Experimental Study on the Asymmetric Traveling Salesman Problem

R. Bhat, D. Jariwala, M. Hrkić, Q. Wang

University of Illinois at Chicago
Department of Computer Science
CS 594 Project proposal

1. INTRODUCTION

In the traveling salesman problem, one is given a set of N cities and for each pair of cities \((c_i, c_j)\) a distance \(d(c_i, c_j)\). The goal is to find a permutation of the cities that minimizes the sum of distances between the consecutive cities in the permutation including the distance between the first and the last city in the permutation.

A lot of the research on heuristics on this problem has concentrated on the symmetric case (the STSP), where \(d(c_i, c_j) = d(c_j, c_i)\) for all pairs of cities \((c_i, c_j)\). For the general not necessarily symmetric case, typically referred to as the “asymmetric TSP” (ATSP), there are fewer publications. As a wide variety of ATSP applications arise in practice, we attempt to attack the ATSP in this project.

Current ATSP heuristics can be divided into three classes: (1) classical tour construction heuristics such as Nearest Neighbor and the Greedy algorithm, (2) local search algorithms based on re-arranging segments of the tour, and (3) algorithms based on patching together the cycles in a minimum cycle cover.

2. PROPOSED ALGORITHMS

We plan to develop a set of core algorithms that will be used on whole problem instances as well as on parts of those instances. Our set of algorithms will contain the following four algorithms: Tour Length Minimization algorithm (TLM) from [5], modified version of TLM with partitions using ideas from [2],
“Full-Blown” version and its heuristic variant with smaller computational complexity. For TLM algorithm one of the input requirements is a fixed tree. In [5] this input tree was based on Minimum Spanning Tree (MST). Although the TLM algorithm does not require symmetric distances MST does, so we need to perform some additional experiments to determine how to generate input fixed tree for TLM. Once these four algorithms are implemented in a modular fashion, we will use them in Multi-level, Regional Refinement and Interleaving flows.

2.1 Preliminary Project Responsibilities

- Acquiring and preparing test instances: Devang, Ben
- Input processing: Ben
- Acquire source code of TLM and modularize it: Milos
- Implementing modified TLM with partitions: Milos
- Acquire and modularize “Full-Blown” version: Devang
- Implement heuristic version of “Full-Blown” alg.: Devang
- Generating fixed tree topology for TLM: Rajeev, Ben, Milos
- Multi-level flow: Rajeev, Ben
- Regional Refinement: Rajeev, Ben
- Interleaving: Devang
- Experiments: Devang, Ben, Rajeev, Milos

3. EXPERIMENTAL METHODOLOGY

The experimental methodology follows what is described in [3] and [4]. There are total thirteen classes of the test instances available from http://www.research.att.com/~dsj/chsp/atsp.html. Out of these, twelve classes have instances that are randomly generated. Each class comprises of ten instances with 100 and 316 cities, three instances with 1,000 cities and one instance with 3,162 cities. The thirteenth class consists of “real-world” instances with number of cities ranging from 43 to 1,000 cities.
The solution quality will be determined in terms of the length of tour and the run time. The length of tour will be compared with Held-Karp bound obtained on the equivalent symmetric instance. The running time of the algorithm implementation would be compared with the results published in [4] employing normalization procedure explained in documentation received along with the benchmark generators.

4. REFERENCES