A Comparative Study Between Vector and Matrix Representations of Chromosomes in TSP

María A. Osorio, Rosaliano Pérez, Flaviano Pérez
School of Computer Sciences,
Autonomous Univesity of Puebla,
Ciudad Universitaria, San Manuel. Puebla, Pue. 72550
México.

Abstract: The data structure utilized to represent the chromosomes and its associated operators has a strong impact in the genetic algorithms performance. The present research has the objective to show the performance differences between the vector and the matrix representation of chromosomes, its operators associated and the parameters value, in the solution of asymmetric travelling salesman problems with genetic algorithms.

Keywords: Travelling Salesman Problem (TSP), Genetic Algorithms, Evolutive Programs, Chromosomes Representation

1. Introduction

The TSP has become a classic problem because it serves to represent a great number of applications in real life, as the colouring sequence in textile industry, the design of insulating material and optic filters, the planning of trajectories in robotics and many other examples that can be represented using sequences [6]. Besides, it may represent a big number of combinatorial problems that cannot be solved in polynomial time and are NP hard.

The exponential nature of the time needed to solve exactly this problem has originated, during the last decades, the development of heuristic algorithms to approximate its optimal solution [6].

The TSP was the focus of attention of several researches that utilized genetic algorithms for its solution. Recently, the research has focused on the best chromosomes structures and the most suitable operators to improve the numerical solutions and to reduce solution times (see Fox y McMahon [4], Grefenstette [8], Jog [10], and Schaffer [14]).

Genetic algorithms are defined as searching techniques that use natural selection and genetic concepts to approximate the global optimum in a searching space. Our objective was to obtain comparative results about the performance of matrix and vector chromosome structures and different genetic operators for TSP.

To relate the experience obtained, we present history, importance and model representation of TSP in section 2. In section 3, we describe the vector chromosome structure for TSP and the operators associated. Section 4 describes different matrix chromosome structures for TSP and their operators. Finally, section 5 contains tables with different parameter values, comparative results and conclusions.

2. Travelling Salesman Problem

The TSP can be formulated saying that the travelling salesman must visit every city in his territory exactly once and then return to the starting point. Given the cost of travel between all cities, he should plan his itinerary for a minimum total cost of the entire tour. It may be formulated as a maximisation problem, if he could obtain a different profit according to its itinerary, and the problem will plan his itinerary for a maximum profit.

Space solution for TSP is the *n*-cities permutation, *n*!. Any simple permutation is a different solution. The optimum is the permutation that correspond to a travel with the minimum cost or the maximum profit. The evaluation function is the sum of the cost or the profit associated with each segment included in the itinerary.

The TSP was already documented in 1759 by Euler. The term 'travelling salesman' was first used in 1932 in a German book written by a veteran travelling salesman. The TSP, in the way we know it now, was introduced by the RAND Corporation in 1948. The TSP also became popular at that time due to the apparition of linear programming and the attempts to solve combinatorial problems.

In 1979, TSP was probed to be NP-hard, a special kind of NP-complete problems (see Garey [5]). If one can find a solution in polynomial time to one of them, it may find it for all NP and then, P=NP. But nobody has been able to find efficient algorithms for NP-complete problems until now.

The TSP can be symmetric or asymmetric. In the symmetric case, departure and return costs are the same and can be represented with a no directed graph. For the asymmetric case, the more common one, the departure and return costs are different and can only be represented by a directed graph. The symmetric problem can be seen as a special case of the asymmetric one. This research was directed to the asymmetric case and all references to TSP correspond to the asymmetric case.

3. Vector Representation

There are three vector representations for TSP: adjacency, ordinal and path. They have one common mutation operator that introduces changes into the tour, but different crossover operators.

The binary representation of tours is not well suited for the TSP. We are interested in the best permutation of cities and the binary representation will no provide any advantage and would require a special repair algorithm, since a change of a single bit may result in an illegal tour, but there are few exceptions that use binary representation, classical operators, crossover and mutation, and report high quality results [11].

The **adjacency representation** describes a tour as a list of *n* cities. The city *j* is listed in the position *i* if and only if the tour leads from city *i* to city *j*. Each tour has only one adjacency list representation; but, some adjacency lists can represent illegal tours. It does not support the classical crossover operator and a repair algorithm may be necessary. It has three crossover operators: alternating edges, subtour chunks, and heuristic crossovers [8].

The **ordinal representation** describes a tour as a list of *n* cities; the *i*-th element of the list is a number in the range from 1 to $n-i+1$. There is an ordered set of cities, which will be used as a reference point for the ordinal representation. The tour is described by an ordinal vector where the first element in the list is the first city visited. If we eliminate the first city in the set, the second element is the next city position in the current cities set and so on. The main advantage is that we can use easily the classical crossover. We can cut any two tours in the ordinal representation, after some position and cross them together, to produce two offspring, each of them a legal tour.

The **path representation**, used in this research, is the most natural representation of a tour. The vector contains an ordered list describing the itinerary of the cities that will be visited in the tour. The operator associated with this representation will be described in the next section.

3.1 Path Representation

The path representation for the tour 5-1-7-8-9-4-6-2-3, is simply (5 1 7 8 9 4 6 2 3). Three crossovers are defined for this representation: partially mapped (PMX), order (OX), and cycle (CX) crossovers.

Crossovers. The partially mapped crossover (PMX) was proposed by Goldberg and Lingle [7], and builds an offspring by choosing a subsequence from the tour in one parent and preserving the order and position of as many cities as possible from the tour in the other parent. A subsequence of a tour is selected by choosing two random cut points, which serve as boundaries for swapping operations. The PMX crossover exploits important similarities in the value and ordering simultaneously when used with an appropriate reproductive plan.

The order crossover (OX), proposed by Davis [3], builds an offspring by choosing a subsequence of the tour in one parent and preserving the relative order of cities in the other. The OX crossover exploits a property of the path representation, that the order of cities (not their positions) are important.

The cycle (CX) crossover, proposed by Oliver in 1987, builds an offspring in such a way that each city (and its position) comes from one of the parents until a chosen point. The offspring will contain the first city in the first parent, then the first city in the second parent, but using the position in the first parent and so on. The remaining cities are filled from the other parent. The CX preserves the absolute position of the elements in the parent sequence. We did not use the CX crossover, because the offspring are mainly influenced by only one parent.

Sometimes, the population is prematurely stabilized with the PMX and OX crossovers because the parents become too similar. We tried several modifications until we found one that allow us to generate offspring different to their parents, avoiding in that way a premature stabilization.

Our modification uses two cut points and the same subsequence between them in the offspring. The rest of the list will have the parents' sequence in a different sense. It is useful in the case where parents become too similar or the same, as in this example: P1= (1 2 3 | 4 5 6 7 | 8 9) and P2 = (1 2 3 | 4 5 6 7 | 8 9). The cut points marked by '|' would produce the following offspring H1= (**x x x** | 4 5 6 7 | **x x**) and H2= (**x x x** | 4 5 6 7 | **x x**), with the same parents segment between cut points. We use P2 sequence after the second cut point and continue with the initial sequence before the first cut point. We get the sequence, 8-9-1-2-3, that will be used from left to right to fill the **x**'s in H1. The result is H1= (8 9 1 | 4 5 6 7 | 2 3). For H2, we can use P1 sequence from right to left, removing cities already present in the list. The resultant sequence is 9-8-3-2-1, that can be used to fill the **x**'s from left to right in H2. The H2= (9 8 3 | 4 5 6 7 | 2 1).

Mutation. Mutation arbitrarily alters one or more genes of a selected chromosome, by a random change with a probability equal to the mutation rate. The intuition behind the mutation operator is the introduction of some extra variability in the population. It is useful to escape from local optima points when population is prematurely stabilized. Proportional probability corresponds to the relative position of the individual in the population. It affects a small percentage of individuals in the population. Mutation operators for TSP are insertion, reciprocal change, inversion and three edges selection.

For insertion operator, we randomly select a city and insert it in a different position in the tour, for reciprocal change operator, we randomly interchange two cities according to their position in the tour, and for inversion operator, we invert the cities' order between two random cut points (Grefenstette [8]).

In three edges operator, we randomly select three edges in the tour, as A=(*a1,a2*), B=(*b1,b2*) and C=(*c1,c2*). Later, we obtain the new tour connecting the city *a1* in edge A with city *b2* in edge B; the city *c1* in egde C with city *a2* in edge A and the city *b1* in edge B with city *c2* in edge C, generating a new tour. Restrictions apply, enforcing an order preservation between A, B y C, and a clockwise sense in the connection (Aho [1]). Because their great ability to improve diversity in the population, we used the insertion and three edge operators.

4. Matrix Representation

Lately, there were three independent attempts to construct an evolution program using matrix representation for chromosomes (Fox and McMahon [4], Seniw [11], and Homaifar and Guan[9]). Fox and McMahon represented a tour as a precedence binary matrix *M*. Matrix element m_{ij} in row *i* and column *j* contains a 1 if and only if the city *i* occurs before city *j* in the tour. The number of ones in this matrix is exactly $n(n-1)/2$; there are zeros in the diagonal where *i* and *j* has the same value and the indexes have the transitivity property: if there is 1 in the positions *ij* and *jk*, there will be 1 in *ik*.

In the second approach described by David Seniw, matrix element m_{ij} in the row *i* and column *j* contains a 1 if and only if the tour goes from city *i* directly to city *j*. There is only one nonzero entry for each row and each column, because there is only one city visited prior and after city *i*. Each complete tour is represented as a binary matrix with only one bit in each row and one bit in each column set to one. Subtours were allowed thinking that natural clustering would take place. At the end, the best chromosome is reduced to a single tour by successively combining pairs of subtours using a deterministic algorithm. The third approach, proposed by Homaifar and Guan [9] offers more advantages and was implemented here.

4.1 Homaifar and Guan's Matrix Representation and Operators

The m_{ij} element of the binary matrix *M* is set to 1 if an only if there is an edge from city *i* to city *j*. The matrix representation for chromosomes P1= (1,3,5,4,2) and P2= (2,4,3,1,5) is shown in Fig. 1.

Crossover. The matrix crossover operator (MX) exchanges all entries of the two parent matrices between two crossover points that cut the matrices vertically. An additional "repair algorithm" is run to cut and connect cycles and to produce a legal tour. Both offspring H1 and H2 in Fig. 1, have crossover points in columns 3 and 4 and are illegal. First step in the repair algorithm moves some ones in matrices in such a way that each row and each column has precisely one 1. For example, H1 has a duplication of ones in row 5, and the 1 in position M_{54} will be moved to position M_{14}, and H2 has a duplication in row 1, and the 1 in position M_{13} will be moved to position M_{53}.

Finally, H1 represents the legal tour (1,4,3,5,2) and H2, subtours (1,5,3) and (4,2). Second step, in the repair algorithm, applies only in second offspring. In this stage, the algorithm cuts and connects subtours to produce a legal tour. The cut and connect phase takes into account the existing edges in the original parents, *i.e.*, edge (5,4) is selected to connect the two subtours. The complete tour (a legal second offspring) is (1,5,4,2,3).

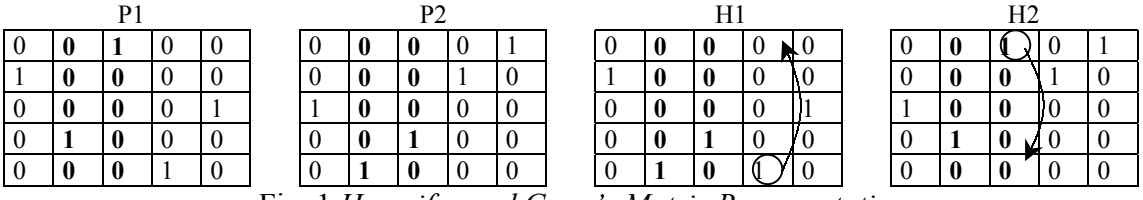

Fig. 1 *Homaifar and Guan's Matrix Representation*

Mutation. It can use the same mutation operators described for the vector type. We used insertion, reciprocal change and three edges selection for the Homaifar and Guan's matrix representation.

5. Computational Results

We used instances, generated with random uniform distributions, with different

sizes and previous known optimal solutions to test different values for each parameter, trying to improve our algorithm performance.

We tested populations with 10, 15, 20 y 25 individuals in each representation to select population size. Solution remained similar for larger sizes while the CPU time increased enormously. We tested probabilities of 50%, 60%, 70% and 80% for vector and matrix representations in Crossover; and 1%, 2% and 3% for matrix representation and 3%, 5% and 7% for vector representation in Mutation.

Best results for vector representation, were obtained with a population size of 20 individuals, modified crossover operator (OX) with a probability of 70%, and the insertion and three edges selection operators for mutation with a probability of 7%.

Best results for the Homaifar and Guan's matrix representation, were obtained with a population size of 10 individuals, insertion, reciprocal change and three edges selection with a crossover probability of 80% and a mutation probability of 1%. We used a random crossover operator in each instance (each operator was applied in approx. 1/3 of the complete set). Mutation operators were a great source of diversity here.

We tested 10 instances to get the average of the relative error. The optimal values were obtained with CPLEX. We reported the relative error between the best solution found by our genetic algorithm using 50,000 generations and the real optimal value.

Table 1 shows relative errors, for vector and matrix structures. For 50,000 generations, we used probabilities of 0.5, 0.6, 0.7 and 0.8. For 100,000 generations we only used 0.7 in vector case and 0.8 in matrix cases, the probabilities that produced the best results for the 50,000 generations cases.

We did the same computational tests for the maximization case with randomly generated instances. The best crossover probabilities were 0.7 for the vector representation and 0.8 for the matrix representation. .

Crossover	Vector Representation					Matrix Representation				
Probability	0.5	0.6	0.7	0.8	0.7	0.5	0.6	0.7	0.8	0.8
Cities	Generations: 50,000				100,000	Number of Generations: 50,000				100,000
20	0.1217	0.1006	0.1086	0.1157	0.0821	0.0833	0.0682	0.0824	0.0484	0.0470
40	0.1312	0.1145	0.1580	0.1201	0.1572	0.1345	0.1528	0.1412	0.1368	0.1302
60	0.1864	0.2177	0.1980	0.2217	0.1830	0.2627	0.2647	0.2441	0.2814	0.2298
80	0.2404	0.2585	0.2414	0.2499	0.2052	0.3170	0.3466	0.3385	0.3286	0.3145

Table 1. *Crossover Probability and Relative Error*

We used the number of iterations that helped us to reach the optimum value with those parameters. Table 2 has the average number of iterations for every ten instances.

Number of generations has a greater impact in the vector representation while it is not affecting too much the performance in the matrix case. Matrix representation has a better performance for problems with a small number of cities and a worse performance for instances with a big number of cities.

We used 50,000 generations, and a crossover probability of 0.7 and 0.8 for the vector and matrix chromosome structures.

Maximization instances can be solved with very little relative errors, while minimization instances produced bigger relative errors. In both cases, the results were better with the vector representation for examples with more than 20 cities.

Number of iterations has a bigger correlation with the number of cities in the matrix representation, while relative error is strongly dependent of the number of cities.

In cases with a high correlation coefficients between the number of cities and the number of iterations needed to approach the best solution, we can use minimum

squares to relate number of cities with 'advised' number of iterations to solve the problem and to know in advance the relative error expected.

For the Maximization type, the best crossover probabilities were 0.7 for the vector representation and 0.8 for the matrix representation. Relative errors for each problem type are in Table 2.

Our results show that data structure, size population and crossover and mutation operators and its specific probability, play a very important role in the solution of TSP with genetic algorithms. This research can be seen as another step to determine the impact of chromosome structures in the quality of the solutions and the CPU time needed to solve the asymmetric TSP.

	Vector Representation				Matrix Representation			
	Minimization		Maximization		Minimization		Maximization	
Cities	Iterations	Rel..Error	Iterations	Rel. Error	Iterations	Rel. Error .	Iterations	Rel. Error .
20	28509	0.0821	6913	0.0036	8902	0.0484	11471	0.0135
40	17679	0.1572	26861	0.0090	37707	0.1368	39871	0.0252
60	42757	0.1830	37912	0.0111	38126	0.2814	40988	0.0651
80	47486	0.2052	47832	0.0145	47117	0.3286	49549	0.0757
Correlation	0.8686	0.9224	0.8941	-0.0597	0.9196	0.9828	0.8692	0.9762

Table 2. *Iterations and Relative Error*

References

[1]Aho, A., Hopcroft, J., and J. Ullman, *Data Structure and Algorithms,* Addison Wesley Longman, 1998.

[2]Chambers, L., *Practical Handbook of Genetic Algorithms,* New Frontiers Vol. II, CRC Press, Inc, 1995.

[3]Davis, L., Applying Adaptive Algorithms to Epistatic Domains*, Proc. Int. Joint Conf. on AI*, 1985, pp. 162-164.

[4]Fox, B. and M. McMahon, *Genetic Operators for Sequencing Problems* in Rawlins, G., 1991, pp. 284-300.

[5]Garey, M. and D. Johnson, *Computers and Intractability*, W.H. Freeman, 1979.

[6]Gass, S., *Encyclopedia of Operations Research and Management Sciences*. Kluwer Academic Publishers, 1997.

[7]Goldberg D. and R. Lingle, Alleles, Loci and the TSP, *Proc. First Int. Conf. on GA*, Lauwrence Erlbaum Asociates, Hillsdale, NJ, 1985, pp. 154-159.

[8]Grefenstette, J., Incorporating Problem Specific Knowledge into Genetic Algorithms, *Proc. Int. Joint Conf. on AI*, 1987, pp. 42-60.

[9]Homaifar, A. and S. Guan, *A New Approach on the Traveling Salesman Problem by Genetic Algorithm*, Tech. Rep., N.Carolina A&T State Univ, 1991.

[10]Jog, P., Suh, J. and D. Gucht, *The Effects of Population Size, Heuristic Crossover, and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem* in Schaffer, 1989, pp. 110-115.

[11]Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1996.

[12]Oliver, I., Smith, D. And J. Holland, A Study of Permutation Crossover Operators on the Traveling Salesman Problem, *Proc. Second Int. Conf. on GA*, Lawrence Erlbaum Associatess, Hillsdale, NJ, 1987, pp. 224-230.

[13]Rawlins, G. (1991). Foundations of Genetic Algorithms, *First Workshop on the Foundations of GA and CS*, Morgan Kaufmann Publishers, San Mateo, CA.

[14]Schaffer, J. (1989). (Editor), *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, San Mateo, CA, 1989.