Using Genetic Algorithms to Optimize the Traveling Salesman Problem

By: Ryan Honig

Abstract

My goal is to create a program that can solve the Traveling Salesman Problem, finding near-optimal solutions for any set of points. I will use genetic algorithms to try to find the optimal paths between the points. I would also like to expand my algorithm so that it can solve both symmetric and asymmetric problems. In the end, after I create a working algorithm that will find near optimal paths, I hope to create a graphic interface that will display the chosen points and the paths through those points as the algorithm runs.

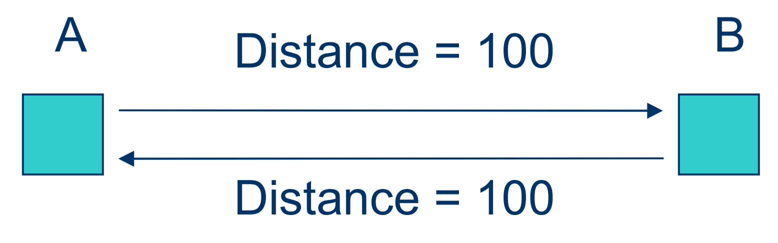
What is the Traveling Salesman Problem

Traveling Salesman Problem (TSP) - a set of points is given. Try to find the shortest path that travels between each point once and returns to the starting point

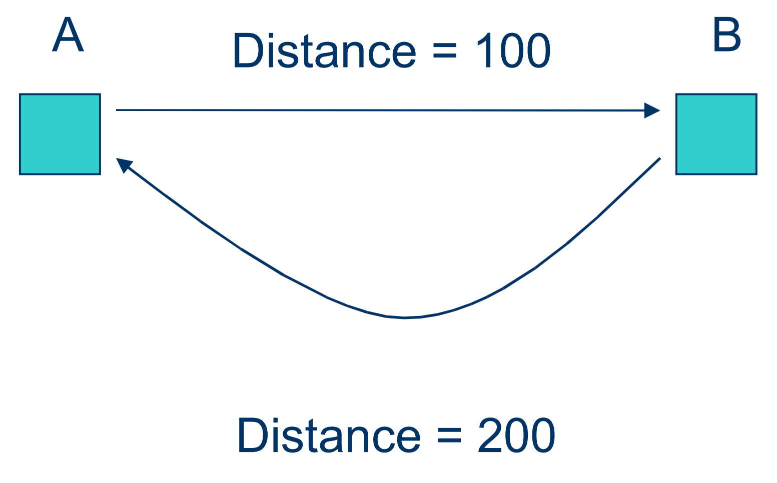
Symmetric TSP - distance between towns A and B is the same as distance between towns B and A.

Asymmetric TSP - distance between towns A and B is different from distance between towns B and A.

Symmetric



Asymmetric



Development

- I have a genetic algorithm that creates a pool, and then uses genetic crossovers within the pool to find the best solution
- I also have a mutation function that has a one in fifty chance of adding a variation into the pool by reversing a segment of a path, this helps to keep the pool from getting filled by copies of the same path
- I also created a heuristic that creates a better pool than the randomized pool, although it runs much slower
- During third quarter, I began work on converting my random pool program so that it can find near optimal solutions to asymmetric traveling salesman problems

Results

Testing the random-pool program against the Heuristically generated pool program

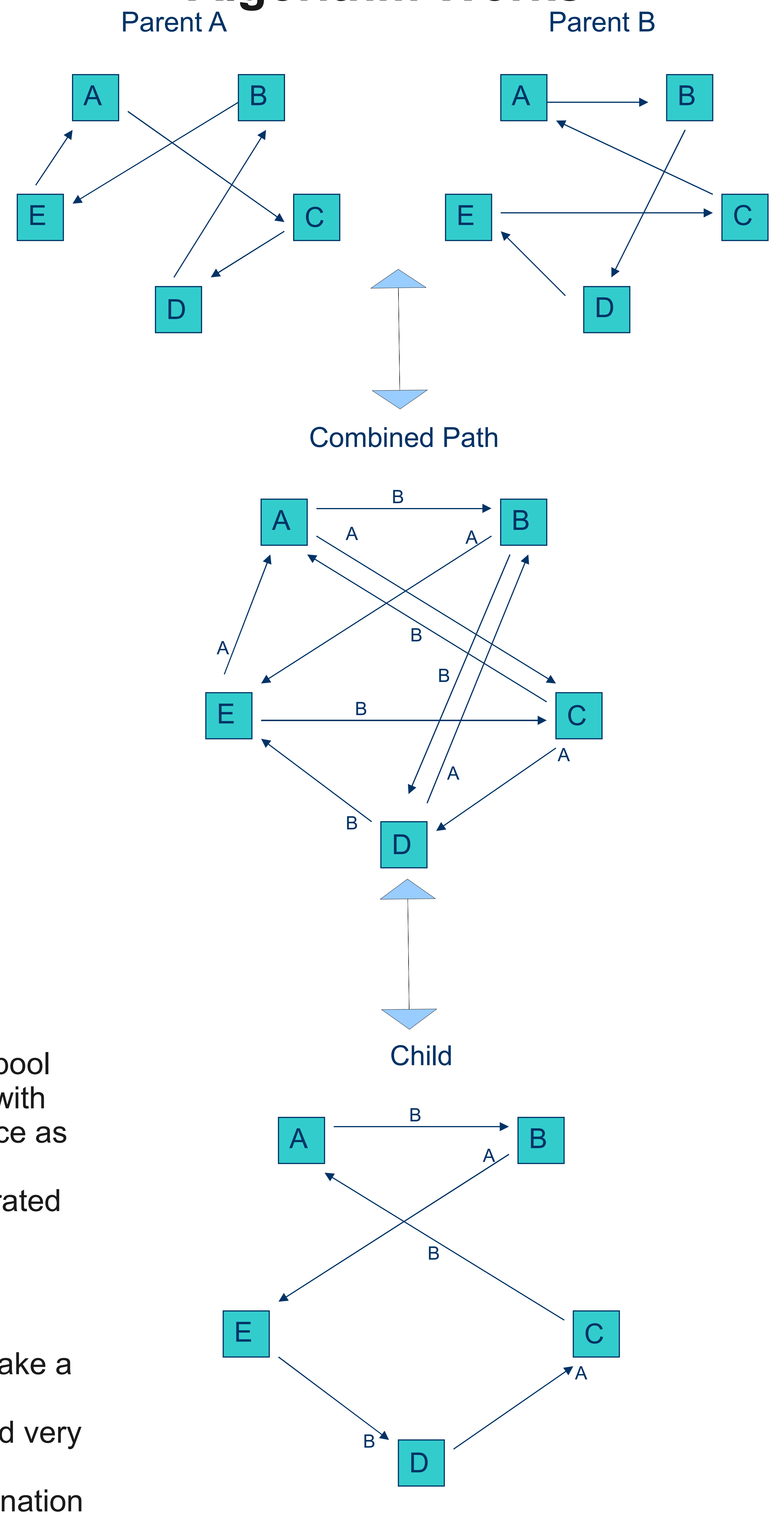
	Random Pool Program		Heuristic Program	
Data Set / Best solution	Average (of 5 runs)	Average Run time	Average (of 5 runs)	Average Run Time
A280: 2579	2780.54	1.75 sec	2729.37	3.03 sec
ATT48: 10628	12017.46	2.31 sec	12104.32	4.71 sec
BAYG29: 1610	1750.92	1.33 sec	1683.84	2.42 sec
BAYS29: 2020	2385.34	1.86 sec	2327.77	2.81 sec
CH130: 6110	6493.65	2.76 sec	6387.37	4.54 sec

- As you can see from my data, while the heuristically-generated pool program found slightly better solutions on most of the data sets, with the exception of data set ATT48, on every case it took almost twice as long to run than the randomly generated pool program did. I am currently not sure whether I will stick to using the randomly generated pool program or the heuristically generated pool program.

Background

- Purely genetic approaches can find near optimal solutions, but take a long time
- Purely heuristic approaches can run very efficiently, but don't find very optimal solutions
- Many of the current best known solution algorithms use a combination of heuristics and genetic algorithms

How My Genetic Algorithm Works


Parent A
Parent B
A
B
C
D
E
Combined Path
Child