1 Traveling Salesman Problem and evolutionary algorithms

A shortest route visiting all cities needs to be found, without visiting the same city twice. This is the traveling salesman problem. While the problem is relatively easy for a few cities, it becomes computationally more demanding as more cities are added to the set. The number of possible routes is the factorial of the number of cities to visit. Three cities have six different ways of visiting each, but with five cities the number is already 120 and ten cities give an boggling number of 3 628 800 different routes. So, given the 48 capitals of the 48 continental states of USA, how can one evaluate all possible routes and find the optimal path before the Universe suffers from heat death?

The answer is you can't. The age of the Universe is 4.354×10^{17} seconds, and 48 cities have approximately 1.24×10^{61} different combinations. Even doing a yotta (10^{24}) evaluations per second, far beyond the abilities of any current supercomputer, will not suffice.

But lest we be discouraged, there are still ways of tackling this problem. One way is to use heuristic methods, such as evolutionary algorithms. They will not give the optimal solution, at least not without considerable time. But they can give an acceptably short route in a short time scale.

2 Partially mapped crossover

In evolutionary computation, a chromosome is a data structure which contains a path through the 48 US capitals. New paths are created through mutation, and parts of existing paths are combined through crossover.

This is a permutation problem, since the chromosomes cannot contain duplicate values. Having duplicate values would mean that a city would be visited twice, and the problem does not allow for such events. Therefore both the mutation and the crossover operators have to be adapted for this problem.

Redesigning the mutation operator is quite straightforward. Places between two randomly chosen cities are switched on the chromosome, thereby preserving unity on the chromosome.

Crossover is a bit more complicated. Since the switching of chromosome parts might bring duplicate values onto one chromosome, special procedures have to be implemented after the switching in order to correct for duplications. Partially matched crossover (PMX) is one way of addressing this problem.

Since I cannot describe PMX any better than what Andy Thomas at the Generation5 website[1] (who by the way has written the best descrip-

[1] `www.generation5.org/content/2001/ga_tsp.asp`

tion of PMX that I found on the internet), I will simply visually quote his description:

Partially Matched Crossover
With this in mind, we need a scheme analogous to simple crossover, but one which preserves the solution viability while allowing the exchange of ordering information. One such scheme is Partially Matched Crossover (PMX). In this scheme, a crossing region is chosen by selecting two crossing sites. We return to our initial 10 allele example, but now mark two crossing sites:
A = 614x829x0735
B = 729x361x5480

Crossover is performed in the centre region to yield the following *transitionary* offspring:

A~ = HH4x361x07H5
B~ = 7HHx829x54H0

The 'H' positions are holes in the transitionary offspring left by the deliberate removal of alleles which would otherwise be replicated in the crossing region. These holes are filled by cross-referencing with the parent of the alternate chromosome. This can best be explained with a step by step example:

1. Take the first hole in A~ at index 0, the missing allele value at this position is 6.
2. Search along B (the alternate parent of A~) and match the first allele encountered with a value of 6. This occurs in B at index 4.
3. Fill H[0] in ~A with the allele found at A[4], a value of 2.

Applying this process to the remaining holes, we obtain the following PMX offspring:
A' = 2943610785
' = 7618295430

With this particulars in mind, I proceeded to solving the TSP US route problem.

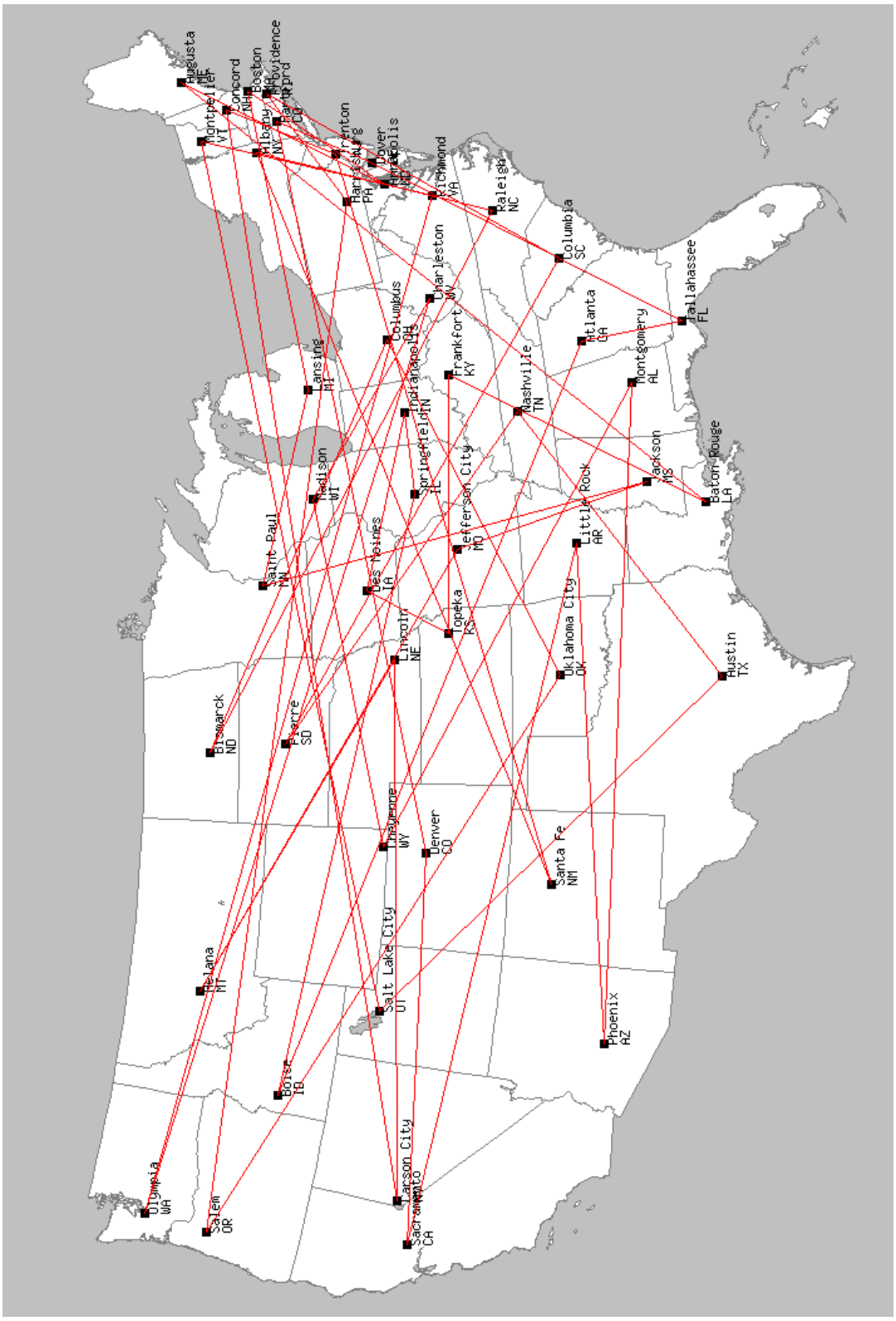
3 Methods and materials

The Euclidean distance between the cities were calculated on the basis of coordinates given by the instructor. A program emplying evolutionary algorithms was written in Perl 5.14 and run on a laptop with Crunchbang Linux and a 1.6 GHz processor.

The parameters for the evolutionary algorithms were a population size of 100, 0.05 mutation rate, 2000 generations, crossover rate 0.95. Crossover was 2-point, and tournament selection was employed with a tournament size of 6.

4 Results

The shortest path the program came to had a length of 47632 (distance unit unknown). The path is visualised in the following picture:


Augusta ME
Montpelier VT
Concord NH
Boston MA
Albany NY
Providence RI
Hartford CT
Trenton NJ
Harrisburg PA
Dover DE
Annapolis MD
Richmond VA
Raleigh NC
Columbia SC
Charleston WV
Tallahassee FL
Atlanta GA
Columbus OH
Frankfort KY
Montgomery AL
Lansing MI
Indianapolis IN
Nashville TN
Springfield IL
Jackson MS
Baton Rouge LA
Madison WI
Jefferson City MO
Little Rock AR
Saint Paul MN
Des Moines IA
Topeka KS
Lincoln NE
Oklahoma City OK
Austin TX
Bismarck ND
Pierre SD
Cheyenne WY
Denver CO
Santa Fe NM
Helana MT
Salt Lake City UT
Phoenix AZ
Boise ID
Carson City
Olympia WA
Salem OR
Sacramento CA

As it can be difficult to apprehend the exact route, the route was also drawn on a circle diagram: