Solving the Traveling Salesman Problem using Genetic Algorithms

Luis F. Copertari

Abstract— **The Traveling Salesman Problem (TSP) is solved using genetic algorithms. Two alternative ways to represent the solution (genes in series and genes in two columns) are used and different mutation rates as well as different population sizes are considered. Two computer programs are created to solve the problem and yield experimental data. The data is analyzed and discussed and conclusions are derived. An equation relating the percentage of the population with the same solution (w) and the mutation rate (g) is discussed. This equation and the tradeoff between population size, mutation rate and chromosome portraying method are the theoretical contributions of this work, as well as an innovative way to solve the TSP, which is a hard problem to solve using traditional optimization, such as linear programming.**

Index Terms— **Traveling Salesman Problem, Genetic Algorithms, Evolutionary Computation**

I. Introduction

The Traveling Salesman Problem (TSP) is a very well-known problem in operations research, in which one salesman tries to visit n cities in a given tour. The problem is to find the sequence of visits that minimizes the sum of all of the distances between cities [1]. Checking all possible tours requires doing n! additions. For a sample problem with 27 cities, that means 1.088886945×10^{28} sums. A modern computer is capable of approximately one billion operations per second [2]. That means it would take 345,283,785,200 years to cover all possibilities a brute force approach. An impossible problem!

This problem can be solved using linear programming [3], but for large values of n it takes a relatively long time (hours, days, months, or even years) to come up with a solution. The approach is to use genetic algorithms. Although genetic algorithms do not guarantee an optimal solution, they provide good solutions and sometimes even the optimal solution in little time. Moreover, a solution (although maybe not the optimal) is always available. It is not necessary to wait until the program finishes. The user may request the best solution found so far at any time.

Manuscript received July 02, 2018

Luis F. Copertari is a professor and researcher at the Autonomous University of Zacatecas (UAZ). Prior to joining UAZ he has worked as research assistant and executive assistant at the Monterrey Institute of Technology and Advanced Studies (ITESM). Then the conducted research and teaching assistant duties at McMaster University while doing his Ph.D. He worked for one month as postdoctoral researcher at the University of Auckland. He has been a full time professor and researcher at UAZ for about sixteen years.

Genetic algorithms rely on evolution to find a solution. Each individual in the population represents one possible solution. The individuals are represented so that it is possible to create new generations by reproducing the best fitted individuals of the previous generation.

Two alternative ways to represent TSP tours were considered. The first one is by arranging the cities to visit in a row. The second one is to arrange the cities in two columns. Each approach entails a different way of reproducing the population. Both approaches were tried with different mutation rates to see which approach leads faster to the solution or provides the optimal solution.

First, the methodology is discussed. Second, the theory on the TSP and genetic algorithms is reviewed. Third, the experimental results are considered. Finally, the findings are discussed and conclusions are derived.

II. Methodology

The methodology is basically the scientific method. The scientific method is a series of steps, which are iterated in a series of successive and ever increasingly closer approximations to the truth. A theoretical approach had to come up based on the observation of the problem under study, test the theory in the laboratory in a series of experiments, and if the results were not within theoretical parameters, there was the need to go back to the theory and try to be more accurate based on the available observations.

Throughout the years, the scientific method has been discussed thoroughly. The specific approach used is based on the ideas presented by Gauch Jr. [4] and Wilson [5], which consists of the following steps:

1. Observation. The direct observation from reality is the basis of all discoveries. The problem was first considered and then it was realized that there are at least two different ways to portray the information: genes in series and genes in two columns.
2. Hypothesis. The right combination of mutation rate, population size, and model (genes in series or genes in columns) are important to determine how good the solution is and how fast the solution is reached.
3. Theory. What is to know about the TSP was considered as well as its implementation using genetic algorithms.
4. Experiment. Two computer programs using Delphi were created to solve the two genetic algorithm problems and obtain data for each representation, each possible mutation rate and each population size.
5. Conclusions. After analyzing the data obtained in the laboratory, the results were considered and discussed. Finally, conclusions are derived.

III. THEORY

A. *The Traveling Salesman Problem*

The Traveling Salesman Problem (TSP) is a classical problem in operations research. There are n cities, and it is supposed a salesman (or as an extension a load, a truck, or a plane) has to visit each and every city in order to minimize the total distance travelled.

The TSP is a NP problem, that is, the number of steps required to solve it grows exponentially as the problem size grows. The above means that to solve the TSP for typical examples from real life, where there is a fleet of hundreds of Wonder or Coke delivery trucks, or any other case that can be thought of, requires a prohibitive number of steps to obtain the solution by solving using linear programming, for example. That is, the computer would take too much time (years maybe) to solve a problem large enough so that it matters in real life.

Table 1 shows the distances between n cities. Notice that it is required to specify the distance between city 1 and city j, where j goes from 2 to n, which is done in the first row. For i=1 and j=1, there is a distance of zero, because the distance between city one and city one is none. However, to avoid going from one city to the same one, a very large distance is assigned, such as 100,000 (for all i = j). In the second row, there is the distance between city 2 and city j (where j goes from 3 to n; notice that j=1 is not included, because the distance between city 2 and city 1 is already in the first row as the distance between city 1 and city 2). And so on. In general, the distance between city i and city j, where j goes from i+1 to n, is denoted as d_{ij}. Notice that in this way the upper triangle of distances is obtained, marked in grey, which is duplicated in the lower triangle of distances. However, it is possible in reality that the distance to go from city i to city j (d_{ij}) may be different than the distance to go from city j to city i (d_{ji}).

Table 1. Distances between cities for the TSP.

City\City	1	2	3	…	n
1	100,000	d_{12}	d_{13}	…	d_{1n}
2	$d_{21}=d_{12}$	100,000	d_{23}	…	d_{2n}
3	$d_{31}=d_{13}$	$d_{32}=d_{23}$	100,000	…	d_{3n}
…	…	…	…	…	…
n	$d_{n1}=d_{1n}$	$d_{n2}=d_{2n}$	$d_{n3}=d_{3n}$	…	100,000

B. *Genetic Algorithms*

Genetic algorithms were developed by John Holland [6] during the 1960s and 1970s and popularized by one of his students, David Goldberg [7]. Genetic algorithms rely on the principles of genetics and natural selection to find solutions. De Jong [8] showed the usefulness of genetic algorithms for function optimization and tried to find optimal parameters for genetic algorithms.

Some of the advantages of genetic algorithms, taken from Haupt & Haupt [9], are:

- Optimizes with continuous or discrete variables.
- Doesn't require derivative information.
- Simultaneously searches from a wide sampling of the cost surface.
- Deals with a large number of variables.
- Is well suited for parallel computers.
- Optimizes variables with extremely complex cost surfaces (they can jump out of a local minimum).
- Provides a list of optimum variables, not just a single solution.
- May encode the variables so that the optimization is done with the encoded variables.
- Works with numerically generated data, experimental data, or analytical functions.

Genetic algorithms are very intriguing and they produce stunning results when traditional optimization techniques, such as linear programming, fail miserably. Sometimes, such as when evaluating an analytical function, calculus may outperform genetic algorithms. Also, the fact that a large population of solutions is required means that traditional methods may find the solution faster. However, if a parallel computer is available, genetic algorithms can take advantage of that because each processor may calculate the fitness of a group of individuals. Genetic algorithms are ideally suited for parallel computing.

There are five steps in the genetic algorithm process:

1. Encoding (genesis): Each individual in the population is generated such that it portrays one possible solution in a given string and it is possible to do the crossover using such encoding scheme.
2. Evaluation: Each solution of the individuals in the population is evaluated to find how well it fits the solution to the problem and each individual is assigned a score.
3. Crossover: Based on the score, only the best individuals are crossed or reproduced to create new individuals.
4. Mutation: In some cases, random changes or twitches are applied to some individuals randomly chosen for such purpose.
5. Decoding: After the process has finished, given so many generations have passed, the solution encoded in the representation of the individual with the best score is decoded and transformed into a solution to the problem. If an optimum (or at least a local optimum) has been found, a relatively large percentage of the population will have found such best solution.

C. *Solving the TSP with Genetic Algorithms*

It is possible to solve the traveling salesman problem using genetic algorithms, although Kirkpatrick, Gelatt and Vecchi [10] use simulated annealing to solve the problem. There are several ways to do the encoding. An encoding known as matrix representation, discussed by Homaifar, Guan and Liepins [11], as well as Michalewicz [12], is possible. In matrix representation, the tours are represented in a binary (0 and 1) matrix so that if there is a one in row i, column j, that means the salesman is going from city i to city j. This kind of matrix is also called a precedence matrix in operations research [1][3].

The matrix shown in Figure 1 indicates that the salesman is

going from city 1 (row 1) to city 2 (because in row 1, the one is in column 2). Also, that from city 2 goes to city 3 and from city 3 to city 1. As can be seen, the tour matrix from Figure 1 is valid because it constitutes a valid tour, that is, there are no repeated sequences such as, for example, having the possibility of going from city 1 to both cities 2 and 3.

Figure 1. Matrix representation using a precedence matrix.

	1	2	3
1	0	1	0
2	0	0	1
3	1	0	0

The problem with the matrix representation is the crossover operation. In crossover (or reproduction) the characteristics of the individual represented by their chromosome (encoding) are mixed to create new individuals. However, matrix representation as it is does not guarantee valid offspring.

The TSP problem can also be represented by a string of integers in two different ways. The first one, given by Homaifar et al. [11], Michalewicz [12], Goldberg [7] and Wroblewski [13], is using a string, such as $a_1a_2a_3\ldots a_n$. The previous string means that the salesman would go from city a_1 to city a_2, from a_2 to a_3, and so on until city a_n, and also from city a_n to city a_1 in order to complete the circuit.

For convenience, the string notation is going to be used. The encoding and decoding operations are very straightforward to do. However, the evaluation, crossover and mutation operations have to be explained. The evaluation operation is simply taking the distance between the first and second cities, adding the distance between the second and third city, and so on until city n, where the distance between city n and city 1 is added to complete the circuit.

The crossover operation is a little bit more complicated to explain. Let say there are strings with n cities. The best first half of individuals is taken and two of them are chosen at random from this pool of candidates. The first individual chosen is the male, the second the female. A random number between 0 and n is obtained. Let say such number is d. This random number indicates the breaking point in the male. If the number obtained is n, it means that all of the numbers in the male string must pass to the offspring (cloning of the male). If the number is zero, it means all the numbers in the female pass intact to the offspring (cloning of the female). If a number between 1 and n-1 is obtained, both individuals have to be combined. The first d numbers from the male are taken, and then all the non-repeating numbers from the female are taken to create a new sequence of numbers in which the left side of the male is preserved.

Consider the example shown in Figure 2, where n=5. The top numbers in the male chromosome are the cutting positions. Zero is at the beginning and five at the end. Let say d = 2. The cutting point in the male is indicated at the top of Figure 2. All the numbers to the left of the cutting point of the male are preserved in the offspring, so that the offspring chromosome begins with 5-3. To complete the sequence in the offspring, all the valid numbers from the female chromosome are taken. The first number in the female is 3, but 3 cannot be used because there is already a 3 from the male between position 1 and 2. The next number in the female is 4, and since 4 is not repeated, it is added. The third number in the female is 5, but that is not valid because the offspring already has a five. The fourth number in the female is 2, and since it is valid (not repeated) it is added. Finally, the fifth number in the female chromosome is 1, which is also valid and added to the offspring. In this way valid chromosomes as offspring can be generated.

Figure 2. Crossover operation for genes in series.

Male

0 1 2 3 4 5
5 - 3 | 1 - 2 - 4

Female

0 1 2 3 4 5
3 - 4 - 5 - 2 - 1

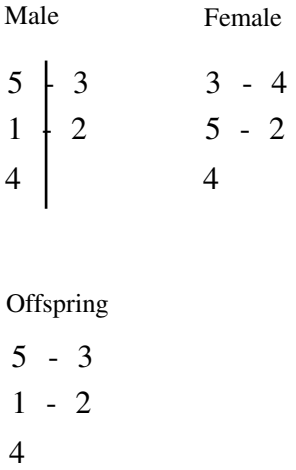
Offspring

0 1 2 3 4 5
5 - 3 - 4 - 2 - 1

It is also important to notice that apparently similar chromosomes may refer to the same tour. Consider for example n=5 and a tour such as 5-3-1-2-4. The chromosome 3-1-2-4-5 represents the same information as 5-3-1-2-4 because city 5 is at the end in the chromosome 3-1-2-4-5, but since the chromosomes are circular, after city 5 follows city 3, which is the same as in chromosome 5-3-1-2-4. The rest of the sequence after this is the same. Another variant is to invert the tours. Chromosome 4-2-1-3-5 is the same as chromosome 5-3-1-2-4 and chromosome 3-1-2-4-5, because the tour is still the same even though the order has been reversed.

Another variant being considered is to arrange cities on columns instead of series. This new arrangement changes the way in which the crossover occurs. Figure 3 shows the same case as the one in Figure 2. Notice that now, genes are arranged in two columns. For reproduction, the left side of the male is taken (5-1-4) and the right side is completed using the sequence from the female chromosome. In this case, the first city is 3, and since there is no 3 in 5-1-4, it is added. The second city is 4, but it is not included since 5-1-4 already contains 4. The third gene is 5, which is also not included since 5 is already in the sequence of the left column in the offspring (5-1-4). Then comes 2, which is included for not being in the left side sequence. Finally, there is 1, which is not included (the chromosome has been completed anyway), because it is already in the left side of the male chromosome (5-1-4).

Figure 3. Crossover operation for genes in columns.



Mutation is very simple to perform. It is just a matter of drawing a random number for each offspring and if it is less than or equal to a given mutation percentage rate, w, then such individual is to mutate. To perform the mutation on any given individual, two random numbers are simply drawn between 1 and n and swap the cities in those positions. Notice that now the positions do not refer to the space between the cities but to the space of the cities themselves.

There is a limit to mutation. Let g be the percentage of the population required to have the same solution in order to finish the run. Also, let w be the mutation rate, that is, the percentage of the population that is expected to mutate in a given iteration. Assume that for a particular generation there is, before mutation, 100% agreement in the solution and that g = 0.70. Since 100% > 70%, if there were no mutation, the run would be finished. Also, let say that the mutation rate is 50%. Since out of the 100%, 50% of the population is expected to mutate, we would end up having only 50% of the population untouched with the original solution. In this situation, the algorithm would never converge. In general, equation (1) applies.

$$g+w < 1 \qquad (1)$$

Equation (1) indicates that in order for the algorithm to have a chance to converge, g+w must be less than one. A value for g of 0.7 (70%) is used. Clearly, a mutation rate of 50% (w=0.5) is too high and the algorithm would never converge under such conditions since 0.7+0.5 = 1.2 > 1.

IV. Testing

To test whether or not a given encoding scheme (it could be genes in series or genes in columns) implies reaching the solution faster or assuring an optimal solution, a number of r tests are going to be conducted. One of the variables in the tests is the encoding mechanism: series or columns. Another variable is the mutation rate: 5%, 10%, 20% or 50%.

There is also a final variable. Considering populations throughout human history, the question concerning the size of the population having an impact in the speed in which the population evolved arises. In the distant past, when our ancestors were in the stone-age period, populations were relatively small. Now there are large populations. Does population size have any effect in the speed the population is evolving? Translating the idea into genetic algorithms, a third variable for the tests was derived: population size (m). A relatively large population size is considered, such as 1,024 individuals and then divide that population by 16, which makes 64 individuals, for a second population size. The number of generations it takes for the population to reach a 70% (g = 0.7) having the same solution is considered. This variable is taken unto account for both scenarios (genes in series and genes in columns) and for all mutation rates (5%, 10%, 20%, and 50%).

For testing, a medium-size problem is used. Twenty-seven of the most important cities in México are being considered. The number each city has been assigned and the names of the cities are given in Table 2.

Table 2. List of cities.

Number	City
1	Acapulco
2	Aguascalientes
3	Cancún
4	Cuernavaca
5	Chihuaha
6	Durango
7	Guadalajara
8	Hermosillo
9	León
10	Manzanillo
11	Mérida
12	México City
13	Morelia
14	Monterrey
15	Nuevo Laredo
16	Oaxaca
17	Pachuca
18	Puebla
19	Querétaro
20	San Luis Potosí
21	Tampico
22	Tijuana
23	Toluca
24	Torreón
25	Veracruz
26	Villahermosa
27	Zacatecas

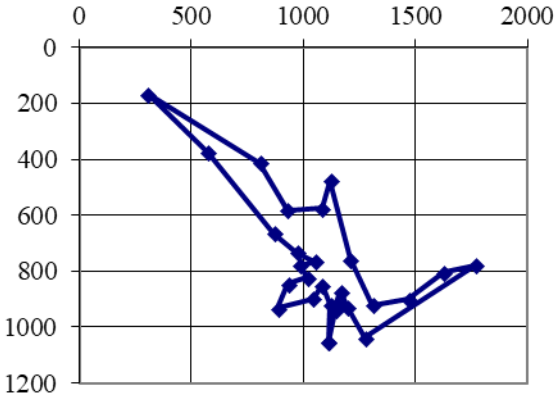
The distances between cities are given in the Appendix.

This problem using linear programming and LINDO/LINGO was solved. The optimal solution to the problem is: 1-23-19-13-10-7-9-2-20-27-6-8-22-5-24-14-15-21-25-26-11-3-16-18-17-12-4. The total distance for this solution is 12,216 km. Notice that this solution is cyclical, and after the last city (4) comes the first city again (1). The fact that the solution is cyclical also means there are 27x2 = 54

alternative ways of showing the same solution.

Positions for the cities were obtained in a map. Figure 4 shows the optimal route on a virtual map of México (coordinates are in pixels). In theory, the optimal solution should have no crossings. It can be seen in the map of Figure 4 that there are no crossings.

Figure 4. Route for the optimal solution.



In order to get an idea of the matching between the routes in Figure 4 and an actual map of México and its routes, take a look at Figure 5.

Figure 5. Map of Mexico's routes.

The average of the number of generations it takes to reach 70% agreement in the solution (run or r) is calculated. There is also another important matter: how many runs should be considered?

To calculate sample size (r), the formula for sample size known as the standard deviation (σ) was used. The formula was taken from a book by Kvanli, Guynes and Pavur [14] and it is detailed in equation (2).

$$r = \left(\frac{z_{\alpha/2}\sigma}{E}\right)^2 \tag{2}$$

$Z_{\alpha/2}$ is the value of the normal distribution for a given confidence interval (α). In this case, $Z_{\alpha/2}$ is equal to 1.96, which implies a confidence of 95%. The value for σ is the standard deviation of the population, which typically is approximated using the standard deviation of the sample. Since there is no such value, it is possible to approximate it using equation (3).

$$\sigma \approx \frac{H-L}{4} \tag{3}$$

The reason for the formula in equation (3) is that it is known as an empirical rule that 95.4% of the population will be between μ-2σ and μ+2σ. Let L = μ-2σ be the minimum value expected in the sample and H = μ+2σ be the maximum value expected, so that H-L = 4σ. Since the statistic indicates the number of runs required, the maximum value for such statistic would be r (H = r) and the minimum value would be zero (L = 0). Substituting H and L into equation (3) yields equation (4).

$$\sigma \approx \frac{r}{4} \tag{4}$$

Substituting σ from equation (4) into equation (2) yields equation (5).

$$r = \left(\frac{z_{\alpha/2}r}{4E}\right)^2 \tag{5}$$

Solving for r from equation (5) yields equation (6).

$$r = \left(\frac{4E}{z_{\alpha/2}}\right)^2 \tag{6}$$

The only parameter that is not being considered is E, which is the absolute error. A value for E of 7 is used. Applying equation (6) yields the sample size: 200, as indicated in equation (7).

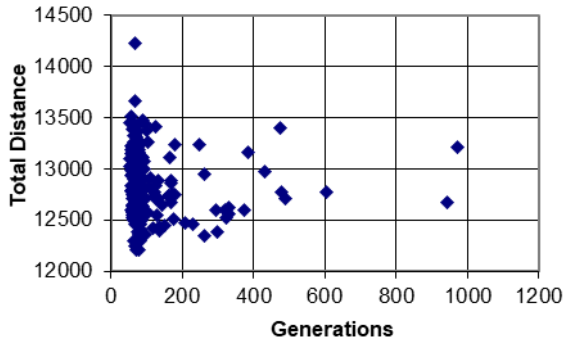
$$r = \left(\frac{4\times7}{1.96}\right)^2 = 204.08 \approx 200 \tag{7}$$

V. Results

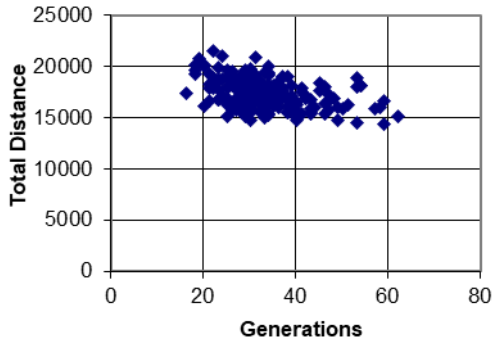
What is a run? There are five steps in genetic algorithms as seen in section 3.B. These steps are: 1. Encoding/Genesis, 2. Evaluation, 3. Crossover, 4. Mutation, and 5. Decoding. After creating a population (step 1), the fitness of all individuals in such population is evaluated (step 2), then comes to reproduce them according to the scheme specific to such sequence (step 3) and some of the individuals mutate according to a given mutation rate (step 4). Then, steps 2, 3, and 4 are iterated a number of times (i) until there is at least g×100% of individuals with the same solution. A run is precisely such sequence of repetitions for a given population. After there is agreement on the solution, the number of generations it took to reach such result, the total distance for the given solution string, as well as the result itself (the sequence of cities to visit) are recorded in files called, by default, output1.txt and output2.txt (step 5). A total of 200 runs are used and the resulting data is recorded. The average of the numbers of generations and total distances for all 200 runs is calculated.

The first set of 200 runs is for a mutation rate of 5% (w=0.05) and a population size of 1,024 (m=1,024) using genes in series. Figure 6a shows an XY scatterplot of the data. Right next to it, in Figure 6b, is the same mutation rate but for a population of 64 (m = 64).

Figure 6. 200 runs for w = 0.05 having genes in series.
a. m = 1,024



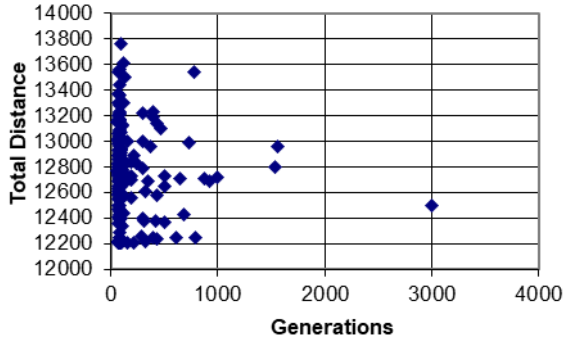
b. m = 64



The best run for m = 1,024 generated a route with the same (minimum) total distance of 12,216 km. The route obtained was 19-13-7-10-9-2-20-27-6-8-22-5-24-15-14-21-25-26-11-3-16-18-17-12-4-1-23. This route is the same as the minimum total distance route obtained using linear programming, except for the pairs 7-10 and 15-14, which are reversed. Notice that the linear programming solution starts with 1-23-19-13…, whereas this solution starts with 19-13… Both are the same except for the two pairs mentioned. Nevertheless, both have the same minimum total distance of 12,216 km.

Figure 7 and Figure 8 have the same results for mutation rates of 10% and 20%, respectively. As expected, there could not be obtained results for a mutation rate of 50% given the fact that g+w would be equal to 1.2, which is greater than 1; see equation (1).

Figure 7. 200 runs for w = 0.10 having genes in series.
a. m = 1,024



b. m = 64

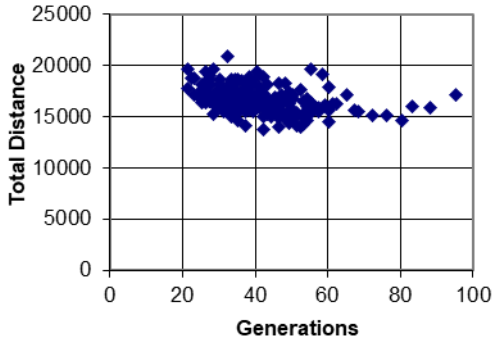
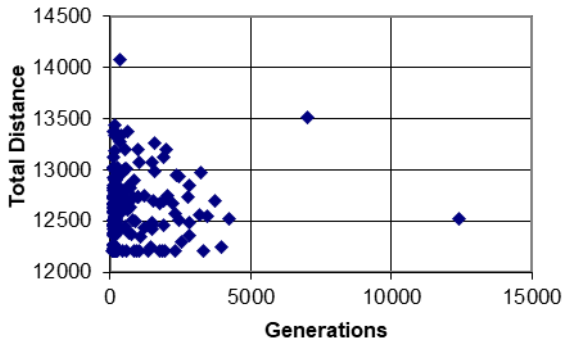
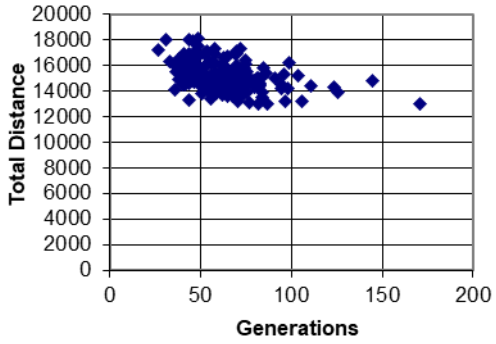


Figure 8. 200 runs for w = 0.20 having genes in series.
a. m = 1,024

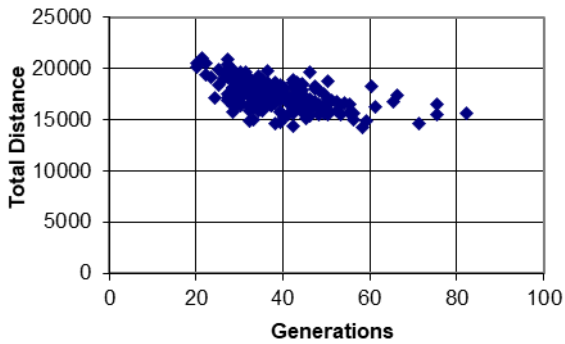


b. m = 64



The same set of results was obtained for genes in columns, as shown in Figure 9, Figure 10, and Figure 11 for mutation rates of 5%, 10%, and 20%, respectively. Once again there could not be results for a mutation rate of 50%.

Figure 9. 200 runs for w = 0.05 having genes in columns.
a. m = 1,024



b. m = 64

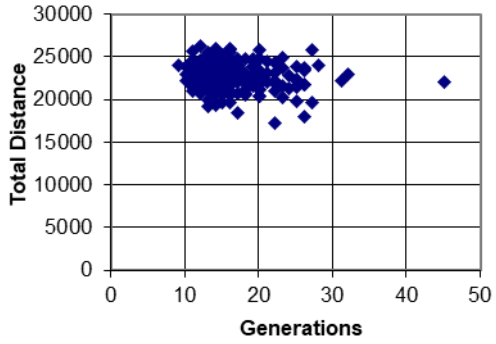
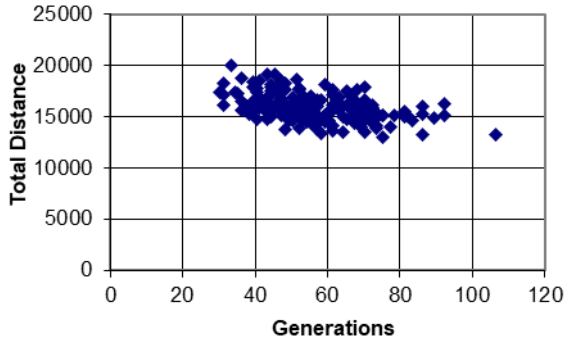


Figure 10. 200 runs for w = 0.10 having genes in columns.
a. m = 1,024



b. m = 64

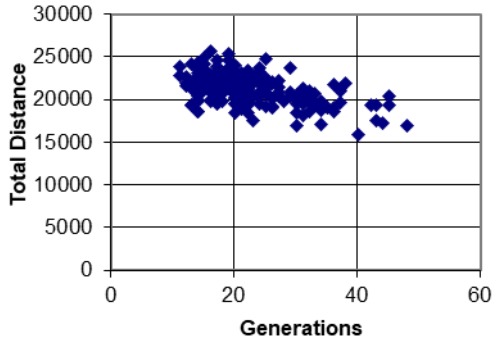
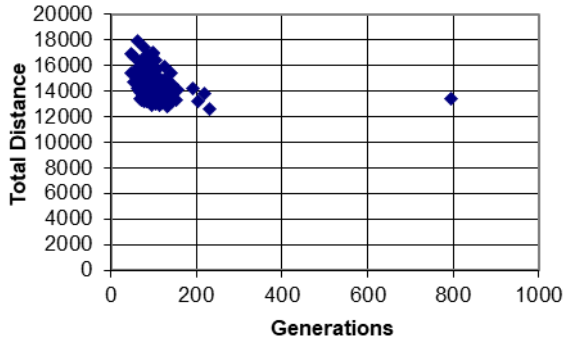
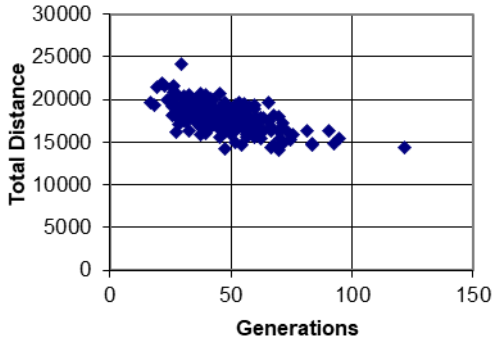


Figure 11. 200 runs for w = 0.20 having genes in columns.
a. m = 1,024

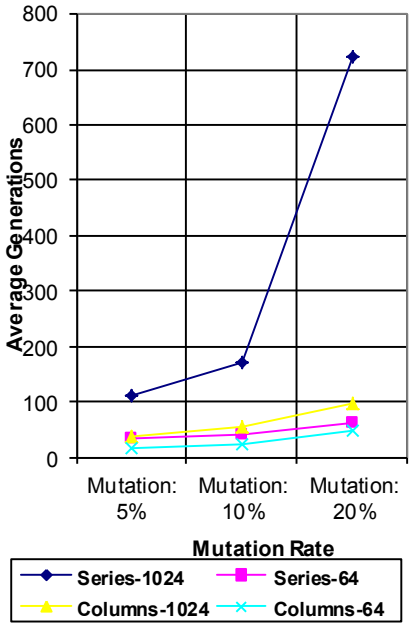


b. m = 64

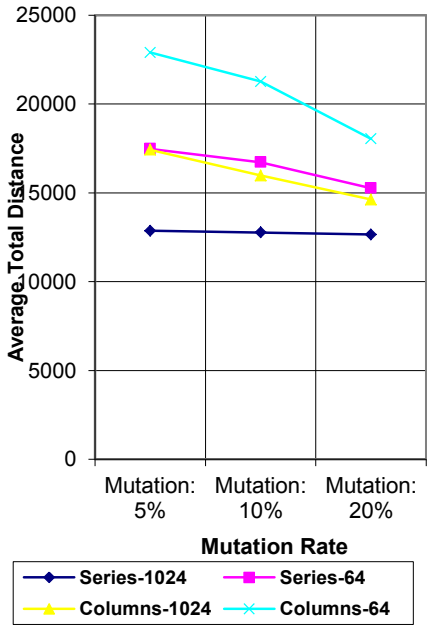


The average number of generations of each set of 200 runs was also calculated, which is shown in Figure 12a, as well as the average total distance of each set of 200 runs, shown in Figure 12b.

Figure 12. Average numbers for each set of 200 runs.
a. Number of generations



b. Total distance



VI. Discussion and Conclusion

From Figure 6 it can be seen that it takes less generations (usually about 40) to converge when the population is small (m = 64) as opposed to a large population (m = 1,024). However, the results obtained are worse when the population is small, which makes sense because a larger population allows the system to consider a wider set of alternatives.

The same can be said for higher mutation rates. Notice that for a mutation of 10% (w = 0.10) and for a mutation of 20% (w = 0.20) several optimal solutions were found: 6 for 10% mutation and 17 for 20% mutation. Also notice that mutation tends to increase the number of generations it takes to reach a solution. Apparently, a higher mutation rate tends to improve the solution, as long as the mutation is not too high to break the rule from equation (1).

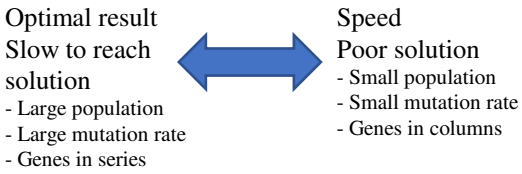
For the case of genes in column, none of the 200 runs of each type of mutation and population size achieved optimal results. Comparing Figure 6 and Figure 9 it can be seen that it takes genes in columns a considerably shorter number of generations to reach agreement on the solution. About 40 generations were required to reach a solution for genes in columns having a mutation rate of 5% and a large population size of 1,024, whereas it took about 100 for genes in series. Apparently, reproducing the population with genes in columns reaches a solution faster. Also, using genes in columns seldom yields an optimal solution.

Figure 12a summarizes the results by considering the average of all the figures (total number of generations) for each case: genes in series with m = 1,024, genes in series with m = 64, genes in columns with m = 1,024 and genes in columns with m = 64. It can be seen that the case with the lowest average number of generations to reach a solution is genes in columns with a small population size, followed by genes in series with a small population size. Next is genes in columns with a large population size and finally genes in series with a large population size. Clearly, if the idea is to reach a solution fast there should be used a small population size. Also, as the mutation rate increases, the average number of generations to reach a solution increases too. The extreme example is the case of genes in series, a large population (m = 1,024) and a high mutation rate (w = 0.20), exceeding 700 generations on average.

Figure 12b shows the same four cases applied to the average total distance of the resulting tour obtained. The case with the minimum averaged total distance is series with a large population closely followed by columns with a large population. The worst cases were series with small population and finally columns with small population. Thus, in order to achieve a better result (minimum total distance of the tour), it is best to use a large population size. Notice also that as the mutation rate increases, the results are better for all four cases.

In short, a small population size increases speed but reduces performance. Also, a large mutation rate increases performance. Working with genes in series increases performance, whereas genes in columns reduces the number of runs required to reach a solution. For optimal performance (better result), it is recommended to use a large population, large mutation rate and genes in series. The mutation rate should be 10% or 5% under the largest value allowed according to equation (1). In the case under study, the population limit (g) is 70%. Thus, the best mutation rate should be 100%-70%-10% = 20% or maybe even 100%-70%-5% = 25%. For optimal speed (faster to the result), it is recommended to use a small population, small mutation rate, and genes in columns. Figure 13 summarizes the results obtained.

Figure 13. Summary of findings.



There is a tradeoff between the population size, the mutation rate, and the chromosome portraying method. If good performance (optimal solution) is the objective, a large population and large mutation rate, but not as large as to break equation (1), should be considered. This of course results in larger solution times, so a balance among the variables should be attained. If a fast result is the objective, a small population size and a small mutation rate should be used, but not so small as to render useless solutions.

Once the target percentage of the population with the same result (g) has been decided, the best mutation rate (w) is simply between 1% and 5% below 1-g; that is between w=1-g-0.01 and w=1-g-0.05.

APPENDIX. DISTANCES AMONG CITIES

Destination/ Origin	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	100000	899	1994	299	1856	1308	934	2337	781	675	1679	388	652	1313
2	899	100000	2158	600	979	433	250	1654	128	549	1843	511	322	588
3	1994	2158	100000	1706	3119	2571	2197	3600	2044	2442	315	1651	1953	2378
4	299	600	1706	100000	1557	1009	635	2038	482	880	1387	89	391	1014
5	1856	979	3119	1557	100000	667	1177	690	1107	1476	2800	1468	1301	783
6	1308	433	2571	1009	667	100000	631	1221	561	930	2252	920	755	563
7	934	250	2197	635	1177	631	100000	1403	230	299	1878	545	268	788
8	2337	1654	3600	2038	690	1221	1403	100000	1633	1630	3281	1949	1671	1473
9	781	128	2044	482	1107	561	230	1633	100000	530	1725	393	194	714
10	675	549	2442	880	1476	930	299	1630	530	100000	2123	791	484	1085
11	1679	1843	315	1387	2800	2252	1878	3281	1725	2123	100000	1332	1634	2063
12	388	511	1651	89	1468	920	545	1949	393	791	1332	100000	302	925
13	652	322	1953	391	1301	755	268	1671	194	484	1634	302	100000	903
14	1313	588	2378	1014	783	563	788	1473	714	1085	2063	925	903	100000
224	1138	1148	2287	1309	938	1697	1010	787	1007	1238	456	177	1494	1710	1528	2602	812	634	477	1537	15
1359	756	454	1179	1245	847	2403	1000	1374	1922	456	965	528	1710	1528	2602	812	634	477	1537	15	16
900	390	88	1391	879	410	1995	592	933	1482	1591	178	1043	1528	2602	812	634	477	1537	15	16	17
1048	425	123	1209	914	516	2072	669	1043	1591	178	1304	1866	296	603	18
708	195	215	1547	592	178	1763	360	701	1247	304	512	2074	168	811	19
517	397	423	1755	635	197	1720	336	499	1045	512	577	1848	574	876	20
530	790	488	1533	1041	603	2003	742	905	1313	577	1085	2647	518	1384	21
2338	2536	2814	4148	2495	2498	865	2268	2086	1555	2903	153	1715	491	285	22
903	238	64	1398	727	373	1885	482	896	1442	153	467	2647	518	1384	23
316	838	998	2326	1013	644	1157	714	247	467	1085	100000	1706	929	775	24
1023	720	418	1040	1209	811	2367	964	1338	1898	476	1359	929	1284	1120	25
1504	1075	773	559	1564	1166	2722	1319	1893	2241	828	878	1284	130	1005	26
458	452	617	1949	527	258	1524	328	303	849	706	2268	130	1005	27

Destination/ Origin	15	16	17	18	19	20	21	22	23	24	25	26	27
1	1537	703	476	477	603	811	876	3202	285	1384	775	1120	1005
2	812	965	528	634	296	168	574	2519	491	518	929	1284	130
3	2602	1494	1710	1528	1866	2074	1848	4465	1715	2647	1359	878	2268
4	1238	456	177	178	304	512	577	2903	153	1085	476	828	706
5	1007	1922	1482	1591	1247	1045	1313	1555	1442	467	1898	2241	849
6	787	1374	933	1043	701	499	905	2086	896	247	1338	1893	303
7	1010	1000	592	669	360	336	742	2268	482	714	964	1319	328
8	1697	2403	1995	2072	1763	1720	2003	865	1885	1157	2367	2722	1524
9	938	847	410	516	178	197	603	2498	373	644	811	1166	258
10	1309	1245	879	914	592	635	1041	2495	727	1013	1209	1564	527
11	2287	1179	1391	1209	1547	1755	1533	4148	1398	2326	1040	559	1949
12	1148	454	88	123	215	423	488	2814	64	998	418	773	617
13	1138	756	390	425	195	397	790	2536	238	838	720	1075	452
14	224	1359	900	1048	708	517	530	2338	903	316	1023	1504	458
15	100000	1583	1124	1272	932	741	754	2582	1127	540	1247	1728	682
16	1583	100000	513	331	669	877	829	3266	518	1450	382	620	1071
17	1124	513	100000	182	232	438	387	2880	152	1020	422	832	630
18	1272	331	182	100000	338	546	496	2937	187	1119	298	650	740
19	932	669	232	338	100000	202	499	2626	195	775	633	988	398
20	741	877	438	546	202	100000	406	2585	408	582	841	1198	196
21	754	829	496	100000	499	406	100000	2868	552	846	493	974	502
22	1127	518	152	1119	775	582	846	2022	970	100000	482	837	593
23	540	1450	1020	1119	775	582	846	2022	970	100000	482	837	593
24	1247	382	422	298	633	841	582	846	2022	970	100000	1414	1769
25	1728	620	832	650	988	1198	974	3587	837	1769	481	100000	1390
26	682	1071	630	740	398	196	502	2389	593	386	1035	1390	100000
27	1537	703	476	477	603	811	876	3202	285	1384	775	1120	1005

ACKNOWLEDGMENT

The author would like to thank the support of the Autonomous University of Zacatecas (UAZ) while doing this research. Also, I would like to thank the help and guidance of colleagues within UAZ.

REFERENCES

[1] Elsayed, Elsayed A. & Boucher, Thomas O. (1994). *Analysis and Control of Production Systems*. Upper Saddle River, New Jersey: Prentice Hall.
[2] Moravec, Hans. (1999). *Robot: Mere Machine to Transcendent Mind*. Oxford, England: Oxford University Press.
[3] Winston, Wayne L. (1994). *Operations Research: Applications and Algorithms*. Belmont, California: Duxbury Press.
[4] Gauch Jr., Hugh G. (2003). *Scientific Method in Practice*,. Cambridge, United Kingdom: Cambridge University Press.
[5] Wilson, Edgar Bright. (1990). *An Introduction to Scientific Research*. New York, NY: Dover Publications, Inc..
[6] Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.
[7] Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimizations and Machine Learning*. Addison-Wesley.
[8] De Jong, K.A. 1975. *Analysis of the behavior of a class of genetic adaptive systems*. Ph.D. Dissertation, University of Michigan, Ann Arbor.
[9] Haupt, Randy L. & Haupt, Sue Ellen. (2004). *Practical Genetic Algorithms*. Hoboken, NJ: John Wiley & Sons, Inc.
[10] Kirkpatrick, S., Gelatt, C. D. & Vecchi, M. P. (1998). "Optimization by simulated annealing", *Science*, *220*, 671-680.
[11] Homaifar, Abdollah, Guan, Shanguchuan & Liepins, Gunar E. (1992). "Schema analysis of the traveling salesman problem using genetic algorithms", *Complex Systems*, *6* (2), 183-217.
[12] Michalewicz, Zbigniew. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag.
[13] Wroblewski, Jakub. (1996). "Theoretical foundations of order-based genetic algorithms", *Fundamenta Informaticae*, *28* (3-4), 423-430.
[14] Kvanli, Alan H., Guynes, C. Stephen & Pavur, Robert J. (1989). *Introduction to Business Statistics*. West Publishing Company.

Luis F. Copertari is a professor and researcher at the Autonomous University of Zacatecas (UAZ). Prior to joining UAZ he has worked as research assistant and executive assistant at the Monterrey Institute of Technology and Advanced Studies (ITESM). Then the conducted research and teaching assistant duties at McMaster University while doing his Ph.D. He worked for one month as postdoctoral researcher at the University of Auckland. He has been a full time professor and researcher at UAZ for about sixteen years.