Computer Systems Research Paper Initial Draft
Using Genetic Algorithms to Optimize the Traveling Salesman Problem
2007-2008

Ryan Honig

November 2, 2007

1 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. In the end, after I create a working alorithm 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.

2 Purpose

The main purpose of my project is to develop my own genetic algorithm that can hopefully find close to optimal solutions for the Traveling Salesman Problem. Once this is done I hope to modify the program to work for asymmetric problems and create a user interface that will graphically display the current problem and run the algorithm to find a solution. This is a good problem to tackle because it is fairly complex and deals both with some complex algorithms and with some higher level math. By finding an efficient and optimal solution to the traveling salesman problem, it can be applied to the larger

NP-complete field of optimization problems which can contribute to many fields of study. The TSP has been around for a long time, but more efficient programs for solving the TSPs are still being created. Many different algorithms have been used to attempt to solve TSPs, including heuristics, genetic algorithms, colony based simulations, and brute force. Heuristics are the best for finding 'good', but not optimal, paths fairly quickly, while genetic algorithms take longer but find more optimal paths. The paper: "New Genetic Local Search Operators for the Traveling Salesman Problem" by Bernd Freisleben and Peter Merz details how a good way to create an algorithm for the Traveling Salesman Problem is to use a basic heuristic to find the initial pool of paths and then use the genetic algorithm on this pool of paths to find a near-optimal solution. I hope to build off of this approach by creating an algorithm that will work for both symmetric and asymmetric TSPs. Another approach that is detailed by Marco Dorigo and Luca Maria Gambardella in "Ant Colonies for the Traveling Salesman Problem" is to use a simulated ant colony to solve a TSP data set. While this is not the most efficient way of solving a TSP, it can find very near-optimal solutions. One of the most interesting articles that I found on the Traveling Salesman Problem is "Genetic Algorithms for the Traveling Salesman Problem: A Review of Representations and Operators". This article has a comparison does a comparison of the different types of algorithms used to solve TSPs and their different way of representing the data. The question that I would like to answer through my project is what combination of algorithms can create the most efficient and optimal traveling salesman program.

3 Development

With my project, I would like to develop an efficient algorithm that can find near-optimal solutions for both symmetric and asymmetric traveling salesman problems and then incorporate it into a user interface that will run the algorithm and display the paths that the algorithm comes up with. My algorithm will be a mix of basic heuristics and the more complex genetic algorithms. I began by creating a program that used a simple genetic algorithm that would reverse a section of a parent path which would then be replaced in the pool if it had a shorter path than the parent. I began testing this with data sets that can be found here: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/. After finding my solu-

tions were off by multiple powers of ten, I discarded that algorithm and began a new one. This new algorithm starts by creating an initial pool of fifty random, legal paths. For each iteration of the genetic algorithm it will then select two parent paths at random to create a child path from. All of the links between each point on the parent path are then compiled into one set of links. The program will then alternate choosing a link from each of the two parents to create the crossover. If the program gets stuck on a node and cannot create a legal link from the parent links, then a greedy algorithm takes over and completes the broken path. I have been testing this new algorithm and I found that I have a bug in my code. In a small fraction of iterations of the genetic algorithm, a section of a path might be duplicated, creating an illegal path. I am working on fixing this bug and have narrowed it down to within the greedy algorithm.

4 Results and Discussion

After testing my initial algorithm that reversed sections of the paths, I was not surprised to find that my solutions to data sets were multiple powers of ten off from the best known solutions. I knew that since my initial algorithm was based off of single parent genetics, it would not work very well. When I began testing my second algorithm, I was surprised to find that my solutions were much lower than the best known solutions. After a little while I realized that this was due to a bug in my program. I hope to continue testing this algorithm soon, after I fix this bug.

5 Bibliography

—Dorigo, Marco and Gambardella, Luca Maria. "Ant colonies for the Traveling Salesman Problem". http://code.ulb.ac.be/dbfiles/DorGam1997bio.pdf

—Freisleben, Bernd and Merz, Peter. "New Genetic Local Search Operators for the Traveling Salesman Problem". http://www.rfai.li.univ-tours.fr/pagesperso/rousselle/do

—Larranaga, P., Kuijpers, C.M.H., Murga, R.H., Inza, I., and Dizdarevic, S. "Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators". http://wedhusprucul.tripod.com/skripsi/tsp.pdf

6 Appendices

1. An Overview of the Traveling Salesman Problem:

The Traveling Salesman Problem is a problem in which a set of points is given and you want to find the shortest path that travels between each point once and then returns to the starting point. A symmetric problem is one in which the distance between towns A and B is the same as the distance between towns B and A. An Asymmetric problem is one in which the distance between towns A and B is different from the distance between towns B and A.