

# USING ADAPTIVE RESONANCE THEORY AND LOCAL OPTIMIZATION TO DIVIDE AND CONQUER LARGE SCALE TRAVELING SALESMAN PROBLEMS

*Samuel A. Mulder*

Dept. of Computer Science  
University of Missouri-Rolla  
Rolla, MO 65401  
[smulder@umr.edu](mailto:smulder@umr.edu)

*Donald C. Wunsch II*

Dept. of Electrical and Computer Eng.  
University of Missouri-Rolla  
Rolla, MO 65401  
[dwunsch@ece.umr.edu](mailto:dwunsch@ece.umr.edu)

**ABSTRACT** - The Traveling Salesman Problem (TSP) is a very hard optimization problem in the field of operations research. It has been shown to be NP-complete, and is an often-used benchmark for new optimization techniques. One of the main challenges with this problem is that standard, non-AI heuristic approaches such as the Lin-Kernighan algorithm (LK) and the chained LK variant are currently very effective and in wide use for the common fully connected, Euclidean variant that is considered here. This paper presents an algorithm that uses adaptive resonance theory (ART) in combination with a variation of the Lin-Kernighan local optimization algorithm to solve very large instances of the TSP. The primary advantage of this algorithm over traditional LK and chained-LK approaches is the increased scalability and parallelism allowed by the divide-and-conquer clustering paradigm. Tours obtained by the algorithm are lower quality, but scaling is much better and there is a high potential for increasing performance using parallel hardware.

## 1. INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the most studied problems in computer science literature. It is an example of the important class of problems known as NP-complete problems. An NP-complete problem is one that is solvable in polynomial time by a non-deterministic algorithm, but not necessarily by a deterministic one. Another way of considering this class of problems is to say that a correct solution may be checked in polynomial time, but no known algorithm can solve the problem in that time. These problems are interesting because it is possible to draw parallels between any two NP-complete problems and show that finding an algorithm to solve one in polynomial time gives you an algorithm to solve the other.

They are also interesting because it has never been proven that NP-complete problems cannot be solved in polynomial time, so it remains as an open question. The most general form of the TSP is to find a Hamiltonian cycle given an arbitrary graph. All known algorithms to solve this problem take greater than polynomial time, but if we have a solution we can check it in  $O(n)$  time. The more specific case of the TSP considered in this paper is also a member of NP-complete, but is more complicated to check.

The Traveling Salesman family of problems is an area in which neural networks have so far been unable to compete with the best non-neural approaches, in accuracy, speed, or scaling. In fact, it seems doubtful whether neural networks will ever approach the accuracy of algorithms like the Lin-Kernighan solution [1] in solving this class of problems. The problem domain is well enough understood, and the problem presents enough information about each instance that the generalization abilities of neural networks are not as useful as with many classes of optimization problems. In addition, Clustering algorithms in general are limited in the accuracy they can achieve on the TSP. If highly accurate tours are required, then Lin-Kernighan and variants [2] seem likely to remain the algorithms of choice. In situations where accuracy is not the primary determinant, however, neural networks may show some promise. This is especially relevant for situations where significant modifications to [1] and [2] may be required, as opposed to minor modifications and retraining for a neural network.

Our goal then is to evaluate what advantages a neural clustering algorithm can bring to bear on the problem and decide how to best exploit those advantages. Adaptive resonance theory provides a very rapid and effective neural clustering model. Functionally, it operates similarly to k-means clustering, with an optimal k determined dynamically by the vigilance factor. The divide and conquer paradigm gives us the flexibility to hierarchically break large problems into arbitrarily small clusters depending on what trade-off between accuracy and speed is desired. In addition, the sub-problems provide an excellent opportunity to take advantage of parallel systems for further optimization. Even without parallel processing, the algorithm developed in this paper has demonstrated that better scaling is possible using a divide and conquer approach.

#### A. Traveling Salesman Problem

Given a complete undirected graph  $G = (V, E)$ , where  $V$  is a set of vertices and  $E$  is a set of edges each relating two vertices with an associated non-negative integer costs  $c(u, v)$ , the most general form of the traveling salesman problem is equivalent to finding any Hamiltonian cycle over  $G$  where such a cycle is known as a tour. The more common form of the problem is the optimization problem of trying to find the shortest Hamiltonian cycle. Both of these problems have been proven to be NP-complete in [3]. These problems are very useful to consider because they map so easily to many real-world applications in a wide range of fields from network routing to cryptography [4-7].

The variation of the TSP that we consider is even more limited. We look only at graphs that can be mapped to a two-dimensional Euclidean coordinate system where the edge weight between two nodes is the distance between them. This system implies that the triangle inequality holds such that  $c(u, w) \leq c(u, v) + c(v, w)$ . We also require that the XY coordinates of each node be given. This variation can be shown to still be NP-complete and therefore to map onto the more general problem [8]. The Euclidean form of the TSP has been widely studied [9-11].

#### B. Adaptive Resonance Theory

Adaptive Resonance Theory was first introduced by Carpenter and Grossberg [12]. The unsupervised variants provide a simple, effective neural clustering algorithm. The original algorithm, designated ART1, used binary input sequences. The variation developed for this solution uses two-input integer sequences where the integers are the XY coordinates of a node. In this form, ART is

functionally similar to k-means clustering except that k is increased dynamically as new patterns are introduced.

#### C. The Lin-Kernighan Local Optimization Algorithm

The current best results on large-scale TSP instances come from variations on an algorithm proposed by Lin and Kernighan in 1973 [1]. This algorithm takes a randomly selected tour and optimizes it by stages until a local minimum is found. This result is saved and a new random tour selected to begin the process again. Given enough time, the optimal tour can be found for any instance of the problem, although in practice this is limited to small instances for an absolute result due to time constraints. Even with extremely large instances, however, reasonably short tours can be found [1].

The optimizations performed by the Lin-Kernighan (LK) algorithm use the concept of  $\lambda$ -optimality. If  $n$  is the number of cities in a TSP, then we know that given any tour we can reach an optimal tour by swapping at most  $n$  edges. Using  $\lambda$ -optimality, we consider the optimal tour that can be reached by swapping only  $\lambda$  edges. As  $\lambda$  grows, the chance that the  $\lambda$ -optimal tour is actually optimal increases. Unfortunately, the processing time required to find  $\lambda$ -optimal tours is intractable for large  $\lambda$ . The largest commonly used values for  $\lambda$  are 2 and 3, although some attempts have been made with  $\lambda$  as 4 or 5 [1].

The algorithm proposed by Lin and Kernighan examines variable  $\lambda$  optimization. In brief, an initial 2 edge swap is chosen. The algorithm then examines whether it would be profitable to make a 3 edge swap, and continues increasing the number of edges as long as the total swap remains an improvement over the initial tour. This process is repeated with each node being considered in turn as a starting point for the initial 2 edge swap until no further improvements can be found. The original algorithm then started with a different random tour and worked from the beginning, always saving the best tour seen. This original algorithm is what is used in our current algorithm. Applying more advanced variants of LK will likely yield even better results.

## 2. ALGORITHM – DIVIDE AND CONQUER

The algorithm developed in this paper combines ART and the LK local optimization algorithm to divide and conquer instances of the TSP. We begin by reading in the cities, stored in TSP-LIB format [13]. The ordering of the cities in memory represents the current tour at any time. This is known as a permutation representation and obviously only works with fully connected variants of the

TSP, as an arbitrary permutation of a non-fully connected graph would not necessarily represent a valid tour. This permutation representation is chosen because weight-matrix representations become intractable for large values of  $n$ , i.e. a 250k-city problem would require around 62.5 GB of memory to store a weight-matrix. Since the problem is Euclidean, it is sufficient to store  $(x,y)$  coordinates for each city and calculate distances on the fly.

The first stage of the algorithm involves sorting the cities into clusters using the ART algorithm described previously. Our variation of ART uses the vigilance parameter to set a maximum distance from the current pattern. A vigilance parameter between 0 and 1 is considered and used as a percentage of the global space to determine the vigilance distance. Values were chosen based on the number and size of individual clusters desired, but typical values ranged from 0.80 to 0.97. The learning rate was set to 0.02. The clusters at this point were still in a random order. The individual clusters were then each passed to a version of the LK algorithm also described above. Since the size of the tours was controlled and kept under a thousand cities, we allowed the LK search depth to be infinite as long as the total improvement for a given swap series remained positive. We also did not restrict the neighborhood size as specified in the original algorithm. This required more time than a limited depth search, but resulted in better overall tour quality. After a first pass through the LK algorithm, a simple intersection removal algorithm was applied. This algorithm is based on the idea that any tour containing an intersection between two edges is demonstrably sub-optimal. Typically after LK is run, few intersections remain, but those that do remain often involve edges crossing large distances. Swapping these edges changes the global search space sufficiently that a new run of LK can often find additional improvement. In short, the intersection removal algorithm is capable of making double-bridge swaps that the LK algorithm is unable to discover. This double-bridge property is the same as that used in the chained LK algorithms described in [2].

After the second pass of the LK algorithm, the given cluster is then merged into the final tour. The first cluster is simply added to an empty tour and each additional cluster is merged. This is accomplished by a simple method of choosing the closest set of two vertices in the new cluster to any two adjacent vertices in the final tour. In order to avoid a global  $O(n^2)$  process, each vertex in the existing final tour is compared to the centroid of the new cluster. The closest  $k$  vertices are chosen and then compared to each vertex in the new cluster. Typical values of  $k$  used were around 100. This kept the

computational complexity of this step down to  $O(n)$  and added very little to the overall running time.

Table 1 – Algorithm Pseudocode

Read data from file
Apply ART algorithm to cluster data
For each cluster
Apply LK optimization
Apply intersection removal
Apply LK optimization
Merge with final tour
End for
Save final tour and time

### 3. RESULTS

The results from our experiments were very positive. In the tables below we compare our clustered variation of the LK algorithm to the original LK algorithm. The LK algorithm used for these comparisons is that same algorithm that we implemented and used on the clusters of our main algorithm. Since the focus of these experiments was to determine the effects of clustering, the LK implementation is not heavily optimized and the running times reflect are only to be used in comparison to each other.

Table 2 - Lin-Kernighan algorithm implemented by author

TSP Size	Tour Length	Time
1000	2.63808e7	5 seconds
2000	3.64058e7	21 seconds
4000	5.07807e7	82 seconds
20000	1.12221e8	4814 seconds

Table 3 – Clustered Lin-Kernighan

TSP Size	Tour Length	Time	% off LK
1000	2.69785e7	1 second	2%
2000	3.80334e7	3 seconds	4%
4000	5.35437e7	8 seconds	5%
20000	1.17e8	98 seconds	4%

Obviously the clustered variation is scaling much better than the original, and tour quality is remaining within 5% or so. Also note that with the clustered version we can smoothly trade time for quality in either direction by adjusting the vigilance factor. Since our implementation of the original LK algorithm fails to scale past a few tens of thousands of cities, largely due to the fact that we used an unlimited neighborhood instead of the finite neighborhood called for in the original algorithm, we compare our clustered algorithm to the well-known Concorde software package. Concorde is widely regarded as the fastest TSP solver for large instances currently in existence. Concorde is described in more detail in [14]. Essentially, it uses the chained LK algorithm. Search depth for LK is greatly limited, but high quality is obtained by repeatedly applying double-bridge

transformations and optimizing. The Concorde package also applies a simple algorithm to generate an initial tour of fair quality before beginning that provides a large speed boost to the LK algorithm. These techniques could also be implemented in our clustered variation giving similar speed and quality improvements. For now we are just interested in the scalability of the relative algorithms.

In Table 5, we can clearly see that while our algorithm is below the quality of tour that the chained-LK algorithm produces, it is scaling nicely. (Our quality-of-result numbers, reported in the last column of table 5, are comparable with previously published neural net results for much smaller TSP problems, see [15].) In fact, our algorithm gives us additional flexibility in terms of changing the vigilance factor to generate different numbers of clusters. To illustrate this, Table 6 shows the results of the 250k city problem as the vigilance parameter is adjusted.

Table 4 – Concorde Chained LK

TSP Size	Tour Length	Time
10000	7.20532e7	38 seconds
20000	1.01468e8	85 seconds
250000	3.58274e8	1380 seconds

Table 5 – Clustered Lin-Kernighan

TSP Size	Tour Length	Time	% off CLK
10000	8.21277e8	57 seconds	14%
20000	1.17e8	98 seconds	15%
250000	4.27169e8	693 seconds	19%

Table 6 – Clustered Lin-Kernighan 250k city problem

Vigilance	Tour Length	Time	% off CLK
0.95	4.09444e8	2451 seconds	14%
0.96	4.18222e8	1077 seconds	17%
0.97	4.27169e8	693 seconds	19%
0.98	4.47744e8	778 seconds	25%

As is clear from the table, increasing the vigilance parameter increases the speed at the expense of tour quality. We also observe the limit of 0.97 for this particular problem past which the time begins to increase instead of decrease. This is the point where the individual constant factors involved with processing each cluster begin to outweigh the advantages of having smaller clusters. The actual vigilance parameter chosen depends largely on the quality of tour needed and the time available.

#### 4. CONCLUSIONS

The main result shown in this paper is the potential of combining neural clustering with traditional local search techniques. A significant speedup was shown by using clustering in the original Lin-Kernighan algorithm. This

speedup was offset by a 2%-5% loss in tour quality that is the result of the inherent limitations of clustering when applied to the TSP. Clustering approaches will typically be unable to locate a global minimum because they assume a certain structure on the underlying data that may be sub-optimal. In most real-world situations, however, finding the absolute global minimum is not a requirement. Most applications prefer speed to tour quality within limitations. While our current implementation does not beat the latest chained-LK techniques with heavy optimization, it does scale at least as well, and gives additional flexibility when trading tour quality for speed. In addition, since the number of global operations is limited in our algorithm, it should scale smoothly with parallel hardware and show significant improvement over current solutions.

#### 5. REFERENCES

- [1] S. Lin, B. W. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem", *Operations Research* 21, 1973, 498-516.
- [2] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for large traveling salesman problems" Technical report, 2000. [http://www.keck.caam.rice.edu/reports/chained\\_lk.ps](http://www.keck.caam.rice.edu/reports/chained_lk.ps)
- [3] T. Cormen, C. Leiserson, R. Rivest. *Introduction to Algorithms*, MIT Press, 1996, 954-960.
- [4] R. Agarwala, D.L. Applegate, D. Maglott, G.D. Schuler, A.A. Schaffler. "A Fast and Scalable Radiation Hybrid Map Construction and Integration Strategy", <http://www.ncbi.nlm.nih.gov/genome/rhmap/>, 2000.
- [5] C. A. Bailey, T. W. McLain, R. W. Beard, "Fuel Saving Strategies for Dual Spacecraft Interferometry Missions," *Journal of the Astronautical Science*, (to appear).
- [6] J. Turino. "ASIC DFT and BIST Alternatives", <http://www.chipcenter.com/asic/tm004.html>, 2002.
- [7] *Worldwide Airport Finder*. <http://www.wapf.com/>, 2002. [8] C.H. Papadimitriou. "The Euclidean Traveling Salesman Problem is NP-Complete", *Theoretical Computer Science* 4, 1977, 237-244.
- [9] M.L. Braun, J.M. Buhmann. "The Noisy Euclidean Traveling Salesman Problem and Learning", *Advances in Neural Information Processing Systems* 14. MIT Press, Cambridge, MA, 2002.
- [10] E.M. Cochrane, J.M. Cochrane, "Exploring competition and co-operation for solving the Euclidean Traveling Salesman Problem by using the Self-Organizing Map." *Artificial Neural Networks*, 7-10 September 1999. 180 – 185, Volume:1.
- [11] S. Aurora. "Polynomial time approximation schemes for {Euclidean} traveling salesman and other geometric problems", *Journal of the ACM* V.45 N.5, 1998, 753-782.
- [12] G. Carpenter, S. Grossberg. "The art of adaptive pattern recognition by a self-organizing neural network", *IEEE Computer*, March 1988, 47-88.
- [13] P. Moscoto. "TSPBIB Home Page". [http://www.densis.fee.unicamp.br/~moscato/TSPBIB\\_home.html](http://www.densis.fee.unicamp.br/~moscato/TSPBIB_home.html), 2002.
- [14] D. Applegate, R. Bixby, V. Chvatal, W. Cook. "Concorde – a code for solving Traveling Salesman Problems". <http://www.math.princeton.edu/tsp/concorde.html>. 2001.
- [15] N. Vishwanathan, D. Wunsch. "A Hybrid Approach to the TSP", *IEEE International Joint Conference on Neural Networks*, Washington, DC, July 2001.