Use of a Colony of Cooperating Agents and MAPLE to Solve the Traveling Salesman Problem

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

Abstract

A new approach for finding optimal solutions to the Traveling Salesman Problem is being reviewed and implemented using MAPLE. The method draws from the way ant colonies manage to establish shortest route paths from their colonies to feeding sources and back. We will compare this method to more traditional methods such as the Lin-Kernighan approach and to other "evolutionary" methods such as simulated annealing and genetic algorithms.

Introduction:

The Traveling Salesman Problem (TSProblem) is a well-known problem in combinatorial optimization. Simply stated it asks the following: given n cities, what is the shortest complete tour possible for a person wanting to visit all cities once and return to the initial city? There are, at least, two versions of this problem, the symmetric TSP where the distances (which could represent geometric distances or cost or some other measure) are the same whether one travels from city A to city B or vice-versa. The non-symmetric case would be the one where the distances are dependent on the direction of travel along the edges joining the different cities. [1] gives some examples of the different variations which center around the original TSProblem. Numerous approaches for solution, interesting in that they are truly independent approaches, abound. In fact, one can locate on the Internet, data sets of cities (some of them containing thousands of cities) which any new algorithm can use as comparative data. One can measure his/her algorithm's effectiveness against a list of top contenders. It is a rich environment where some approaches are good at solving one specific type of TSProblem, while others, perhaps not as speedy, are more robust and adapt to wider requirements. For instance the Ant System will work well in both the symmetric and non-symmetric case. Keep in mind that an exhaustive search for the minimum path would require $\frac{1}{2}(n-1)!$ (15 cities $\rightarrow \frac{1}{2}(14)! = 4.35 \times 10^{10}$ cases) computational steps. The number of steps grow faster than the number of cities raised to any finite power and thus becomes unmanageable very quickly. All practical methods are heuristic in nature and will usually provide an optimal path rather than a global minimum. [2,3,4] are several good places to start investigating if you are interested in the TSProblem.

Goal:

We wanted to implement, using Maple as a platform, an "evolutionary" algorithm that articles [4,5] referred to as an Ant System. This is part of an effort to introduce undergraduate students to the TSProblem and the different methods of solution available. This is in the context of the development of a special course, opened to motivated students in search of research experiences that help them prepare oral presentations at conferences (such as an undergraduate research symposium) and write undergraduate research papers. We felt that the students would be more likely to become "engaged" if a connection with the real world was forthcoming. Several such approaches, for instance use of

simulated annealing, neural networks, genetic algorithms, are appealing because they attempt to solve complex problems by incorporating in their mode of solutions processes which are observed at work in the world around us. They offer excellent entry points for curious students who are, then, more likely to investigate well-known algorithms such as the Lin-Kernighan approach [6] for solving the TSProblem. In the case of ants, although individual interactions may be simple (one ant following the chemical scent of another), together they can determine the shortest route to a food source.

Definitions:

n represents the number of cities.

m represents the number of ants. Contrary to our initial reaction we will in fact use only $m = n$ ants, starting the process with one ant per city.

$d_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{\frac{1}{2}}$ represents the Euclidian distance between cities i and j, i.e. edge (i,j). The information is kept in the form of a distance matrix W.

$\eta_{ij} = \frac{1}{d_{ij}}$ represents the visibility on edge (i,j). This quantity will be raised to power β which we will use to control the degree of visibility. This means that we are letting a given (artificial) ant make some local decisions and possibly opt for the nearest city at a given step in the process.

τ_{ij} represents the intensity of the trail on edge (i,j). This is a measure of the amount of pheromone, the chemical marker deposited on the trail by each ant. The higher the level of pheromone on a given edge, the more likely a given ant will follow that edge. This quantity will be raised to power α which we will use to control the reliance on scent. The way in which this quantity will be handled in our process is not fully intuitive. In fact, let us explain how time will be handled in our simulation. At each time step, the m ants, in unison, will travel to the next allowable city (the choice is based on a stochastic decision). That part of the simulation takes place over a period of n steps, at the end of which each ant will have completed a tour. At that moment (at moments $n, 2n, 3n, \ldots$ when cycles will have been completed) the trails' scent level will be updated. The update rule for the pheromone levels is $\tau_{ij}(t+n) = \rho\tau_{ij}(t) + \Delta\tau_{ij}$ where $\rho < 1$ is such that $(1-\rho)$ represents the evaporation of scent between time t and $t+n$ and where $\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^k$ measures the additional trail traffic with

$$\Delta\tau_{ij}^k = \left\{ \begin{array}{ll} \frac{Q}{L_k} & \text{if } k\text{-th ant uses edge } (i,j) \\ 0 & \text{otherwise} \end{array} \right\}$$

Q is a constant and L_k is the tour length of the k-th ant so that, the shorter the tour, the more the chemical reinforcement.

$$p_{ij}^k(t) = \left\{ \begin{array}{ll} \frac{[\tau_{ij}(t)]^\alpha [\eta_{ij}(t)]^\beta}{\sum_{k \in allowed} [\tau_{ik}(t)]^\alpha [\eta_{ik}(t)]^\beta} & \text{if edge } (i,j) \text{ was not previously visited by ant } k \\ 0 & \text{otherwise} \end{array} \right\}$$

represents the transition probability. This means that, at each time step t, ant k will determine, in a probabilistic fashion, which city it will visit next. This probability is based somewhat on the distances to the nearest cities that it has not visited yet as well as on the amount of pheromone present at that moment on the different allowed edges. Therefore the transition probability is a trade-off between visibility (which says that close towns should be chosen with high probability, thus implementing a greedy constructive heuristic) and trail

intensity at time t (that says that edge (i,j) is highly desirable if it has seen a lot of traffic), thus implementing the autocatalytic process. An autocatalytic, i.e. positive feedback, process is a process that reinforces itself, in a way that causes very rapid convergence and, if no limitation mechanism exists, leads to explosion!

The central data structure, in this algorithm, is an array referred to as $tabu_k$. It contains, for ant k, a growing list of all the cities it has already visited.

Algorithm:

A detailed description of the algorithm can be found in [6] and a MAPLE program attempting to implement the algorithm is available by sending an e-mail to bruno.guerrieri@famu.edu. In a nutshell, think of n ants (one per city) taking a day (figuratively speaking) to go to the next city in their own tour. That initial group of ants will find all scent levels to be the same on all trails, set at some initial minimal non-zero value. Let us also assume, for sake of argument, that there are 30 cities. Each morning, the ants roll a weighted die (taking into account proximity and scent level) to determine the next (new) city they will visit. Records of ant choices, city visited, trail scent levels are rigorously kept. At the end of the first month the first cycle is completed and all visited trails are ”chemically” updated. At the beginning of the new month, a new set of n ants starts a new cycle, the difference being that trail levels are not uniform (meaning the same on all trails) anymore. The above process is repeated month after month for a given number of cycles.

Conclusion:

We are now trying to adjust the different parameters (in particular α, β, ρ, Q mentioned above) to speed up the convergence process to its optimal answer. A thorough investigation of the parameter space would call attention to bad solutions and stagnation situations where all ants take the same tour. There is, in fact, a multitude of quantities and situations to be investigated that could sustain efforts carried out by different students. We mentioned the study of the parameter space. One can also investigate and see if ”daily” update of scent is superior to ”monthly” update, not to mention the fact that the TSPproblem has so many variations itself, for instance the unsymmetric case or the constrained one [7]. Concerning the speed of convergence, we are still in the analysis phase of this algorithm not having been able to investigate the parameter space as thoroughly as we would have liked to (we hope to entice more students!). So, our rendition of the algorithm is not competitive yet with approaches using the Lin-Kernighan or simulated annealing methods. We should state that comparisons based solely on speed of execution are not always fair because the ”robustness” of a method is also an important factor. By robustness, we mean minor changes in the algorithm allow it to ”solve” variations of the TSPproblem and perhaps even address other combinatorial optimization problems altogether. More importantly, moving away from the traditional TSPproblem where all the initial information is static, we are now in search of algorithms that will face the same complexity and, in addition, will need to adapt to varying situations as time elapses. For instance, the constant updating of a pheromone trail enables real ants to adapt to changes in the environment. It would be interesting to compare how gracefully the Lin-Kernighan algorithm and the Ant System would recover when, deep into the solution process, a ”deus ex machina” traffic overloading were to develop on some of the edges.

References:

[1] The Traveling Salesman Problem: PCBs, Punch Presses... and Pachinko, M.

Kobayashi, S. Misono, K. Iwano, SIAM News, Vol. 27, No. 10, December 1994.

[2] TSPBIB Home Page, http://www.densis.fee.unicamp.br/~moscato/TSPBIB_home.html.

[3] The Traveling Salesman Problem, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys, Wiley-Interscience Series in Discrete Mathematics, 1985.

[4] Swarm Smarts, E. Bonabeau and G. Theraulaz, Scientific American, March 2000, pp. 73-79.

[5] The Ant System: Optimization by a Colony of Cooperating Agents, M. Dorigo, V. Maniezzo, A. Colorni, IEEE Transactions on Systems, Man, and Cybernetics-PartB, Vol. 26, No. 1, 1996, pp. 1-13.

[6] An Effective Heuristic Algorithm for the Traveling-Salesman Problem, S. Lin and B.W. Kernighan, Operations Research, 21, 1973, pp. 498-516.

[7] Numerical Recipes in C (2nd Ed.), W.H. Press, S.A. Teukolsy, W.T. Vetterling, B.P. Flannery, Cambridge University Press, Chapter 10, Section 10.9 on Simulated Annealing Methods. This topic can also be found in other Numerical Recipes books written by these authors.