Machine Learning 1

ICS2207

Assignment Report

Marc Ferriggi (286397M)

April 10, 2018

Contents

1 Introduction **1**

2 Genetic Algorithms **2**

3 Ant-Colony Optimisation **4**

4 Results and Comparisons **6**

5 Conclusion **6**

6 Statement of Completion **6**

7 Plagiarism Declaration Form **6**

8 Appendix **6**

1 Introduction

The Travelling Salesman Problem is a very common problem in the field of operations research. This has been studied extensively by mathematicians, computer scientists, and many great minds yet its complexity is still unknown [1]. The problem statement is given as follows:

"Given a collection of cities and the cost of travel between each pair of them, the travelling salesman problem, or TSP for short, is to find the

cheapest way of visiting all of the cities and returning to your starting point." [1]

Traditional methods for solving this problem come in three types; calculus based methods, exhaustive search methods and random search methods [3]. These methods, however, pose a large number of problems such as the algorithm converging to a local optimum or running in exponential time. Machine Learning Algorithms such as Genetic Algorithms (GAs) or the Ant-Colony Optimisation method (ACO) can be used to find accurate approximations to the solutions to the TSP and other similar optimisation problems.

In order to solve this problem using a Genetic Algorithm (Section 2) and the ACO method (Section 3), a few assumptions were made. Firstly, it was assumed that the data provided will come from a symmetric instance of TSP, i.e. the distance d from city c_i to city c_j $d(c_i, c_j) = d(c_j, c_i)$ $\forall i, j \in [1, n]$. Another assumption that's being made is that the "closed" version of TSP will be solved for this task, i.e. the salesman will end in the city where he started.

In Section 2 of this paper, Genetic Algorithms shall be defined and a method to solve the TSP using such an algorithm will be proposed. Section 3 shall define Ant-Colony Optimisation and propose a method for solving the TSP using this method. Section 4 will then compare the outcomes of the two algorithms and these results shall be discussed in detail. All code used will be listed in the Appendix (Section 8).

2 Genetic Algorithms

Genetic algorithms are designed to simulate a biological process [2], thus most of the terminology refers to the algorithm's biological counterpart. The components that make up GAs are as follows:

- an objective (or fitness) function
- a population of chromosomes
- a selection operator on the chromosomes

- a crossover function which produces a new generation of chromosomes
- a random mutation function

The objective function is the function that the algorithm is trying to optimise. In Genetic Algorithms, this is often referred to as a *fitness* function, this term is in fact taken from evolutionary theory [2]. For the TSP, the fitness function is the sum of the distances between the points, the TSP in fact deals with minimising this sum in order to be able to find the shortest path. Given that the data under study is given in coordinate format [4], the fitness function used for the TSP is given in Equation 1 [2].

$$D = \sum_{k=1}^{n} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \tag{1}$$

A chromosome refers to a value that will be considered as the candidate solution to the optimisation problem. In the case of the TSP, the chromosome could be a permutation of the cities which represent the order that the salesman visits each city. Historically, chromosomes were encoded as a bit string, however it would make more sense not to encode it in this way for the TSP, this however will require a change in the crossover and mutation functions [2].

The selection operator refers to the method used to select which chromosomes are to be chosen for reproduction. In general, a fitter chromosome should be more likely to be selected. In the case of TSP, the probability function must be changed slightly from the original definition since the objective of the TSP is to minimize the fitness function (cost) and not maximize it, thus care must be taken to reverse the probability function.

Once the "fittest" chromosomes are selected, the crossover operator is then used to "combine" them. This operator "resembles the biological crossing over and recombination of chromosomes to create two offspring" [2]. In the case of the TSP, special care needs to be taken when designing this function and the classical way of defining this operator will not work since each city needs to be visited only once [5]. The technique which shall be used in the implementation for the TSP is known as the "cycle" crossover (as taken from [2]) and works as follows:

1. Initially, a random location is chosen in the length of the chromosome.

2. The two parent chromosomes exchange integers at this point to create the offspring.

3. If the integers are the same value then the offspring is the same as the parent and the algorithm terminates.

4. Otherwise, each offspring now has a duplicate integer, so switch the duplicate integer in the first offspring with the integer in the same location in the second offspring.

5. Repeat the above step until there are no duplicates in the first offspring (and thus the second offspring).

6. Both are now valid permutations.

The importance of the mutation step is to reduce the probability of the algorithm converging to a local optimum [5]. This is achieved by causing the algorithm to maintain diversity in the population, however it can cause it to converge more slowly [2]. Again, since the chromosome for the TSP is not encoded as a bit string, the mutation operator changes slightly from that described by Haupt in the original definition of a Genetic Algorithm. The implemented mutation operator randomly chooses two integers in a chromosome from the new generation and swaps them [2]. This will happen with a rather low probability in order to increase the rate of convergence.

3 Ant-Colony Optimisation

The Ant-colony Optimisation algorithm forms part of a class of population-based algorithms known as swarm intelligence (SI) that considers the collective behaviour of the population and individual solutions [6]. A general ACO algorithm has the following form [7]:

```
Set parameters, initialize pheromone trails
SCHEDULE ACTIVITIES
        ConstructAntSolutions
        DaemonActions {optional}
        UpdatePheromones
END_SCHEDULE_ACTIVITIES
```

The parameters initialised for my implementation of the Ant Colony Optimisation Algorithm are as follows:

1. no_of_ants: the number of ants in each iteration.
2. max_iter: the maximum number of iterations.
3. evaporation_rate: the rate at which the pheromones evaporate.
4. alpha: a parameter for calculating the probability of an ant selecting a certain route. Alpha gives more weighting to the pheromone values.
5. beta: a parameter for calculating the probability of an ant selecting a certain route. Beta gives more weighting to the heuristic value.
6. q0: this parameter gives the probability of using the previous ants' experience over selecting a random path.

Since the task at hand is to write an algorithm to solve the symmetric TSP and R is optimised for working with matrix algebra, the distances between each city were calculated as an initialisation step in order to avoid having to compute these values each time on every ant run. The distances between each city were stored in a symmetric matrix thus only the lower triangular matrix needed to be worked out using for loops.

In order to construct the ant solutions, initially, each ant is placed at random on a city. At each step, the ant then chooses the next city based on the probability matrix given in Equation 2.

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha . [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i^k} [\tau_{il}(t)]^\alpha . [\eta_{il}]^\beta} \; if \; j \in \mathcal{N}_i^k \tag{2}$$

Where $p_{ij}^k(t)$ is the probability that ant k goes to city j from city i at iteration t, $\tau_{ij}(t)$ is the pheromone value on edge ij at iteration t, η_{ij} is the heuristic value on edge ij and $\mathcal{N}_i^k$ is the set of all possible nodes reachable by ant k from node i which in our case will be the set of all possible nodes not yet visited by ant k [8].

The ant with the best solution after each iteration will then update the

pheromone values of each edge and is defined as seen in Equation 3 [6].

$$\begin{aligned} \tau_{ij} &= (1-\rho)\tau_{ij} + \rho \sum_{k=1}^{m} \triangle \tau_{ij}^{k}, \\ \triangle \tau_{ij}^{k} &= \frac{1}{L^{k}} \end{aligned} \tag{3}$$

Where m is the number of ants, and L^k is the length of the best tour of that iteration. All pheromone values also evaporate at a rate defined by the parameter after each iteration.

4 Results and Comparisons

5 Conclusion

6 Statement of Completion

7 Plagiarism Declaration Form

8 Appendix

References

[1] uwaterloo. *The Problem.*
http://www.math.uwaterloo.ca/tsp/problem/index.html.
[6th March 2018].

[2] Carr, J. *An Introduction to Genetic Algorithms.*
https://www.whitman.edu/Documents/Academics/Mathematics/2014/carrjk.pdf.
[16th May 2014].

[3] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning.*
[1989].

[4] Unknown. *MP-TESTDATA - The TSPLIB Symmetric Traveling Salesman Problem Instances.*
http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html

[5] Haupt, R.L., & Haupt, S.E. (2004). *Practical Genetic Algorithms* (2nd ed.). Hoboken: Wiley.

[6] Tseng, S.P., et. al. *A fast Ant Colony Optimization for traveling salesman problem.*
https://ieeexplore-ieee-org.ejournals.um.edu.mt/document/5586153/?reload=true.
[23rd July 2010].

[7] Dorgio, M. *Ant colony optimization.*
http://www.scholarpedia.org/article/Ant_colony_optimization.
[2007].

[8] Garg D., Shah S. *ANT COLONY OPTIMIZATION FOR SOLVING TRAVELING DALESMAN PROBLEM. International Journal of Computer Science and System Analysis*, vol. 5, no. 1, pp. 23-29, 2011.
available: http://www.serialsjournals.com/serialjournalmanager/pdf/1330336909.pdf