Network and Complex Systems
ISSN 2224-610X (Paper) ISSN 2225-0603 (Online)
Vol.6, No.3, 2016

Genetic Algorithm to Traveling Salesman Problem

Afshan butt[1], Vanita Ben Dhagat[2] and Dr.Bhanu pratap tripathi[3]

1 Bhilai Institute of Technology ,Kendari, Raipur

2 Jai Narayan College of Technology, Bhopal

3 Govt. N.P.G. Science College, Raipur

Abstract:

In this paper, we apply a genetic algorithm to TSP. Since in TSP, a tour must pass through edges in E' (E) at least once, it is necessary to involve E' and the information of direction in the chromosome. However, if we use the existing chromosome structure, the length of the chromosome becomes 2 jE0j and the size of the solution space becomes 2jE0j jE0j!. In the previous study, since the chromosome uses two kinds of information (E' and the direction), the results and the time to find a near-optimal solution vary according to the method of applying genetic operators. To resolve these defects, this paper proposes a new structure of chromosome for TSP.

1. Introduction

For an undirected graph G=(V,E), the Rural Postman Problem (RPP) is a problem that finds a minimum cost tour that must pass through edges in E'(E) at least once. RPP, like Traveling Salesman Problem (TSP), is known as an NP-Complete problem. There are many local optima in NP problems. Hence, it is necessary to search the whole solution space to find a global optimum. However, it is difficult both to search the whole solution space and to find a global optimum. Therefore, we usually find a near-optimal solution using some algorithm, such as Simulated Annealing (SA)[Greedy Heuristic ,Neural Network, and genetic algorithms (GA)

In this paper, each chromosome is represented by a string of nodes within a Hamiltonian graph. Hence, it is necessary to transform a RPP into a Hamiltonian graph before chromosomes are constructed.

In order to transform a graph of TSP into a Hamiltonian graph, we transform edges in the E' (E) into nodes (V H) and connect these nodes completely. Each chromosome is obtained from

a string of nodes in V H. While transforming, the paths in the RPP between the nodes included in V H are inserted in some tables and these tables are used in decoding that is a procedure for obtaining a final routing. In simulation, we compare the performance of the proposed genetic algorithm with an SA and a Greedy Heuristic algorithm.

In a genetic algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

Genetic algorithms find application in bioinformatics, phylogenetics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields.

A typical genetic algorithm requires:

1. a genetic representation of the solution domain,
2. a fitness function to evaluate the solution domain.

A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary

programming; a mix of both linear chromosomes and trees is explored in gene expression programming.

The fitness function is defined over the genetic representation and measures the *quality* of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The *fitness* of the solution is the sum of values of all objects in the knapsack if the representation is valid, or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used.

Once the genetic representation and the fitness function are defined, a GA proceeds to initialize a population of solutions (usually randomly) and then to improve it through repetitive application of the mutation, crossover, inversion and selection operators.

2. Initialization

Initially many individual solutions are (usually) randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, allowing the entire range of possible solutions (the *search space*). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

3. Hamiltonian Graph and TSP

4. Hamiltonian Graph

Hamiltonian graphs are graphs that contain spanning cycles. The problem of determining whether a graph has a Hamiltonian cycle is called the Hamiltonian cycle problem which is known to be NP Complete [1, 10].

In general, it is difficult to determine if a graph has a Hamiltonian cycle, but there are necessary and sufficient conditions for a graph to be a Hamiltonian graph [10].

Path and Cycle Length Condition Let G=(V,E) be a graph of order jV j 3, and suppose that for every pair of nonadjacent vertices u and v in G

$$\deg(u) + \deg(v) \quad jV\ j:$$

Closure Condition for Hamiltonian A graph G=(V,E) is Hamiltonian if and only if the closure of G is Hamiltonian. From the above conditions, all the completely connected graphs are Hamiltonian. In this paper, according to the above conditions, we obtain the chromosome after transforming a RPP into a completely connected graph which is a Hamiltonian graph and apply the transformed chromosome to genetic algorithm.

4. Rural Postman Problem

Figure 1 shows a traveling path of RPP. In Figure 1,a-a', b-b', c-c', d-d', and e-e' are the edges in E'(E) that must be passed at least once in the path. a'b, b'c, c'd, and d'e are the intermediate paths from which shortest traveling path should be determined .

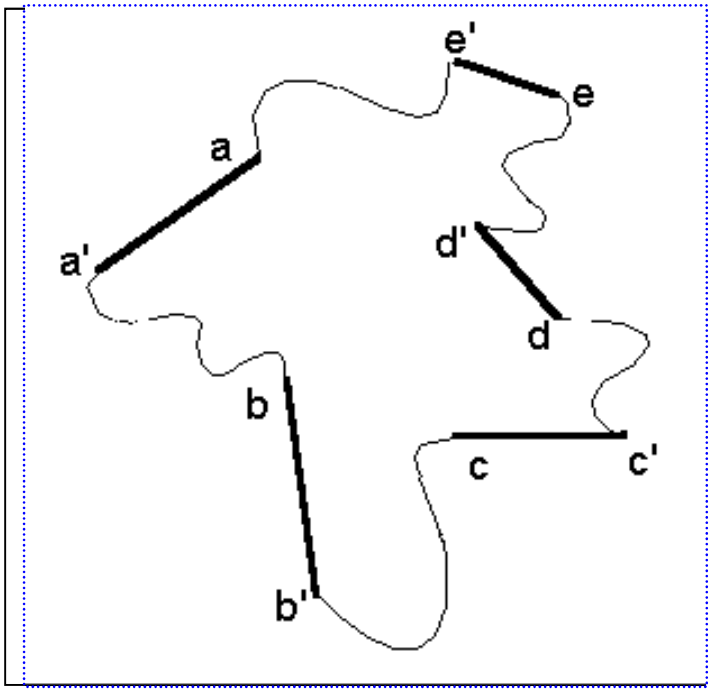

Figure 1: The traveling path

5. Genetic Algorithm for TSP

5.1 The Existing Structure of Chromosome The existing structure of chromosome (ECS) consists of two kinds of strings. One is for describing the visiting order of the edges in E' and

the other, represented by a set of binary codes, is for describing the decoding information. For example, assume that E' = f1; 2; 3; 4; 5g, where 1, 2, 3, 4 and 5 denote edges (a, a'), (b, b'), (c, c'), (d, d'), and (e, e'), respectively, and 0 and 1 denote directions of the edges. If the decoding information of an element is 1, the direction of the tour is reverse. For example, assume that the following p, q describe the structure of a chromosome.

p	1	2	3	4	5
q	1	0	1	0	1

Table 1

The 1(q(1)) of edge 1 (p(1)) means that in the tour we travel from a to a', and the 0 (q(2)) of edge 3 (p(2)) denotes a path from node c' to c, because the decoding information is 1.

6. Genetic Operators

6.1 Crossover

Crossover is an operator that exchanges some strings in two selected chromosomes appropriately and a pair of new chromosomes is produced. The order of strings is important in our problem. Hence, we use Partialy Matched Exchange (PMX) proposed by Goldberg and Lingle and the PMX method is presented well

6.2 Mutation

In this paper, two mutation methods are applied. The first is that two selected points marked by '|' are swapped (mutation1). The second is that the substring between two cut points marked by 'j' along the length of the chromosome is inversed (mutation2). In the following examples, o means a chromosome generated from p.

mutation1. Reciprocal exchange

p = (9876543210)

o = (9876321450)

mutation2. Inversion

p = (987 j 6543 j 210) o = (987 j 6321 j 450):

7. Experimental Results

The GA was programmed in MATLAB version 7.5 and tested on an IBM PC Pentium Pro with 8 randomly generated problems. Table 4 describes the problems applied to GA. In GA, the size of the population is 100, and we evolve the population for 1000 generations. The selection scheme in this paper is the roulette wheel method according to fitness function. Crossover (PMX) rate is 0.6, the mutation1 Reciprocal Exchange) rate is 0.03, and the mutation2 (Inversion) rate is 0.04.Table 5 describes the comparison of GA with the existing chromosome structure (ECS) and GA with the proposed chromosome structure (PCS). For most test problems, PCS produces better near-optimal solution and shows fast convergence.

Figure 2 Figure 4 illustrate the convergence of GAs according to scaling factors for problems 1, 2, 3 and 6. In these figures, we can see that PCS is less affected by scaling factors than ECS. Table 6 shows results of comparison of PCS with an SA and a Greedy Heuristic algorithm. We implement the SA and the Greedy Heuristic algorithm described in [7] and apply them to TSP. The Greedy Heuristic algorithm produced the worst results among the 3 algorithms because it finds only a local-minima in 4 cases. On the other hand, the SA, which can find a global-minimum, produced the best results. In GA, the test problems except 4 and 7 produced the same results as the SA, because both Problems 4 and 7 have premature convergences

Problem	$\|V\|$	$\|E\|$	$\|E'\|$
1	10	18	10
2	15	23	7
3	17	36	10
4	40	82	19
5	50	63	17
6	30	49	20
7	79	128	20
8	45	76	12

Table2: Test problems

8. Conclusions

This paper introduced a genetic algorithm and proposed the structure of chromosome for RPP. We organized the chromosomes using the nodes of Hamiltonian graph that RPP is transformed into. Hence, the length of the proposed chromosome was shorter than that of the existing chromosome structure and the size of solution space could be reduced. Table

Problem	ECS		PCS		[4]
	cost	generation	cost	generation	
1	22	21	22	5	
2	955.34	76	955.34	6	
3	31	13	31	5	[5]
4	1198.59	768	1176.67	692	
5	920	140	920	242	
6	36	96	36	12	
7	912.03	12	909.21	780	[6]
8	2024.26	368	2024.26	26	

. Table 3: ECS and PCS

Problem	Greedy	PCS	SA
1	22	22	22
2	963.55	955.34	955.34
3	32	31	31
4	1194.02	1176.67	1137.50
5	940	920	920
6	36	36	36
7	911.30	909.21	908.17
8	2024.26	2024.26	2024.26

Table 4: Comparison PCS with SA and Greedy Heuristic

Table 3 show that the GA using the proposed chromosome structure is less affected by scaling factors than the GA using the existing chromosome structure. In this paper, we compared GA with an SA and a Greedy Heuristic algorithm. Table 4 shows that the GA produces effective results in obtaining near-optimal solutions. In the future, we can apply the proposed GA to directed graphs and real world problems such as network design, garbage collection and other routing problems.

REFERENCES

1. Bondy, J. A. and Murty, U. S. R., Graph Theory with Applications, Macmillan Press Ltd., 1977.

2. Fausett, L., Fundamentals of Neural Networks Architectures, Algorithms, and Applications, Prentice Hall, 1994.

3. Freeman, J. A. and Skapura, D. M., Neural Net-works Algorithms, Applications, and Programming Techniques, Addison-Wesley, 1979.

4. Garey, M. R. and Johnson, D. S., Computers and Intractability - A Guide to the Theory of NP-Completeness, p. 213, FREEMAN, 1979.

5. Goldberg, D. E., Genetic Algorithm in Search,Optimization " Machine Learning, Addison Wesley, 1989.

6. Goldberg, D. E. and Lingle, R., Alleles, loci,and the Traveling Salesman Problem., Proceedings of an International Conference on Genetic Algorithms and Their Applications, pp. 154-149,1985.

7. Greening, D. R., Simulated Annealing with Errors, PhD. paper, University of California, 1995.

8. Haykin, S., Neural Networks - A Comprehen sive Foundation, Macmillan Publishing Company, 1994.

9. Ladd, S. R., Genetic Algorithms in C++, M&T Books, 1995.

10. Lenstra, J. K. and Rinnooy Kan, A. H. G., On General Routing Problems., Networks, Vol. 6, pp.273-280, 1976.

11. Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs, Second Edition, Springer-Verlag, 1994.

12. Srinivas, M. and Patnaik, L. M., Genetic Algorithms: A Survey, Computer, pp. 17-26, June 1994.

13. Nering, E.D and Tucker, A.W., 1993, *Linear Programming and Related Problems*, Academic Press, Boston, MA.

14. Ahuja, Ravindra K.; Magnanti, Thomas L.; and Orlin, James B. (1993). *Network Flows: Theory, Algorithms and Applications*. Prentice Hall.
15. Bertsekas, Dimitri P. (1999). *Nonlinear Programming* (2nd ed.). Athena Scientific.
16. Bonnans, J. Frédéric; Gilbert, J. Charles; Lemaréchal, Claude; Sagastizábal, Claudia A. (2006). *Numerical optimization: Theoretical and practical aspects*. Universitext (Second revised ed. of translation of 1997 French ed.). Berlin: Springer-Verlag. pp. xiv+490.
17. Cook, William J.; Cunningham, William H.; Pulleyblank, William R.; Schrijver, Alexander (November 12, 1997). *Combinatorial Optimization* (1st ed.). John Wiley & Sons.
18. Lawler, Eugene (2001). "4.5. Combinatorial Implications of Max-Flow Min-Cut Theorem, 4.6. Linear Programming Interpretation of Max-Flow Min-Cut Theorem". *Combinatorial Optimization: Networks and Matroids*. Dover. pp. 117–120. ISBN 0486414531.
19. Ghare, P.M., Schrader, G.F.: A model for exponentially decaying inventories, J. Ind. Eng., Vol.**14**, pp.238-243, 1963.
20. Orden, A.: The transshipment problem. Manage. Sci. 2, 276–285 (1956)
21. Zhao, H., Despande, V., Rayan, J.K.: Emergency transshipment in decentralized dealer network.
22. Herer, Y.T., Tzur, M.: The dynamic transshipment problem. Nav Res Logist Q 48, 386–408 (2001)