Genetic Algorithm for Solving the Travel Salesman Problem

By:
Vikaskumar Chaudhary
Niraliben Dipakkumar Mistry
Harshkumar Mehta
Bhargav Dineshbhai Patel
Mohasina Shaikh
Gaurav Pathak

CPSC-5506 Project

Research Report submitted in Partial Fulfilment
of the Requirements of the Master's Course
in Computational Science
(Introduction to Computational Science, CPSC-5506)

Department of Mathematics and Computer Science
Faculty of Sciences, Engineering and Architecture
Laurentian University
Sudbury, Ontario, Canada

Winter 2021

Table of Contents

Abstract
This research report is the result of implementation of travelling salesman problem (TSP) using genetic algorithm (GA) in python carried out by the authors. TSP is one of the most intensively studied problem in optimization. The main attraction of TSP is a salesman visiting all the cities in his tour at the least possible cost. In genetic algorithms crossover and mutation are the preferred technique to solve the optimization problem using survival for the fittest idea. The implementation can solve the travelling salesman problem up to 29 cities in < 2 minutes on a standard testbed with 8 GB of RAM. For our experimental investigation, results have shown that genetic algorithms lead to a good optimization as high as 70 percent even with less population in consideration.

Keywords
Travelling Salesman Problem, Genetic Algorithms, Path Representation, Optimization.

1. Chapter 1: THE TRAVEL SALESMAN PROBLEM

1.1 Introduction
Travelling salesman problem are known classical permutation based combinatorial optimisation problems which has been extensively studied over past few decades. This program requires a colossal amount of system resources to be solved efficiently, as when the size of a problem increases the solution space also increases exponentially. The problem objective is to find the shortest route for the travelling salesman who, starting from his home city must visit every city given on the list precisely once and then return to his home city this problem is mainly focused to find the shortest such trip. The city visit ends back at the starting city, this problem is known as NP-hard as it cannot be solved in polynomial time [1] [2]. The coordinate of the city are known in advance, in order to find the pairwise distance between the cities. The main difficulty is the immense number of possible tours for n cities: (n-1)! /2 [3]

1.2 The Travel salesman Problem
Travelling salesman problem is relatively an old problem. The idea of this problem was introduced as early as 1759 by Euler like TSP where a knight visits each square of chessboard

exactly once in his tour. Although the term '*travelling salesman*' coined in early 1930's in a German book written by a travelling salesman [3].

Over the years TSP has occupied the interest of numerous researchers. The reason for this is the lack of a polynomial time algorithm to resolve the problem Although there are several techniques available to solve the travelling salesman problem [4], [5] but these techniques are not optimized. TSP is applicable on a variety of routing and scheduling problems [6]. Multiple heuristic approaches have been developed to solve TSP as described in [7]. Using genetic algorithm the first researcher to tackle the travelling salesman problem was Brady [8]. The genetic algorithm provided by researchers to solve the travelling salesman problem up to 531 cities have provided very good results but the solution was not optimal [9] [10].

Recently there is increasingly many reasons now to believe that TSP is very hard [11]. There is evidence that there is no polynomial time algorithm for obtaining the exact solution even if the distance is restricted [12] and a solution for guaranteed accuracy [13]. The problem is verify whether the solution is optimal exactly or approximately is also seems to be intractable [11].

1.2.1 Genetic Algorithm

Genetic Algorithm are adaptive search technique based on principal and mechanism of natural selection and the survival of the fittest.GA gained popularity from Holland's study in 1975 [14] of adaption in artificial and natural systems in search problems. In recent years numerous papers have been published on the optimization of NP-hard problems in different application domains such as computer science, biology, telecommunication. GA operate on an iterative fixed size population or pool of candidate solution. The candidate solution represents an encoding of the problem which is like the chromosomes of a biological system.

Each chromosome is associated with the fitness value. It is the ability of chromosome which determine the ability to survive and further produce an offspring. Space search problem are represented as '*individuals*' which are represented by character of strings referred as '*chromosomes'*. Integer and floating point can also be used [15].

Part of space search which is to be examined is called '*population*'. Genetic algorithm working described in (Figure 1). To start, an initial population is chosen along with this the

quality of population is determined and then evaluating everyone with fitness function. In further iterations parents are selected from the population and in turn these parents produce children which are further added back to the population. Offspring are generated through a process called crossover and mutation. It can further be defined as the operations which define the child production process and mutation process are known as crossover operator and mutation operator.

Offspring are generally placed back into the population thus replacing other individuals. Mutation helps algorithm to explore the new stats by avoiding the local optima [3]. Crossover increases the average quality of the population thus by choosing the adequate crossover and mutation operators, the probability that genetic algorithm will produce the nearly optimal solution will increases with respect to the increased number of iterations. GA algorithm relies on three genetic operators: *selection, crossover, mutation.* The selection operation use the fitness value to select the parents of next generation [15].

BEGIN AGA
Make initial population at random.
WHILE NOT stop **DO**
BEGIN
Select parents from the population.
Produce children from the selected parents.
Mutate the individuals.
Extend the population adding the children to it.
Reduce the extend population.
END
Output the best individual found.
END AGA

Figure 1. The pseudo-code of the Abstract Genetic Algorithm (AGA) [3].

We generate the finite set of individuals which we called *'population'*. The size of population set is predetermined before applying the genetic algorithm procedure. An individual characterised by the set of variables is known as '*gene*'. We calculate the fitness of everyone which is commonly done by calculating the sum of Euclidean distance between cities in the

solution. During selection process the initial population get chosen arbitrarily among the possible individuals.

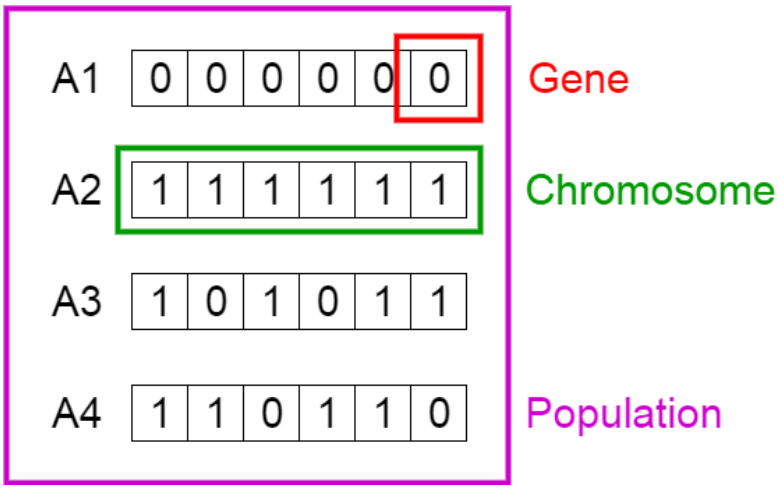

Figure 2. Representation of population Chromosome and Genes

Gene is joined to form a set of string usually known as chromosome depicted in figure 2. In genetic algorithm the fitness is defined by using a fitness function, it determines how fit is an individual to compete with other individuals by assigning a fitness score to everyone. The probability of selecting individual based on fitness score highlights that individual is selected for reproduction. The selection phase usually selects the individual who are fittest so that their genes can be passed to next generation. The classical crossover operation was proposed by Holland in 1975 [14], as shown below where two solutions of 6 cities are available for travelling salesman problem:

(000 001 010 011 100 101) and
(101 100 011 010 001 000) [3]

Randomly among the strings a crossover point is selected from where the string is broken into two separate parts, considering we have chosen the below crossover point highlighted with pipe.

(000 001 010 | 011 100 101) and
(101 100 011 | 010 001 000) [3]

After recombining the parts result in two separate offspring:

(000 001 010 010 001 000) and

(101 100 011 011 100 101) [3]

The mutation operator which was developed by Holland [14] alters one or more bit with the probability equivalent of mutation rate. A tour represented by string 1-2-3-4-5-6:

(000 001 010 011 100 101):

Consider the last and second last bits are selected for mutation, hence these bits will change its value from 0 to 1 and 1 to 0:

(000 001 010 011 100 110):

Bibliography

[1] S. Lin and B. W. Kernighan, 'An Effective Heuristic Algorithm for the Traveling-Salesman Problem', *Oper. Res.*, vol. 21, no. 2, pp. 498–516, Apr. 1973, doi: 10.1287/opre.21.2.498.

[2] N. Biggs, 'THE TRAVELING SALESMAN PROBLEM A Guided Tour of Combinatorial Optimization', *Bull. Lond. Math. Soc.*, vol. 18, no. 5, pp. 514–515, 1986, doi: https://doi.org/10.1112/blms/18.5.514.

[3] P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, 'Genetic Algorithms for the Travelling Salesman Problem: A Review of Representations and Operators', *Artif. Intell. Rev.*, vol. 13, no. 2, pp. 129–170, Apr. 1999, doi: 10.1023/A:1006529012972.

[4] H. Talbi, A. Draa, and M. Batouche, 'A new quantum-inspired genetic algorithm for solving the travelling salesman problem', in *2004 IEEE International Conference on Industrial Technology, 2004. IEEE ICIT '04.*, Dec. 2004, vol. 3, pp. 1192-1197 Vol. 3, doi: 10.1109/ICIT.2004.1490730.

[5] D. Goldberg, 'Genetic Algorithms in Search Optimization and Machine Learning', 1988, doi: 10.5860/choice.27-0936.

[6] M. M. Flood, 'The Traveling-Salesman Problem', *Oper. Res.*, vol. 4, no. 1, pp. 61–75, Feb. 1956, doi: 10.1287/opre.4.1.61.

[7] R. Johnson and M. G. Pilcher, 'The traveling salesman problem, edited by E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B Shmoys, John Wiley & Sons,

Chichester, 1985, 463 pp’, *Networks*, vol. 18, no. 3, pp. 253–254, 1988, doi: https://doi.org/10.1002/net.3230180309.

[8] R. M. Brady, ‘Optimization strategies gleaned from biological evolution’, *Nature*, vol. 317, no. 6040, Art. no. 6040, Oct. 1985, doi: 10.1038/317804a0.

[9] H. Mühlenbein, M. Gorges-Schleuter, and O. Krämer, ‘Evolution algorithms in combinatorial optimization’, *Parallel Comput.*, vol. 7, no. 1, pp. 65–85, Apr. 1988, doi: 10.1016/0167-8191(88)90098-1.

[10] H. Braun, ‘On solving travelling salesman problems by genetic algorithms | SpringerLink’. https://link.springer.com/chapter/10.1007/BFb0029743 (accessed Mar. 27, 2021).

[11] C. H. Papadimitriou, ‘The Euclidean travelling salesman problem is NP-complete’, *Theor. Comput. Sci.*, vol. 4, no. 3, pp. 237–244, Jun. 1977, doi: 10.1016/0304-3975(77)90012-3.

[12] R. M. Karp, ‘Reducibility among Combinatorial Problems’, in *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*, R. E. Miller, J. W. Thatcher, and J. D. Bohlinger, Eds. Boston, MA: Springer US, 1972, pp. 85–103.

[13] S. Sahni and T. Gonzales, ‘P-complete problems and approximate solutions’, in *15th Annual Symposium on Switching and Automata Theory (swat 1974)*, Oct. 1974, pp. 28–32, doi: 10.1109/SWAT.1974.22.

[14] T. M. Press, ‘Adaptation in Natural and Artificial Systems | The MIT Press’. https://mitpress.mit.edu/books/adaptation-natural-and-artificial-systems (accessed Mar. 26, 2021).

[15] K. Katayama, H. Sakamoto, and H. Narihisa, ‘The efficiency of hybrid mutation genetic algorithm for the travelling salesman problem’, *Math. Comput. Model.*, vol. 31, no. 10, pp. 197–203, May 2000, doi: 10.1016/S0895-7177(00)00088-1.