FLEXMAP – A Neural Network For The Traveling Salesman Problem With Linear Time And Space Complexity

Bernd FRITZKE and Peter WILKE

Universität Erlangen-Nürnberg
Lehrstuhl für Programmiersprachen
Martensstr. 3
D-8520 Erlangen
Germany

Tel. (++49) +9131/857830
Fax (++49) +9131/39388
fritzke@immd2.informatik.uni-erlangen.de

Abstract

We present a self-organizing "neural" network for the traveling salesman problem. It is partly based on the model of Kohonen. Our approach differs from former work in this direction as no ring structure with a fixed number of elements is used. Instead a small initial structure is enlarged during a distribution process. This allows us to replace the central search step, which normally needs time $O(n)$, by a local procedure that needs time $O(1)$. Since the total number of search steps we have to perform is $O(n)$ the runtime of our model scales linear with problem size. This is better than every known neural or conventional algorithm. The path lengths of the generated solutions are less than 9 percent longer than the optimum solutions of solved problems from the literature.

1 Introduction

The traveling salesman problem (TSP) is perhaps the most popular problem of combinatorial optimization. One variant is the Euclidean TSP: given n cities described by their coordinates in the plane, what is the shortest round trip that visits each city exactly once?

Since the TSP belongs to the class of $\mathcal{NP}$-complete problems, optimum solutions can not be guaranteed to be found in polynomial time. Thus a number of heuristics have been proposed to find approximate solutions[11,12]. Recently also "neural" approaches based on different kinds of collective computation have gained interest[1,2,8].

The best of these algorithms show approximately quadratic time complexity. E.g. the well known algorithm of Lin & Kernighan[11] scales with $n^{2.2}$. This makes it difficult to treat large problems which occur in many practical applications (e.g. steering a drilling machine).

In this contribution we present a new algorithm, denoted as FLEXMAP, which is linear in time and space and produces solutions within 9 percent from optimum for all test problems it was applied to.

2 The Growing Ring Structure

The central element of our concept is a ring of cells. Initially it consists only of three cells. The ring is placed on the plane by attaching a two-dimensional position vector pos(q) to every cell q. Successively new cells are inserted. Before a new cell is inserted, we distribute the current structure such that:

a) The cell positions are near to the city positions.

b) Neighboring cells in the structure have similar positions.

Fig. 1 shows an example for a small TSP. The initial structure can be seen on the left. New inserted cells are shown as white circles. A valid tour is found, when every city has a different nearest cell in the plane.

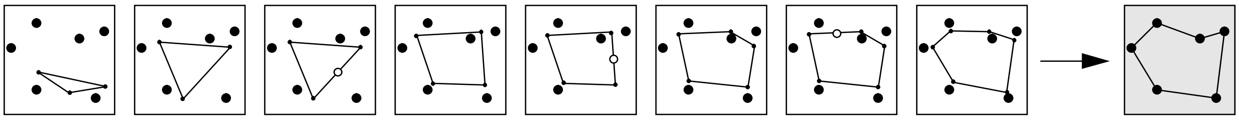

Figure 1: Development of a ring structure for a 6 city TSP

The cities are then visited in the order of their associated nearest cells in the ring structure.

Now we have to answer two questions:

- How can we distribute the ring structure such that the conditions a) and b) are satisfied?
- Where should new cells be inserted?

Distribution. A distribution step is performed as follows (see also fig. 2):

- Choose a city c at random.
- Determine the cell bmu (best matching unit) lying nearest to c in the plane.
- Move bmu and its two direct neighbors in the ring structure towards c by a fraction of the total distance ε_{bmu} resp. $\varepsilon_{neighbor}$.

This is very similar to the adaption steps taking place in the model of Kohonen[9]. The difference is, that Kohonen uses a decreasing neighborhood and also decreasing adaption parameters whereas we use only neighborhood of degree one and fixed parameters. However, if we choose those parameters sufficiently small we can transfer his result that after some distribution steps the systems gets into an equilibrium state with both conditions a) and b) being satisfied rather well and not much cell movement taking place anymore.

Insertion. During the distribution we can compile information about where to insert the next cell. Each time when a cell is bmu for the current city c we add its distance from the current city to a local error variable:

$$err(bmu) := err(bmu) + ||pos(bmu) - pos(c)||$$

Such an error variable is attached to each cell. A high value of this variable indicates, that the cell lies in a region of the plane, where the cell/city-ratio is low. New cells should be inserted in such regions, since for a valid solution every city must have a different nearest cell.

The complete algorithm is shown in fig. 3. Due to the condition in step 3 a solution can be found at the earliest when the ring structure has reached size n. However, in all our simulations $2n$ to $3n$ cells had to be generated, until the tour was found. Considering this empiric result, that the number of cells and therefore also the number of iterations over step 1-3 is $O(n)$, it is easy to determine the overall complexity of the algorithm. Step 1 requires the inspection of all cells to find the bmu for the current city. This is done $n_{distribution}$ times and thus step 1 requires $O(n_{distribution}n) = O(n)$ since $n_{distribution}$ is a constant. In step 2

Figure 2: Example of a distribution step

Init Start with a ring structure consisting of three cells at random positions. Initialize the error variables to 0.0.

Step 1 Perform a constant number $n_{distribution}$ of distribution steps. At every step update the error variable of the *bmu*.

Step 2 Find the edge e_{worst} connecting two cells v_1 and v_2, such that the *edge error*

$$err(e_{worst}) := err(v_1) + err(v_2)$$

is maximum for all edges in the structure. Insert a new cell c_{new} in the center of e_{worst}. Initialize the error variable of the new cell as

$$err(c_{new}) := \frac{1}{3}\,err(v_1) + \frac{1}{3}\,err(v_2)$$

Diminish the error variables of v_1 and v_2 such that the total error of v_1, v_2 and c_{new} stays constant:

$$err(v_1) := \frac{2}{3}\,err(v_1), \qquad err(v_2) := \frac{2}{3}\,err(v_2)$$

Step 3 If for every two cities the associated nearest cells are different, a solution has been found. Otherwise continue with Step 1.

Figure 3: The O(n^2) algorithm for the growing ring structure

every edge has to be visited to find the edge with maximum error. The number of edges is equal to the number of cells and so this requires time O(n). In step 3 for every city the nearest cell has to be found which requires (at least) time O(n).

Since each of step 1-3 requires (at least) O(n) and this loop is repeated O(n) times the time complexity of the described algorithm is (at least) O(n^2). The space requirement is O(n) since we only have to reserve storage for the n cities, the O(n) cells and some local variables.

3 Making the Algorithm Linear

To improve on the quadratic time behaviour of the algorithm we have to replace each of step 1-3 by a faster procedure. Graphical visualization of our algorithm shows, that in the beginning the ring structure is distributed quickly over the whole area and then changes become more local as the number of cells increases. This behavior lead to the idea to replace the global search for the *bmu* in step 1 with the following local approximation:

For every city we keep a reference to that cell, which most recently was its *bmu*. When the city is chosen another time during a distribution step the search for the *bmu* is restricted to the referenced cell and a neighborhood up to degree $k_{neighbor}$ in the ring structure (see fig. 4). Thereby is $k_{neighbor}$ a constant and thus this search can be done in O(1).

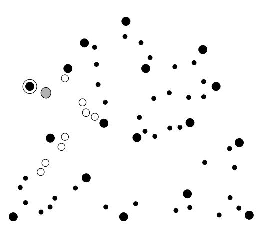

Figure 4: Local search for the best matching unit: the previous *bmu* is shown as shaded circle and the local neighbors (here up to degree 4) as white circles

To avoid the linear search in step 2 for the edge with maximum error, we consider the fact that only those edges get a high error value which connect cells that are often *bmu*. We therefore restrict the search for e_{worst} to those edges that are adjacent to a *bmu* in one of the distribution steps in step 1. Since there are only $n_{distribution}$ such distribution steps this can be done in constant time. Obviously we can not guarantee to find always the edge with maximum error but we are likely to find an edge with a high error value.

The check in step 3 whether a solution has been found can be made obsolete by the following modification of our algorithm. When the current *bmu* has been *bmu* for the same city c for a certain number of times, we assume that for this city the mapping on the ring structure has stabilized and "pin" the cell onto the city. The cell is placed on the exact position of the city and is not moved anymore by following distribution steps. The city is removed from the set of "free" cities used for distribution steps. When no free city is left a solution has been found. Book keeping for the *bmu* and pinning can be done in constant time and thus step 3 has been replaced by an O(1) operation.

Now there are only constant time operations left in a loop which has to be executed $O(n)$ times. This means that we can generate approximate solutions to traveling salesman problems in time $O(n)$.

4 Linear Tour Improvement

An improvement of the generated solutions can often be achieved by the following local heuristic:

> For every sequence of k cities in the solution tour compute all ($k!$) permutations of these cities and replace the actual sequence by that permutation which leads to the shortest tour.

Obviously only for small values of k this is feasible since $k = n$ would result in a complete enumeration of the TSP. We usually choosed k = 4 thereby improving our solutions by 0 to 2 percent. Since for a fixed k the time needed for one sequence is O(1) and there are n such sequences, the time complexity of this heuristic is linear.

5 Simulation Results

Time is only one criterion, the other one is the quality of the generated solutions (i.e. path length). To find out, whether the choice of the computational short cuts is justified and does not lead to bad solutions, a number of simulations were performed.

In the following we denote as FLEXMAP the algorithm consisting of the linear tour construction procedure and the described linear tour improvement procedure. All implementations have been done in C++ on a risc work station (Sun SPARCstation 2). The ring structure was realized as a doubly linked list.

We performed experiments with the following problems from the literature for which optimum solutions are known (the column FLEXMAP indicates the best solution found by our algorithm):

name	ref.	cities	optimum	FLEXMAP
HT30	[8]	30	424	424
EIL08	[3]	51	426	434
EIL10	[3]	101	629	642
KRO30	[10]	150	26524	27293
KRO32	[10]	200	29368	30315

name	ref.	cities	optimum	FLEXMAP
LK318	[11]	318	42029	43941
PCB442	[13]	442	50779	53475
ATT532	[12]	532	27686	29294
TK1002	[13]	1002	259045	277152
TK2392	[13]	2392	378032	411147

Furthermore we generated random problems of size 700, 1200, 1600 and 2000. This was mainly to have a sufficiently dense sequence of problem sizes for run time studies. To get comparative values for run time and solution quality we implemented the 2-opt method, a well known heuristic with quadratic time behavior. Since 2-opt is a tour improvement method it needs a valid initial tour. We used the nearest-neighbor-heuristic to generate these initial tours. The resulting two-phase method is denoted in the following as NN-2-OPT.

We performed 20 experiments with every combination of algorithm and problem. The simulation parameters for our algorithm were: $n_{distribution} = 100$, $\varepsilon_{bmu} = 0.05$, $\varepsilon_{neighbor} = 0.05$, $k_{neighbor} = 50$. Pinning was performed, whenever a cell had been 100 times *bmu* for the same city. It should be noted that we did not have to change parameter values when the problem size changes. We performed all described simulations with the above set of parameter values. Fig. 5 shows the measured mean run time for the algorithms. For

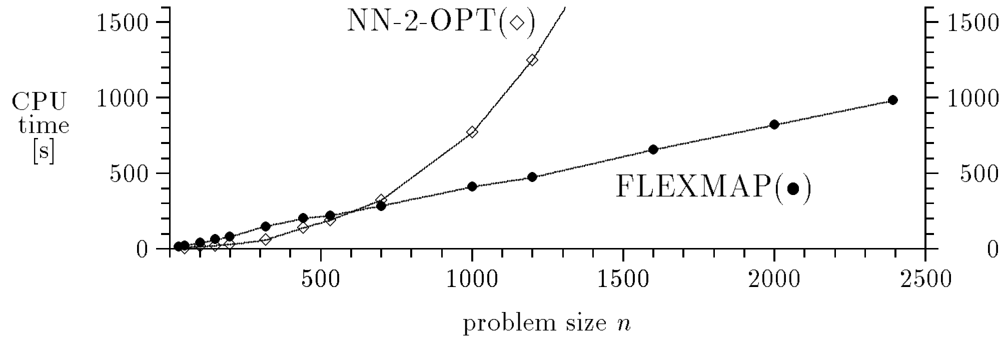

Figure 5: Run time behaviour

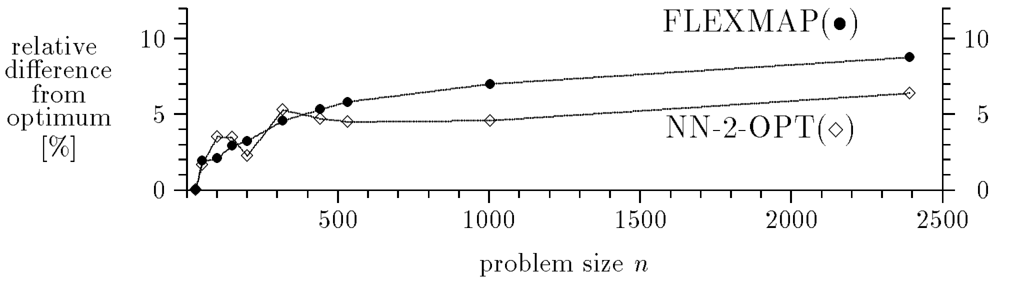

Figure 6: Quality of the generated solutions

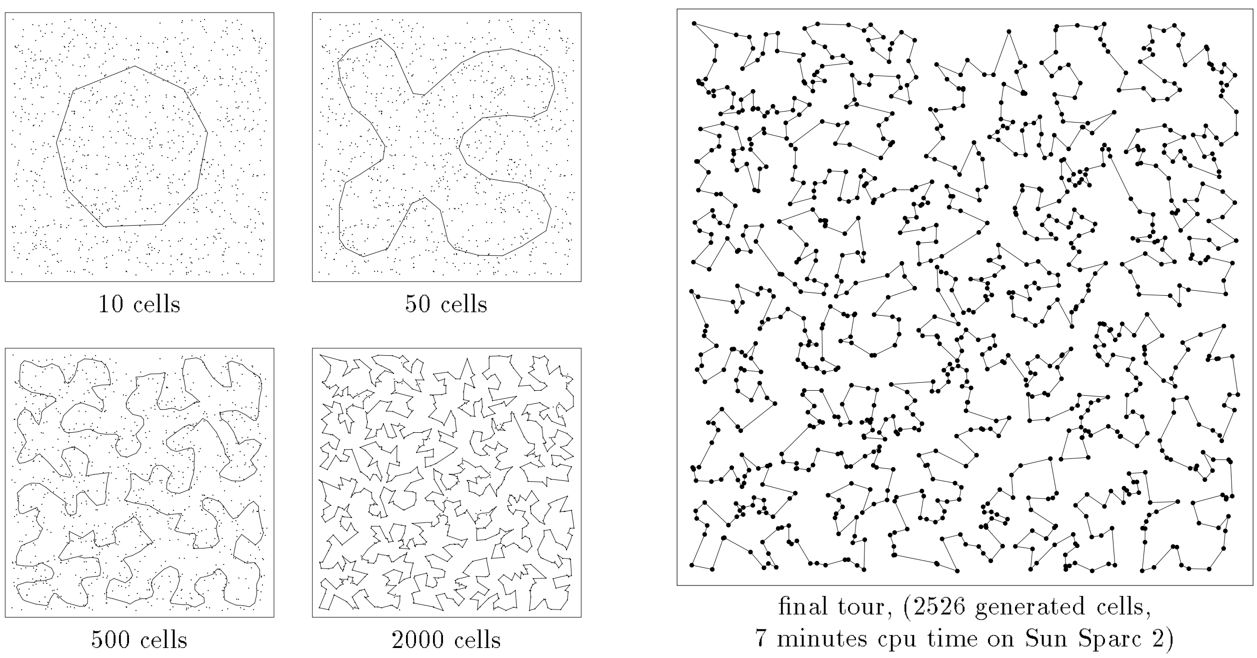

Figure 7: Development of a tour for a 1000-city problem

small problem sizes our approach is slower than the conventional heuristic. However, the time complexity is clearly linear and so for problems larger than a threshold value (in this case 700) the heuristic is outperformed with respect to time. E.g. for the 2392-city problem 5037 seconds were needed by NN-2-OPT in contrast to 980 seconds used by FLEXMAP.

The solutions generated by our approach are slightly worse than the solutions generated by the heuristic. Fig. 6 shows the quality of the best solutions found. Both procedures seem to generate better solutions for small problems, but the difference from optimum increases slowly and stays (for FLEXMAP) within 9 percent for the largest of the test problems ($n = 2392$). The number of cells that had to be generated to find a solution for a problem of size n was in all our experiments between $2n$ and $3n$, mostly very close to $2.5n$. In fig. 7 some snap shots of a simulation run and the final solution for a large TSP is shown.

6 Conclusion and Further Research

We presented a neural network for the traveling salesman problem that has linear time and space complexity. While the linear space requirements are also a property of some other networks (e.g. [4]) the linear time complexity is unprecedented by any conventional or neural algorithm. The treatment of problems with more than a million cities seems to become possible. The quality of the generated solutions is within 9 percent from optimum for all test problems and can probably be further increased. A new linear tour improvement procedure is currently investigated.

Related work dealing with growing two- and higher-dimensional cell structures has already been done and has lead to very versatile variants of Kohonen's self-organizing feature maps[6,7] and to a new clustering method[5].

References

[1] Burr, D.J., "An Improved Elastic Net Method for the Traveling Salesman Problem," *IEEE Conf. on Neural Networks, San Diego* (1988), 69–76.

[2] Durbin, R. & D. Willshaw, "An analogue approach to the travelling salesman problem using an elastic net method," *Nature* 326 (1987), 689–691.

[3] Eilon, S., C.D.T Watson-Gandy & N. Christofides, "Distribution Management: Mathematical Modeling and Practical Analysis," *Operational Research Quarterly* 20 (1969), 309.

[4] Favata, F. & R. Walker, "A study of the application of Kohonen-type neural networks to the travelling Salesman Problem," *Biological Cybernetics* 64 (1991), 463–468.

[5] Fritzke, B., "Unsupervised clustering with growing cell structures," *to appear in: Proc. of IJCNN-91*, Seattle (1991).

[6] Fritzke, B., "Let it grow - self-organizing feature maps with problem dependent cell structure," *to appear in: Proc. of ICANN-91*, Helsinki (1991).

[7] Fritzke, B. & O. Richter, "Using hypertetrahedron structures for multi-dimensional feature mappings," *submitted to NIPS*, Denver (1991).

[8] Hopfield, J. J. & D. W. Tank, ""Neural" computation of decisions in optimization problems," *Biological Cybernetics* 52 (1985), 141–152.

[9] Kohonen, T., "Self-organized formation of topologically correct feature maps," *Biological Cybernetics* 43 (1982), 59–69.

[10] Krolak, P.D., W. Felts & G. Marble, "A man-machine approach toward solving the traveling salesman problem," *Communications of the ACM* 14 (1971), 327–334.

[11] Lin, S. & B.W. Kernighan, "An effective heuristic for the travelling-salesman problem," *Optimization Research* 21 (1973), 498–516.

[12] Padberg, M.W. & G. Rinaldi, "Optimization of a 532-City Symmetric Traveling Salesman Problem by Branch and Cut," *Operations Research Letters* 6 (1987), 1–7.

[13] Padberg, M.W. & G. Rinaldi, "A branch & cut algorithm for the resolution of large-scale symmetric traveling salesman problems," IASI Research Report 247, 1988.