

Using `time.time`, I recorded the amount of time it took to complete each algorithm for 1000 iterations and recorded the best cost it came up with. I repeated this for 25 runs and plotted the results as below. All parameters used are listed in the algorithm description



Final Tours



Discussion questions:

How many “solutions” did your algorithms generate during their searches?

Simulated annealing only generates one new solution each iteration so in total 1000.

My implementation of evolutionary algorithm started with 20, and generated 20 new solutions each iteration which gives a total of $20 + 20 \times 1000$ so 20020.

My implementation of beam search started with 20 and generated 20 new solutions each iteration so also a total of 20020.

How many solutions are there for the TSP with 25 cities?

$25! \approx 15,511,210,043,330,985,984,000$

15 quintillion if we want to put it into words

What percentage of all solutions did your algorithms search through?

Simulated annealing: $6.446950284384474e-23$

Evolutionary+Beam: $1.2906794469337715e-21$

Safe to say, a very *miniscule* amount.

What are some of the benefits and difficulties of each of your search algorithms?

Simulated annealing is always faster than the other algorithms but since it just picks the better solution of two (mostly), it can fall into some bad solutions and it's not that consistent.

The evolutionary algorithm and beam search for population-based search are virtually identical in the way that I implemented them, but based on the specific perturbation or mutation done, this would affect it. If a state were defined as a robot location on its path to a goal, evolutionary would converge slower, but it may find a better solution.

Why do these algorithms not find an optimal solution every time?

In early iterations it's only looked or looking at a few solutions, so an optimal solution would be found by random luck. In simulated annealing, it only ever compares two solutions so it might throw away a solution that might have been made better with one change and pick one that seemed better at the time. This can lead it to bad solutions early on, but this is fixed with more iterations. Likewise the other two algorithms look at multiple solutions so they may find an optimal path faster. There are fun cost x iteration graphs in the code attached that show this in more detail.