ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

A Combined Genetic Algorithm for Symmetric Travelling Salesman Problem

Baiduryya Sarkar
Department of BCA
The Heritage Academy
Kolkata, India

Avik Mitra
Department of BCA
The Heritage Academy
Kolkata, India

Somnath Bhattacharyya
Department of BCA
The Heritage Academy
Kolkata, India

Abstract: There has been significant work on solving Travelling Salesman Problem and its variants using heuristic approach as the algorithms for finding exact solutions are computationally hard. Among the heuristics, genetic algorithms have shown promising result in terms simplicity in implementation and computational complexity. In this paper, we propose Combined Genetic Algorithm that uses partially mapped crossover and exchange mutation to progressively eliminate the weaker solution. Computational performance in our setting has shown quadratic time complexity.

Keywords: Travelling Salesman Problem; NP-Hard; Optimization; Heuristic; Genetic Algorithm

I. INTRODUCTION

Travelling Salesman Problem (TSP) is a NP-hard optimization problem [1]. In TSP, minimum cost to tour of n cities is to be found where the costs to travel between each pair of cities are known; the problem has a constraint that each city has to be visited exactly once. It is most intensively studied [2] optimization problem and has wide range of applications like, tour planning [3], wiring problem in computer [4], vehicle routing problem [5] (extension of TSP), etc. TSP has many variants [6]: when the cost to travel from city i to city j, *c(i, j)* is same as cost to travel from city j to city i, c(j, i), i.e., c(i,j) = c(j,i), then the TSP is called *symmetric TSP (sTSP)*; if c(i,j)≠ c(j, i), then the TSP is called *asymmetric TSP (aTSP)*; if there are multiple salesmen satisfying that each city is visited exactly once, then the TSP becomes *multiple TSP (mTSP)*. If the costs between pair of cities represent Euclidean distances between the cities then the *Euclidean TSP* is an NP-Complete problem [7]. Many solution approaches has been proposed in literature and these approaches can be broadly classified into (Fig. 1) two categories: *Approaches for Exact solutions* [8]-[11] and *Heuristic approaches* [12]-[17]. The approaches for finding *exact solution* tries to find the optimum route; since the problem is NP-hard and hence the time complexity of these approaches are nearer to O((n-1)!). For example, the time complexities for [9] and [10] are proportional to $n^2 2^n$, which though much lesser than (n-1)!, it is significantly high for moderate values of n. The *heuristic approaches* find approximate solution to TSP, i.e., it finds tour or tours whose cost is close to but may be different than the optimum cost. All the heuristic approaches initially assume tour(s), called approximate solutions, and then improve these approximate solutions to get better solution(s); the steps continue until there is no improvement in the consecutive approximate solution(s) or the number of steps exceeds a predetermined value. Among the heuristic approaches, Genetic Algorithm [18] (*GA*) approach has been very effective in solving TSP because of its simplicity and it is reported to perform better than the other heuristic techniques under linear constraints [19]-[22]. For solution using GA, initially a set of tours is assumed as solutions, these solutions are termed as *chromosomes*. These chromosomes are then tested for fitness using a *fitness function*, and then the 'fit' chromosomes are recombined (called *crossover*), to get next set of chromosomes, that is, new chromosomes are created for next generation. Often each of these chromosomes may undergo self transformation called *mutation* with P_m as *probability of mutation*. The formation of generations of chromosomes representing solutions is continued until it is found that fitness of newly formed generation cannot be improved further. The various approaches for TSP using GA can be distinguished by the nature of chromosomes, crossover methodology and mutation probability [23]. In this paper, we propose Combined GA algorithm to solve sTSP, in which we provide new technique to produce more fit solutions and rejecting the unfit ones, thus reducing the number of steps hence reducing the cost of implementation. Rest of the paper is organized as follows: section II explains the Combined GA; section III discusses the results obtained after implementing the proposed algorithm; followed by conclusion and references in section IV and V respectively.

II. COMBINED GA FOR TSP

In this section, we describe the proposed Combined GA for solving sTSP, which combines *partially mapped crossover* [24] and *exchange mutation* [25]. In subsequent sub-sections we define the components of the algorithm: (1) *Solution representation* in chromosomes; (2) *Initialization*, i.e., generating chromosomes representing probable solutions to the TSP problem; (3) *Selection* using fitness function; (4) *Crossover*; (5) *Mutation* and *Updating*.

A. *Representation and Initialization*

We represent the chromosomes as tours of the cities where each city appears exactly once. Therefore, length of each chromosome is equal to the number of cities in the TSP. For example, if there are 5 cities number 1 through 5, a

chromosome could be (5, 4, 3, 1, 2). We now generate tours (chromosomes) for TSP. The generated chromosomes form the first generation of population. The population is improved by selection using fitness function, crossover and mutation.

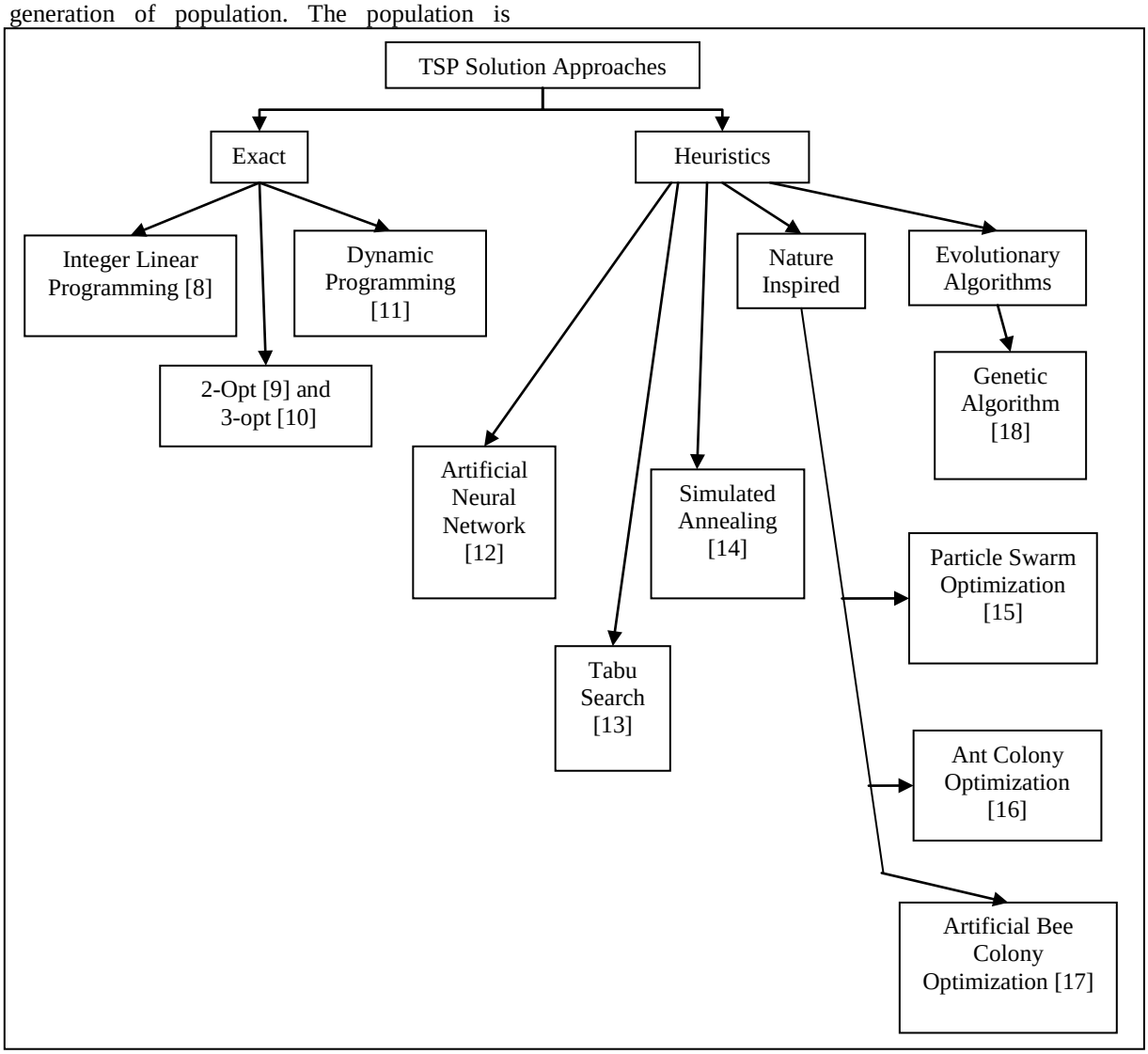

Figure 1. Solution Approaches for Travelling Salesman Problem (TSP)

B. Selection using Fitness Function

The fitness function first finds the path lengths of chromosomes of the present generation. The *path length of a chromosome* is the sum of costs between two consecutive cities along a chromosome. Then four chromosomes among the population is found: chromosome having minimum path length (say *minLengthC*), second minimum path length (say *secondMinLengthC*), maximum path length (say *maxLengthC*) and, second maximum path length (say *secondMaxLengthC*). In case of a tie, one of the chromosomes is chosen arbitrary. The chromosomes minLengthC and secondMinLengthC are now used for crossover.

C. Crossover

The crossover technique used is the *partially mapped crossover* (PMX). Given two selected (section II.B) chromosomes, two crossing points (say *position-1* and *position-2*, where position-2 > position-1) which are set to be same for both the chromosomes, are chosen. The parts within position-1 and position-2 are first swapped between the chromosomes creating two children chromosomes. Each child has one crossover zone and two non-crossover zones. Now these new chromosomes may not represent a feasible tour, i.e., it may contain repeated cities. Therefore, each of these two children are checked for duplicate cities and modified if needed, in two <non-crossover zone, crossover zone> zone-pairs of a chromosome: if there is duplicate city, one in crossed over region and another in non –crossover region, then the duplicate city in the non-crossover zone is replaced by a city from the other child such that that city is unique to the former child. Therefore for the two children, there are four checking for duplicate cities. The newly formed feasible children may now undergo mutation.

D. Mutation and Updating

After PMX (section II.C), one of the children chromosomes is selected at random for mutation. Then, two random positions within the children chromosome are chosen and the cities from those positions are swapped, thus causing mutation. The mutated and non-mutated children chromosomes now replace the maxLengthC and secondmaxlengthC chromosomes (section II.B). This replacement updates the population that contains chromosomes with lesser path lengths than the previous population.

E. Procedure for Combined GA

The procedure uses the operations explained in section II.A to section II.D. The costs between *n* cities are taken in adjacency matrix. Initially, population of N (≤n!) unique chromosomes is generated, where length of a chromosome is *n*. At each iteration: population is tested for fitness and two chromosomes from the population are selected (section II.B), followed by crossover (section II.C) and mutation (section II.D) operations on these two chromosomes and the resulting chromosomes replace the chromosomes that have highest and second highest path lengths. This step eliminates the 'weaker' chromosomes and creates an improved population for next iteration (or generation). Iterations are carried out for pre-determined value of number of generations.

III. RESULTS AND DISCUSSIONS

We have implemented our algorithm using C programming language and run in Dev-C++ IDE supporting GCC 4.9.32 (32-bit release). The IDE is installed on Windows 8 Pro with Intel Pentium G2020 running at 2.90 GHz and 2GB RAM. The number of cities is varied from 10 to 200 with the interval of 10. The population size (N) and the pre-determined value of number of generations are both set to 100. For each value of number of cities, the programme is run 1000 times and the average of these 1000 CPU times is taken. The time required to run the algorithm is measured using clock() function and the CLOCKS_PER_SEC macro. The obtained CPU time is given in Table 1. It is to be noted that the programme, which becomes a process, competes and cooperates with other processes of the operating system and we have not used any specialized system for our run, nor have we considered the process scheduling algorithm of the operating system. The variation of average CPU time with number of cities is shown in Fig. 2. From the figure and its regression line, we can say that our algorithm has quadratic time complexity which is an agreement with [5] and significant improvement over [26] where Integer Linear Programming is used to solve the sTSP.

Table I. CPU Times for the Implementation

Number of Cities	*Average CPU Time*
10	0.000718
20	0.001422
30	0.002314
40	0.003437
50	0.004349
60	0.005985
70	0.007079
80	0.009125
90	0.010531
100	0.012593
110	0.014656
120	0.017470
130	0.020001
140	0.022345
150	0.025220
160	0.029155
170	0.031439
180	0.034893
190	0.038329
200	0.042409

IV. CONCLUSION

We have proposed a Combined Genetic Algorithm for solution of symmetric Travelling Salesman Problem, where we have used a multi-point crossover scheme- partially mapped crossover and exchange mutation. The algorithm progressively removes the less fit chromosomes, i.e., the tours with high path lengths are removed. The performance of the algorithm is quadratic proving superiority of heuristic schemes over the exact algorithms.

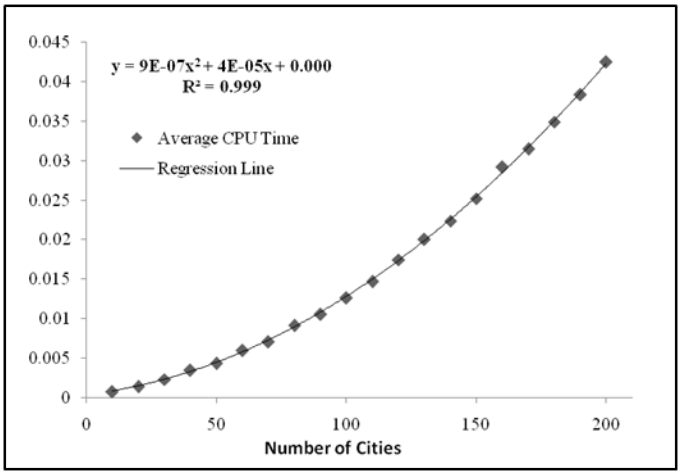

Figure 2. CPU time variation with number of cities for Combined GA

V. REFERENCES

[1] G.J. Woeginger, "Exact algorithms for NP-hard problems- A survey", Combinatorial Optimization – Eureka, You Shrink!, pp 185-207, Springer, 2003.

[2] G. Jati, "Evolutionary firefly algorithm for travelling salesman problem", Adaptive and Intelligent Systems, pp 393-403, 2011.

[3] A. Piwonska and J. Koszelew, "A memetic algorithm for a tour planning in the selective travelling salesman problem on a road network", ISMIS 2011.

[4] J.K. Lenstra and A.H.G. Rinnoy Kan, "Some simple applications of the travelling salesman problem", Operational Research Quarterly, vol. 26, issue no. 4, pp 717-733, 1975.

[5] G. Laporte, "The vehicle routing problem: an overview of exact and approximate algorithms", European Journal of Operational Research, vol. 59, issue no. 3, pp 345-358, 1992.

[6] G. Gutin and A.P. Punnen, "The travelling salesman problem and its variations", Springer Science and Business Media, 2007.

[7] C.H. Papadimitriou, "The Euclidean travelling salesman problem is NP-Complete", Theoretical Computer Science, vol. 4, issue no. 3, pp 237-244, June 1977.

[8] C.E. Miller, A.W. Tucker, and Richard A. Zamelin, "Integer programming formulation of travelling salesman problems", Journal of the ACM, vol. 7, issue no. 4, pp 326-329, 1960.

[9] G.A. Croes, "A method for solving travelling salesman problem", Operations Research, vol. 6, issue no. 6, pp 791-812, December, 1958.

[10] S. Lin, "Computer solutions of the travelling salesman problem", The Bell System Technical Journal, vol. 44, issue no. 10, pp 2245-2269, 1965.

[11] M. Held and R.M. Karp, "A dynamic Programming approach to sequencing problems", Journal of the Society

for Industrial and Applied Mathematics, vol 10, issue no. 1, pp 196-210, March 1962.

[12] J.Y. Potvin, "State-of-the-art Survey—the travelling salesman problem: a neural network perspective", OPRSA Journal of Computing, vol. 5, issue no. 4, pp 328-348, 1993.

[13] M. Gendreau, L. Gilbert, and S. Frederic, "A tabu search heuristic for the undirected selective travelling salesman problem", European Journal of Operational Research, vol. 106, issue no. 2, pp 539-545, 1998.

[14] J.R.A. Allwright, "A distributed implementation of simulated annealing for the travelling salesman problem", Parallel Computing, vol. 10, issue no. 3, pp 335-338, May 1989.

[15] K.P. Wang, L. Wuang, C.G. Zhou, and W. Pang, "Particle swarm optimization for travelling salesman problem", International Conference on Machine Learning and Cybernetics, 2003.

[16] M. Dorigo and L.M. Gambardella, "Ant colonies for travelling salesman problem", Biosystems, vol. 43, issue no. 2, pp 73-81, July 1997.

[17] D. Karaboga and B. Gorkemli, "A combinatorial artificial bee colony algorithm for travelling salesman problem", IEEE INISTA 2011.

[18] H. Braun, "On solving travelling salesman problems by genetic algorithms", International Conference on Parallel Problem Solving from Nature, 1990.

[19] R. Hassan, B. Cohanim, O. Week, and G. Venter, "A comparison of particle swarm optimization and genetic algorithm", 6th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference. 2005.

[20] J.N.D. Gupta, R.S. Sexton, "Comparing back-propagation with a genetic algorithm for neural network Training", Omega, vol. 26, issue no. 7, pp 679-684, 1999.

[21] R.S. Sexton, R. E. Dorsey and J.D. Johnson, "Optimization of neural networks: A comparative analysis of the genetic algorithm and simulated annealing", European Journal of Operational Research, vol. 114, issue no. 3, pp 589-601, 1999.

[22] M.A. Arostegui, S.N. Kadipasaoglu, and B.M. Khumawala, "An empirical comparison of tabu search, simulated annealing and genetic algorithms for facilities location problems", International Journal of Production Economics, vol. 103, issue no. 2, pp 742-754, 2006.

[23] P. Larranaga, 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", Artificial Intelligence Review, vol. 13, issue no. 2, pp 129-170, 1999.

[24] D.E. Goldberg, R. Lingle, "Alleles, loci, and the travelling salesman problem", Proceedings of an International Conference on Genetic Algorithms and Their Applications, vol. 154, pp 154-159, July 1985.

[25] W. Banzhaf, "The "molecular" travelling salesman", Biological Cybernetics, vol. 64, issue no. 1, pp 7-14, November 1990.

[26] M. Grotschel, "On the symmetric travelling salesman problem: solution of a 120-city problem", Combinatorial Optimization, pp 61-77, 1980.