Genetic Algorithm for Solving the Travel Salesman Problem

By:
Vikaskumar Chaudhary
NiralibenDipakkumar 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

List of Figures

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 8GB 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 knownclassicalpermutation 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 approacheshave 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].

BEGINAGA

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.

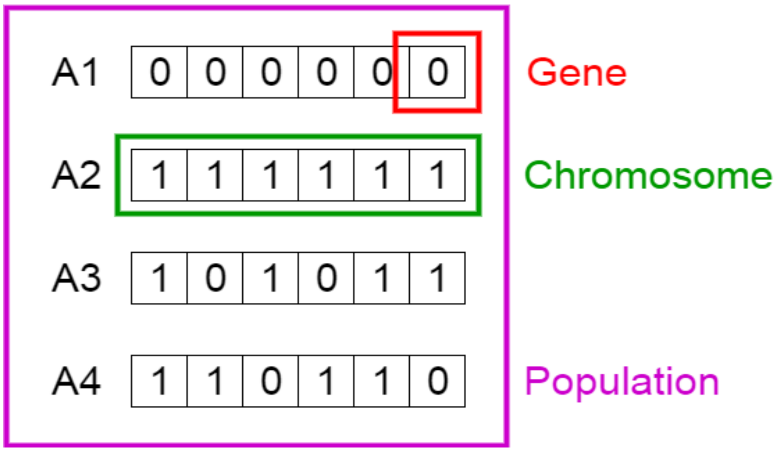

Figure 2. Representation of population Chromosome and Genes

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):

1.3 Extension of the Travel Salesman Problem

We can extend the problem using different approaches or methods to find out best possible solution of the TSP.

1.3.1 Quantum Computing:

In early 80, Richard Feynman's observed that certain quantum mechanical effects cannot be simulated efficiently on a computer. His observation led to speculation that computation in general could be done more efficiently if it used this quantum effects. This speculation proved justified in 1994 when Peter Shor described a polynomia1 time quantum algorithm for factoring numbers.

In quantum systems, the computational space increases exponentially with the size of the system which enables exponential parallelism. This parallelism could lead to exponentially faster quantum algorithms than possible classically [16].

1.3.2 The Proposed Algorithm:

We introduce here a new algorithm inspired from both genetic programming and quantum computing fields to find the shortest Hamiltonian circuit relating N cities. The symmetry of the problem has no special importance. The algorithm deals indifferently with symmetric and asymmetric instances of the TSP.

The algorithm has as input data the distances between each pair of cities. These distances are arranged within a square matrix D of NxN element. The element D[i, j] denotes s the distance between the city labelled i and the one labelled j. The figure below gives the distances matrix of a TSP instance (“gr24” [16]).

1	257	187	91	150	80	130	134	243	185	214	70	272	219	293	54	211	290	268	261	175	250	192	121
257	0	196	228	112	196	167	154	209	86	223	191	180	83	50	219	74	139	53	43	128	99	228	142
187	196	0	158	96	88	59	63	286	124	49	121	315	172	232	92	81	98	138	200	76	89	235	99
91	228	158	0	120	77	101	105	159	156	185	27	188	149	264	82	182	261	239	232	146	221	108	84
150	112	96	120	0	63	56	34	190	40	123	83	193	79	148	119	105	144	123	98	32	105	119	35
80	196	88	77	63	0	25	29	216	124	115	47	245	139	232	31	150	176	207	200	76	189	165	29
130	167	59	101	56	25	0	22	229	95	86	64	258	134	203	43	121	164	178	171	47	160	178	42
134	154	63	105	34	29	22	0	225	82	90	68	228	112	190	58	108	136	165	131	30	147	154	36
243	209	286	159	190	216	229	225	0	207	313	173	29	126	248	238	310	389	367	166	222	349	71	220
185	86	124	156	40	124	95	82	207	0	151	119	159	62	122	147	37	116	86	90	56	76	136	70
214	223	49	185	123	115	86	90	313	151	0	148	342	199	259	84	160	147	187	227	103	138	262	126
70	191	121	27	83	47	64	68	173	119	148	0	209	153	227	53	145	224	202	195	109	184	110	55
272	180	315	188	193	245	258	228	29	159	342	209	0	97	219	267	196	275	227	137	225	235	74	249
219	83	172	149	79	139	134	112	126	62	199	153	97	0	134	170	99	178	130	69	104	138	96	104
293	50	232	264	148	232	203	190	248	122	259	227	219	134	0	255	125	154	68	82	164	114	264	178
54	219	92	82	119	31	43	58	238	147	84	53	267	170	255	0	173	190	230	223	99	212	187	60
211	74	81	182	105	150	121	108	310	37	160	145	196	99	125	173	0	79	57	90	57	39	182	96
290	139	98	261	144	176	164	136	389	116	147	224	275	178	154	190	79	0	86	176	112	40	261	175
268	53	138	239	123	207	178	165	367	86	187	202	227	130	68	230	57	86	0	90	114	46	239	153
261	43	200	232	98	200	171	131	166	90	227	195	137	69	82	223	90	176	90	0	134	136	165	146
175	128	76	146	32	76	47	30	222	56	103	109	225	104	164	99	57	112	114	134	0	96	151	47
250	99	89	221	105	189	160	147	349	76	138	184	235	138	114	212	39	40	46	136	96	0	221	135
192	228	235	108	119	165	178	154	71	136	262	110	74	96	264	187	182	261	239	165	151	221	0	169
121	142	99	84	35	29	42	36	220	70	126	55	249	104	178	60	96	175	153	146	47	135	169	0

A solution for a TSP dealing with N cities is a circuit which relates in a suitable order these cities. So we can represent the solution by an NxN matrix “A” associating to every city its range in the circuit, i.e. if A (i, j) = 1, j is the i^{th} visited city. Elsewhere if d (i, j) = 0. The figure below gives a representation of the best solution for the problem above.

Starting with the initial population, we apply cyclically a set of4 operations followed by a measurement (figure 3)

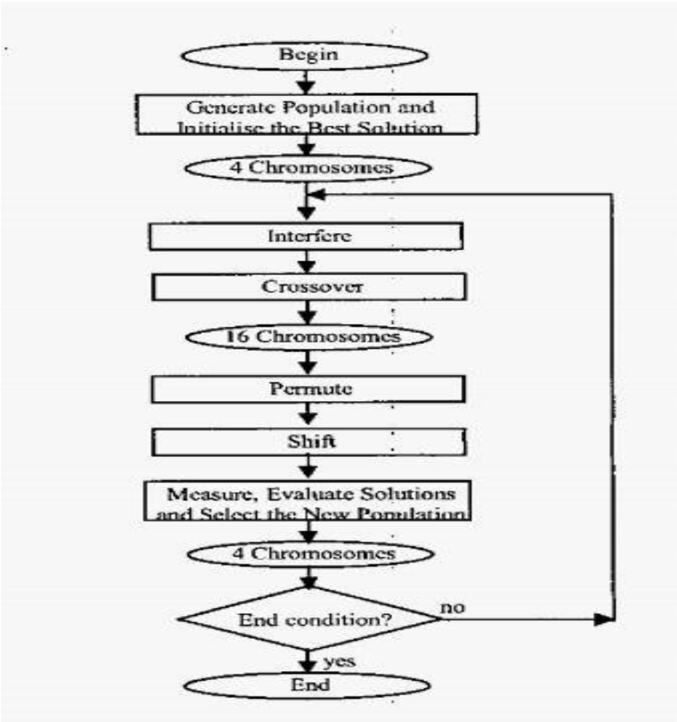

Figure 3: The proposed algorithm

There are some other approaches to find out efficient solution for the travel salesman problem. The approaches are mentioned below:

I) An analogue approach to the travelling salesman problem using an elastic net method

II) The co-adaptive neural network approach to the Euclidean Travelling Salesman Problem

III) The travel salesman problem using the Brute-Force approach (Naive approach) which uses nearest neighbour method

1.4 Conclusion for the Extension of the Travel Salesman Problem

We have suggested a new algorithm inspired from both genetic algorithms and quantum computing to solve the travelling salesman problem as a representative of combinatorial optimisation problems class. Our algorithm provides a great diversity by using quantum coding of solutions, i.e. all the solutions exist within each chromosome and what change are the probabilities to have one of them as a result of a measurement. Therefore, the size of the population does not need to be great. So, we have chosen to have only 4 chromosomes at the origin of each generation. Another advantage is that the interference provides in some way a guide for the population individuals and reinforces therefore the algorithm convergence. This has allowed obtaining good solutions after a small number of iterations. Introducing permutation and shifting operations has improved the algorithm's performance by permitting it to avoid been blocked in local minima.

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.

[16] http://www.informatik.uni-heidclberg.dc/groups/omopt‘ softwarci TSPLIB95/STSP.html