MAPLE-BASED ALGORITHMS FOR THE TRAVELING SALESMAN PROBLEM (Continuation)

Bruno Guerrieri
Florida A&M University
bruno.guerrieri@famu.edu

Introduction:

This is a sequel to a paper titled "Maple-Based Traditional and Evolutionary Algorithms for the Traveling Salesman Problem" (TSProblem) [1]. The objective was to propose the traveling salesman problem as a basis for a series of investigations that students, working in groups, could conduct. This would have to take place either in a seminar form or under the umbrella of a Research Experience for Undergraduates such as the one we conducted over a period of three years at Florida A&M University. The primary objectives of the investigation was to introduce scientific mathematics majors to undergraduate research while attempting to stay within range of their capabilities, give them a different impression of mathematics that they experienced in the classroom and also show them the "implementation" of the mathematics to a computing environment such as Maple. It was an ambitious program whose weak point was the computer implementation. Students who had a good basis in programming could create a very satisfactory first approach, such as the Nearest-Neighbor Approach to solve the TSProblem. For these students to then be able to carry out something similar using another method of solution such as, say, the genetic algorithm or the branch-and-bound method turned out to be a little bit ambitious for a 8 week program, primarily because Maple was, after all, a new computing environment for them. Seduced by the multitude of approaches to solve this problem, the author, with help from various students from the program went on to create a series of modules, each introducing some very interesting mathematics. Students, at times, had to be heavily coaxed, because of the computing challenges but stayed on task because of the nature of the program and their desire to have a workable oral presentation on the final day of the 8-week workshop which was conducted the way a mathematical conference would. It did help that there were other such programs operating in conjunction with ours so that the conference involve at least twenty presentations of fifteen minutes each running in two concurrent sessions.

In the previous paper we quickly alluded to the Lin-Kernighan method of solution as well as simulated annealing, the Ant System approach, and the use of a genetic algorithm. In this paper we want to quickly refer to two other methods we investigated, namely the Branch-and-Bounds and the Hopfield-Tank approaches. We make no claim that what we present is up-to-date results. We simply wanted to end a combination of efforts to create modules suitable for use in situations such as the ones described above. The Maple routines that one may be interested in getting from us are basically operational but some of them need to be fine-tuned. Again, we do not pretend that they are optimized and, in certain situations, we may not be able to explain why a particular instance of one of the methods may not converge to an near-optimal solution. It is well-known that the neural implementation based on the Hopfield-Tank approach is having major difficulty in converging to a solution for a number of cities in the double digits. There have been attempts at trying to improve the situation in ways that we do not pursue. We feel that the method itself is interesting in its own right and is a good seed to plant into a willing student.

Goal:

We want to implement, using Maple as a platform, a Branch-and Bounds algorithm and a Neural-Network one that will provide optimal solutions for the TSProblem. Please refer to the previous paper for a review of the problem itself, the difficulty with obtaining an optimal solution and a small description of the characteristics of the different methods used to solve the TSProblem.

Branch-and-Bounds Method:

The method followed was taken from [2]. Basically one conducts a tree search. Starting with the weight matrix one starts with a process of reduction, a sort of lowering of the "water level". This process of reduction calls for subtracting a value, call it q, from a column or a row so as to create a zero in that row or column. Traversing the weight matrix in such a fashion as to stop only once on a given row and a given column and coming back to the point of departure is equivalent to performing a Hamiltonian tour (we are looking for the shortest one). In the process of reduction the relative costs of the different tours remain the same while all tours see their costs diminished by an amount q. One repeats the reduction process (subtracting a value from a given row or column until a zero is obtained and not creating any negative entries). The key point is that the total amount subtracted, $(\sum q_i)$, will be a lower bound on the cost of any solution. The second key point involves splitting the set of solutions into two classes about a particularly selected edge (left subtree containing the selected edge and right subtree not containing that edge), **namely the one that causes the greatest increase in the lower bound of the right subtree**. The left subtree node contains an updated weight matrix in which the pertinent row and column have been removed and several adjustments performed (see the reference for a clear detailing of the process). Reduction is repeated until termination and can be implemented in a recursive manner. In the best of circumstances the process is carried out $(N-2)$ times and one obtains the optimal tour. In general though, the process will not be as smooth and backtracking into the tree to investigate earlier right nodes with a lower value for the lower bound than the current one may be necessary. In the worse case scenario one will basically have to investigate all possible tours. It is important to have students working several examples by hand until they understand the mechanics of the process. They, then, stand a better chance to a have a good understanding of the process at a higher level and will go further in the programming implementation of the method.

Neural Network Approach:

In this case one must come up with a circuit, a set of neural interconnections that "represents" the problem. We think of the neurons in our artificial network as small amplifiers connected to one another through feedback circuits [4], a synapse being a connection between two amplifiers. Inhibiting and excitatory connections are established via amplifiers with negative and positive output voltages respectively. The behavior of such a circuit can be described by an equation that embodies the evolution of the circuit, namely

$$\frac{du_i}{dt} = -u_i + \sum_{j=1}^{N} T_{ij} V_j + I_i \qquad (1)$$

where N represents the number of neurons, u_i the input voltage over time of amplifier i, $T_{ij}V_j$ the sum of the output voltages from each of the other amplifiers, and I_i the current input to

the amplifier from the power supply. The neural network simulation is carried out by solving the above system of equations which will evolve to a steady-state if $T_{ij} = T_{ji}$ as was shown by Hopfield. At this steady-state, the energy of the system is at a minimum. Hopfield and Tank determined what would be the energy equation they could associate with the Traveling Salesman Problem and from that what would the evolutionary equations be. The structure of these equations then dictated what the circuit connections would be like. The energy equation, as set by Hopfield and Tank, is as follows:

$$E = \frac{a}{2}\sum_x\sum_i\sum_{i\neq j} V_{xi}V_{xj} + \frac{b}{2}\sum_i\sum_x\sum_{x\neq y} V_{xi}V_{yi} + \tag{2}$$
$$\frac{c}{2}(\sum_x\sum_i V_{xi} - N)^2 + \frac{d}{2}\sum_x\sum_{y\neq x}\sum_i d_{xy}V_{xi}(V_{y,i+1} + V_{y,i-1})$$

where V_{xi} is the output voltage of the neuron that represents city number x in position i. The first two terms ensure that the steady state represents the right kind of solution (one term in each row and one in each column), the third one ensures that we get the right number of terms in the final tour and the fourth one forces the solution toward shorter tours. Recall that we are trying to minimize the energy function. This minimization procedure gives rise to the system of differential equations to be solved. As mentioned before, it is the structure of that system that helps one determine the way in which the circuit should be built if one were to build one and run the experiment electronically. Otherwise, one can solve the problem numerically to determine the solution. The output presents itself in the form of a square matrix made of zeros and ones (after convergence) that indicate what the tour should be. This representation calls for an $n \times n$ matrix where n is the number of cities. In other words a TSProblem involving $n = 5$ cities will have $N = n^2 = 25$ neurons. The literature, subsequent to this Hopfield-Tank article, seems to indicate that convergence is not easy to establish unless one considers a really tiny number of cities. We have not pursued this avenue at this time.

Conclusion:

We hope that the totality of the methods presented in this paper and the previous one will give one many directions to investigate. It is interesting to see how the TSProblem is often referred to in the literature and how new algorithms involved with the subject matter are tested on this perennial problem. As indicated earlier, we had individual students working on the separate approaches. We would meet in groups as well as individually and the commonality of the subject matter was most useful. We will, of course, be very much willing to share any of our Maple worksheets with anyone interested provided they understand that these worksheets could stand improvements. The author is in the process of compiling while updating all the work done on the subject in a booklet to be published.

References:

[1] Maple-Based Traditional and Evolutionary Algorithms for the Traveling Salesman Problem, 14th Annual International Conference on Technology in Collegiate Mathematics, Addison-Wesley (2001)

[2] Discrete Optimization Algorithms. M. M. Syslo, N. Deo, J. S. Kowalik, Prentice-Hall, 1983.

[3] Neurons, Analog Circuits, and the Traveling Salesperson, Bev and Bill Thompson, AI EXPERT, July 1987.

[4] Neural Computation of Decisions in Optimization Problems, J.J. Hopfield and D.W. Tank, Biological Cybernetics, Vol. 52, 1985, pg 141.