The Traveling Salesman Problem: A Neural Network Perspective

Jean-Yves Potvin

Centre de Recherche sur les Transports
Université de Montréal
C.P. 6128, Succ. A,
Montréal (Québec)
Canada H3C 3J7
potvin@iro.umontreal.ca

Abstract. This paper surveys the "neurally" inspired problem-solving approaches to the traveling salesman problem, namely, the Hopfield-Tank network, the elastic net, and the self-organizing map. The latest achievements in the neural network domain are reported and numerical comparisons are provided with the classical solution approaches of operations research. An extensive bibliography with more than one hundred references is also included.

Introduction

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization problem, which is simple to state but very difficult to solve. The problem is to find the shortest possible tour through a set of N vertices so that each vertex is visited exactly once. This problem is known to be NP-complete, and cannot be solved exactly in polynomial time.

Many exact and heuristic algorithms have been devised in the field of operations research (OR) to solve the TSP. We refer readers to [15, 64, 65] for good overviews of the TSP. In the sections that follow, we briefly introduce the OR problem-solving approaches to the TSP. Then, the neural network approaches for solving that problem are discussed.

Exact Algorithms

The exact algorithms are designed to find the optimal solution to the TSP, that is, the tour of minimum length. They are computationally expensive because they must (implicitly) consider

all feasible solutions in order to identify the optimum. The exact algorithms are typically derived from the integer linear programming (ILP) formulation of the TSP

$$\text{Min} \quad \sum_i \sum_j d_{ij} x_{ij}$$

subject to:

$$\sum_j x_{ij} = 1 \quad , i=1,...,N$$

$$\sum_i x_{ij} = 1 \quad , j=1,...,N$$

$$(x_{ij}) \in X$$

$$x_{ij} = 0 \text{ or } 1 \quad ,$$

where d_{ij} is the distance between vertices i and j and the x_{ij}'s are the decision variables: x_{ij} is set to 1 when arc (i,j) is included in the tour, and 0 otherwise. $(x_{ij}) \in X$ denotes the set of subtour-breaking constraints that restrict the feasible solutions to those consisting of a single tour.

Although the subtour-breaking constraints can be formulated in many different ways, one very intuitive formulation is

$$\sum_{i,j \in S_V} x_{ij} \leq |S_v| - 1 \qquad (S_v \subset V;\ 2 \leq |S_v| \leq N-2) \ ,$$

where V is the set of all vertices, S_v is some subset of V and $|S_v|$ is the cardinality of S_v. These constraints prohibit subtours, that is, tours on subsets with less than N vertices. If there were such a subtour on some subset of vertices S_v, this subtour would contain $|S_v|$ arcs. Consequently, the left-hand side of the inequality would be equal to $|S_v|$, which is greater than $|S_v|-1$, and the constraint would be violated for this particular subset. Without the subtour-breaking constraints, the TSP reduces to an assignment problem (AP), and a solution like the one shown in Figure 1 would then be feasible.

Branch and bound algorithms are commonly used to find an optimal solution to the TSP, and the AP-relaxation is useful to generate good lower bounds on the optimum value. This is true in particular for asymmetric problems, where $d_{ij} \neq d_{ji}$ for some i,j. For symmetric problems, like the Euclidean TSP (ETSP), the AP-solutions often contain many subtours with only two vertices. Consequently,

these problems are better addressed by specialized algorithms that can exploit their particular structure. For instance, a specific ILP formulation can be derived for the symmetric problem which allows for relaxations that provide sharp lower bounds (e.g., the shortest spanning one-tree[46]).

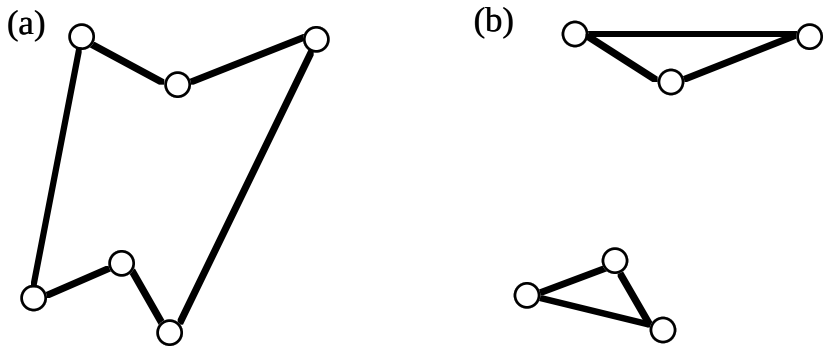


Fig. 1. (a) Solving the TSP, (b) Solving the assignment problem.

It is worth noting that problems with a few hundred vertices can now be routinely solved to optimality. Also, instances involving more than 2,000 vertices have been addressed. For example, the optimal solution to a symmetric problem with 2,392 vertices was identified after two hours and forty minutes of computation time on a powerful vector computer, the IBM 3090/600.[76,77] On the other hand, a classical problem with 532 vertices took five and a half hours on the same machine, indicating that the size of the problem is not the only determining factor for computation time. We refer the interested reader to [64] for a complete description of the state of the art with respect to exact algorithms.

Heuristic Algorithms

Running an exact algorithm for hours on an expensive computer may not be very cost-effective if a solution, within a few percent of the optimum, can be found quickly on a microcomputer. Accordingly, heuristic or approximate algorithms are often preferred to exact algorithms for solving the large TSPs that occur in practice (e.g., drilling problems).

Generally speaking, TSP heuristics can be classified as tour construction procedures, tour improvement procedures, and composite procedures, which are based on both construction and improvement techniques.

(a) *Construction procedures.* The best known procedures in this class gradually build a tour by selecting each vertex in turn and by inserting them one by one into the current tour. Various metrics are used for selecting the next vertex and for identifying the best place to insert it, like the proximity to the current tour and the minimum detour.[88]

(b) *Improvement procedures.* Among the local improvement procedures, the k-opt exchange heuristics are the most widely used, in particular, the 2-opt, 3-opt, and Lin-Kernighan heuristics.[67,68] These heuristics locally modify the current solution by replacing k arcs in the tour by k new arcs so as to generate a new improved tour. Figure 2 shows an example of a 2-opt exchange. Typically, the exchange heuristics are applied iteratively until a local optimum is found, namely a tour which cannot be improved further via the exchange heuristic under consideration. In order to overcome the limitations associated with local optimality, new heuristics like simulated annealing and tabu search are being used.[25,39,40,60] Basically, these new procedures allow local modifications that increase the length of the tour. By this means, the method can escape from local minima and explore a larger number of solutions.

The neural network models discussed in this paper are often compared to the simulated annealing heuristic described in [60]. In this context, simulated annealing refers to an implementation based on the 2-opt exchanges of Lin[67], where an increase in the length of the tour has some probability of being accepted (see the description of simulated annealing in Section 3).

(c) *Composite procedures.* Recently developed composite procedures, which use both construction and improvement techniques, are now among the most powerful heuristics for solving TSPs. Among the new generation of composite heuristics, the most successful ones are the CCAO heuristic,[41] the GENIUS heuristic,[38] and the iterated Lin-Kernighan heuristic.[53]

For example, the iterated Lin-Kernighan heuristic can routinely find solutions within 1% of the optimum for problems with up to

10,000 vertices.[53] Heuristic solutions within 4% of the optimum for some 1,000,000-city ETSPs are reported in [12]. Here, the tour construction procedure is a simple greedy heuristic. At the start, each city is considered as a fragment, and multiple fragments are built in parallel by iteratively connecting the closest fragments together until a single tour is generated. The solution is then processed by a 3-opt exchange heuristic. A clever implementation of this procedure solved some 1,000,000-city problems in less than four hours on a VAX 8550.

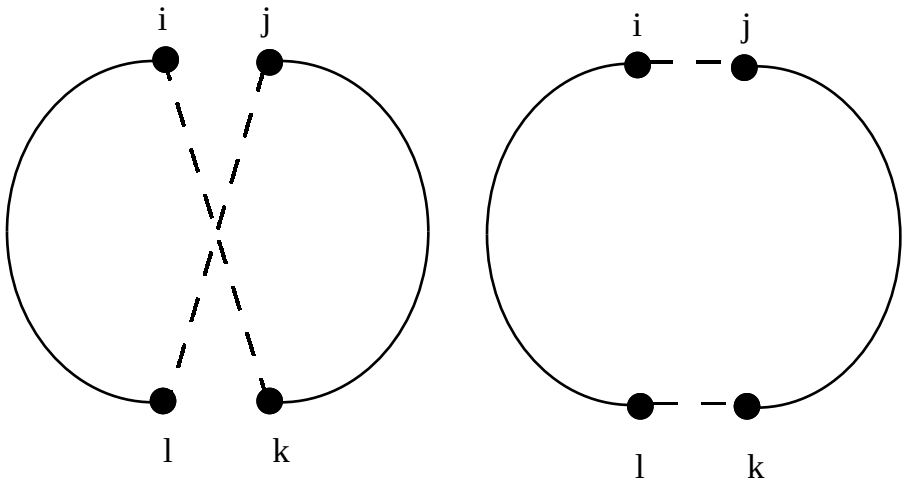


Fig. 2. Exchange of links (i,k),(j,l) for links (i,j),(k,l).

Artificial Neural Networks

Because of the simplicity of its formulation, the TSP has always been a fertile ground for new solution ideas. Consequently, it is not surprising that many problem-solving approaches inspired by artificial neural networks have been applied to the TSP.

Currently, neural networks do not provide solution quality that compares with the classical heuristics of OR. However, the technology is quite young and spectacular improvements have already been achieved since the first attempts in 1985.[51] All of these efforts for solving a problem that has already been quite successfully addressed by operations researchers are motivated, in part, by the fact that artificial neural networks are powerful parallel devices. They are made up of a large number of simple elements that can process their inputs in parallel. Accordingly, they lend themselves naturally to implementations on parallel computers. Moreover, many neural

network models have already been directly implemented in hardware as "neural chips."

Hence, the neural network technology could provide a means to solve optimization problems at a speed that has never been achieved before. It remains to be seen, however, whether the quality of the neural network solutions will ever compare to the solutions produced by the best heuristics in OR. Given the spectacular improvements in the neural network technology in the last few years, it would certainly be premature at this time to consider this line of research to be a "dead end."

In the sections that follow, we review the three basic neural network approaches to the TSP, namely, the Hopfield-Tank network, the elastic net, and the self-organizing map. Actually, the elastic nets and self-organizing maps appear to be the best approaches for solving the TSP. But the Hopfield-Tank model was the first to be applied to the TSP and it has been the dominant neural approach for solving combinatorial optimization problems over the last decade. Even today, many researchers are still working on that model, trying to explain its failures and successes. Because of its importance, a large part of this paper is thus devoted to that model and its refinements over the years.

The paper is organized along the following lines. Sections 1 and 2 first describe the Hopfield-Tank model and its many variants. Sections 3 and 4 are then devoted to the elastic net and the self-organizing map, respectively. Finally, concluding remarks are made in Section 5. Each basic model is described in detail and no deep understanding of neural network technology is assumed. However, previous exposure to an introductory paper on the subject could help to better understand the various models.[61] In each section, computation times and numerical comparisons with other OR heuristics are provided when they are available. However, the OR specialist must understand that the computation time for simulating a neural network on a serial digital computer is not particularly meaningful, because such an implementation does not exploit the inherent parallelism of the model. For this reason, computation times are often missing in neural network research papers.

A final remark concerns the class of TSPs addressed by neural network researchers. Although the Hopfield-Tank network has been applied to TSPs with randomly generated distance matrices,[106]

virtually all work concerns the ETSP. Accordingly, Euclidean distances should be assumed in the sections that follow, unless it is explicitly stated otherwise.

The reader should also note that general surveys on the use of neural networks in combinatorial optimization may be found in [22, 70]. An introductory paper about the impacts of neurocomputing on operations research may be found in [29].

Section 1. The Hopfield-Tank Model

Before going further into the details of the Hopfield model, it is important to observe that the network or graph defining the TSP is very different from the neural network itself. As a consequence, the TSP must be mapped, in some way, onto the neural network structure.

For example, Figure 3a shows a TSP defined over a transportation network. The artificial neural network encoding that problem is shown in Figure 3b. In the transportation network, the five vertices stand for cities and the links are labeled or weighted by the inter-city distances d_{ij} (e.g., $d_{NY,LA}$ is the distance between New York and Los Angeles). A feasible solution to that problem is the tour Montreal-Boston-NY-LA-Toronto-Montreal, as shown by the bold arcs.

In Figure 3b, the Hopfield network[50] is depicted as a 5x5 matrix of nodes or units that are used to encode solutions to the TSP. Each row corresponds to a particular city and each column to a particular position in the tour. The black nodes are the activated units that encode the current solution (namely, Montreal is in first position in the tour, Boston in second position, NY in third, etc.).

Only a few connections between the units are shown in Figure 3b. In fact, there is a connection between each pair of units, and a weight is associated with each connection. The signal sent along a connection from unit i to unit j is equal to the weight T_{ij} if unit i is activated. It is equal to 0 otherwise. A negative weight thus defines an inhibitory connection between the two units. In such a case, it is unlikely that both units will be active or "on" at the same time, because the first unit that turns on immediately sends an inhibitory signal to the other unit through that connection to prevent its activation. On the other hand, it is more likely for both units to be on at the same time if the connection has a positive weight. In such a

case, the first unit that turns on sends a positive excitatory signal to the other unit through that connection to facilitate its activation.

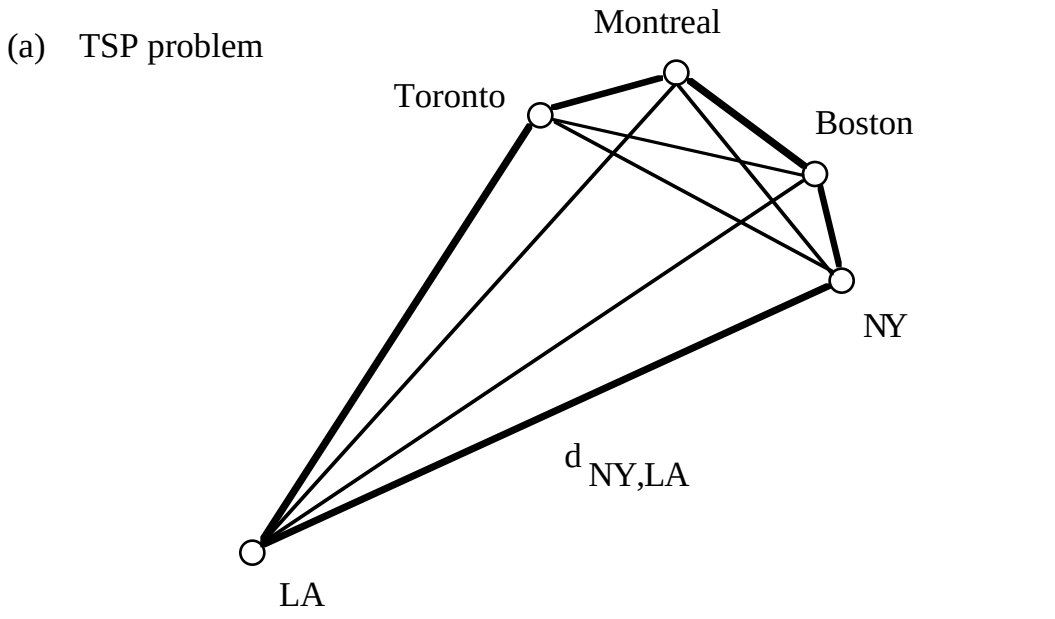


Fig. 3. Mapping a TSP onto the Hopfield network.

In the TSP context, the weights are derived in part from the inter-city distances. They are chosen to penalize infeasible tours and, among the feasible tours, to favor the shorter ones. For example, $T_{LA5,NY5}$ in Figure 3b denotes the weight on the connection between the units that represent a visit to cities LA and NY both in the fifth position on a tour. Consequently, that connection should be inhibitory (negative weight), because two cities cannot occupy the same exact position. The first unit to be activated will inhibit the other unit via that connection, so as to prevent an infeasible solution to occur.

In Section 1.1, we first introduce the Hopfield model, which is a network composed of binary "on/off" or "0/1" units, like the artificial neural network shown in Figure 3b. We will then describe the Hopfield-Tank model, which is a natural extension of the discrete model to units with continuous activation levels. Finally, the application of the Hopfield-Tank network to the TSP will be described.

1.1 The Discrete Hopfield Model

The original Hopfield neural network model[50] is a fully interconnected network of binary units with symmetric connection weights between the units. The connection weights are not learned but are defined a priori from problem data (the inter-city distances in a TSP context). Starting from some arbitrarily chosen initial configuration, either feasible or infeasible, the Hopfield network evolves by updating the activation of each unit in turn (i.e., an activated unit can be turned off, and an unactivated unit can be turned on). The update rule of any given unit involves the activation of the units it is connected to as well as the weights on the connections. Via this update process, various configurations are explored until the network settles into a stable configuration. In this final state, all units are stable according to the update rule and do not change their activation status.

The dynamics of the Hopfield network can be described formally in mathematical terms. To this end, the activation levels of the binary units are set to zero and one for "off" and "on," respectively. Starting from some initial configuration $\{V_i\}_{i=1,...,L}$, where L is the number of units and V_i is the activation level of unit i, the network relaxes to a stable configuration according to the following update rule

$$\text{set } V_i \text{ to } 0 \quad \text{if } \Sigma_j T_{ij} V_j < \theta_i$$
$$\text{set } V_i \text{ to } 1 \quad \text{if } \Sigma_j T_{ij} V_j > \theta i$$
$$\text{do not change } V_i \text{ if } \Sigma_j T_{ij} V_j = \theta_i \ ,$$

where T_{ij} is the connection weight between units i and j, and θ_i is the threshold of unit i.

The units are updated at random, one unit at a time. Since the configurations of the network are L-dimensional, the update of one unit from zero to one or from one to zero moves the configuration of the network from one corner to another of the L-dimensional unit hypercube.

The behavior of the network can be characterized by an appropriate energy function. The energy E depends only on the activation levels V_i (the weights T_{ij} and the thresholds θ_i are fixed and derived from problem data), and is such that it can only decrease as the network evolves over time. This energy is given by

$$E = -1/2 \ \Sigma_i \Sigma_j \ T_{ij} V_i V_j \ + \ \Sigma_i \ \theta_i V_i \ . \qquad (1.1)$$

Since the connection weights T_{ij} are symmetric, each term $T_{ij}V_iV_j$ appears twice within the double summation of (1.1). Hence, this double summation is divided by 2.

It is easy to show that a unit changes its activation level if and only if the energy of the network decreases by doing so. In order to prove that statement, we must consider the contribution E_i of a given unit i to the overall energy E, that is,

$$E_i = - \ \Sigma_j \ T_{ij} V_i V_j \ + \ \theta_i V_i \ .$$

Consequently,

$$\text{if } V_i = 1 \text{ then } \ E_i = \ - \ \Sigma_j \ T_{ij} V_j + \theta_i$$
$$\text{if } V_i = 0 \text{ then } \ E_i = \ 0 \ .$$

Hence, the change in energy due to a change ΔV_i in the activation level of unit i is

$$\Delta E_i = - \ \Delta V_i \ (\Sigma_j \ T_{ij} V_j \ - \ \theta_i) \ .$$

Now, ΔV_i is one if unit i changed its activation level from zero to one, and such a change can only occur if the expression between the parentheses is positive. As a consequence, ΔE_i is negative and the energy decreases. This same line of reasoning can be applied when a unit i changes its activation level from one to zero (i.e., $\Delta V_i = -1$). Since the energy can only decrease over time and the number of configurations is finite, the network must necessarily converge to a stable state (but not necessarily the minimum energy state). In the next section, a natural extension of this model to units with continuous activation levels is described.

1.2 The Continuous Hopfield-Tank Model

In [51], Hopfield and Tank extended the original model to a fully interconnected network of nonlinear analog units, where the activation level of each unit is a value in the interval [0,1]. Hence, the space of possible configurations $\{Vi\}_{i=1,...,L}$ is now continuous rather than discrete, and is bounded by the L-dimensional hypercube defined by $V_i = 0$ or 1. Obviously, the final configuration of the network can be decoded into a solution of the optimization problem if it is close to a corner of the hypercube (i.e., if the activation value of each unit is close to zero or one).

The main motivation of Hopfield and Tank for extending the discrete network to a continuous one was to provide a model that could be easily implemented using simple analog hardware. However, it seems that continuous dynamics also facilitate convergence.[47]

The evolution of the units over time is now characterized by the following differential equations (usually called "equations of motion")

$$dU_i/dt = \Sigma_j T_{ij}V_j + I_i - U_i, \quad i=1,...,L \qquad (1.2)$$

where U_i, I_i and V_i are the input, input bias, and activation level of unit i, respectively. The activation level of unit i is a function of its input, namely

$$V_i = g(U_i) = 1/2\ (1 + \tanh U_i/U_o) = 1/\ (1+ e^{-2U_i/U_o})\ . \qquad (1.3)$$

The activation function g is the well-known sigmoidal function, which always returns a value between 0 and 1. The parameter U_o is

used to modify the slope of the function. In Figure 4, for example, the U_o value is lower for curve (2) than for curve (1).

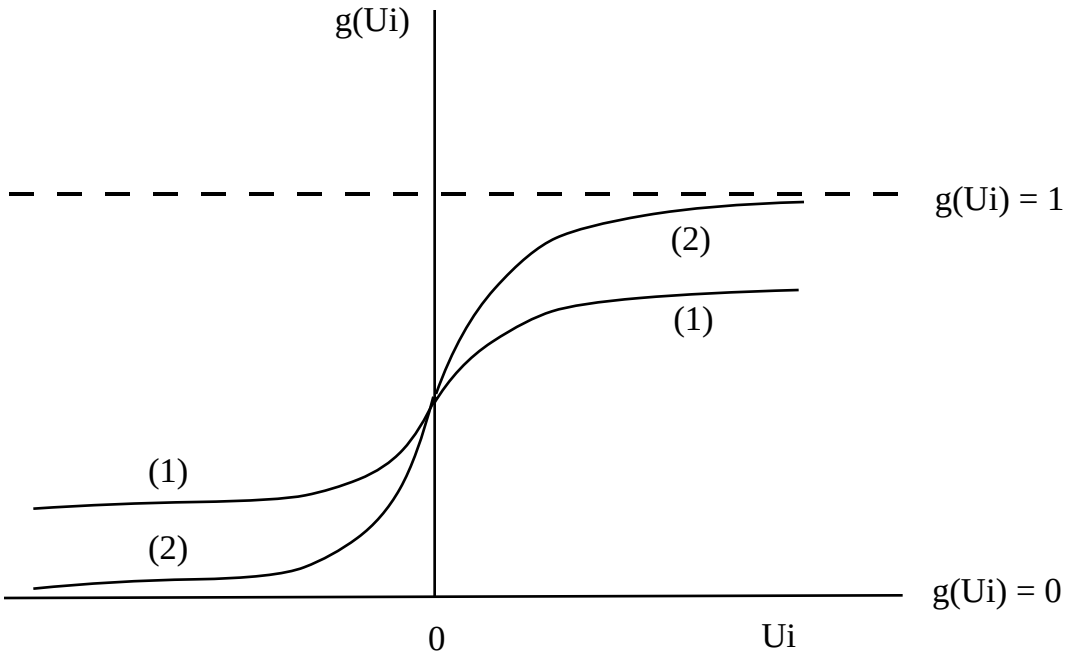


Fig. 4. The sigmoidal activation function.

The energy function for the continuous Hopfield-Tank model is now

$$E = -1/2 \sum_i \sum_j T_{ij} V_i V_j - \sum_i V_i I_i + \int_0^{V_i} g^{-1}(x)dx \, . \qquad (1.4)$$

Note in particular that $dU_i/dt = -dE/dV_i$. Accordingly, when the units obey the dynamics of the equations of motion, the network is performing a gradient descent in the network's configuration space with respect to that energy function, and stabilizes at a local minimum. At that point, $dU_i/dt = 0$ and the input to any given unit i is the weighted sum of the activation levels of all the other units plus the bias, that is

$$U_i = \sum_j T_{ij} V_j + I_i \, .$$

1.3 Simulation of the Hopfield-Tank Model

In order to simulate the behavior of the continuous Hopfield-Tank model, a discrete time approximation is applied to the equations of motion

$$\{U_i(t+\Delta t) - U_i(t)\} / \Delta t = \sum_j T_{ij} V_j(t) + I_i - U_i(t) ,$$

where Δt is a small time interval. This formula can be rewritten as

$$U_i(t+\Delta t) = U_i(t) + \Delta t \left(\sum_j T_{ij} V_j(t) + I_i - U_i(t)\right) .$$

Starting with some initial values $\{U_i(0)\}_{i=1,...,L}$ at time t=0, the system evolves according to these equations until a stable state is reached. During the simulation, Δt is usually set to 10^{-5} or 10^{-6}. Smaller values provide a better approximation of the analog system, but more iterations are then required to converge to a stable state.

In the literature, the simulations have mostly been performed on standard sequential machines. However, implementations on parallel machines are discussed in [10, 93]. The authors report that it is possible to achieve almost linear speed-up with the number of processors. For example, a Hopfield-Tank network for a 100-city TSP took almost three hours to converge to a solution on a single processor of the Sequent Balance 8000 computer.[10] The computation time was reduced to about 20 minutes using eight processors.

1.4 Application of the Hopfield-Tank Model to the TSP

In the previous sections, we have shown that the Hopfield-Tank model performs a descent towards a local minimum of the energy function E. The "art" of applying that model to the TSP is to appropriately define the connection weights T_{ij} and the bias I_i so that the local minima of E will correspond to good TSP solutions.

In order to map a combinatorial optimization problem like the TSP onto the Hopfield-Tank model, the following steps are suggested in [83, 84]:

(1) Choose a representation scheme which allows the activation levels of the units to be decoded into a solution of the problem.

(2) Design an energy function whose minimum corresponds to the best solution of the problem.
(3) Derive the connectivity of the network from the energy function.
(4) Set up the initial activation levels of the units.

These ideas can easily be applied to the design of a Hopfield-Tank network in a TSP context:

(1) First, a suitable representation of the problem must be chosen. In [51], the TSP is represented as an NxN matrix of units, where each row corresponds to a particular city and each column to a particular position in the tour (see Figure 1).

If the activation level of a given unit V_{Xi} is close to 1, it is then assumed that city X is visited at the ith position in the tour. In this way, the final configuration of the network can be interpreted as a solution to the TSP. Note that N^2 units are needed to encode a solution for a TSP with N cities.

(2) Second, the energy function must be defined. The following function is used in [51]

$$\begin{aligned} E = \quad & A/2 \left(\Sigma_X \Sigma_i \Sigma_{j \neq i} V_{Xi} V_{Xj}\right) \\ & + B/2 \left(\Sigma_i \Sigma_X \Sigma_{Y \neq X} V_{Xi} V_{Yi}\right) \\ & + C/2 \left(\Sigma_X \Sigma_i V_{Xi} - N\right)^2 \\ & + D/2 \left(\Sigma_X \Sigma_{Y \neq X} \Sigma_i \, d_{XY} V_{Xi} \left(V_{Yi+1} + V_{Yi-1}\right)\right) , \quad (1.5) \end{aligned}$$

where the A, B, C, and D parameters are used to weight the various components of the energy.

The first three terms penalize solutions that do not correspond to feasible tours. Namely, there must be exactly one activated unit in each row and column of the matrix. The first and second terms, respectively, penalize the rows and columns with more than one activated unit, and the third term requires a total of N activated units (so as to avoid the trivial solution $V_{Xi}=0$ for all Xi). The fourth term ensures that the energy function will favor short tours over longer ones. This term adds the distance d_{XY} to the energy value when cities X and Y are in consecutive positions in the tour (note that subscripts are taken modulo N, so that $V_{X,N+1}$ is the same as V_{X1}).

(3) Third, the bias and connection weights are derived. To do so, the energy function of Hopfield and Tank (1.5) is compared to the generic energy function (1.6), which is a slightly modified version of (1.4): each unit has now two subscripts (city and position) and the last term is removed (since it does not play any role here)

$$E = -1/2 \sum_{Xi} \sum_{Yj} T_{XiYj} V_{Xi} V_{Yj} - \sum_{Xi} V_{Xi} I_{Xi} \quad . \qquad (1.6)$$

Consequently, the weights T_{XiYj} on the connections of the Hopfield-Tank network are identified by looking at the quadratic terms in the TSP energy function, while the bias I_{Xi} is derived from the linear terms. Hence,

$$\begin{aligned} T_{XiYj} &= - A \, \delta_{XY} \, (1- \delta_{ij}) \\ &\quad - B \, \delta_{ij} \, (1- \delta_{XY}) \\ &\quad - C \\ &\quad - D \, d_{XY} \, (\delta_{j,i+1} + \delta_{j,i-1}) \ , \end{aligned}$$

$$I_{Xi} = + CN_e \ ,$$

where $\delta_{ij}=1$ if $i=j$ and 0 otherwise.

The first and second terms in the definition of the connection weights stand for inhibitory connections within each row and each column, respectively. Hence, a unit whose activation level is close to 1 tends to inhibit the other units in the same row and column. The third term is a global inhibitor term. The combined action of this term and the input bias I_{Xi}, which are both derived from the C term in the energy function (1.5), favor solutions with a total of N activated units. Finally, the fourth term is called the "data term" and prevents solutions with adjacent cities that are far apart (namely, the inhibition is stronger between two units when they represent two cities X, Y in consecutive positions in the tour, with a large inter-city distance d_{XY}).

In the experiments of Hopfield and Tank, the parameter N_e in the definition of the bias $I_{Xi} = CN_e$ does not always correspond exactly to the number of cities N. This parameter is used by Hopfield and Tank to adjust the level of the positive bias signal with respect to the negative signals coming through the other connections, and it is usually slightly larger than N. Note

finally that there are $O(N^4)$ connections between the N^2 units for a TSP with N cities.

(4) The last step is to set the initial activation value of each unit to 1/N plus or minus a small random perturbation (in this way the sum of the initial activations is approximately equal to N).

With this model, Hopfield and Tank were able to solve a randomly generated 10-city ETSP, with the following parameter values: A=B=500, C=200, D=500, N_e=15. They reported that for 20 distinct trials, using different starting configurations, the network converged 16 times to feasible tours. Half of those tours were one of the two optimal tours. On the other hand, the network was much less reliable on a randomly generated 30-city ETSP (900 units). Apart from frequent convergence to infeasible solutions, the network commonly found feasible tours with a length over 7.0, as compared to a tour of length 4.26 generated by the Lin-Kernighan exchange heuristic.[68]

Three years later, it was claimed in [105] that the results of Hopfield and Tank were quite difficult to reproduce. For the 10-city ETSP of Hopfield and Tank, using the same parameter settings, the authors report that on 100 different trials, the network converged to feasible solutions only 15 times. Moreover, the feasible tours were only slightly better than randomly generated tours. Other experiments by the same authors, on various randomly generated 10-city ETSP problems, produced the same kind of results.

The main weaknesses of the original Hopfield-Tank model, as pointed out in [105] are the following.

(a) Solving a TSP with N cities requires $O(N^2)$ units and $O(N^4)$ connections.

(b) The optimization problem is not solved in a problem space of O(N!), but in a space of $O(2^{N^2})$ where many configurations correspond to infeasible solutions.

(c) Each valid tour is represented 2N times in the Hopfield-Tank model because any one of the N cities can be chosen as the starting city, and the two orientations of the tour are equivalent for a symmetric problem. This phenomenon is referred to as "2N-degeneracy" in neural network terminology.

(d) The model performs a gradient descent of the energy function in the configuration space, and is thus plagued with the limitations of "hill-climbing" approaches, where a local optimum is found. As a consequence, the performance of the model is very sensitive to the initial starting configuration.

(e) The model does not guarantee feasibility. In other words, many local minima of the energy function correspond to infeasible solutions. This is related to the fact that the constraints of the problem, namely that each city must be visited exactly once, are not strictly enforced but rather introduced into the energy function as penalty terms.

(f) Setting the values of the parameters A, B, C, and D is much more an art than a science and requires a long "trial-and-error" process. Setting the penalty parameters A, B, and C to small values usually leads to short but infeasible tours. Alternatively, setting the penalty parameters to large values forces the network to converge to any feasible solution regardless of the total length. Moreover, it seems to be increasingly difficult to find "good" parameter settings as the number of cities increases.

(g) Many infeasible tours produced by the network visit only a subset of cities. This is due to the fact that the third term in the energy function (C term) is the only one to penalize such a situation. The first two terms (A and B terms), as well as the fourth term (D term), benefit from such a situation.

(h) It usually takes a large number of iterations (in the thousands) before the network converges to a solution. Moreover, the network can "freeze" far from a corner of the hypercube in the configuration space, where it is not possible to interpret the configuration as a TSP solution. This phenomenon can be explained by the shape of the sigmoidal activation function which is very flat for large positive and large negative U_i's (see Figure 2). Consequently, if the activation level V_i of a given unit i is close to zero or one, even large modifications to U_i will produce only slight modifications to the activation level. If a large number of units are in this situation, the network will evolve very slowly, a phenomenon referred to as "network paralysis."

Paralysis far from a corner of the hypercube can occur if the slope of the activation function is not very steep. In that case, the flat regions of the sigmoidal function extend further and

affect a larger number of units (even those with activation levels far from zero and one).

(i) The network is not adaptive, because the weights of the network are fixed and derived from problem data, rather than taught from it.

The positive points are that the model can be easily implemented in hardware, using simple analog devices, and that it can also be applied to non-Euclidean TSPs, in particular, problems that do not satisfy the triangle inequality and cannot be interpreted geometrically.[106] This is an advantage over the geometric approaches that are presented in Sections 3 and 4.

Section 2. Variants of the Hopfield-Tank Model

Surprisingly enough, the results of Wilson and Pawley did not discourage the community of researchers but rather stimulated the search for ways to improve the original Hopfield-Tank model. There were also numerous papers providing in-depth analysis of the model to explain its failures and propose various improvements to the method. [4,5,6,7,17,24,56,86,87,90,108]

The modifications to the original model can be classified into six distinct categories: modifications to the energy function, techniques for estimating "good" parameter settings, addition of hard constraints to the model, incorporation of techniques to escape from local minima, new problem representations, and modifications to the starting configurations. We now describe each category, and emphasize the most important contributions.

2.1 Modifications to the Energy Function

The first attempts were aimed at modifying the energy function to improve the performance of the Hopfield-Tank model. Those studies, which add or modify terms to push the model towards feasible solutions, are mostly empirical.

(a) In [18, 75, 78, 95], the authors suggest replacing either the third term (C term) or the three first terms (A, B, and C terms) of the original energy function (1.5) by

$$F/2 \ \Sigma_X (\Sigma_i V_{Xi} - 1)^2 \ + \ G/2 \ \Sigma_i (\Sigma_X V_{Xi} - 1)^2 .$$

This modification helps the model to converge towards feasible tours, because it heavily penalizes configurations that do not have exactly one active unit in each row and each column. In particular, it prevents many solutions that do not incorporate all cities. Note that a formulation where the A, B, and C terms are replaced by the two new terms can be implemented with only $O(N^3)$ connections. In [14], the author proposes an alternative approach to that problem by adding an additional excitatory bias to each unit so as to get a larger number of activated units.

(b) In [18], the authors suggest the addition of a new penalty term in the energy function (1.5). This term attempts to drive the search out from the center of the hypercube (and thus towards the corners of the hypercube) so as to alleviate the network paralysis problem. The additional term is

$$F/2 \sum_{Xi} V_{Xi} (1-V_{Xi}) = F/2 \left(N^2/4 - \sum_{Xi} (V_{Xi} - 1/2)^2\right) \ .$$

In the same paper, they also propose a formulation where the inter-city distances are only used in the linear components of the energy function. Hence, the distances are provided to the network via the input bias rather than encoded into the connection weights. It is a great advantage for a hardware implementation, like a neural chip, because the connection weights do not change from one TSP instance to another and can be fixed into the hardware at fabrication time. However, a new representation of the problem with $O(N^3)$ units is now required.

Brandt and his colleagues[18] report that the Hopfield-Tank model with the two modifications suggested in (a) and (b) consistently converged to feasible tours for randomly generated 10-city ETSPs. Moreover, the average length was now much better than the length of randomly generated tours. Table 1 shows these results for problems with 10, 16, and 32 cities. Note that the heading "Manual" in the Table refers to tours constructed manually by hand.

Number of Cities	Number of Problems	Average Tour Length		
		Brandt	Manual	Random
10	21	2.97	2.92	5.41
16	10	3.70	3.57	9.77
32	4	5.48	4.57	17.12

Table 1. Comparison of Results for Three Solution Procedures

2.2 Finding Good Settings for the Parameter Values

In [44, 45], the authors experimentally demonstrate that various relationships among the parameters of the energy function must be satisfied in order for the Hopfield-Tank model to converge to feasible tours. Their work also indicates that the region of good settings in the parameter space quickly gets very narrow as the number of cities grows. This study supports previous observations about the difficulty to tune the Hopfield-Tank energy function for problems with a large number of cities.

Cuykendall and Reese[28] also provide ways of estimating parameter values from problem data, as the number of cities increases. In [4, 5, 6, 7, 27, 57, 58, 74, 86, 87] theoretical relationships among the parameters are investigated in order for feasible tours to be stable. The work described in these papers is mostly based on a close analysis of the eigenvalues and eigenvectors of the connection matrix.

Wang and Tsai[102] propose to gradually reduce the value of some parameters over time. However, time-varying parameters preclude a simple hardware implementation. Lai and Coghill[63] propose the genetic algorithm, as described in [49], to find good parameter values for the Hopfield-Tank model.

Along that line of research, the most impressive practical results are reported in [28]. The authors generate feasible solutions for a 165-city ETSP, by appropriately setting the bias of each unit and the U_0 parameter in the sigmoidal activation function (1.3). Depending on the parameter settings, it took between one hour and 10 hours of computation time on an APOLLO DN4000 to converge to a solution. The computation times were in a range of 10 to 30 minutes for

another 70-city ETSP. Unfortunately, no comparisons are provided with other problem-solving heuristics.

2.3 Addition of Constraints to the Model

The approaches that we now describe add new constraints to the Hopfield-Tank model, so as to restrict the configuration space to feasible tours.

(a) In [81, 97, 98] the activation levels of the units are normalized so that $\sum_i V_{Xi} = 1$ for all cities X. The introduction of these additional constraints is only one aspect of the problem-solving methodology, which is closely related to the simulated annealing heuristic. Accordingly, the full discussion is deferred to Section 2.4, where simulated annealing is introduced.

(b) Other approaches are more aggressive and explicitly restrict the configuration space to feasible tours. In [96], the authors calculate the changes required in the remaining units to maintain a feasible solution when a given unit is updated. The energy function is then evaluated on the basis of the change to the updated unit and all the logically implied changes to the other units. This approach converges consistently to feasible solutions on 30-city ETSPs. The tours are only 5% longer on average than those generated by the simulated annealing heuristic.

In [69], the author updates the configurations using Lin and Kernighan's exchange heuristic.[68] Foo and Szu[35] use a "divide-and-conquer" approach to the problem. They partition the set of cities into subsets and apply the Hopfield-Tank model to each subset. The subtours are then merged back together into a single larger tour with a simple heuristic. Although their approach is not conclusive, the integration of classical OR heuristics and artificial intelligence within a neural network framework could provide interesting research avenues for the future.

2.4 Incorporation of Techniques to Escape from Local Minima

The Hopfield-Tank model converges to a local minimum, and is thus highly sensitive to the starting configuration. Hence, various modifications have been proposed in the literature to alleviate this problem.

(a) In [1, 2, 3], a Boltzmann machine[48] is designed to solve the TSP. Basically, a Boltzmann machine incorporates the simulated annealing heuristic[25,60] within a discrete Hopfield network, so as to allow the network to escape from bad local minima.

The simulated annealing heuristic performs a stochastic search in the space of configurations of a discrete system, like a Hopfield network with binary units. As opposed to classical hill-climbing approaches, simulated annealing allows modifications to the current configuration that increase the value of the objective or energy function (for a minimization problem). More precisely, a modification that reduces the energy of the system is always accepted, while a modification that increases the energy by ΔE is accepted with Boltzmann probability $e^{-\Delta E/T}$, where T is the temperature parameter. At a high temperature, the probability of accepting an increase to the energy is high. This probability gets lower as the temperature is reduced. The simulated annealing heuristic is typically initiated at a high temperature, where most modifications are accepted, so as to perform a coarse search of the configuration space. The temperature is then gradually reduced to focus the search on a specific region of the configuration space.

At each temperature T, the configurations are modified according to the Boltzmann update rule until the system reaches an "equilibrium." At that point, the configurations follow a Boltzmann distribution, where the probability of the system being in configuration s' at temperature T is

$$P_T(s') = \frac{e^{-E(s')/T}}{\sum_s e^{-E(s)/T}} . \qquad (2.1)$$

Here, E(s') is the energy of configuration s', and the denominator is the summation over all configurations. According to that probability, configurations of high energy are very likely to be observed at high temperatures and much less likely to be observed at low temperatures. The inverse is true for low energy configurations. Hence, by gradually reducing the temperature parameter T and by allowing the system to reach equilibrium at each temperature, the system is expected to ultimately settle down at a configuration of low energy.

This simulated annealing heuristic has been incorporated into the discrete Hopfield model to produce the so-called Boltzmann machine. Here, the binary units obey a stochastic update rule, rather than a deterministic one. At each iteration, a unit is randomly selected and the consequence of modifying its activation level (from zero to one or from one to zero) on the energy is evaluated. The probability of accepting the modification is then

$$\frac{1}{1 + e^{\Delta E/T}},$$

where ΔE is the modification to the energy.

This update probability is slightly different from the one used in simulated annealing. In particular, the probability of accepting a modification that decreases the energy of the network (i.e., $\Delta E < 0$) is not one here, but rather a value between 0.5 and one. However, this new update probability has the same convergence properties as the one used in simulated annealing and, in that sense, the two expressions are equivalent.

Aarts and Korst[1,2,3] design a Boltzmann machine for solving the TSP based on these ideas. Unfortunately, their approach suffers from very slow convergence and, as a consequence, only 30-city TSPs have been solved with this model.

(b) In [43], the Boltzmann Machine is generalized to units with continuous activation levels. A truncated exponential distribution is used to compute the activation level of each unit. As for the discrete Boltzmann machine, the model suffers from slow convergence, and only small 10-city ETSPs have been solved.

(c) The research described in [81, 97, 98], which is derived from the mean-field theory, is probably the most important contribution to the literature relating to the Hopfield-Tank model since its original description in [51]. The term "mean-field" refers to the fact that the model computes the mean activation levels of the stochastic binary units of a Boltzmann machine.

This section focuses on the model of Van den Bout and Miller,[97,98] but the model of Peterson and Soderberg[81] is similar. We first introduce the iterative algorithm for updating the

configurations of the network. Then, we explain the relationships between this model and the Boltzmann machine.

The neural network model introduced in [97] is characterized by a new simplified energy function

$$E = d_{max}/2 \left(\sum_i \sum_X \sum_{Y \neq X} V_{Xi} V_{Yi}\right) + \sum_X \sum_{Y \neq X} \sum_i d_{XY} V_{Xi} (V_{Yi+1} + V_{Yi-1}) . \qquad (2.2)$$

The first summation penalizes solutions with multiple cities at the same position, while the second summation computes the tour length. Note that the penalty value is weighted by the parameter d_{max}.

Starting from some arbitrary initial configuration, the model evolves to a stable configuration that minimizes (2.2), via the following iterative algorithm:

1. Set the temperature T.
2. Select a city X at random.
3. Compute
$$U_{Xi} = - d_{max} \sum_{Y \neq X} V_{Yi} - \sum_{Y \neq X} d_{XY}(V_{Yi+1} + V_{Yi-1}), \quad i=1,...,N.$$
4. Compute
$$V_{Xi} = \frac{e^{U_{Xi}/T}}{\sum_j e^{U_{Xj}/T}} , \quad i=1,...,N .$$
5. Evaluate the energy E.
6. Repeat Steps 2 to 5 until the energy no longer decreases (i.e., a stable configuration has been reached).

We note that the activation levels always satisfy the constraints

$$\sum_i V_{Xi} = 1 , \text{ for all cities X.}$$

Accordingly, each value V_{Xi} can be interpreted as the probability that city X occupies position i. When a stable configuration is reached, the activation levels V_{Xi} satisfy the following system of equations (called the "mean-field equations")

$$V_{Xi} = \frac{e^{U_{Xi}/T}}{\sum_j e^{U_{Xj}/T}}, \qquad (2.3)$$

where $U_{Xi} = - dE/dV_{Xi}$ (see Step 3 of the algorithm).

In order to understand the origin of the mean-field equations, we must go back to the evolution of a discrete Hopfield network with binary units, when those units are governed by a stochastic update rule like the Boltzmann rule (see the description of the simulated annealing heuristic and the Boltzmann machine in point (a)).

It is known that the configurations of that network follow a Boltzmann distribution at equilibrium (i.e., after a large number of updates). Since the network is stochastic, it is not possible to know what the exact configuration will be at a given time. On the other hand, the average or mean activation value of each binary unit at Boltzmann equilibrium at a given temperature T is a deterministic value which can be computed as follows

$$<V_{Xi}> = \sum_s P_T(s) V_{Xi}(s) = \sum_{s_{Xi}} P_T(s_{Xi}) \ .$$

In this equation, the summations are restricted to the configurations satisfying $\Sigma_j V_{Xj} = 1$, for all cities X (so as to comply with the model of Van den Bout and Miller), $P_T(s)$ is the Boltzmann probability of configuration s at temperature T, $V_{Xi}(s)$ is the activation level of unit Xi in configuration s, and s_{Xi} denotes the configurations where $V_{Xi} = 1$. Hence, we have

$$<V_{Xi}> = \frac{\sum_{s_{Xi}} e^{-E(s_{Xi})/T}}{\sum_j \sum_{s_{Xj}} e^{-E(s_{Xj})/T}} \ .$$

In this formula, the double summation in the denominator is equivalent to a single summation over all configurations s, because each configuration contains exactly one activated unit in $\{X_j\}_{j=1,...,N}$ ($\Sigma_j V_{Xj} = 1$ and V_{Xj} is either zero or one).

Now, we can apply the so-called "mean-field approximation" to $<V_{Xi}>$. Rather than summing up over all configurations, we assume that the activation levels of all units that interact with a given unit X_j are fixed at their mean value. For example, rather than summing up

over all configurations s_{Xi} in the numerator (configurations where $V_{Xi}=1$), we fix the activation levels of all the other units to their mean value. In this way, the summation can be removed. By applying this idea to both the numerator and the denominator, and by observing that $-U_{Xi}$ is the contribution of unit Xi to the energy (2.2) when $V_{Xi}=1$, the expression can be simplified to

$$<V_{Xi}> = \frac{e^{<U_{Xi}>/T}}{\sum_j e^{<U_{Xj}>/T}} ,$$

where

$$<U_{Xi}> = - d_{max} \sum_{Y \neq X} <V_{Yi}> - \sum_{Y \neq X} d_{XY}(<V_{Yi+1}> + <V_{Yi-1}>) .$$

These equations are the same as the equations of Van den Bout and Miller (2.3). Hence, the V_{Xi} values computed via their iterative algorithm can be interpreted as the mean activation levels of the corresponding stochastic binary units at Boltzmann equilibrium (at a given temperature T). At low temperatures, the low energy configurations have high Boltzmann probability and they dominate in the computation of the mean values $<V_{Xi}>$. Hence, the stable configuration computed by the algorithm of Van den Bout and Miller is expected to be of low energy, for a sufficiently small parameter value T, because the stable configuration is composed of those mean values $<V_{Xi}>$.

As noted in [98], all the activation levels are the same at high temperatures, that is, $V_{Xi} \rightarrow 1/N$ when $T \rightarrow \infty$. As the temperature parameter is lowered, each city gradually settles into a single position, because such configurations correspond to low energy states. In addition, the model also prevents two cities from occupying the same position, because a penalty of $d_{max}/2$ is incurred in the energy function. If the parameter d_{max} is set to a value slightly larger than twice the largest distance between any two cities, the network can find a configuration with lower energy simply by moving one of the two cities into the empty position. Feasible tours are thus guaranteed through the combined actions of the new energy function and the additional constraints imposed on the activation levels V_{Xi} (once again, for a sufficiently small parameter value T).

It is clear that the key problem is to identify a "good" value for the parameter T. By gradually decreasing the temperature, Van den

Bout and Miller identified a critical value T_c where all the energy minimization takes place. Above the critical value T_c, the units rarely converge to zero or one and feasible tours do not emerge. Below T_c, all the tours generated are feasible, and the best tours emerge when T is close to T_c. Obviously, the critical temperature value is highly dependent on the particular TSP to be solved. In [97, 98], Van den Bout and Miller describe a methodology for estimating that value from the inter-city distances. Using various T and d_{max} parameter values, their best tour on a 30-city TSP had a length of 26.9, as compared to 24.1 for a tour obtained with the simulated annealing heuristic.

Peterson and Soderberg[81] test a similar model on much larger problems, ranging in size from 50 to 200 cities. They observe that the length of the tours generated by the neural network approach are about 8% longer on average than the tours generated with a simulated annealing heuristic. Moreover, no tour was more than 10% longer than the corresponding simulated annealing tour. However, the average tour lengths are not provided (rather, the results are displayed as small histograms). Also, no computation times are reported.

It is quite interesting to note that Bilbro et al.[13] have shown that the evolution of the above model is equivalent to the evolution of the Hopfield-Tank network governed by the equations of motion (see Section 1). However, convergence to a stable configuration is much faster by solving the mean-field equations. This increased convergence speed explains the successes of Peterson and Soderberg who routinely found feasible solutions to the TSP with up to 200 cities, those being the largest problems ever solved with models derived from the work of Hopfield and Tank.

(d) Mean-field annealing refers to the application of the mean-field algorithm, as described in (c), with a gradual reduction of the temperature from high to low values, as in the simulated annealing heuristic.[13,19,80,107] As pointed out in [97], this approach is of little use if the critical temperature, where all the energy minimization takes place, can be accurately estimated from problem data. If the estimate is not accurate, then it can be useful to gradually decrease the temperature.

(e) In [8, 26, 66], random noise is introduced into the activation level of the units in order to escape from local minima. The random

noise follows a normal distribution with mean zero, and its intensity is modulated by a temperature parameter which is gradually reduced, as in the simulated annealing heuristic.

(f) Finally, Wells[103] describes a network of neural oscillators. The network evolves towards sustained oscillations between equivalent configurations, as opposed to the classical Hopfield-Tank model which is evolving towards a single stable configuration.

2.5 Using New Problem Representations

In [54, 55], a new problem representation based on an integer linear programming formulation of the TSP is proposed. Each unit V_{XY} now stands for an arc (X,Y) in the graph, and that unit is on when arc (X,Y) is in the tour. This "adjacency representation" requires $O(N^2)$ units and only $O(N^3)$ connections. Moreover, the inter-city distances now occur in the linear terms of the energy function and are thus provided to the network via the input bias

$$E = A/2 \sum_X (\sum_{X \neq Y} V_{XY} - 2)^2 + B/2 \sum_Y (\sum_{X \neq Y} V_{XY} - 2)^2$$

$$+ C/2 \sum_X \sum_Y V_{XY} d_{XY} .$$

In this new energy function, the first two terms penalize cities that do not have exactly two neighbors.

Basically, this model solves the assignment problem associated to the TSP when the subtour-breaking constraints are relaxed. Accordingly, the network usually ends up with multiple subtour solutions. In order to address that problem, a second binary layer is connected to the Hopfield-Tank layer, so as to break the stability of multiple subtour solutions and drive the network towards feasible solutions.

Xu and Tsai[106] use the same representation and energy function. However they address the multiple subtours problem in a different way. When the neural network reaches a multiple subtour solution, they add a higher-order connection between the units involved in a subtour, so as to forbid any later occurrence of that subtour (a higher-order connection can involve an arbitrary number of units, as opposed to a standard connection which involves only two units). Higher-order connections are incrementally added in this way, until the network stabilizes with a feasible solution. That

procedure is usually slow to converge. In the worst case, all possible subtours must be forbidden for a valid solution to emerge!

In practice, the authors often stop the procedure before a feasible solution is reached, and then merge the remaining subtours into a single tour via the "patching" heuristic proposed in [59]. This simple heuristic connects two subtours by deleting one link in each subtour and by introducing two new links to generate a single larger tour. In Figure 5, for example, the links (i,j) and (k,l) have been replaced by the links (i,l) and (j,k). The links to be replaced are chosen so as to minimize the length of the new enlarged tour. As noted by Xu and Tsai, the number of solutions with multiple subtours tends to increase when the number of cities increases, and the patching heuristic thus plays an important role in large problems.

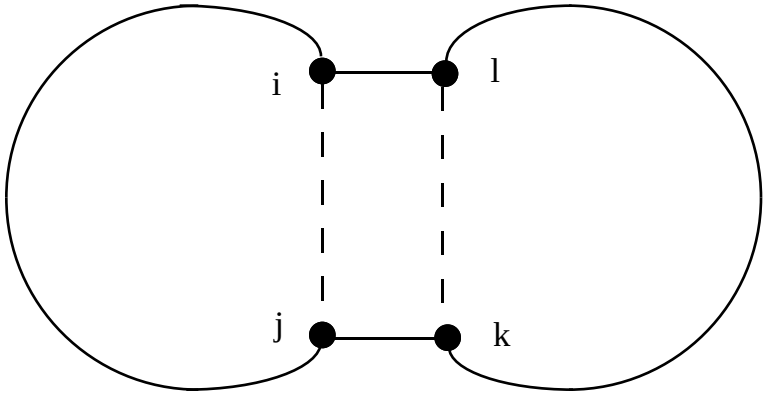


Fig. 5. The patching heuristic.

Euclidean problems with up to 150 cities have been addressed with this approach. However, the authors have shown that better solutions can be generated by applying 25 distinct runs of a 2-opt exchange heuristic from different randomly generated initial solutions, and by taking the best overall solution.

The authors also apply their model to TSPs with randomly generated distance matrices. In that context, the model provided better solutions than the 2-opt (30 runs) and the Lin-Kernighan heuristics (30 runs). These results are supported by the probabilistic analysis in [59], where it is shown that solving the assignment problem first and then applying the patching heuristic provides near optimal solutions to TSPs with random distance matrices.

2.6 Starting from Different Initial Configurations

In [98], the authors suggest starting from a good initial configuration by forcing cities that are close in distance to be adjacent in the tour (which means that some units are preset to one). This technique is called "heuristic seeding." Other authors suggest starting from the center of the unit hypercube, that is, setting the initial activation value of all units to 1/2.[44]

This discussion shows that the Hopfield-Tank model and its many variants are still restricted to relatively small TSPs with 200 cities or less. The main results with respect to this model are summarized in Table 2.

Paper	Largest TSP (Number of Cities)	Length of Tour from Hopfield-Tank Models	TSP Heuristic	Length of Tour from TSP Heuristic
Hopfield and Tank (1985)	30	5.07	LK	4.26
Brandt et al. (1988)	32	5.48	MT	4.57
Cuykendall and Reese (1989)	165	9.409-9.933[a]	NA	NA
Peterson and Soderberg (1989)	200	NA	SA	NA
Xu and Tsai (1991)	100	592.3	TO	570.2[c]
	150	9.529[b]	LK	9.616[d]

[a] Computation time of 1 to 10 hours on an Apollo DN4000
[b] Best of 5 runs with post-optimization by Lin-Kernighan
[c] Best of 25 runs starting from randomly generated tours
[d] Best of 30 runs starting from randomly generated tours

LK Lin-Kernighan

MT Manual Tour
NA Not Available
SA Simulated Annealing
TO 2-opt

Table 2. Comparison of Results Produced by Hopfield-Tank Models and TSP Heuristics

Although the work in [81, 97, 98] has shown that it is possible to design models that consistently converge to feasible tours (one of the problems with the original formulation), those tours do not yet compare with the tours generated by other TSP heuristics, like simulated annealing. Furthermore, the convergence is quite slow when the model is executed on a sequential computer. These observations have led individuals to explore new lines of research, in particular, elastic nets and self-organizing maps.

Section 3. The Elastic Net

The elastic net approach[32] is fundamentally different from the approach of Hopfield and Tank. It is a geometrically inspired algorithm, originating in the work of Willshaw and Von der Malsburg[104], and is closely related to the self organizing map, as described in Section 4.

The elastic net algorithm is an iterative procedure where M points, with M larger than the number of cities N, are lying on a circular ring or "rubber band" originally located at the center of the cities. The rubber band is gradually elongated until it passes sufficiently near each city to define a tour. During that process two forces apply: one for minimizing the length of the ring, and the other one for minimizing the distance between the cities and the points on the ring. These forces are gradually adjusted as the procedure evolves.

Figures 6a, 6b and 6c show how the elastic net typically evolves over time. In the figure, the small black circles are the points located on the ring which are migrating towards the cities. The final solution is shown in Figure 6d.

The elastic net algorithm will now be introduced more formally. Let X_i be the position of the ith city, i=1,...,N and let Y_j be the position of the jth point on the ring, j=1,...,M. In order to get a feasible tour, one point in the ring should converge to each city. Accordingly, at

each step, the ring points are updated in parallel according to the following equations

$$\Delta Y_j = \alpha \sum_i w_{ij} (X_i - Y_j) + \beta K (Y_{j+1} + Y_{j-1} - 2Y_j) , \quad j=1,...,M \qquad (3.1)$$

where

$$w_{ij} = \frac{\phi(d_{X_i Y_j}, K)}{\sum_k \phi(d_{X_i Y_k}, K)} ,$$

$$\phi(d,K) = e^{-d^2/2K^2} .$$

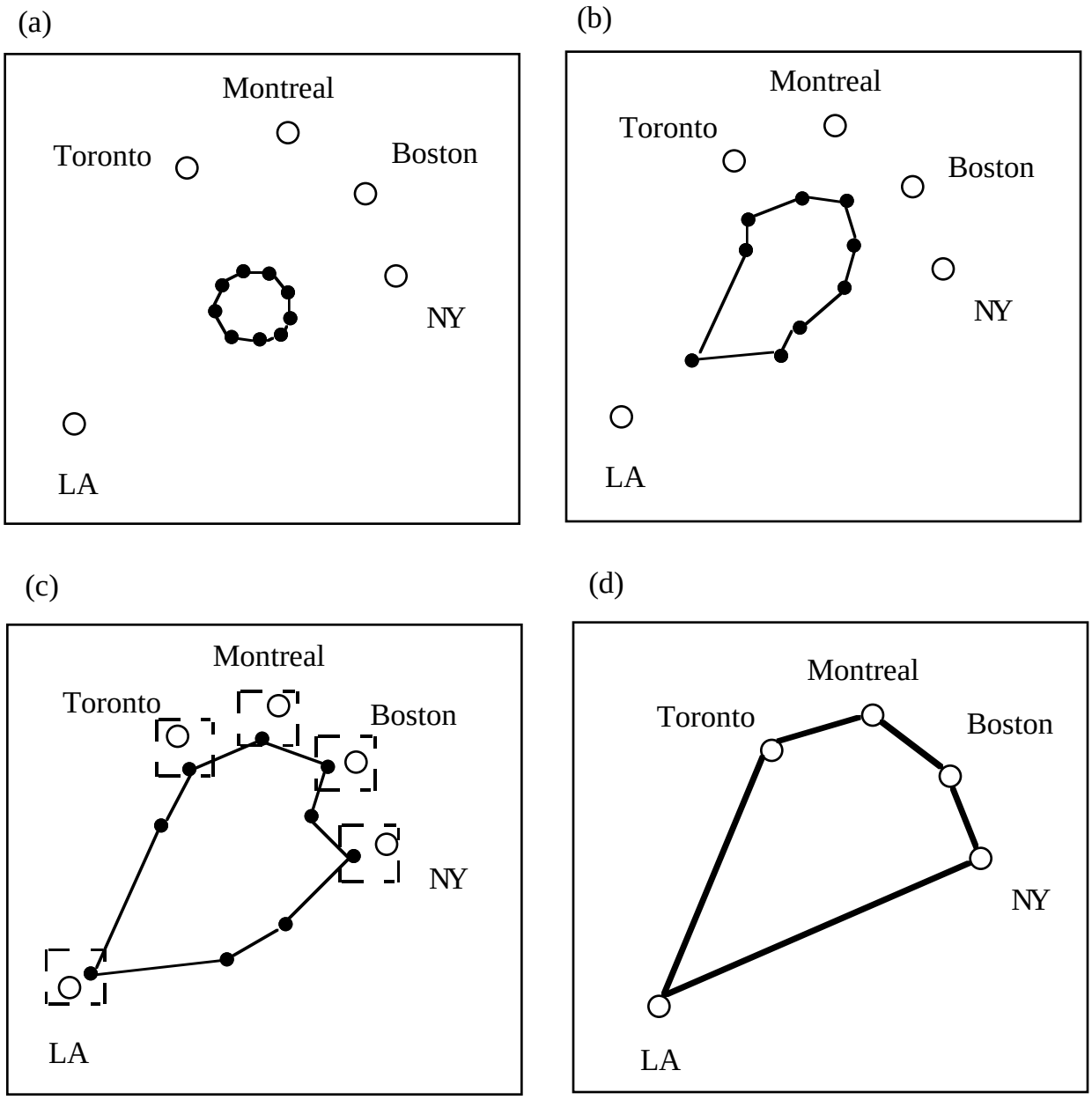


Fig. 6. Evolution of the elastic net over time (a) (b) (c) and the final tour (d).

In equation (3.1), α and β are constant parameters, K is the "scale" parameter, $d_{X_i Y_j}$ is the Euclidean distance between X_i and Y_j, and w_{ij} is a measure of the "influence" of city i on point j. The α term drives the points on the ring towards the cities, so that for each city i there is at least one ring point j within distance K. The β term is aimed at keeping neighboring points together so as to produce a short tour.

Figure 7 illustrates the two forces that impact point j. The first force (1), derived from the α term in the update equations, is quite easy to understand and is aimed at driving point j towards city i. The second force (2), derived from the β term, can be more easily interpreted by considering the equivalence

$$Y_{j+1} + Y_{j-1} - 2Y_j = (Y_{j+1} - Y_j) + (Y_{j-1} - Y_j) \ .$$

Accordingly, this force defines a tension on the ring that keeps neighboring points together.

The update equations can be expressed as the derivatives of an appropriate energy function E, namely

$$\Delta Y_j = -K \, dE/dY_j \ ,$$

where

$$E = - \alpha K \sum_i \ln \sum_j \phi(d_{X_i Y_j}, K) + \beta/2 \sum_j d_{Y_j Y_{j+1}}^2 \ . \qquad (3.2)$$

Consequently, this algorithm finds a local minimum of the energy function (3.2) by performing a gradient descent in the (continuous) two-dimensional Euclidean space. Note also that when $K \to 0$ and $M/N \to \infty$, the global minimum of the energy function solves the TSP. Accordingly, as the algorithm evolves, the scale parameter K is gradually reduced (like the temperature parameter in the simulated annealing heuristic). When K is 0, the energy reduces to the sum of the squared distances between the points on the ring. This is equivalent to minimizing the tour length at the limit $M/N \to \infty$.

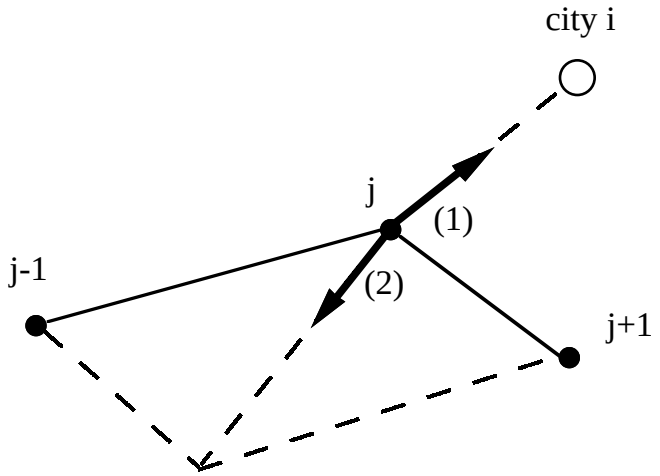


Fig. 7. Forces that impact ring point j.

With this algorithm, Durbin and Willshaw[32] solved five different sets of randomly generated 50-city ETSPs. It took about 1,250 iterations to converge and the tour lengths were, on average, only 2% longer than those obtained with the simulated annealing heuristic (average of five distinct runs). The authors report that the computation times for both the elastic net and the simulated annealing heuristic are approximately the same. The elastic net algorithm also solved a 100-city ETSP and generated a tour about 1% longer than the best tour obtained after eight million trials of the simulated annealing heuristic. In this experiment, the following parameter settings were used: $M = 2.5N$, $\alpha=0.2$, $\beta=2.0$, $K_{max} = 0.2$, and $K_{min} = 0.02$.

In a follow-up paper,[31] Durbin and his colleagues analyze how the shape of the energy function (3.2) is modified by the value of the scale parameter. It is shown that when $K \rightarrow 0$, the local mimima of the energy function are such that each city X_i is matched by at least one Y_j. Moreover, each X_i is matched by exactly one Y_j if parameters α and β satisfy the condition $\alpha/\beta < A$, where A is the shortest distance from a point being attracted to some city to any other point not being attracted to that city.

As pointed out by Simmen[92], the above conditions ensure that each city is matched by exactly one point on the ring, but a point can still match two or more cities, if those cities are sufficiently close. Although the problem could be solved by using so many points on the ring that the typical spacing between neighboring points would be much less than the minimum inter-city distance, Simmen provides additional conditions on the α/β ratio to achieve the same result, and avoid the computational burden associated with a large M.

Two variants of this algorithm have been proposed in the literature.

(a) Burr[23] reports improvements to the convergence speed of the elastic net algorithm, by using the new update equations

$$\Delta Y_j = \alpha \sum_i w_{ij} (X_i - Y_j) + \alpha \sum_i s_{ij} (X_i - Y(X_i)) + \beta K (Y_{j+1} + Y_{j-1} - 2Y_j) ,$$

where

$Y(X_i)$ is the closest ring point to X_i ,

$$s_{ij} = \frac{\phi(d_{Y(X_i)Y_j}, K)}{\sum_k \phi(d_{Y(X_i)Y_k}, K)} .$$

The first and third terms in the new update equations correspond to the two original terms in (3.1). The middle term is added by the author to favor short tours by keeping the points evenly distributed on the ring. The update equations are applied with a varying K parameter whose value is computed at each iteration, and derived from the average squared distance between each city and its closest point on the tour.

The author reports that on the five sets of 50-city ETSPs described in [32], the solutions are of similar quality as those of the original elastic net algorithm. However, it took only 30 iterations on average to converge to a solution, as compared to 1,250 iterations for the model of Durbin and Willshaw.

(b) In [16], the ring points are updated by only considering the cities that have a significant influence on them. The authors report that this filtering mechanism reduces the computation times without affecting the quality of the solution. The authors have solved problems ranging in size from 100 to 1,000 cities. Their solutions are about 5% to 10% longer than those of the Lin-Kernighan heuristic. However, on the 1,000-city problems, the elastic net algorithm took only half the time of the Lin-Kernighan heuristic.

To summarize this section, a benchmark study in [79] compares five different problem-solving approaches on randomly generated ETSPs ranging in size from 50 to 200 cities. In this study, the elastic net and the model of Peterson and Soderberg[81], a variant of the Hopfield-Tank model, are compared. The main results are shown in Table 3.

These results clearly indicate that the elastic net is superior to the model of Peterson and Soderberg. Another comparison in favor of the elastic net can be established by comparing the figures in Durbin and Willshaw[32] and in Hopfield and Tank[51]. On the same 30-city ETSP, the best tour generated by the Hopfield-Tank model had a length of 5.07, while the elastic net found a tour of length 4.26 on a single trial (the same tour as the one found by the Lin-Kernighan exchange heuristic[68]).

As a final remark, interesting relationships can be established between the elastic net model and the Hopfield-Tank network. In particular, Simic[91] demonstrates that the elastic net and the Hopfield-Tank model can both be derived from statistical mechanics. The differences observed between the two approaches stem from different ways of handling the constraints. The elastic net, by its very nature, strictly enforces some TSP constraints, while the Hopfield-Tank model transforms these constraints into penalties. Yuille[109] provides a similar study relating elastic nets to deformable templates, which are simple parameterized shapes that dynamically evolve to fit an image.

Number of Cities	Average Tour Length	
	EN	PS[a]
50	5.62	6.61
100	7.69	8.58
200	11.14	12.66

[a] Best of five runs on each problem

EN Elastic Net
PS Peterson and Soderberg

Table 3. Comparison of Results for the Elastic Net and the Model of Peterson and Soderberg

Section 4. The Self-organizing Map

The self-organizing maps are instances of so-called "competitive neural networks," which are used by many unsupervised learning systems to cluster or classify data. As opposed to supervised models, where a desired response is externally provided to the network, the unsupervised systems must find useful correlations in the input data by themselves. Hence, they are said to be "self-organizing." They self-organize in an adaptive way, by gradually adjusting the weights on their connections (as opposed to the Hopfield model, where the connection weights are fixed).

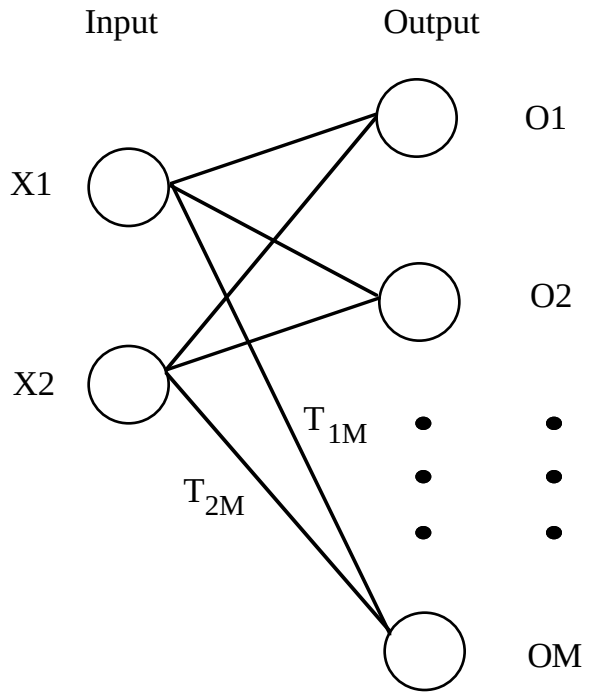
Conceptually, this model is closely related to the elastic net. In both cases, a ring is defined and gradually modified until it passes sufficiently near each city to define a tour. However, the two approaches differ in the way they update the coordinates of the points on the ring.

In Section 4.1, we first describe the competitive networks in general terms. Then, we introduce the self-organizing map and describe its application to the TSP.

4.1 Competitive Networks

A typical two-layer competitive network with two input units and M output units is depicted in Figure 8. Since we are considering two-dimensional TSPs in this paper, we have shown only two input units. However, there can be a large number I of inputs. In Figure 8, T_{1M} and T_{2M} denote the weights on the connections from the two input units to output unit M, and $T_M = (T_{1M}, T_{2M})$ is the weight vector of unit M.

The aim of the network is to group a set of I-dimensional input patterns into K clusters ($K \leq M$). The classes or clusters are not known in advance and, consequently, the network must discover by itself structure in the input data. At the end of the self-organization process, each cluster is associated with a distinct output unit. More precisely, each output unit gets activated only for a particular subset of input patterns (defining a cluster).



$$T_M = (T_{1M}, T_{2M})$$

Fig. 8. A two-layer network.

The model is quite simple and works in the following way. After initializing the network with small random connection weights T_{ij}, the input patterns are presented sequentially to the network. For each presentation of a given I-dimensional pattern X, the network initiates the competition among the output units, and the winning unit j* is the one which maximizes its net input, namely

$$net_{j^*} = max_j \; net_j, \quad j=1,...,M$$

where $net_j = \sum_i T_{ij} X_i$.

The winning unit thus associates the current input pattern to cluster j* and updates its weight vector $T_j^* = \{T_{ij}^*\}_{i=1,...,I}$, so as to get closer to that input pattern. The update rule is

$$T_j^{*\,new} = T_j^{*\,old} + \mu \, (X - T_j^{*\,old}) \; ,$$

where μ is a fractional learning rate. Typically, the other output units do not modify their weight vectors.

In order to get a stable classification, the whole set of input patterns must be presented many times to the network (referred to as "passes" or "cycles" through the set of inputs). As the network evolves, the weight vectors of the output units move in the I-dimensional space and tend to settle after a while into distinct regions of that space. When this phenomenon occurs, the winning output unit associated with each input pattern does not change from one pass through the data to another and the clustering process stabilizes. The final weight vector of each unit is then interpreted as the prototype vector of a cluster (that is, the centroid or "average pattern" of the cluster).

4.2 The Self-organizing Map

A competitive network groups the inputs on the basis of similarity, namely via the correlation between the input patterns X and the weight vectors T_j. In the TSP context, the input patterns are the two-dimensional coordinates of the cities, and consequently, the Euclidean distance must be used as the similarity measure (maximizing the inner product is equivalent to minimizing the Euclidean distance only when all input and weight vectors are normalized to the same length). Moreover, the objective is now to map these coordinates onto a ring of output units, so as to define an ordering of the cities.

Those observations suggest the use of Kohonen's self-organizing map.[62] This model is aimed at generating mappings from higher to lower dimensional spaces, so that the relationships among the inputs are reflected as faithfully as possible in the outputs (topological mapping). As a consequence, topological relationships are established among the output units, and those relationships are respected when the weights are updated. Namely, the output units that are "close" to the winning unit, according to the topology, also update their weight vectors so as to move in the same direction as the winning unit.

In the TSP context, a set of two-dimensional coordinates, locating the cities in Euclidean space, must be mapped onto a set of one dimensional positions in the tour, so as to preserve the neighborhood relationships among the cities. For example, the competitive network depicted in Figure 8 can be transformed into a self-organizing map,

by choosing $M \geq N$ and by linking the output units to form the ring O_1-O_2-...-O_M-O_1 (each output unit has two immediate neighbors).

The evolution of the self-organizing map is quite similar to the evolution of the competitive network of Section 4.1. However, there are two main differences. First, the winning output unit is the one whose weight vector is closest in Euclidean distance to the vector of coordinates of the current city (current input pattern). Second, the neighbors of the winning unit also move their weight vector in the same direction as the winning unit, but with decreasing intensity along the ring.

As a consequence, the winning unit j* satisfies

$$d_{X,T_{j^*}} = \min_j d_{X,T_j} \ , \ j=1,...,M \ ,$$

and the connection weights are then updated according to the formula

$$T_j^{\ new} = T_j^{\ old} + \mu \ G(d'_{jj^*}) \ (X - T_j^{old}) \ , \qquad (4.1)$$

where $d_{X,Tj}$ is the Euclidean distance between input vector X and weight vector T_j, and G is a function of the "proximity" of output units j and j* on the ring. Usually, the units on the ring are indexed from 1 to M, and d'_{jj^*} is based on that indexing, so that

$$d'_{jj^*} = \min \ (|j - j^*|, M - |j - j^*|) \ ,$$

where |x| denotes absolute value. In particular, the distance between a given unit and its two immediate neighbors on the ring is one.

Obviously, the value of G must decrease as the d'_{jj^*} value increases. Moreover, the modifications to the weight vectors of the neighboring units should ideally be reduced over time. As an example, the function G suggested in [21] is

$$\begin{aligned} G &= (1 - d'_{jj^*}/L)^{\beta} \ , \text{ if } d'_{jj^*} < L \\ &= 0 \qquad\qquad\quad \ , \text{ otherwise} \ , \end{aligned}$$

where L= N/2, and the value of β is increased after each complete pass through the set of coordinates. Note that G is always equal to one for the winning unit j*, and the value decreases for the other units j as d'_{jj^*} and β increase.

Multiple passes through the entire set of coordinates are performed until there is a weight vector sufficiently close to each city. Figure 6 is still a good way to visualize the evolution of the self-organizing map. However, the little black circles on the ring now represent the output units, and the location of each output unit O_j is determined by its weight vector T_j.

Different self-organizing maps are described in the literature based on that scheme.

(a) Fort[36] was one of the first with Angeniol et al.,[9] Hueter,[52] and Ritter and Schulten[85] to apply Kohonen's ideas to the TSP. Fort notes, in particular, that the speed of convergence can be increased by reducing the neighborhood of the winning unit and by reducing the modification to the weights of the neighboring units over time. In his experiments, Fort uses 2N output units and solves problems with up to 400 cities. With respect to the 400-city problem, he generates a tour of length 15.73, as compared to an estimated optimum of 14.98 derived from Stein's formula[94].

(b) The work of Angeniol et al.[9] is based on the distinctive feature that units in the ring are dynamically created and deleted. A unit is duplicated if it wins for two different cities after a complete pass through the set of coordinates. It is deleted, if it does not win after three complete passes. Starting with a single unit in the ring, the authors report that up to twice as many units as cities may be created during the procedure.

In their weight update equations, there is no learning rate parameter μ (it is implicitly set to one), but there is an additional parameter K in the definition of the function G, namely

$$G(d'_{jj}*, K) = 1/\sqrt{2}\ (e^{-d'_{jj}*^2/K^2}) \ .$$

When $K \to \infty$, all units move towards the current city with strength $1/\sqrt{2}$. On the other hand, when $K \to 0$ only the winning unit j* moves toward the city. Consequently, the K value is decreased after each pass through the set of coordinates according to the formula

$$K \leftarrow (1- \alpha)\, K \ ,$$

with $0 < \alpha < 1$.

The authors report good results on the five sets of 50-city problems of Durbin and Willshaw.[32] Their results also show that a single run of the elastic net algorithm is superior to a single run of the self-organizing map (see Table 4). The same kind of results are reported in [21].

City Sets	Average Tour Length				
	EN	SA	SA+3	SOM	SOM+
1	5.98	5.88	5.84	6.06	5.84
2	6.09	6.01	5.99	6.25	6.00
3	5.70	5.65	5.57	5.83	5.58
4	5.86	5.81	5.70	5.87	5.60
5	6.49	6.33	6.17	6.70	6.19

EN Elastic Net
SA Simulated Annealing
SA+3 Best solution found by SA and many distinct runs of 3-opt starting from randomly generated tours
SOM Self-organizing Map ($\alpha = 0.2$)
SOM+ Best solution found by SOM over 4,000 different runs (by processing the cities in various orders)

Table 4. Comparison of Results for Five Solution Procedures on Five Sets of 50-city Problems

The authors also solved a randomly generated 1,000-city ETSP with their model. It took 20 minutes on a "classical sequential machine" to solve the problem with α set to 0.2. Slightly better solutions were found by setting α to 0.02, but the computation time increased to 12 hours. Unfortunately, no comparisons are provided with other problem-solving heuristics for the 1,000-city problem.

Fritzke and Wilke[37] describe a similar approach where units in the ring are dynamically created. They also introduce various simplifications in order to reduce complexity (e.g., the neighborhood of the winning unit only includes its two immediate neighbors). The authors were able to solve nine problems taken from the OR literature ranging in size from 8 to 2,392 cities. Since the optimum

for each one of these problems is known, they observed that their tours were never more than 10% longer than the optimum. However, the difference increases steadily from the 8-city problem (3%) to the 2,392-city problem (10%). It took about three hours to solve the largest problem on a SPARC 2 workstation.

The authors also observe that better solutions are produced by constructing a tour with the nearest neighbor heuristic and by applying the 2-opt exchange heuristic to this initial tour. On the 2,392-city problem, the solution obtained was about 6% larger than the optimum.

(c) Favata and Walker[34] add a third "normalizing" dimension to the coordinates of the cities, so that all the input vectors have the same length. In their experiments, the authors note that it sometimes occurs that two or more cities are mapped to the same unit in the ring. They interpret that phenomenon as meaning that the network does not care about the local ordering of those cities in the solution. For small problems, the use of a local optimization procedure for defining a complete ordering slightly improved the solutions. On larger problems, however, an arbitrary insertion proved to be equally good.

Favata and Walker were able to solve problems with up to 10,000 cities with their approach. On randomly generated 30-city ETSPs, their solutions are about 5% longer than the solutions produced by the simulated annealing heuristic. On a 1,000-city problem, they got a tour of length 26.82 after half an hour of computation time on a VAX 3600, as compared to a length of 25.95 with the simulated annealing heuristic after one hour of computation. They report that their approach is 10 times faster than simulated annealing on the 10,000-city problem, but no additional details are provided.

(d) In [20, 21], the authors use the "conscience" mechanism[30] to solve the problem related to the mapping of multiple cities to the same unit in the ring. Basically, the conscience mechanism inhibits units that are winning too often, so that other units also get a chance to win. The conscience mechanism avoids the artificial "deletion/duplication" step of the procedure described in [9], and the number of units on the ring is fixed to the number of cities N

throughout the algorithm. The computational results show that their approach (called the "guilty net") is better than the Hopfield-Tank model, but is outperformed by the elastic net on four sets of randomly generated problems ranging in size from 10 to 100 cities (see Table 5). The authors also note that their model converges much faster than the elastic net (500 iterations on the 100-city problems, as compared to 7,000 for the elastic net).

Number of Cities	Average Tour Length				
	EN	GN	HT	SA	Optimum[a]
10	2.45	2.78	2.92	2.74	2.43
30	4.51	4.81	5.12	4.75	4.21
50	5.58	6.21	6.64	6.13	5.43
100	8.23	8.81	NA	8.40	7.68

a Optimum estimated via Stein's formula[94]

EN Elastic Net
GN Guilty Net
HT Hopfield-Tank
NA Not Available
SA Simulated Annealing

Table 5. Comparison of Results for Five Solution Procedures on Random Problems

(e) Matsuyama[71] adds a new term, previously introduced in Durbin and Willshaw,[32] to the weight update equations (4.1)

$$T_j^{new} = T_j^{old} + \mu\, G(d'_{jj^*})\,(X - T_j^{old}) + \beta\,(T_{j+1} + T_{j-1} - 2T_j) \quad .$$

As previously noted in the description of the elastic net algorithm, the last term is aimed at shortening the length of the ring. It acts in synergy with the second term to allow the neighbors of the winning unit to move in the same direction as the winning unit.

Table 6 summarizes the main results with respect to the self-organizing map.

Paper	Largest TSP (Number of Cities)	Length of Tour from Self-organizing Map	TSP Heuristic	Length of Tour from TSP Heuristic
Fort (1988)	50 400	6.035 15.73	SA NA	5.735 NA
Angeniol et al. (1988)	1,000	18,036-18,800[a]	NA	NA
Fritzke and Wilke (1991)	2,392	NA[b]	NN+TO	NA
Favata and Walker (1991)	1,000 10,000	26.82[c] NA	SA NA	25.95 NA
Burke and Damany (1992)	100	8.81	SA EN	8.40 8.23

[a] Computation time of 20 minutes to 12 hours on a sequential computer (not identified)

[b] Computation time of 3 hours on a SPARC 2 workstation

[c] Computation time of 30 minutes on a VAX 3600

EN Elastic Net
NA Not Available
NN+TO Nearest Neighbor + 2-opt
SA Simulated Annealing

Table 6. Comparison of Results Produced by Self-Organizing Maps and TSP Heuristics

As a final remark, it is interesting to note that Van den Bout and Miller[99] describe a way to implement Kohonen's self-organizing map in hardware with VLSI components (i.e., a neural chip).

Assuming a clock frequency of 10 MHz, they report that their neural chip could process an input vector each 6.9 microseconds, regardless of the number of units in the network.

Section 5. Concluding Remarks

In summary, the self-organizing map and the elastic net can now find good solutions to very large ETSPs with up to 10,000 cities. Both approaches outperform the Hopfield-Tank network, which is restricted to relatively small problems with 200 cities or less (but can be applied as well to problems with no geometric interpretation). The results in the literature also indicate that the elastic net is better than the self-organizing map with respect to solution quality.[9,21]

On the other hand, the gap between the elastic net or the self-organizing map and the best heuristics of OR is still quite large. For example, the study of Fritzke and Wilke[37] shows that the tours generated by their self-organizing map are 5% to 10% longer than the optimal tours for ETSPs with 500 to 2,392 cities. These results can hardly be compared with the tours generated by the heuristics of Johnson and Bentley.[12,53] In the first case, solutions within 1% of the optimum are reported for problems with 10,000 cities, while in the second case, solutions within 4% of the optimum are reported on problems with 1,000,000 cities.

However, the rapid evolution of neural network technology can hardly be overlooked. Just eight years ago, Hopfield and Tank were struggling on a 10-city problem! The aim of this paper is to inform the OR community about the rapid progress made in the neural network field, and to demonstrate its potential for new research developments, in particular, via harmonious integration of OR and neural network technologies. As an example, Wacholder and his colleagues[100,101] used the well-known equivalence between the TSP and the multiple traveling salesman problem[11] to design Hopfield-Tank models for the latter problem.

Recently, various extensions to the TSP have been considered by neural network researchers. In [33, 42, 72, 73], multiple rings of output units are used to solve the multiple traveling salesman problem and the vehicle routing problem with the self-organizing map. In [89], an extension of the Hopfield-Tank model for dynamic TSPs is described. In these problems, the traveling costs change according to the time of the day. Finally, in [82], a neural network is

trained to perform a vehicle dispatching task. The network learned to weight various criteria used by human experts to evaluate dispatching situations and to assign service requests to vehicles. This application is well suited to neural network technology, because many subjective assessments are made by the dispatcher, and neural networks are known to be particularly useful for such ill-defined problems.

Acknowledgments. I would like to thank Professor Guy Lapalme and three anonymous referees for their very useful comments. Thanks also to Dr. Bruce L. Golden who gave to me some interesting papers on the subject. Finally, this research would not have been possible without the financial support of the Natural Sciences and Engineering Council of Canada (NSERC) and the Fonds pour la Formation de Chercheurs et l'Aide a la Recherche of Quebec Government (FCAR).

References

1. E.H.L. Aarts, J.H.M. Korst (1987), "Boltzmann Machines and their Applications", in Parallel Architectures and Languages in Europe, Lecture Notes in Computer Science 258, Volume 1, J.W. de Bakker, A.J. Nijman and P.C. Treleaven Eds, Springer-Verlag, Berlin, pp. 34-50.

2. E.H.L. Aarts, J.H.M. Korst (1989), "Boltzmann Machines for Travelling Salesman Problems", European Journal of Operational Research 39, pp. 79-95.

3. E.H.L. Aarts, J.H.M. Korst (1989), Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, Wiley, New-York, NY.

4. S. Abe (1989), "Theories on the Hopfield Neural Networks", in Proceedings of the Int. Joint Conf. on Neural Networks, Washington, DC, pp. I-557-564.

5. S. Abe (1991), "Global Convergence and Suppression of Spurious States of the Hopfield Neural Networks", in Proceedings of the Int. Joint Conf. on Neural Networks, Singapore, pp. 935-940.

6. S.V.B. Aiyer, M. Niranjan, F. Fallside (1990), "A Theoretical Investigation into the Performance of the Hopfield Model", IEEE Transactions on Neural Networks 1(2), pp. 204-215.

7. S.V.B. Aiyer, M. Niranjan, F. Fallside (1990), "On the Optimization Properties of the Hopfield Model", in Proceedings of the Int. Neural Network Conf., Paris, France, pp. 245-248.

8. Y. Akiyama, A. Yamashita, M. Kajiura, H. Aiso (1989), "Combinatorial Optimization with Gaussian Machines", in Proceedings of the Int. Joint Conf. on Neural Networks, Washington, DC, pp. I-533-540.

9. B. Angeniol, G. Vaubois, J.Y. Le Texier (1988), "Self-Organizing Feature Maps and the Travelling Salesman Problem", Neural Networks 1, pp. 289-293.

10. N. Bagherzadeh, T. Kerola, B. Leddy, R. Brice (1987), "On Parallel Execution of the Traveling Salesman Problem on a Neural Network Model", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. III-317-324.

11. M. Bellmore, S. Hong (1974), "Transformation of Multi-Salesman Problem to the Standard Traveling Salesman Problem", Journal of the ACM 21, pp. 500-504.

12. J. Bentley (1992), "Fast Algorithms for Geometric Traveling Salesman Problems", ORSA Journal on Computing 4(4), pp. 387-411.

13. G. Bilbro, R. Mann, T. Miller, W. Snyder, D.E. Van den Bout, M. White (1989), "Opimization by Mean Field Annealing", in Advances in Neural Information Processing Systems I, D. Touretzky Eds, Morgan Kaufmann, San Mateo, CA, pp. 91-98.

14. A. Bizzarri (1991), "Convergence Properties of a Modified Hopfield-Tank Model", Biological Cybernetics 64, pp. 293-300.

15. L. Bodin, B.L. Golden, A. Assad, M. Ball (1983), "Routing and Scheduling of Vehicles and Crews: The State of the Art", Computers & Operations Research 10(2), pp. 63-211.

16. M.C.S. Boeres, L.A.V. de Carvalho (1992), "A Faster Elastic Net Algorithm for the Traveling Salesman Problem", in Proceedings of the Int. Joint Conf. on Neural Networks, Baltimore, MD, pp. II-215-220.

17. P. Bozovsky (1990), "Discrete Hopfield Model with Graded Response", in Proceedings of the Int. Joint Conf. on Neural Networks, San Diego, CA, pp. III-851-856.

18. R.D. Brandt, Y. Wang, A.J. Laub (1988), "Alternative Networks for Solving the Traveling Salesman Problem and the List-Matching Problem", in Proceedings of the IEEE Int. Conf on Neural Networks, San Diego, CA, pp. II-333-340.

19. T. Bultan, C. Aykanat (1991), "Parallel Mean Field Algorithms for the Solution of Combinatorial Optimization Problems", in Artificial Neural Networks, vol. 1, T. Kohonen, K. Makisara, O. Simula, J. Kangas Eds, North-Holland, Amsterdam, pp. 591-596.

20. L.I. Burke (1992), "Neural Methods for the Traveling Salesman Problem: Insights from Operations Research", presented at the ORSA-TIMS Joint Meeting, San Francisco, CA, November 1992.

21. L.I. Burke, P. Damany (1992), "The Guilty Net for the Traveling Salesman Problem", Computers & Operations Research 19 (3/4), pp. 255-265.

22. L.I. Burke, J.P. Ignizio (1992), "Neural Networks and Operations Research: An Overview", Computers & Operations Research 19(3/4), pp. 179-189.

23. D.J. Burr (1988), "An Improved Elastic Net Method for the Traveling Salesman Problem", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp, I-69-76.

24. L.A.V. de Carvalho, V.C. Barbosa (1990), "A TSP Objective Function that Ensures Feasibility at Stable Points", in Proceedings of the Int. Neural Network Conf., Paris, France, pp. 249-253.

25. V. Cerny (1985), "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", Journal of Optimization, Theory and Applications 45, pp. 41-55.

26. J.H. Cervantes, R.R. Hildebrant (1987), "Comparison of Three Neuron-Based Computation Schemes", in Proceedings of the IEEE Int. Conf on Neural Networks, San Diego, CA, pp. III-657-671.

27. W.I. Clement, R.M. Inigo, E.S. McVey (1988), "Synaptic Strengths for Neural Simulation of the Traveling Salesman Problem", in Proceedings of Applications of Artificial Intelligence, SPIE 937, Orlando, FL, pp. 373-380.

28. R. Cuykendall, R. Reese (1989), "Scaling the Neural TSP Algorithm", Biological Cybernetics 60, pp. 365-371.

29. J.W. Denton, G.R. Madey (1989), "Impact of Neurocomputing on Operations Research", in Impacts of Recent Computer Advances on Operations Research, Publications in Operations Research Series 9, R. Sharda, B.L. Golden, E. Wasil, O. Balci and W. Stewart Eds, Elsevier Science Publishing Co., New-York, NY, pp. 302-312.

30. D. Desieno (1988), "Adding a Conscience Mechanism to Competitive Learning", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. I-117-124.

31. R. Durbin, R. Szeliski, A. Yuille (1989), "An Analysis of the Elastic Net Approach to the Traveling Salesman Problem", Neural Computation 1, pp. 348-358.

32. R. Durbin, D. Willshaw (1987), "An Analogue Approach to the Traveling Salesman Problem using an Elastic Net Method", Nature 326, pp. 689-691.

33. H. El-Gahziri (1991), "Solving Routing Problems by a Self-Organizing Map", in Artificial Neural Networks, T. Kohonen, K. Makisara, O. Simula, J. Kangas Eds, North-Holland, Amsterdam, pp. 829-834.

34. F. Favata, R. Walker (1991), "A Study of the Application of Kohonen-type Neural Networks to the Traveling Salesman Problem", Biological Cybernetics 64, pp. 463-468.

35. Y.P.S. Foo, H. Szu (1989), "Solving Large-Scale Optimization Problems by Divide-and-Conquer Neural Networks", in Proceedings of the Int. Joint Conf. on Neural Networks, Washington, DC, pp. I-507-511.

36. J.C. Fort (1988), "Solving a Combinatorial Problem via Self-Organizing Process: An Application of the Kohonen Algorithm to the Traveling Salesman Problem", Biological Cybernetics 59, pp. 33-40.

37. B. Fritzke, P. Wilke (1991), "FLEXMAP: A Neural Network for the Traveling Salesman Problem with Linear Time and Space Complexity", in Proceedings of the Int. Joint Conf. on Neural Networks, Singapore, pp. 929-934.

38. M. Gendreau, A. Hertz, G. Laporte (1992), "New Insertion and Post-Optimization Procedures for the Traveling Salesman Problem", Operations Research 40, pp. 1086-1094.

39. F. Glover (1989), "Tabu Search, Part I", ORSA J. on Computing 1(3), pp. 190-206.

40. F. Glover (1990), "Tabu Search, Part II", ORSA J. on Computing 2(1), pp. 4-32.

41. B.L. Golden, W.R. Stewart (1985), "Empirical Analysis of Heuristics", in The Traveling Salesman Problem. A Guided Tour of Combinatorial Optimization, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan and D.B. Shmoys Eds, Wiley, Chichester, pp. 207-249.

42. M. Goldstein (1990), "Self-Organizing Feature Maps for the Multiple Traveling Salesmen Problem", in Proceedings of the Int. Neural Network Conf. Paris, France, pp. 258-261.

43. K. Gutzmann (1987), "Combinatorial Optimization Using a Continuous State Boltzmann Machine", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. III-721-733.

44. S.U. Hedge, J.L. Sweet, W.B. Levy (1988), "Determination of Parameters in a Hopfield/Tank Computational Network", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. II-291-298.

45. S.U. Hedge, J.L. Sweet, W.B. Levy (1989), "Determination of Parameters in a Hopfield/Tank Computational Network", in Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton, T. Sejnowski Eds, Morgan Kaufmann, San Mateo, CA, pp. 211-216.

46. M. Held, R.M. Karp (1970), "The Traveling-Salesman Problem and Minimum Spanning Trees", Operations Research 18, pp. 1138-1162.

47. J. Hertz, A. Krogh and R.G. Palmer (1991), Introduction to the Theory of Neural Computation, Addison-Wesley, Redwood City, CA.

48. G.E. Hinton, T.J. Sejnowski (1986), "Learning and Relearning in Boltzmann Machines", in Parallel Distributed Processing: Explorations in the MicroStructure of Cognition, vol. 1, D.E. Rumelhart and J.L. McClelland Eds, The MIT Press, Cambridge, MA, pp. 282-317.

49. J.H. Holland (1992), Adaptation in Natural and Artificial Systems, The MIT Press, Cambridge, MA.

50. J.J. Hopfield (1982), "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", in Proceedings of the National Academy of Sciences, volume 79, pp. 2554-2558.

51. J.J. Hopfield, D.W. Tank (1985), "Neural Computation of Decisions in Optimization Problems", Biological Cybernetics 52, pp. 141-152.

52. G.J. Hueter (1988), "Solution of the Traveling Salesman Problem with an Adaptive Ring", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. I-85-92.

53. D.S. Johnson (1990), "Local Optimization and the Traveling Salesman Problem", in Automata, Languages and Programming, Lecture Notes in Computer Science 443, G. Goos and J. Hartmanis Eds, Springer-Verlag, Berlin, pp. 446-461.

54. A. Joppe, H.R.A. Cardon, J.C. Bioch (1990), "A Neural Network for Solving the Traveling Salesman Problem on the Basis of City Adjacency in the Tour", in Proceedings of the Int. Joint Conf. on Neural Networks, San Diego, CA, pp. III-961-964.

55. A. Joppe, H.R.A. Cardon, J.C. Bioch (1990), "A Neural Network for Solving the Traveling Salesman Problem on the Basis of City Adjacency in the Tour", in Proceedings of the Int. Neural Network Conf., Paris, France, pp. 254-257.

56. A.B. Kahng (1989), "Traveling Salesman Heuristics and Embedding Dimension in the Hopfield Model", in Proceedings of the Int. Joint Conf. on Neural Networks, Washington, DC, pp. I-513-520.

57. B. Kamgar-Parsi, B. Kamgar-Parsi (1987), "An Efficient Model of Neural Networks for Optimization", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. III-785-790.

58. B. Kamgar-Parsi, B. Kamgar-Parsi (1990), "On Problem-solving with Hopfield Neural Networks", Biological Cybernetics 62, pp. 415-423.

59. R.M. Karp, J.M. Steele (1985), "Probabilistic Analysis of Heuristics", in The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, E.L Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys Eds, Wiley, Chichester, pp. 181-205.

60. S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi (1983), "Optimization by Simulated Annealing", Science 220, pp. 671-680.

61. K. Knight (1990), "Connectionist Ideas and Algorithms", Communications of the ACM 33(11), pp. 59-74.

62. T. Kohonen (1988), Self-Organization and Associative Memory, Springer-Verlag, Berlin.

63. W.K. Lai, G.C. Coghill (1992), "Genetic Breeding of Control Parameters for the Hopfield-Tank Neural Net", in Proceedings of the Int. Joint Conf. on Neural Networks, Baltimore, MD, pp. IV-618-623.

64. G. Laporte (1992), "The Traveling Salesman Problem: An Overview of Exact and Approximate Algorithms", European Journal of Operational Research 59(2), pp. 231-247.

65. E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, D.B. Shmoys (1985), The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization, Wiley, Chichester.

66. B.C. Levy, M.B. Adams (1987), "Global Optimization with Stochastic Neural Networks", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. III-681-689.

67. S. Lin (1965), "Computer Solutions of the Traveling Salesman Problem", Bell System Technical Journal 44, pp. 2245-2269.

68. S. Lin, B. Kernighan (1973), "An Effective Heuristic Algorithm for the Traveling Salesman Problem", Operations Research 21, pp. 498-516.

69. R. Lister (1989), "Segment Reversal and the TSP", Proceedings of the Int. Joint Conf. on Neural Networks, Washington, DC, pp. I-424-427.

70. C.K. Looi (1992), "Neural Network Methods in Combinatorial Optimization", Computers & Operations Research 19 (3/4), pp. 191-208.

71. Y. Matsuyama (1990), "Competitive Self-Organization and Combinatorial Optimization: Applications to the Traveling Salesman Problem", in Proceedings of the Int. Joint Conf. on Neural Networks, San Diego, CA, pp. III-819-824.

72. Y. Matsuyama (1991), "Self-Organization via Competition, Cooperation and Categorization Applied to Extended Vehicle Routing Problems", in Proceedings of the Int. Joint Conf. on Neural Networks, Seattle, WA, pp. I-385-390.

73. Y. Matsuyama (1992), "Self-Organization Neural Networks and Various Euclidean Traveling Salesman Problems", Systems and Computers in Japan 23(2), pp. 101-112.

74. S. Mehta, L. Fulop (1990), "A Neural Algorithm to Solve the Hamiltonian Cycle Problem", in Proceedings of the Int. Joint Conf. on Neural Networks, San Diego, CA, pp. III-843-849.

75. E. Mjolsness (1987), "Control of Attention in Neural Networks", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. II-567-574.

76. M. Padberg, G. Rinaldi (1988), "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems", Technical Report R. 247, Istituto di Analisi dei Sistemi ed Informatica, Consiglio Nazionale delle Ricerche, Roma.

77. M. Padberg, G. Rinaldi (1990), "Facet Identification for the Symmetric Traveling Salesman Problem", Mathematical Programming 47, pp. 219-257.

78. J.H. Park, H. Jeong (1990), "Solving the Traveling Salesman Problem Using an Effective Hopfield Network", in Proceedings of the Int. Neural Network Conf., Paris, France, pp. 291-292.

79. C. Peterson (1990), "Parallel Distributed Approaches to Combinatorial Optimization: Benchmark Studies on Traveling Salesman Problem", Neural Computation 2, pp. 261-269.

80. C. Peterson, J.R. Anderson (1987), "A Mean Field Theory Learning Algorithm for Neural Networks", Complex Systems 1(5), pp. 995-1019.

81. C. Peterson, B. Soderberg (1989), "A New Method for Mapping Optimization Problems onto Neural Networks", Int. J. of Neural Systems 1, pp. 3-22.

82. J.Y. Potvin, Y. Shen, J.M. Rousseau (1992), "Neural Networks for Automated Vehicle Dispatching", Computers & Operations Research 19(3/4), pp. 267-276.

83. J. Ramanujam, P. Sadayappan (1988), "Optimization by Neural Networks", Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. II-325-332.

84. J. Ramanujam, P. Sadayappan (1989), "Parameter Identification for Constrained Optimization Using Neural Networks", in Proceedings of the 1988 Connectionist Models Summer School, D. Touretzky, G. Hinton, T. Sejnowski Eds, Morgan Kaufmann, San Mateo, CA, pp. 154-161.

85. H. Ritter, K. Schulten (1988), "Kohonen's Self Organizing Maps: Exploring their Computational Capabilities", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. I-109-116.

86. L. Rong, L. Ze-Min (1991), "Determination of the Parameters in a Modified Hopfield-Tank Model on Solving TSP", in Proceedings of the Int. Joint Conf. on Neural Networks, Singapore, pp. 2455-2460.

87. L. Rong, L. Ze-Min (1992), "Parameters Rules of the Hopfield-Tank Model on Solving TSP", in Proceedings of the Int. Joint Conf. on Neural Networks, Baltimore, MD, pp. IV-492-497.

88. D. Rosenkrantz, R. Sterns, P. Lewis (1977), "An Analysis of Several Heuristics for the Traveling Salesman Problem", SIAM J. Computing 6, pp. 563-581.

89. K. Shinozawa, T. Uchiyama, K. Shimohara (1991), "An Approach for Solving Dynamic TSP's Using Neural Networks", in Proceedings of the Int. Joint Conf on Neural Networks, Singapore, pp. 2450-2454.

90. B. Shirazi, S. Yih (1989), "Critical Analysis of Applying an Hopfield Neural Net Model to Optimization Problems", in Proceedings of the IEEE Int. Conf. on Systems, Man and Cybernetics, Cambridge, MA, pp. 210-215.

91. P.D. Simic (1990), "Statistical Mechanics as the Underlying Theory of Elastic and Neural Optimisations", Network 1, pp. 89-103.

92. M.W. Simmen (1991), "Parameter Sensitivity of the Elastic Net Approach to the Traveling Salesman Problem", Neural Computation 3, pp. 363-374.

93. S.D. Simmes (1987), "Program Parallel Computers using Energy Minimization Algorithms", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. II-205-211.

94. D. Stein (1977), "Scheduling Dial-a-Ride Transportation Systems: An Asymptotic Approach", Ph.D. Thesis, Harvard University, Cambridge, MA.

95. H. Szu (1988), "Fast TSP Algorithm Based on Binary Neuron Output and Analog Neuron Input Using the Zero-Diagonal Interconnect Matrix and Necessary and Sufficient Constraints of the Permutation Matrix", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. II-259-266.

96. D.A. Thomae, D.E. Van den Bout (1990), "Encoding Logical Constraints into Neural Network Cost Functions", in Proceedings of the Int. Joint Conf. on Neural Networks, San Diego, CA, pp. III-863-868.

97. D.E. Van den Bout, T.K. Miller III (1988), "A Traveling Salesman Objective Function that Works", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. II-299-303.

98. D.E. Van den Bout, T.K. Miller III (1989), "Improving the Performance of the Hopfield-Tank Neural Network through Normalization and Annealing", Biological Cybernetics 62, pp. 129-139.

99. D.E. Van den Bout, T.K. Miller III (1989), "TInMann: The Integer Markovian Artificial Neural Network", in Proceedings of the Int. Joint Conf. on Neural Networks, Washington, DC, pp. II-205-211.

100. E. Wacholder, J. Han, R.C. Mann (1988), "An Extension of the Hopfield-Tank Model for Solution of the Multiple Traveling Salesmen Problem", in Proceedings of the IEEE Int. Conf. on Neural Networks, San Diego, CA, pp. II-305-324.

101. E. Wacholder, J. Han, R.C. Mann (1989), "A Neural Network Algorithm for the Multiple Traveling Salesman Problem", Biological Cybernetics 61, pp. 11-19.

102. S.D. Wang, C.M. Tsai (1991), "Hopfield Nets with Time-Varying Energy Functions for Solving the Traveling Salesman Problem", in Proceedings of the Int. Joint Conf. on Neural Networks, Singapore, pp. 807-812.

103. D.M. Wells (1992), "Solving Degenerate Optimization Problems Using Networks of Neural Oscillators", Neural Networks 5, pp. 949-959.

104. D.J. Willshaw, C. Von der Malsburg (1979), "A Marker Induction Mechanism for the Establishment of Ordered Neural Mappings: Its Application to the Retinotectal Problem", Philosophical Transactions of the Royal Society, Series B 287, pp. 203-243.

105. G.V. Wilson, G.S. Pawley (1988), "On the Stability of the Travelling Salesman Problem Algorithm of Hopfield and Tank", Biological Cybernetics 58, pp 63-70.

106. X. Xu, W.T. Tsai (1991), "Effective Neural Algorithms for the Traveling Salesman Problem", Neural Networks 4, pp. 193-205.

107. C.S. Yu, W.D. Lee (1992), "Parallel Mean Field Annealing Neural Network for Solving the Traveling Salesman Problem", in Proceedings of the Int. Joint Conf. on Neural Networks, Baltimore, MD, pp. IV-532-536.

108. T.W. Yue, L.C. Fu (1990), "Ineffectiveness in Solving Combinatorial Optimization Problems Using a Hopfield Network: A New Perspective from Aliasing Effect", in Proceedings of the Int. Joint Conf on Neural Networks, San Diego, CA, pp. III-787-792.

109. A.L. Yuille (1990), "Generalized Deformable Models, Statistical Physics, and Matching Problems", Neural Computation 2, pp. 1-24.