The Algorithm

Monte Carlo Popularity with 2-Opt

Presented as an efficient alternative to finding acceptable solutions to

The Travelling Salesman Problem

Submitted to

Dr. Tony Martinez

For

CS 312: Algorithm Analysis

Brigham Young University

Provo, Utah

December 11, 2014

By

Tommy Jensen, Sean Lane, and Epper Marshall

# Abstract

The Monte Carlo Popularity with 2-Opt algorithm (MCP2Opt) is a hybrid of algorithm methodologies attempting to combine efficiency and accuracy in solving TSP. In this paper, we will analyze the paradigm behind Popularity, how a Monte Carlo modification makes it reliably productive, and how the 2-Opt heuristic (not created by the authors) helps to make answers more accurate.

# I Introduction

This paper introduces an algorithm that was originally created by the authors as well as work done by others. An explanation of the Nearest Neighbor algorithm (known in class as “Greedy”) will set the stage for the Popularity algorithm’s paradigm as a greedy algorithm. Then, Popularity’s polynomial efficiency and its unique solution-finding efficiency when used with the Monte Carlo algorithm design will be discussed. Additionally, the author’s caching implementation of 2-opt will be shown to have great running time improvements over naively-implemented 2-opt without sacrificing any of the gains made by the brute force approach. Finally, the empirical analysis of MCP2Opt’s performance against Greedy and Branch and Bound will conclude this paper.

# II Greedy (Nearest Neighbor)

The Greedy algorithm (often known as “Nearest Neighbor”) is a commonly used approximate solver for TSP because it is easy to implement and gets an answer very quickly. However, it has the drawbacks associated with other local-search algorithms; namely, that it can get stuck in local minima and never find an even close to optimal solution. In fact, city layouts exist that Greedy will find uniquely worst possible solutions for! Nevertheless, the authors found it useful as a source of inspiration.

### Implementation

The authors implemented Greedy according to its well-known pseudocode:

1. Pick a random starting city
2. Find the shortest edge that goes to an unvisited city, and travel it
3. Repeat Step 2 until the original starting city is reached, unless no routes are available, then continue
4. If a dead-end was found, start over at Step 1 with a different start city

Using this pseudocode, Greedy becomes an excellent base-line for comparison for other TSP solving algorithms.