Solving the Travelling Salesman Problem with the Excel Sort Function and Visual Basic for Applications

Richard J. Perle
Department of Finance and Computer Information Systems,
Loyola Marymount University,
One LMU Drive
Los Angeles, CA, USA 90045
rperle@lmu.edu
310.338.2929

Abstract
This paper develops and tests the performance of a new and novel heuristic algorithm for solving the Travelling Salesman Problem. The algorithm is relatively easy to understand and apply because it is implemented within Excel using the sort function and the Visual Basic for Applications programming language. The method has an error function that varies linearly with the problem size. Empirical testing shows that it works well for small problems, but it could be extended to solve larger ones.

Keywords: Traveling salesman problem, Heuristic algorithm, Excel VBA

1. Introduction

The classic Travelling Salesman Problem (TSP) describes the situation where a salesperson wants to leave his/her home city, visit a number of other cities and then return home. Each city, which constitutes a node in a Cartesian coordinate graph of the problem, is to be visited only once. The travel route from city to city constitutes a tour. Obviously, there exists an optimal, shortest-tour solution which is to be found. The TSP is an important problem to solve because there are so many other practical Business Operations Management situations that can be modeled and solved as a TSP. Some examples are: scheduling airliners (Clarke, et. al., 1997), delivery vehicle routing (Laporte, 1992), plotting the drilling of holes in circuit boards Grotschel, et. al., 1991), picking orders in a warehouse (Ratlif and Resenthal, 1983) and sequencing jobs in a machine shop (Lenstra, et. al., 1977).

A characteristic of any TSP is that it is difficult to solve to a provable optimum. Optimal solutions can be found by branch-and bound integer programming methods, described by Lawler, et. al. (1985), branch-and-cut methods, described by Junger, et al. (1995) and by explicit enumeration of all possible solutions. However, theseare not viable methodologies for solving large problems as the number of possible solutions is a factorial function of the number of nodes, resulting in unacceptably large solution times. As a result of the difficulty in finding optimal solutions, practical and useful near-optimal solutions to TSPs can be obtained by the use of heuristic algorithms. There are three general types of TSP heuristics; (1) construction methods, (2) improvement methods and (3) metaheuristic methods. Construction methods start at an arbitrary node and then select succeeding nodes according to a criterion such as cheapest or shortest distance. Well-known examples of construction methods are variations of the Nearest Neighbor Greedy (NNG) algorithm (Rosenkrantz, et. al., 1977) which will be used for comparison purposes in this paper.

Improvement methods start with a feasible tour and then make changes in an effort to find a shorter tour. The 2-Opt, 3-Opt and Lin-Kernighan (1973) algorithms are examples of improvement methods. Metaheuristics such as simulated annealing, tabu search, genetic algorithms and artificial neural networks search neighborhoods for local optima and then use that information to search for better solutions without getting trapped in any one local neighborhood. A good description of basic metaheuristic methods can be found in Reeves (1993).

The Excel Sort (ES) algorithm described in this paper is a hybrid method. It starts with a feasible tour constructed by a simple node-to-node process, then systematically generates a subset of additional complete, feasible tours, ultimately selecting the best tour from the set of feasible tours. It has the advantage that the software may be developed and implemented by using the Excel sort function along with an Excel Macro and VBA. The practical usefulness of any non-optimal heuristic solution, of course, would depend on its expected accuracy which in this paper is measured as the expected percent over an optimal or benchmark tour.

The next two sections of this paper will present the ES algorithm process logic by way of example, and then the mathematics of the general model will be developed. Its robustness and accuracy will then be benchmarked against actual TSP data for which a very good or optimum solution is known.

2. The Excel Sort Algorithm Procedure

The ES algorithm is an eight step procedure as listed below. Steps 2, 4 and 5 are implemented by an Excel sort command. Steps 2-7 are executed iteratively in an Excel VBA program loop in a search for a best solution tour distance value, which is step 8.

The ESS Algorithm:

1. Assign a unique number to each node
2. Sort the nodes on the X-values, from low to high.
3. Based on the X-values, separate the nodes into two equal size sets.
   Left-most set = nodes with the smallest X-values.
   Right-most set = node with the largest X-values.
4. Sort the left-most set of nodes on the Y-values, from low to high.
5. Sort the right-most set of nodes on the Y-values, from high to low.
6. Connect the two sets to identify the complete tour and calculate the tour distance value.
7. Incrementally rotate the Set of nodes 180/n degrees, then repeat steps 3 - 6 until the set has been rotated a total of (180 - 180/n) degrees.
8. Select the solution with the best tour length.

As a simple numerical example consider a set of ordered pair X-Y values for a ten node TSP, shown in Table 1, where the pairs are sorted in ascending X-value order. The objective is to start at any one node and then travel from node-to-node, visiting each node only one time, and end up back at the starting node.

Figure 1 shows a graphic example of the first six steps of the algorithm applied to the data in Table 1. The dashed linesin Table 1 and Figure 1 separate the 10 nodes into two equal size sets of five nodes each, designated as the left-most set and the right-most set (steps 2 and 3). The arrows show the tour path that the first six steps of the first iteration of the ES algorithm would find from start to finish. Step 4 sorts the left-

most set of nodes on the Y-values from low to high; step 5 sorts the right-most set on the Y-values, from high to low. The tour path by node number is: 3-5-1-4-2-7-10-6-9-8-3. Its length is 314.20.

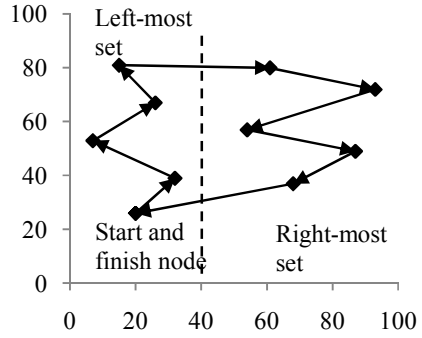

Figure 1: Solution Path for Table 1 Data

It can be inferred from Figure 1 that an advantage of the ES algorithm compared to other solution algorithm is that there will be no crossing-paths within either set. Crossing paths will add to the tour length, assuring that it is not optimal. The only time a crossing path could (but not necessarily) be generated is on the two paths that connect the sets together; the upper-most arrow and the lower-most arrow in Figure 1.

The entire set of nodes is now rotated about its geometric center in successive iterations of the ES algorithm. The rotation does not change the distance between any pair of nodes. Steps 2 through 6 in the algorithm are then repeated to generate a new solution. For example, the nodes in Figure 1 are rotated 90 degrees clockwise and a new ES solution is generated from the new, rotated X-Y coordinates shown in Table 2.

Table 2. Rotated X-Y values.

Node	New X-value	New Y-value
1	16.6	82.8
2	29.6	70.8
3	43.6	95.8
4	57.6	76.8
5	71.6	83.8
6	70.6	41.8
7	62.6	9.8
8	47.6	48.8
9	39.6	15.8
10	27.6	34.8

The new tour path by node number as shown in Figure 2 is: 9-10-2-1-3-5-4-8-6-7-9. The tour length is reduced to 262.83 which is a 31.69 percent improvement over the tour in Figure 1. It is also optimal, proved by explicit enumeration. Even though 90 degrees is the best rotation angle there is no way to know, a priori, that this is the case. So the objective in the ES algorithm is to rotate the nodes incrementally, attempting to generate a new solution with each incremental rotation until a best, but not necessarily optimal solution is found.

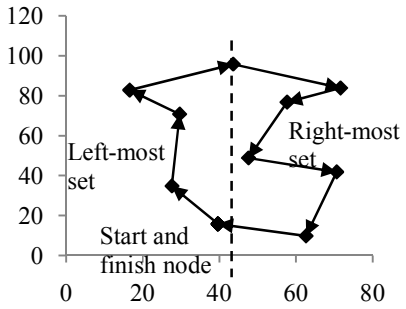

Figure 2: Solution Path for Table 2 Data

The condition that creates a new solution path occurs when a node passes from the right-most set to the left-most set and vice-versa due to rotation. In effect the two sets evolve by exchanging a pair of nodes at each incremental rotation. Consequently, the largest number of different solutions that the rotation process is capable of generating is n, the number of nodes. The set of n unique solutions, if they are all detected, is generated over a total rotation of only 180 degrees. Rotating the nodes beyond 180 and on to 360 degrees simply generates a repeat of the zero to 180 degree rotation solutions, except the tour path is in the opposite direction. But the tour length is obviously the same in either direction. There are a number of rules that could be used to determine the incremental angle of rotation, but the simplest rule, used here, is to rotate 180/n degrees in contiguous increments, starting from zero to (180 - 180/n) degrees. The zero degree position is the original configuration of the nodes.

Another question to address is how to handle an instance where there is an odd number of nodes. Obviously the set of nodes cannot be divided into two sets, each with the same number of nodes. The easiest way to handle this is to arbitrarily assign one extra node to either set, which is done here. There are more sophisticated ways to solve the problem such as creating a "dummy" node as a duplicate of an existing node which will add nothing to any tour length because the sorting process should always connect the dummy node to its real node. However, this could perturb the geometric center of the nodes leading to unknown effects. The general ESalgorithm and the node rotation methodology will now be developed.

3. The Excel Rotation Method

Assume the Cartesian coordinates, $x_i y_i$ (i = 1 to n) in a two-dimensional flat plane for a TSP with n nodes are known. The symmetric Euclidian distance, d_{ij} between any two nodes, i and j ($i \neq j$) is calculated as:

$$d_{ij} = SQRT[(x_i - x_j)^2 + (y_i + y_j)^2] \qquad (1)$$

The ordered pair, $x_c y_c$ is defined as the geometric center of the set of nodes and is calculated as the mean of the X-values and the Y-values:

$$x_c = [\sum_{i=1}^{n} x_i]/n \qquad (2)$$

$$y_c = [\sum_{i=1}^{n} y_i]/n \qquad (3)$$

Consider now a translated coordinate system centered on x_cy_c in the original Euclidian plane which is divided into four quadrants (Q1 to Q4) as shown in Figure 3. An arbitrary node, x_iy_i and its vector is shown in Quadrant 2 along with its angle, θ_i relative to the positive X-axis which is defined as zero degrees. The Euclidian distance from x_cy_c to x_iy_i is h_i and calculated as :

$$h_i = SQRT[(x_i - x_c)^2 + (y_i - y_c)^2] \quad (4)$$

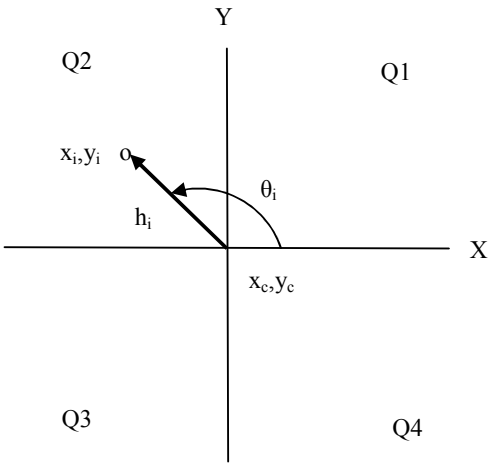

Figure 3: Translated X-Y Coordinates

The calculation of any value of θ_i depends on which quadrant the node is in.

If: $x_i - x_c >= 0$ and $y_i - y_c >= 0$, the node is in Q1,and:

$$\theta_i = \sin^{-1}[(y_i - y_c)/h_i] \quad (5)$$

If: $x_i - x_c < 0$ and $y_i - y_c >= 0$, the node is in Q2, and:

$$\theta_i = 180 - \sin^{-1}[(y_i - y_c)/h_i] \quad (6)$$

If: $x_i - x_c < 0$ and $y_i - y_c < 0$, the node is in Q3, and:

$$\theta_i = 180 - \sin^{-1}[(y_i - y_c)/h_i] \quad (7)$$

If: $x_i - x_c >= 0$ and $y_i - y_c < 0$, the node is in Q4, and:

$$\theta_i = 360 + \sin^{-1}[(y_i - y_c)/h_i] \quad (8)$$

Once the set of θ_i have been calculated the entire set of nodes is incrementally rotated about the geometric center by angle, φ_k (k = 0 to n-1), where k is the kth iteration out of n total iterations.

$$\varphi_k = (180/n)k \quad (9)$$

If k = 0, then $\varphi_k = 0$, and the set of nodes is in its original, non-rotated position. After each rotation, each node's new vector angle relative to the zero degree position in Figure 3 is $(\theta_i - \varphi_k)$, assuming clockwise rotation. New coordinates, x_iy_i in the original Euclidian plane can then be calculated for each rotated node:

$$x_i = h_i[\cos(\theta_i - \varphi_k)]x_c \quad (10)$$

$$y_i = h_i[\sin(\theta_i - \varphi_k)]y_c \quad (11)$$

Of course the distance, d_{ij} between any two rotated nodes is the same as the original, non-rotated distances:

$$d_{ij} = d_{ij} \quad (12)$$

And the tour distance value, v_k for the kth contiguous rotational iteration is:

$$v_k = \sum d_{ijk} \quad (13)$$

where i and j are determined by the final sort sequence which becomes known after step 7 in the ES algorithm. The best overall solution, v^* is then selected in step 8 of the ES algorithm as:

$$v^* = \min\{v_k\} \quad (14)$$

which occurs at a rotation angle of φ^*.

4. Results and Discussion

In a preliminary study, 30 sets of 10 nodes each were randomly generated to test the viability of the ES Algorithm. The true optimum tour was found in 14 of the cases with an average error of 0.68 percent. In this section additional results are presented on the performance of the ES algorithm compared to the performance of the NNG algorithm, but for larger problems. Instances of up to 76 nodes were selected from TSPLIB (Reinelt, 1995) for which the X-Y coordinates and the optimal or best tour were known and available. NNG and ES solutions were generated for each instance and compared to the TSPLIB tourlength. Overall results are presented in Table 3.

Table 3. Selected instances from TSPLIB with known X-Y coordinates and optimal solutions

TSPLIB Instance	Number of nodes	% over TSPLIB, NNG	% over TSPLIB, ES at $\varphi = 0$	φ^*, degrees	% over TSPLIB, ES at $\varphi = \varphi^*$
ulysses16	16	52.70	9.84	0.00	9.84
ulysses22	22	27.82	12.84	163.64	10.69
wi29	29	31.83	53.69	43.45	5.15
dj38	38	46.47	67.12	123.16	34.16
att48	48	20.89	118.55	127.50	28.01
berlin52	52	19.08	86.02	72.69	38.26
st70	70	19.34	80.11	146.57	58.66
pr76	76	41.89	122.08	61.58	60.14

It can be seen in Table 3 that the ES algorithm at φ^* performed better than the NNG algorithm for instances of 38 nodes, or less. For these four instances, rotation of the set of nodes from $\varphi = 0$ to $\varphi = \varphi^*$ improved the solution tour from an average of 35.87 percent over TSPLIB to 14.96 percent over TSPLIB. As might be expected, it can be seen in Table 3 that the error increases with the number of nodes. The data in the second and last columns in Table 3 can be used to estimate the expected accuracy of the ES algorithm, relative to TSPLIB, as a function of the number of nodes. Defining:

p = % over TSPLIB, ES at $\varphi = \varphi^*$
n = number of nodes

Simple linear regression generates the following equation with adjusted $R^2 = 0.89$.

$$p = 0.93(n) - 10.56 \qquad (15)$$

Obviously, the p value in (15) cannot be less than zero. The p-value is equal to zero when the n-value is 11.35. Given that the n-value must be integer, the error predictor equation is modified accordingly and approximated as:

$$p = \begin{cases} \sim 0 \text{ for } n <= 11 \\ 0.93(n) - 10.56 \text{ for } n >= 12 \end{cases} \qquad (16)$$

5. Conclusion

A new algorithm for solving the TSP has been developed and tested. The algorithm can be implemented with only a basic knowledge of trigonometry, Excel macros and VBA programming. The algorithm performs well for small problems with the error increasing linearly with the problem size. This decrease in accuracy is caused by the sort solution methodology which naturally searches for the outside perimeter defined on the graph of the nodes for all rotation angles. This can lead to excessive back and forth, zigzag travel in larger problems. Accuracy could likely be increased by dividing the complete set of nodes into more than two sets, allowing the sort procedure to delve more deeply into the interior region of the node graph with less overall travel within each set. It is expected that the error predictor equation (16) would hold within each pair of sets, even as the overall number of nodes in the instance increases. Connecting the pairs of sets into a complete tour would likely contribute additional error. This is a topic for future research.

6. References

Christophides, N. (1976).*Worst Case Analysis of a New Heuristic for the Travelling Salesman Problem. Report 388*. Graduate School of Industrial Administration, CMU.

Clarke, L., Johnson, E., Nemhauser, G.,& Zhongxi,Z. (1997). The Aircraft Rotation Problem. *Annals of Operations Research,* 69(0), 33-46.

Grotschel, M., Junger, M.,& Reinelt, G. (1991). Optimal Control of Plotting and Drilling Machines: A Case Study. *Mathematical Methods of Operations Research*, 35(1), 61-84.

Junger, M. G., Reinelt, G., & Rinaldi, G., (1995).The Travelling Salesman Problem. *Handbook in Operations Research and Management Science*, M. O. Ball, Magnant, T. L., Monma, C. L.,& G. L. Nemhauser (eds.). Amsterdam: North Holland.

Laporte, G. (1992). The Vehicle Routing Problem: An Overview of Exact and Approximate Algorithms. *European Journal of Operational Research,* 59(3), 345-58.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. J. R., & Shmoys, B. B. (eds.). (1985).*The Travelling Salesman: a Guided Tour of Combinatorial Optimization,*Chichester, England: J. Wiley and Sons.

Lenstra, J. K., Kan, A. H. J. R.,& Brucker P. (1977).Complexity of Machine Scheduling Problems*,Annals of Discrete Mathematics*. (343-62).Amsterdam: North Holland.

Lin, S.,& Kernighan, B. W. (1973).An Effective Heuristic Algorithm for the Travelling Salesman Problem.*Operations Research*,(21), 498-516.

Reeves, (ed.). (1993). *Modern Heuristic Techniques for Combinatorial Problems*. New York: Wiley & Sons.

Ratlif, H. D.,& Rosenthal, A.S. (1983). Order-Picking in a Rectangular Warehouse; A Solvable case for the Traveling Salesman Problem.*Operations Research*, 31(3), 507-21.

Rosenkrantz. D. J., Stearns, R. E.,& Lewis, P. M. (1977). An Analysis of several Heuristics for the Travelling Salesman Problem. *SIAM Journal of Computing,* 6 (5), 63-581.