Chapter 15

ARTIFICIAL NEURAL NETWORKS FOR COMBINATORIAL OPTIMIZATION

Jean-Yves Potvin
Département d'informatique et de recherche opérationnelle
and Centre de recherche sur les transports
Université de Montréal
C.P. 6128, succursale Centre-ville
Montréal (Québec), Canada H3C 3J7
E-mail: potvin@iro.umontreal.ca

Kate A. Smith
School of Business Systems
Faculty of Information Technology
Monash University
P.O. Box 63B
Victoria 3800, Australia
E-mail: kate.smith@infotech.monash.edu.au

1 INTRODUCTION

Artificial neural networks (ANNs) are extremely simplified models of their biological counterpart, the network of natural neurons known as the human brain. Consequently, we are not interested here in the study of the brain, which is the subject of neuroscience. Rather, we are interested to know what these networks of artificial units are, what they can do and how. ANNs were originally developed to provide a fundamentally new and different approach to information processing, when an algorithmic procedure for solving a problem is not known. As opposed to programmed computing, ANNs are capable of internally developing information processing capabilities for solving a problem when fed with appropriate information about the problem. Thus, they are often referred to as learning or adaptive models.

The history of ANNs dates back to the paper of McCulloch and Pitts (1943), when simple types of neural networks were shown to be able to learn arithmetic or logical functions. Important successes were witnessed in the late 50's and early 60's, with the development of the perceptron model and the first neurocomputers (Rosenblatt, 1958). By the end of the 60's, however, the field collapsed: a book by Minsky and Papert (1969), demonstrating that even the simple exclusive-or logical function could not be implemented with a perceptron, was devastating and diverted away research funding. After a dark period, ANNs emerged again in the early 80's with the support of John Hopfield, a renowned scientist, and the publication of an important book by Rumelhart and McClelland (1986), which introduced the backpropagation neural network model to the scientific community. This model extended the capabilities of its ancestor, the perceptron, allowing it to learn a much larger class of functions (including the exclusive-or logical function). Since that time, the field has continually expanded.

The first successes with ANNs were reported for the most part in pattern recognition, classification and prediction tasks. Application of ANNs to combinatorial optimization problems (COPs) dates back to 1985 when Hopfield and Tank solved small instances of the Traveling Salesman Problem (TSP) with a Hopfield neural network (Hopfield and Tank, 1985). The TSP is a classical combinatorial optimization problem, which is simple to state but difficult to solve. Basically, the objective is to find the shortest possible tour (or Hamiltonian cycle) through a set of N vertices so that each vertex is visited exactly once. This problem is known to be NP-hard (Garey and Johnson, 1979; Nemhauser and Wolsey, 1988), and cannot be solved exactly in polynomial time. 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 ANNs have been applied to the TSP (Potvin, 1993). The work of Hopfield and Tank was a first attempt in this direction and it generated much excitement in the neural network and operations research communities alike.

Many other types of COPs have since been tackled with ANNs in different application areas: routing and transportation, scheduling, cutting stock and packing, timetabling, telecommunications, and many others (Burke and Ignizio, 1992). In some cases, the results obtained are competitive with those reported with alternative techniques. In other cases, the results are not yet convincing. It is clear that the computational paradigm of ANNs, which is inherently parallel, distributed and adaptive cannot be fully exploited on current computer hardware. Their behavior must be simulated, thus leading to excessively large computation times. The development of suitable hardware for these models (often called neurocomputers) would thus be an important step toward their full recognition.

The classical backpropagation neural network model, although well suited for many learning tasks is not really indicated for combinatorial optimization. Consequently, ANNs applied to COPs are mostly based on three alternative models: Hopfield-Tank (H–T) and its variants, the elastic net (EN) and the self-organizing map (SOM). H–T provides a natural way to model many COPs and has been widely applied. EN and SOM provide an alternative, more efficient, approach for low-dimensional geometrical problems, like the TSP. These models are reviewed in the following.

2 HOPFIELD NEURAL NETWORKS

In his seminal paper of 1982, John Hopfield described a new way of modeling a system of neurons capable of performing computational tasks (Hopfield, 1982). Using a collection of binary-state neurons and a stochastic updating algorithm, these computational tasks were initially related to storage and retrieval of embedded memories. The computational capabilities of Hopfield's original model were expanded in Hopfield (1984) when he proposed a continuous version, and proved convergence of the model by demonstrating that the dynamics of the model minimized a constructed Liapunov function over time. From here, it became clear that Hopfield networks could be used to minimize any function provided the network parameters were set appropriately. The fact that the continuous version of the Hopfield network was designed to be implemented using electrical circuits also promised rapid computational ability.

This section first presents the two Hopfield neural network models: the discrete and stochastic model of 1982, and the continuous and deterministic model of 1984. The method of Hopfield and Tank (1985) for mapping a combinatorial optimization problem onto a Hopfield network is then described, using the TSP as an example. The section continues with a discussion of the criticisms of the approach. We then briefly review some of the many modifications and extensions that have been made to the original model and approach in an attempt to overcome these limitations. Finally, the reported performance of these Hopfield network models for combinatorial optimization across a range of benchmarked problems is discussed.

2.1 Discrete and Stochastic Hopfield Network

The original Hopfield network, as described in Hopfield (1982) comprises a fully interconnected system of n computational elements or *neurons*. In the following description, Hopfield's original notation has been altered where necessary for consistency. The strength of the connection, or weight, between neuron i and neuron j is determined by W_{ij}. This weight may be positive or negative depending on whether the neurons act in an excitatory or inhibitory manner (or zero if there is no interaction). Each neuron has an internal state u_i and an external state v_i. While the internal states are continuous valued, the external states are binary for this discrete model. The relationship between the internal and external states of the neurons can be shown as:

$$u_i(t+1) = \sum_{j=1}^{n} W_{ij} v_j(t) + I_i \tag{1}$$

$$v_i(t+1) = f(u_i) = \begin{cases} 1 & \text{if } u_i > 0 \\ 0 & \text{if } u_i \leq 0 \end{cases} \tag{2}$$

where I_i is a constant external input to neuron i and $f()$ is the transfer function between internal and external states. The connection weights W are also constant, and the only variable elements in the network are the internal and external states of the neurons that are updated over time. From equations (1) and (2) it is clear that the internal state of each neuron is calculated as the weighted sum of inputs from its connected neurons, with an additional constant input. The neuron will "fire" (as evidenced by an external state of 1), if it receives sufficient stimulation from its connecting neurons, otherwise the neuron's external state will be zero representing a dormant or "non-firing" state.

The neurons update themselves over time in a random sequence, thus the model is said to be discrete and stochastic. As the network updates, and provided the weight matrix is symmetric with non-negative diagonals, the following energy function is guaranteed to be minimized until the system converges to one of its stable states.

$$E_{\mathrm{d}} = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} v_i v_j - \sum_{i=1}^{n} I_i v_i \tag{3}$$

Using the terminology of operations research, the system of equations (1) and (2) perform a gradient descent on the energy function (3), with the neuron states $\boldsymbol{v}$ converging to one of its local minima. If the values of the weights $\boldsymbol{W}$ and external inputs $\boldsymbol{I}$ are fixed appropriately, this process can be used to minimize any quadratic function of binary variables.

2.2 Continuous and Deterministic Hopfield Network

Hopfield's subsequent modifications to the original 1982 model were driven by considerations of biological plausibility. In the biological system, u_i lags behind the instantaneous outputs v_j of the other neurons because of the input capacitance C_i of the cell membrane, the trans-membrane resistance R_i, and the finite impedance $R_{ij} = W_{ij}^{-1}$ between the output v_j and the cell body of neuron *i*. The external states of the neurons are now continuous valued between 0 and 1, rather than binary in the earlier model, and represent an average "firing rate". Hopfield (1984) modeled this more biologically based system using the following resistance-capacitance differential equation to determine the rate of change of u_i, and hence the time evolution of the continuous Hopfield network:

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \sum_{j=1}^{n} W_{ij} v_j - \frac{u_i}{\tau} + I_i \tag{4}$$

$$v_i = f(u_i) \quad \text{and} \quad \tau = R_i C_i \tag{5}$$

where the transfer function $f()$ is now a continuous sigmoidal function such as:

$$v_i = f(u_i) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_i}{T}\right)\right) \tag{6}$$

and T is a parameter used to control the slope of the transfer function. τ is the value of the time constant of the amplifiers, and without loss of generality can be assigned a value of unity, provided the time step of any discrete-time simulation of equation (4) is considerably smaller than unity. This same set of equations represents a resistively connected network of electronic amplifiers, and thus the system can be implemented with electrical circuits. We refer the interested reader to (Hopfield, 1984) for details of this implementation.

Similar to the original discrete model, the dynamics of this continuous model also minimize an energy function over time guaranteed to converge to stable states. This energy function is:

$$E_c = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij} v_i v_j - \sum_{i=1}^{n} I_i v_i + \int_0^{v_i} f_i^{-1}(\mathrm{v})\,\mathrm{d}\mathrm{v} \tag{7}$$

Hopfield (1984) showed that provided the weight matrix is symmetric, this function is a Liapunov function for the system of equations (4) and (5). Furthermore, if the slope of the transfer function (6) is particularly high (i.e., T is near zero), then the transfer function (6) approximates the behavior of the discrete version given by equation (2), and the integral term of equation (7) vanishes. Consequently, the local minima of E_c coincide with the local minima of E_d, and all these local minima lie at the vertices of the unit hypercube resulting in binary values for $\boldsymbol{v}$. Thus, for T near zero, the continuous Hopfield network converges to a 0–1 solution in $\boldsymbol{v}$ which minimizes the energy function E_{d} given by (3).

Thus, there are two Hopfield neural network models available: a discrete version and a continuous version. The continuous version can either be implemented using electrical circuits, or simulated on a digital computer using an approximation to the differential equation (4) such as the Euler approximation. The accuracy of this approximation depends on parameters like the time step of the discretization, and affects the degree to which the discretized dynamics converge on the Liapunov energy function. Clearly, a small time step will approximate the dynamics well, ensuring gradient descent on the energy function. This issue is discussed further in Section 4, along with a variety of other practical issues.

2.3 Adaptation to Solve Combinatorial Optimization Problems

In 1985, John Hopfield teamed together with David Tank to extend the applications of his model to include solving combinatorial optimization problems (Hopfield and Tank, 1985). Hopfield and Tank (H–T) realized that networks of neurons with this basic organization could be used to compute solutions to specific optimization problems by selecting weights and external inputs which appropriately represent the function to be minimized and the desired states of the problem. The updating of the neurons according to the differential equations given by (4) and (5) (or even the discrete versions (1) and (2)) ensures that both the energy function and the optimization problem are simultaneously minimized over time. The analog nature of the neurons and the hardware implementation of the updating procedure could be combined to create a rapid and powerful solution technique.

Using the method proposed by Hopfield and Tank, the network energy function is made equivalent to the objective function of the optimization problem needing to be minimized, while the constraints of the problem are included in the energy function as penalty terms.

Consider the quadratic formulation of the N-city TSP, given the binary decision variable

$$X_{ij} = \begin{cases} 1 & \text{if city } i \text{ is in position } j \\ 0 & \text{otherwise} \end{cases}$$

and the constant distance matrix d_{ik} representing the distance between cities i and k:

$$\text{minimise} \sum_{i=1}^{N} \sum_{\substack{k=1 \\ k \neq i}}^{N} \sum_{j=1}^{N} d_{ik} X_{ij} (X_{k,j+1} + X_{k,j-1}) \tag{8}$$

$$\text{subject to} \sum_{i-1}^{N} X_{ij} = 1 \quad \text{for all } j \tag{9}$$

$$\sum_{j=1}^{N} X_{ij} = 1 \quad \text{for all } i \tag{10}$$

$$X_{ij} \in \{0, 1\} \quad \text{for all } i, j \tag{11}$$

Apart from being a well benchmarked problem, the TSP is a useful problem to consider since its form is that of a quadratic assignment problem. Thus the methods used by Hopfield and Tank for mapping the optimization problem onto a Hopfield neural network can be generalized to a wide range of problems with similar constraint and objective types.

The first step is to construct an energy function representation of the complete optimization problem using a penalty parameter approach, so that all objective functions and constraints are integrated into a single function which needs to be minimized. This is achieved by observing that a constraint of the form (9) can be enforced by ensuring minimization of the quantity

$$\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N} X_{ij}X_{kj} \quad \text{for all } j.$$

That is, a constraint requiring a single “1” in each column can be enforced by minimizing the pairwise product of elements in each column. If there is no more than one “1” in the column, then this term will be at its minimum value of zero. If there is more than one “1” in the column, then this term will be greater than zero. A similar term can be constructed to enforce the row constraint (10). Naturally, these terms will also be zero if there are no “l”s in each row or column as well. Since we need exactly one “1” per column and row, we will also need an additional term to force N elements of the solution matrix X to be “1”.

The complete set of constraints can therefore be enforced through minimization of penalty terms, and when we add the objective function to these terms, we arrive at the H–T energy function for the TSP:

$$E = \frac{A}{2}\sum_{j=1}^{N}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N} X_{ij}X_{kj} + \frac{B}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{\substack{l=1\\l\neq j}}^{N} X_{ij}X_{il} + \frac{C}{2}\left(\sum_{i=1}^{N}\sum_{j=1}^{N} X_{ij} - N\right)^{2}$$

$$+ \frac{D}{2}\sum_{i=1}^{N}\sum_{\substack{k=1\\k\neq i}}^{N}\sum_{j=1}^{N} d_{ik}X_{ij}(X_{k,i+1} + X_{k,i-1}) \qquad (12)$$

X_k,j+1 + X_k,j-1

The first two terms enforce no more than one “1” per column and row respectively, the third term ensures that there are N elements “on” in the solution matrix, and the final term minimizes the tour length. The penalty parameters A, B, C and D need to be fixed at values that reflect the relative importance of these terms in the minimization process. If A, B and C are not large enough relative to D, then the resulting solution may be infeasible. Similarly, if D is not large enough, the solution may be feasible but the tour length may be larger than the optimal value. Hopfield and Tank (1985) used values of $A = B = D = 500$ and $C = 200$ to balance these terms.

Now that the energy function has been constructed, the next step is to derive the Hopfield network weights and external inputs so that the energy function is minimized by the network dynamics. For this we need to expand and rearrange the energy function (12) so that it is in the same form as the standard Hopfield energy function E_{d}

$$E_{\mathrm{d}} = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N} W_{ijkl}X_{ij}X_{kl} - \sum_{i=1}^{N}\sum_{j=1}^{N} I_{ij}X_{ij} \tag{13}$$

which has been modified from (3) to reflect the fact that the neurons X_{ij} for our TSP problem are two dimensional, compared to the linear array of neurons v_i used in the standard Hopfield network. Once the forms of these two functions (12) and (13) are similar, the network weights $\boldsymbol{W}$ and external inputs $\boldsymbol{I}$ can be read as the coefficients of the quadratic and linear terms respectively. To ensure equivalence of the two functions, the summations of each term in (12) need to be extended across all relevant dimensions *(i,j,k,l* for quadratic terms and *i, j* for linear terms). Thus the Kronecker-Delta symbol is incorporated into each term of (12) where necessary:

$$\delta_{ab} = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{if } a \neq b \end{cases}$$

Expanding (12) and rearranging the terms into quadratic, linear, and constant terms, thus yields:

$$\begin{aligned} E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\sum_{l=1}^{N}[& -A\delta_{ik}(1-\delta_{jl}) - B\delta_{jl}(1-\delta_{ik}) - C \\ & - D\delta_{ik}(\delta_{l,j+1} + \delta_{l,j-1})]X_{ij}X_{kl} - \sum_{i=1}^{N}\sum_{j=1}^{N}[CN]X_{ij} + \frac{CN^2}{2} \end{aligned} \tag{14}$$

Dd_i,k(1-δ_i,k)(δ_l,j+1, δ_l,j-1)

Comparing (14) to the standard Hopfield energy function (13) then, it is clear that the network parameters are:

$$\begin{aligned} W_{ijkl} &= -A\delta_{ik}(1-\delta_{jl}) - B\delta_{jl}(1-\delta_{ik}) - C - D\delta_{ik}(\delta_{l,j+1} + \delta_{l,j-1}) \\ I_{ij} &= CN \end{aligned} \tag{15}$$

Dd_i,k(1-δ_i,k)(δ_l,j+1, δ_l,j-1)

The constant term in equation (14) can be ignored since it merely reflects a shift upwards and does not affect the location of the minima of the function.

Now that the network weights and external inputs are determined, the Hopfield network can be initialized (using random values for the initial states), and updated according to equations (4) and (5) (or equations (1) and (2) for a purely discrete version). This updating is guaranteed to minimize the Hopfield energy function (13), and since this function is equivalent to the TSP energy function (12) and (14), then the resulting solution matrix $\boldsymbol{X}$ will provide a local minima of the TSP energy function. The quality and feasibility of this solution depends on the choice of penalty parameters *A, B, C* and *D*, as well as the initialization of the neurons, and the accuracy with which the dynamics of the differential equation (4) can be simulated if the continuous model is chosen.

The complete procedure is summarized in pseudocode form below:

```
Step 0:  Preliminary Tasks
   0.1 Construct an energy function for the optimization
       problem using a penalty parameter approach.
   0.2 Expand energy function and infer network weights and
       external inputs.
Step 1:  Initialization Tasks
   1.1 Initialize neuron states to random values.
   1.2 Select A,B,C,D.
   1.3 Select T, the parameter of the continuous transfer
       function, and the value of the discrete time step if
       simulating the continuous model.
Step 2:  If energy function has converged to local minimum
         proceed to Step 5, otherwise proceed to step 3
Step 3:  Repeat n times:
   3.1 Randomly choose a neuron i to update (if using
       discrete time dynamics).
   3.2 Update $u_i$ and $v_i$ using equations (1)-(2) or (3)-(4).
Step 4:  Go back to Step 2.
Step 5:  Examine final solution matrix and determine
         feasibility and optimality.
Step 6:  Adjust parameters A,B,C,D if necessary to obtain a
         satisfactory solution, re-initialize neuron states,
         and repeat from Step 2.
```

Clearly, one of the main limitations of the H–T approach to solving combinatorial optimization is the difficulty in choosing appropriate penalty parameters. In addition to this difficulty, the dynamics of the Hopfield network perform a gradient descent on the energy function, and thus converge to the first local minimum they encounter. Coupling these two issues, it seems likely that the H–T approach may yield solutions of poor quality. Wilson and Pawley (1988) first published these findings nearly three years after Hopfield and Tank's original paper was published. In doing so, they raised serious doubts as to the validity of the H–T approach to solving optimization problems, which seemingly served to dampen the enthusiasm surrounding the field.

2.4 Extensions

Since Wilson and Pawley's results were published, it has been widely recognized that the H–T formulation is not ideal, even for problems other than the TSP. The problem of optimally selecting the penalty parameters is not trivial and much work has been done to try to facilitate this process (Hedge et al., 1988; Kamgar-Parsi, 1992; Lai and Coghill, 1992). Many other researchers believed that the H–T energy function needed to be modified before any progress would be made, and considerable effort has also been spent in this area (Brandt et al., 1988; Van den Bout and Miller, 1988). One obvious improvement to the H–T approach to solving the TSP is to reduce the number of terms needed to represent the constraints by using the form $(\sum_{i=1}^{N} X_{ij} - 1)^2$ to represent the column constraints, for example. This eliminates the need for the third term in equation (12), thus the penalty parameter C is also eliminated.

Perhaps the most important breakthrough in the field, however, came from the valid subspace approach of Aiyer et al. (1990), and the subsequent work of Gee (1993). Their idea is to represent the constraint set as a hyperplane, and encourage the solution to lie upon it. This is achieved by including a single term in the energy function for the constraints which attempts to minimize the deviation between the solution matrix and the constraint plane, or valid subspace. A single penalty parameter needs to be selected, which if large enough, will guarantee the feasibility of the final solution.

Some researchers have also attempted to address the limitations of the H–T approach by considering alternative representations of constraints, suitable values for penalty parameters, and other modeling issues. The majority of other researchers in the field, however, have focused on the limitation of the Hopfield network dynamics. By extending the network dynamics to include stochasticity and hill-climbing capabilities, various methods have emerged that attempt to avoid the many local minima of the energy function.

The variations of the Hopfield network that have been proposed can be broadly categorized as either deterministic or stochastic. The deterministic approaches include problem specific enhancements such as the "divide and conquer" method of Foo and Szu (1989) for solving the TSP, deterministic hill-climbing such as the "rock and roll" perturbation method of Lo (1992), and the use of alternative neuron models within the Hopfield network such as the winner-take-all neurons used by Amartur et al. (1992) to improve the feasibility of the solutions. Stochastic approaches address the problem of poor solution quality by attempting to escape from local minima. There are basically four main methods found in the literature to embed stochasticity into the Hopfield network:

1. replace sigmoidal transfer function with a stochastic decision-type function;
2. add noise to the weights of the network;
3. add noise to the external inputs of the network;
4. any combination of the above methods.

The *Boltzmann machine* (Aarts and Korst, 1989; Hinton et al., 1984) utilizes the first method based on a discrete Hopfield model. The inputs are fixed, but the discrete transfer function is modified to become probabilistic. Much like simulated annealing (Kirkpatrick et al., 1983), the consequence of modifying the binary transfer level of each neuron is evaluated according to the criteria of the Boltzmann probability factor. This model is able to escape from local minima, but suffers from extremely large computation times. In order to improve the efficiency and speed of the Boltzmann machine, Akiyama et al. (1989) proposed Gaussian machines which combine features of continuous Hopfield networks and the Boltzmann machine. Gaussian machines have continuous outputs with a deterministic transfer function like the Hopfield network, but random noise is added to the external input of each neuron. This noise is normally distributed (or Gaussian) with a mean of zero and a variance controlled by a temperature parameter *T*. However, based upon Szu's fast simulated annealing (Szu and Hartley, 1987) which uses Cauchy noise to generate new search states and requires only a $T/\log(T)$ cooling schedule, the Cauchy machine (Szu, 1988; Takefuji and Szu, 1989) was proposed as an improvement to solution quality. The Cauchy distribution is thought to yield a better chance of convergence to the global minimum than the Gaussian distribution. Furthermore, Cauchy noise produces both local random walks and larger random leaps in solution space, whereas Gaussian noise produces only local random walks (Takefuji and Szu, 1989). The noise is incorporated into the transfer function, while the outputs of the Cauchy machine are binary. In the high-gain limit of the stochastic transfer function *(T* near zero), the Cauchy machine approaches the behavior of the discrete and deterministic Hopfield network. Another stochastic approach which has been very successful is mean-field annealing (Peterson and Soderberg, 1989; Van den Bout and Miller, 1989, 1990), so named because the model computes the mean activation levels of the stochastic binary Boltzmann machine.

2.5 Performance

Hopfield and Tank successfully applied their approach to several optimization problems including an analog-to-digital converter, a signal decision circuit, and a linear programming model (Tank and Hopfield, 1986). It was, however, their results for the combinatorial TSP that attracted the most attention. Hopfield and Tank (1985) simulated a network of 10 cities (100 neurons), chosen at random on the interior of a 2-dimensional unit square. Their results for small-sized problems were quite encouraging. For a 10 city problem, and for 20 random starts, 16 converged to valid tours. About 50% of the trials produced one of the two known shortest tours. Hopfield and Tank then studied a 30 city (900 neuron) problem. Since the time required to simulate the differential equations on a computer scales worse than $O(n^3)$, their results were fragmentary. They were unable to find appropriate penalty parameters to generate valid tours, and commented that "parameter choice seems to be a more delicate issue with 900 neurons than with 100". In fact, their best solution was around 40% away from the best known solution of Lin and Kernighan (1973) on the same 30 city problem.

Since then, the many modifications to the original H–T approach have seen considerable improvement in these results. A recent fuzzy modification of Aiyer's subspace approach yielded nearest-city quality tours for up to 100 randomly generated cities (Wolfe, 1999). Peterson and Soderberg reported solutions for 200 cities using a mean field annealing neural network that were only slightly worse than simulated annealing results (Peterson and Soderberg, 1993). These results are still a long way from those that can be obtained by well known heuristics. For example, the iterated Lin-Kernighan heuristic can routinely find solutions within 1 % of optimal for problems with thousands of cities (Johnson, 1990). Even other neural network approaches such as the deformable template methods discussed in the next section yield considerably better results than the Hopfield variations seem capable of.

The advantage of the H–T approach to combinatorial optimization however lies in its generalization abilities. The H–T approach can be applied to any combinatorial optimization problem that can be formulated within quadratic terms. It does not rely on the geometry of the problem like many of the TSP heuristics or the deformable template methods. The many variations of the Hopfield network that have emerged over the last decade or so have been applied to a wide range of classical combinatorial optimization problems including assignment problems, constraint satisfaction problems, graph problems, integer programming, and scheduling problems to name a few. We refer the interested reader to Smith (1999) for a survey of these and other applications. Many of the results are competitive with other metaheuristic approaches. One of the deficiencies of the literature in this area, however, is the fact that few studies are established as comparative analyses, aimed to determine the competitiveness of the proposed neural network approach with the best known heuristic or metaheuristic approaches to the same problem. This makes a true evaluation of the performance of Hopfield-type models difficult. As Looi (1992) noted, "although there is a large collection of operations research based and other methods for solving all of these problems, comparisons between neural network methods with existing methods have been lacking". Solutions to this problem in the form of guidelines for experiments have now been published (Barr et al., 1995; Hooker, 1995) and we hope that researchers will soon provide enough studies of this nature that an accurate evaluation of the performance and potential of Hopfield-type neural networks on a wide variety of problems can be established.

3 DEFORMABLE TEMPLATES

Elastic nets (EN) and Self-organizing maps (SOM), often referred to as deformable templates, provide alternatives for solving low-dimensional problems with a geometric interpretation, like the Euclidean TSP. These models are fundamentally different from H–T, as they evolve in a low-dimensional continuous search space. In the following, we describe both models for solving the Euclidean TSP. We then establish some relationships between the two models and present a few extensions.

3.1 Elastic Net

The elastic net (EN) of Durbin and Willshaw (1987), originated from a previous work by Willshaw and von der Malsburg (1979). It is an iterative procedure where *M* points, with *M* typically larger than the number of vertices (or cities) *N*, are lying on a circular ring or "rubber band" originally located at the center of gravity of the vertices. The rubber band is gradually elongated until it is sufficiently close to each vertex to define a tour. During that process two forces apply: one for minimizing the length of the ring, and the other for minimizing the distance between the vertices and the ring. These forces are gradually adjusted as the procedure evolves.

Figure 15.1 (a)–(c) 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 vertices in the Euclidean plane. When there is a point on the ring sufficiently close to each vertex, a solution is obtained, as shown in Figure 15.1(d). This model will now be presented more formally, using a pseudocode notation. Let X_i be the coordinates of vertex *i*, $i = 1,\ldots, N$, Y_j the coordinates of ring point *j*, $j = 1,\ldots, M$, and $d_{X_iY_j}$ the Euclidean distance between *i* and *j*. We have:

```
Step 0: $K \leftarrow K_0; Y_j \leftarrow Y_j^0,\ j = 1,\ldots,M$;
Step 1:  Repeat rep times
    1.1 Update the coordinates $Y_j$ of ring
        point $j, j=1,\ldots,M$;
    1.2 If $\min_{j=1,\ldots,M} d_{X_iY_j} \le \varepsilon, i = 1,\ldots,N$, then STOP;
Step 2: $K \leftarrow \alpha K (0 < \alpha < 1)$;
Step 3: Go back to Step 1.
```

Step 0 initializes the scale parameter *K* (see below) and selects an initial location for the points on the ring. In Step 1, the points migrate towards the vertices through

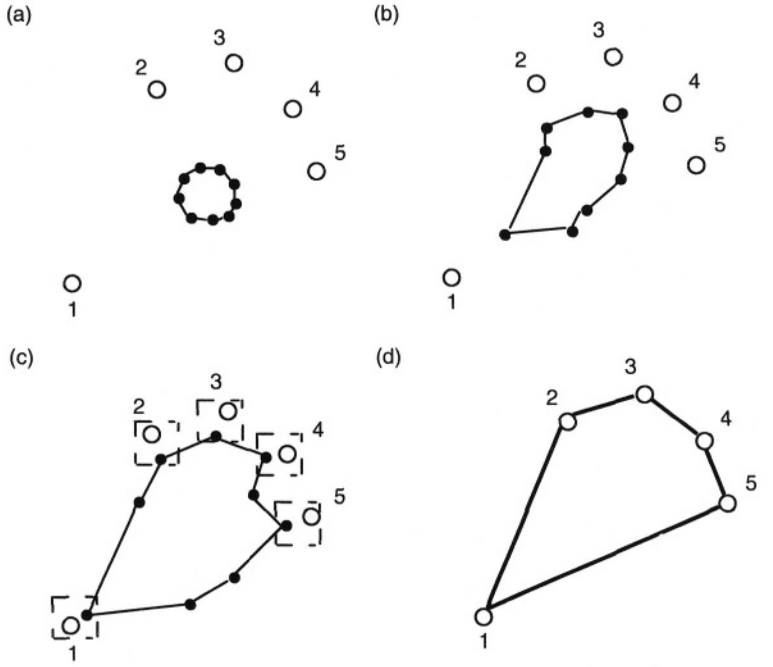

Figure 15.1. Evolution of the elastic net over time (a)–(c) and the final tour 1-2-3-4-5 (d).

an iterative procedure governed by parameter *K*. After a fixed number of iterations, related to the size of the problem, the value of parameter *K* is slightly reduced and the migration process is pursued with this new value. This is repeated until there is a point on the ring sufficiently close to each vertex, as specified by the tolerance ε. An alternative or additional stopping criterion for EN is to iterate until some preset $K_{\min}$ value is reached and then, to associate each vertex with the closest ring point. Parameter *K* is reminiscent of the temperature parameter in the simulated annealing algorithm, as its value must be progressively reduced according to a prespecified "cooling schedule" to obtain a good solution to the problem.

In Step 1.1, the coordinates of each ring point *j* are updated as follows:

$$Y_j \leftarrow Y_j + \Delta Y_j$$

where

$$\Delta Y_j = \alpha \sum_{i=1,\ldots,N} w_{ij} \left(X_i - Y_j\right) + \beta K \left(Y_{j+1} + Y_{j-1} - 2Y_j\right) \qquad (16)$$

$$w_{ij} = \frac{\phi\left(d_{X_i Y_j}, K\right)}{\sum_{k=1,\ldots,M} \phi\left(d_{X_i Y_k, K}\right)}, \quad i = 1, \ldots, N \qquad (17)$$

$$\phi(d, K) = e^{-d^2/2K^2} \qquad (18)$$

where α, β are constant parameters and w_{ij} is a normalized measure of the "attraction" of vertex *i* on ring point *j*. In equation (16), the α term drives the points on the ring

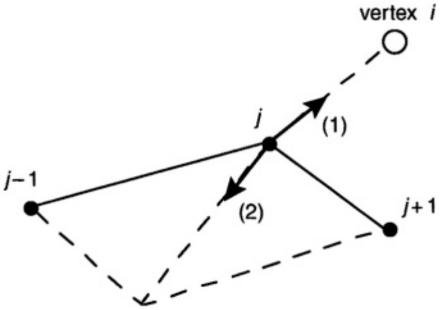

Figure 15.2. Forces that apply on ring point *j*.

towards the vertices, and the β term keeps neighboring points on the ring together during the migration process to produce a short tour (i.e., neighboring points are associated with vertices that are close in distance). These two forces are illustrated in Figure 15.2.

In this figure, force (1) is derived from the α term, and drives point *j* towards vertex *i*. Force (2), which is derived from the β term, is more easily understood by considering the equivalence

$$Y_{j+1} + Y_{j-1} - 2Y_j = (Y_{j+1} - Y_j) + (Y_{j-1} - Y_j). \tag{19}$$

It thus defines a tension on the ring that keeps neighboring points together. Through parameters α and β, the relative strength of the two forces can be regulated.

It is work noting that the update equations (16) can be expressed as the derivative of an appropriate energy function E_K, namely

$$\Delta Y_j = -K \frac{\partial E_K}{dY_j} \tag{20}$$

where

$$E_K = -\alpha K \sum_{i=1,\ldots,N} \ln \sum_{j=1,\ldots,M} \phi\left(d_{X_i Y_j}, K\right) + \frac{\beta}{2} \sum_{j=1,\ldots,M} d^2_{Y_j Y_{j+1}} \tag{21}$$

This algorithm thus finds a local minimum of this energy function by performing a gradient descent in a continuous two-dimensional Euclidean space. When *K* approaches 0 and the ratio of *M* to *N* approaches infinity, minimizing the energy is equivalent to minimizing the total length of the ring and, thus, the solution value. Since the shape of the energy function and its local minima change with *K,* the function is gradually modified through a slow reduction of the value of parameter *K* until the minima correspond to good TSP tours.

3.2 Self-organizing Map

A self-organizing map is an instance of the so-called competitive neural networks (Kohonen, 1982, 1988). It is composed of a layer of input units fully connected to a layer of output units, the latter being organized according to a particular topology, such as a ring structure. Self-organizing maps basically produce topological mappings

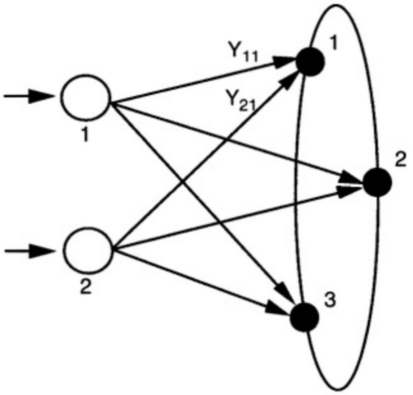

Figure 15.3. A SOM.

from high-dimensional input spaces to low-dimensional output spaces. In the case of a ring structure, the p-dimensional input vectors are associated with output units or 1-dimensional positions on the ring. The mapping is such that two input vectors that are close in the input space will be associated with units that are close on the ring.

In Figure 15.3, a SOM with P = 2 white input units and M = 3 black output units (indexed from 1 to 3) is illustrated.

In Figure 15.3, Y_{11} and Y_{21} denote the weights on the connections from the two input units to output unit 1, and $Y_1 = (Y_{11}, Y_{21})$ is the weight vector associated with output unit 1. In a TSP context, each input vector corresponds to the coordinates of a vertex. We then want vertices that are close in distance to be associated with units that are close on the ring to produce a short tour.

This is obtained through the following iterative adjustment of the connection weights. Let assume that we have a SOM with two input units and M output units on a ring, each with weight vector $Y_j = (Y_{1j}, Y_{2j}), j = 1, \ldots, M, M \geq N$. Let $X_i = (x_i, y_i)$ be the coordinates of vertex i, $i = 1, \ldots, N$ and $d_{X_i Y_j}$, the Euclidean distance between vertex i and output unit j. Then, we have:

```
Step 0: Initialization. $i \leftarrow 0$;   $Y_j \leftarrow Y_j^0$;   $j = 1, \ldots, M$;
Step 1: Competition.
    1.1 $i \leftarrow (i + 1) \bmod (N + 1)$ ;
    1.2 $o_j \leftarrow d_{X_i Y_j}, j = 1, \ldots, M$;
    1.3 $o_j \leftarrow \min_{j=1,\ldots,M}\{o_j\}$;
Step 2: Weight adjustment.
    $Y_j \leftarrow Y_j + \mu f(j, j^*)(X_i - Y_j), j = 1, \ldots, M, 0 < \mu < 1$;
Step 3: If $\min_{j=1,\ldots,M} d_{X_i Y_j} \leq \varepsilon, i = 1, \ldots, N$, then STOP;
Step 4: Go back to Step 1.
```

In Step 1, the winning output unit j^* is the one with the closest weight vector (in Euclidean distance) to the current vertex. In Step 2, function f is typically a decreasing function of the lateral distance between output units j and j^* on the ring (i.e., if there are k units on the ring between the two, the lateral distance is k+1) and its range is the interval [0,1]. Thus, the weight vector of the winning unit j^* and the weight vectors of units that are close to j^* on the ring all move towards the current vertex, but with decreasing intensity as the lateral distance to the winning unit increases. Typically, function f is modified as the algorithm unfolds to gradually reduce the magnitude of the weight adjustment. At the start, all units that are close to the winning unit on the ring "follow" that unit in order to move in the same area. At the end, only the weight vector of the winning unit significantly moves towards the current vertex.

This iterative adjustment procedure is repeated, through multiple passes over the set of vertices (c.f., the modulo operator in Step 1.1) until there is a weight vector sufficiently close to each vertex. Other or additional stopping criteria may be considered like a fixed number of passes through the vertices or a stable competition, when it is observed that the winning unit for each vertex does not change anymore from one pass to another. The association of each vertex with a weight vector (i.e., an output unit with an index or position on the ring) produces a tour. If we consider the two-dimensional weight vector of each output unit on the ring as the location of that unit in the Euclidean space, Figure 15.1 is still a good way of visualizing the evolution of the SOM with the output units on the ring migrating towards the vertices.

3.3 Extensions

Since both EN and SOM exploit the geometry of the problem, they have mostly been applied to the TSP. A few extensions are reported in the literature for the mutiple TSP (Goldstein, 1990) and vehicle routing problems (Ghaziri, 1991, 1996; Matsuyama, 1991; Vakhutinsky and Golden, 1994; Potvin and Robillard, 1995). In these applications, mutiple tours are formed through the migration in parallel of multiple rings. In the case of the VRP, the capacity or time constraints typically break the geometrical nature of the problem and the algorithm must be modified accordingly. The SOM described in (Ghaziri, 1991), for example, involves competition for the current vertex at two different levels: one within each ring based on geometric properties and the other among the rings to take into account the capacity constraints. Some recent papers have also generalized the SOM for applications in non geometric contexts, like the multidimensional knapsack problem and the generalized quadratic assignment problem (Glover, 1994; Smith, 1995). These approaches depart from the traditional ring structure and exploit more complex mappings and topologies.

3.4 Performance

Both SOM and EN are closely related. They both involve migration of a ring towards vertices, although the mechanism for updating the location of the ring points is different. In the case of EN, the update is defined through the minimization of an appropriate energy function. The SOM does not require such a function.

It is difficult to get an accurate picture of the performance of the two models from the current literature. The results are scattered in different journals from different research area. Computation times are often missing and comparisons with alternative methods are rare. Furthermore, the problems are not taken from standard benchmarks. Overall, the literature indicates that both EN and SOM outperform the Hopfield model on the TSP. For problems with up to a few hundred cities, EN often ends up with slightly better solutions than SOM (Angeniol et al., 1988). However, it is also more computationally expensive, as it requires more iterations to converge (c.f., the slow cooling schedule of scale parameter K). SOM scales up better and has been applied on much larger problems. In (Favata and Walker, 1991), for example, the authors report results on problems with up to 10,000 vertices. The solutions obtained were about 5% worse than those produced by a simulated annealing heuristic.

4 PRACTICAL ISSUES

While the previous sections presented a review of Hopfield neural networks, elastic nets and self-organizing maps, this section aims to discuss some of the more practical issues affecting their performance. These have been separated into issues affecting the efficiency of the models, and further issues that have proven to be problematic for researchers and need careful consideration.

4.1 Hopfield Model

4.1.1 Efficiency Issues

The efficiency of the Hopfield network for solving combinatorial optimization depends significantly on the way in which the problem is mapped onto the network. The encoding of the variables, constraints, and objective function into the Hopfield energy function and the values of the penalty parameters combine to determine the complexity of the energy surface. This, in turn, affects the degree to which the Hopfield network dynamics are able to search for local minima.

Problem Representation It is interesting to note that all of the Hopfield network applications to the TSP have been based on the formulation used by Hopfield and Tank which uses a decision variable denoting if a city belongs to a given position in the tour sequence. The operations research community however has based many of its TSP methods on another formulation: here the decision variable X_{ij} denotes if city i follows city j in the tour sequence. This alternative formulation results in a linear objective function, but some complex constraints are needed to avoid sub-tours. Hopfield and Tank's method cannot readily be applied to the linear formulation due to the difficulty of encoding this constraint (Smith et al., 1996). Nevertheless, this raises the issue of the existence of alternative formulations for a given problem, and the degree to which this impacts the efficiency of any resulting Hopfield networks. A recent study of a Hopfield network application to school timetabling has shown that, where two formulations exist for a problem, the chosen formulation greatly impacts the dimensionality of the neurons, the number of constraints needed to represent the problem, and the complexity of the energy surface (Smith et al., 1999).

Energy Function There are frequently several different ways of enforcing constraints in an energy function. As was shown in Section 2.4, Hopfield and Tank's TSP row and column constraints can be rewritten to explicitly state that each row and column must sum to one, thus eliminating the need for their third term in the energy function (Brandt et al., 1988). For other constraint types also, there are often several different approaches to incorporating them into the energy function (Peterson and Soderberg, 1993). Other researchers avoid the need for representing constraints in the energy function by modifying the network dynamics. This is the approach taken by mean field annealing (Van den Bout and Miller, 1989), where a row constraint stating that the neurons must sum to one is handled by normalizing the neuron states in each row. The aim in exploring different representations of constraints is to reduce the number of terms and parameters needed in the energy function, as well as perhaps to reduce the number of local minima generated by these terms. Other terms often added to the energy function include one of the form $\sum_{i=1}^{N} v_i(1 - v_i)$ which is designed to encourage convergence to binary valued neuron states. This term essentially adds a concave term to the original energy surface, thus altering the convexity and driving the solution towards the vertices of the unit hypercube.

Penalty Factors The choice of penalty factor values affect the contour of the energy function surface, and thus greatly affect the ability of the Hopfield network to find local minima of the optimization problem. For many types of constraints, the penalty factors can be treated as equivalent (e.g., penalty factors for row and column constraints in the TSP should be identical, since these constraints are equally important and equally difficult to satisfy). This observation can often reduce the search space for the optimal penalty factor combination. Many researchers have attempted to eliminate the need for trial and error parameter selection by examining the theoretical balancing of terms in the energy function. For the TSP, Hedge et al. (1988) showed that, while some regions of parameter space can be identified that yield better quality results, the size of these regions diminishes rapidly as the problem size increases. Kamgar-Parsi and Kamgar-Parsi (1992) developed a systematic method for selecting the penalty factors based on analyzing the dynamical stability of feasible solutions. Trial and error searching however does not necessarily preclude a systematic method. The efficiency of searching for optimal penalty parameter values can be improved by adopting the following systematic approach: first find values for the penalty factors that provide a feasible solution, holding the objective function penalty factor constant at unity. Once a combination of penalty factors has been found that consistently yields feasible solutions, slowly start to increase the objective function factor in an attempt to produce less expensive feasible solutions. As soon as feasibility is lost, the bounds on this parameter can be established. This much reduced search space can then be explored in more detail to obtain the combination of penalty factors that yields consistently feasible and optimal solutions.

4.1.2 Problematic Issues

There are a number of important issues that researchers should be aware of when developing Hopfield network models for solving combinatorial optimization problems. Some of these issues are rarely addressed in the available literature, yet they dramatically affect the degree to which the Hopfield network follows its theoretical behavior.

Diagonal Weights A common misconception in the literature is that zero diagonals of the weight matrix $(W_{ii} = 0)$ are necessary for the stable states of the continuous model to coincide with those of the original discrete model. This belief has no doubt evolved due to Hopfield's original simplifying assumption that $W_{ii} = 0$. In fact, there are no restrictive conditions on the diagonals of **W** for the continuous model to converge to a minimum of E_c. If $W_{ii} < 0$ however, that minimum may lie in the interior of the hypercube, due to the convexity of the energy function. In this case, annealing techniques are usually employed to drive the solution trace towards the vertices.

Unlike the continuous model however, non-zero diagonal elements of the discrete Hopfield network do not necessarily allow Liapunov descent to local minima of E_d.

This is because the change in energy E_d due to a change in output level v_i is

$$\Delta E_d = -(u_i \Delta v_i) - \tfrac{1}{2} W_{ii}(\Delta v_i)^2 \tag{22}$$

Since $u_i \geq 0$ results in $\Delta v_i \geq 0$, and $u_i < 0$ results in $\Delta v_i \leq 0$ under the discrete model, the first term on the right-hand side of (22) is always negative. The second term is positive however for $W_{ii} < 0$. Consequently, ΔE_d is only negative provided

$$||\Delta v_i|| \leq 2\frac{||u_i||}{||W_{ii}||} \tag{23}$$

Thus, for non-negative diagonal elements, convergence of the discrete model is guaranteed. For negative diagonals, however, convergence may not be possible since Δv_i is large for discrete dynamics. The continuous model does not suffer from these problems if implemented using the differential equation system (4) and (5). When these equations are simulated on a computer, however, the time discretization means that this model too may become unstable if the changes to neuron states are too large.

Consequently, it is important to be aware that, for problems with negative diagonals of the weight matrix (this includes many practical applications, as well as Hopfield and Tank's original TSP formulation), strict Liapunov descent for even the continuous model will become harder to simulate and cannot be guaranteed during simulation in the extreme high-gain limit (i.e., T near zero) of the continuous transfer function (which approximates the discrete case). If $W_{ii} < 0$, the best we can do for simulation purposes is to make sure the gain of the transfer function is not so high that we are approximating the discrete model too effectively. We can then use an annealing technique to drive the network towards a vertex solution. Of course, these problems are not encountered in a hardware implementation where the differential equation (4) does not need to be simulated.

There is also a relationship worth noting between the diagonal elements of the weight matrix and the shape of the energy surface. As mentioned above, adding a term of the form $\lambda \sum_{i=1}^{N} v_i(1 - v_i)$ to the energy function serves to encourage convergence to the vertices of the unit hypercube. The process of hysteretic annealing proposes starting with a negative value for λ and slowly increasing to positive so that the energy surface is converted from convex to concave, driving the solutions out to the vertices. The idea of slowly decaying away the diagonal weight elements has also been shown to lead to chaotic dynamics, which have been useful for escaping local minima of the energy function (Chen and Aihara, 1995).

Simulation While the continuous Hopfield network is designed to be implemented in hardware, most researchers simulate its behavior on a computer. This involves using a discrete-time approximation to the differential equation (4), such as the Euler approximation:

$$u_i(t+1) = u_i(t) + \Delta t \frac{du_i}{dt} = u_i(t)\left[1 - \frac{\Delta t}{\tau}\right] + \sum_{j=1}^{n} W_{ij} v_j(t) + I_i \tag{24}$$

If the time-step Δt is too large, the dynamics of the simulated network will not closely approximate the continuous model, and Liapunov descent cannot be guaranteed. This can be seen also by equations (22) and (23) since a large time step produces a large change in neuron states $\boldsymbol{v}$. If the time-step is too small, convergence can be guaranteed but the time required to simulate the network will be increased. For a TSP with N cities, it has been shown (Kamgar-Parsi and Kamgar-Parsi, 1987) that a time-step for the simulation of (4) as small as $\Delta t < O(1/N^2)$ is required, slowing down simulations immensely for large problem sizes, and making the discretization of the continuous model quite ineffective. The value chosen for the transfer function parameter T also needs to be selected with care, since it too can produce large changes in neuron states $\boldsymbol{v}$ and result in oscillations. Kamgar-Parsi and Kamgar-Parsi (1987) have proposed a discrete-time Hopfield network model with continuous dynamics that satisfies the condition in (23) and produces quick convergence.

Initial States Since the Hopfield network is a gradient descent technique, the initial states of the neurons play a large role in determining the final solution quality. Certainly, the discrete-time versions of the networks incorporate some stochasticity since neurons are selected for updating at random (only the continuous model implemented in hardware is truly deterministic). Nevertheless, this degree of stochasticity is rarely enough to avoid the high dependence of the solution on the initial states. Most researchers report their results based on multiple random and unbiased initializations to avoid this dependence. Some work has been conducted to identify heuristic methods for more effective initializations (Lai and Coghill, 1994; Naphade and Tuzun, 1995; Schwartz et al., 1991).

Updating Mode For any discrete-time version of the Hopfield network (discrete model or simulated continuous model), there are two ways of updating the neurons: asynchronously (sequentially) or synchronously (in parallel). Asynchronous updating is the method described in Section 2, whereby a neuron is chosen at random and its internal and external states are updated before the next random neuron is chosen. Liapunov descent is guaranteed under these conditions (provided equation (23) holds true). For synchronous updating however, all neurons update their internal states, and then the updated values are used to simultaneously update the external states. Under this updating mode, convergence cannot be guaranteed and the network may become trapped in a 2-cycle (Bruch, 1990). Most researchers use the asynchronous mode of updating to avoid such oscillations.

Solution Integrality For combinatorial optimization problems the final solution needs to be binary, even if the method used approaches this solution from within the unit hypercube. Using the discrete Hopfield model is appropriate for ensuring binary solutions, but the problems with convergence for many practical problems where $W_{ii} < 0$ have resulted in most researchers turning to the continuous model. There are several approaches to encouraging binary solutions within the continuous model. Firstly, we can use a gradient of the transfer function with T near zero (which approximates the discrete transfer function (2)), but this risks producing convergence problems like the discrete model, as discussed above. Secondly, we can add an additional term to the energy function to drive towards the vertices. While this introduces another parameter to select, some additional advantages can be found due to the chaotic effect on the dynamics and the hill-climbing this creates (Chen and Aihara, 1995). Other methods have been proposed including a variety of rounding approaches and heuristics for interpretation of final non-integer solutions (Wolfe, 1999).

Termination Criteria One of the factors that has made it difficult for researchers to reproduce Hopfield and Tank's original results is that they omitted certain critical details in their paper about the method used to simulate the differential equation, and the termination criteria. Wilson and Pawley (1988) experimented with three different termination criteria in an effort to reproduce the results: the network simulation was terminated if (*i*) a valid tour was found, (*ii*) the network had frozen as measured by no neuron $\boldsymbol{v}$ values changing by more than 10^{-35} since the last update, and (*iii*) more than 1000 updating iterations had elapsed (a "time-out" test, useful for catching cyclic and oscillatory convergence). Wilson and Pawley found that 1000 iterations were sufficient for their experiments, and increasing the "time-out" range to 10,000 iterations did not produce any improvement in results. It is important to be aware, however, that the quality of the reported results is usually affected greatly by the termination criteria selected, and researchers need to be sure to report these accurately.

4.2 Elastic Net

4.2.1 Efficiency Issues

In the EN algorithm, updating the position of the ring points is computationally expensive, as the update of a single point depends on the position of every vertex, through the attraction coefficients w_{ij}. Furthermore, these coefficients must be recomputed at each iteration. For $M \approx N$, the complexity of each iteration is thus $O(N^2)$. Different approaches have been proposed to reduce this burden.

Filtering A natural way of addressing the problem without degrading too much solution quality is to consider only vertices that have a significant impact on the ring points (Boeres et al., 1992; Vakhutinsky and Golden, 1995). In other words, attraction coefficients that are not of sufficient magnitude, because their vertices are too far from the corresponding ring points, are filtered out. A large number of coefficients may be eliminated in this way because the function $\phi(d, K)$ decreases quickly as the square of the distance d^2 grows.

Hierarchical EN The idea of the hierarchical EN (Vakhutinsky and Golden, 1995) is to divide the area containing the vertices into smaller subareas and to replace the vertices in each subarea by a single vertex located at their center of gravity. As the algorithm unfolds, the subareas are progressively reduced until each subarea contains a single vertex. It is reported that working on smaller aggregated problems at the start allows the algorithm to find the general shape of the solution more quickly.

4.2.2 Problematic Issues

When implementing an elastic net algorithm, the following issues should be taken care of:

Normalization of Coordinate Vectors The behavior of EN may be quite different from one problem instance to another. Some authors have noted that a more robust algorithmic behavior is obtained by normalizing the coordinate vectors of the vertices (Favata and Walker, 1991).

Initial Ring The number of ring points *M* to be used is clearly related to the number of vertices *N*. However, using too many points leads to a loss of efficiency. In the literature, *M* is usually set around 2.5*N*. The initial position of the ring is typically around the center of gravity of the vertices. Good results are also reported when the points on the ring correspond to the convex hull of the vertex set (Burke, 1994).

Ring Migration When the value of parameter *K* is large, the energy function is rather smooth, but as this value is reduced a multimodal energy landscape emerges,

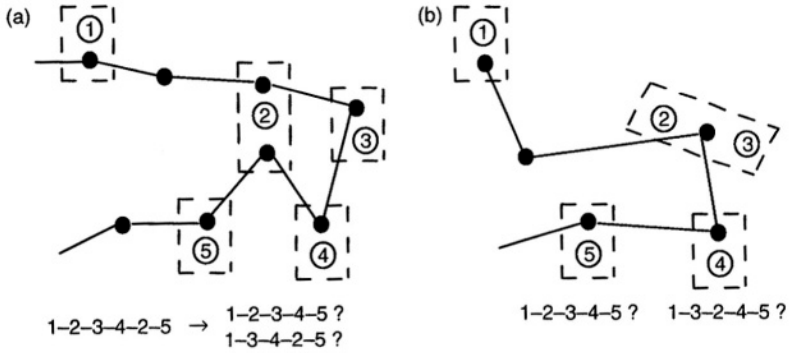

Figure 15.4. Solution ambiguity.

where good local minima should correspond to good tours. In order to obtain this result, parameter K must be slowly reduced to avoid some form of twisting or crossover of the ring (which typically leads to long tours). For example, a problem with 100 vertices was solved in (Durbin and Willshaw, 1987) by setting K to an initial value of 0.2 and by reducing it by 1% every 25 iterations until a value in the range 0.01–0.02 was obtained.

Interpretation of the Final Configuration Two or more different ring points may be associated with the same vertex if they are all within the tolerance of that vertex (this is sometimes referred to as a spike). Conversely, a ring point may be associated with two or more vertices. In such cases, the solution is not well defined. In Figure 15.4(a), two ring points are within the tolerance of vertex 2, and two different sequences 1-2-3-4-5 and 1-3-4-2-5 are obtained depending of the ring point chosen. In Figure 15.4(b), a single ring point is associated with vertices 2 and 3. Hence, it is impossible to know if vertex 2 is visited before or after vertex 3. One possible way of solving these problems is through postprocessing. For example, all valid solutions obtainable with the current configuration can be considered and the best overall solution is taken. Trying to avoid this phenomenon during the course of the algorithm is rather difficult. Some theoretical studies indicate that an appropriate setting of the β/α ratio (so that it is about one-half of the average inter-point distance on the ring) is likely to lead to fully-specified solutions (Simmen, 1991).

4.3 Self-organizing Map

4.3.1 Efficiency Issues

As noted by different authors (Burke and Damany, 1992; Favata and Walker, 1991), significant gains in efficiency are obtained by reducing the number of ring points that move towards the current vertex (i.e., those that are close the the winning point) and also by reducing the magnitude of the move. In Step 2, the update mechanism is governed by function $f(j, j^*)$ which has a form like:

$$\begin{aligned} f(j,j^*) &= \left(1 - \frac{d''_{jj^*}}{L}\right)^{\beta}, \quad if\ d''_{jj^*} < L \\ &= 0, \qquad\qquad\qquad otherwise \end{aligned} \tag{25}$$

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

In this definition, d'' is the lateral distance on the ring between point j and winning point j^* (assuming that the points on the ring are indexed from 1 to M), and $|x|$ is the absolute value of x. This function is such that it always returns a value of 1 for the winning point, and this value decreases as the lateral distance from the winning point increases; when the lateral distance goes beyond parameter L, the points do not move at all. The value of parameter β is increased from one pass to another to progressively reduce the magnitude of the move of neighboring units. At the end, when β is sufficiently large, only the winning unit moves towards the current vertex and separates from its neighbors to fix the solution.

4.3.2 Problematic Issues

Since the SOM is closely related to EN, many issues mentioned above for EN still hold here. We thus focus on a specific issue, which is related to the mechanics of the SOM for updating the ring location.

Ring Point Freezing Solution ambiguity can occur, either because many ring points fall within the tolerance of a given vertex or a single ring point falls within the tolerance of two or more vertices. However, due to the specific methodology adopted for stretching the ring, which is based on a competition between ring points, it may also happen that a number of points will freeze at their initial location, because they never win any competition. Consequently, partially-defined tours are obtained, where a number of vertices are ⟨⟨orphans⟩⟩ (i.e., do not have any close ring points). Different techniques have been proposed to alleviate this problem:

— In Angeniol et al. (1988), the implementation is based on the distinctive feature that ring points are dynamically created and deleted. A point is duplicated if it wins for two different vertices after a complete pass through the set of vertices. It is deleted, if it does not win after three complete passes. Through this mechanism, vertices are less likely to end up alone. Starting with a single point on the ring, the authors report that up to twice as many points as vertices may be created during the procedure.

— In Burke and Damany (1992), a conscience mechanism proposed by Desieno (1988) replaces the dynamic creation and deletion of ring points. A penalty is added to the distance between a ring point and a vertex, based on the number of times that point has won the competition in the past. Consequently, frequent winners are heavily penalized in favor of other units. Basically, Step 1.2 (competition) in the SOM algorithm of Section 3.2 is modified as follows for a given vertex i and ring point j:

$$o_j \leftarrow d_{X_i Y_j} + \gamma b_j,$$

where b_j is the penalty or bias associated with ring point j. This penalty is typically the fraction of competitions won by ring point j in the past. Good results are reported with a number of ring points now equal to the number of vertices, leading to substantial savings in computation time.

— Parameter γ that weighs the penalty with respect to the true distance in the conscience mechanism, is reportedly difficult to tune. In (Burke, 1994), a vigilant net is proposed where ring points are turned off if they win too often to let others win. Basically, the number of wins is recorded for each unit and that unit is turned off for the remaining of the pass through the set of vertices, if this number exceeds some threshold value (known as the vigilant parameter). At the start of the next pass, the winning score of all units is reset to zero. The vigilance parameter is large initially, to allow the vertices to win freely at the start, and is progressively reduced until it reaches a value of one, to let the ring points separate and converge towards distinct vertices.

5 CONCLUDING REMARKS

This chapter has reviewed the two main types of neural network models that can be used for combinatorial optimization: Hopfield networks and the deformable template models of elastic nets and self-organizing maps. The review has covered both the theoretical aspects of their application to combinatorial optimization problems, as well as discussing a variety of practical considerations that affect the performance of the models.

From a metaheuristics viewpoint, neural networks can be seen as an alternative technique with the current potential to match the performance of better known algorithms such as tabu search and simulated annealing. This potential relies on due consideration of the aforementioned range of issues affecting the success and efficiency of the methods. The deformable template methods are well suited to solving low dimensional problems with geometric interpretation like the TSP. The Hopfield network method generalizes to a broad range of combinatorial problems, but the cost of this generalization is a reduction in efficiency and scalability. Certainly, current developments in hardware implementation of neural architectures should see some of these limitations relaxed in future years. The advantage of neural networks over other metaheuristic techniques could then be more fully demonstrated.

ACKNOWLEDGMENTS

This work was partly supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) and the Quebec Fonds pour la Formation de Chercheurs et l' Aide a la Recherche (FCAR). This support is gratefully acknowledged.

REFERENCES

Aarts, E.H.L. and Korst, J. (1989) *Simulated Annealing and Boltzmann Machines.* John Wiley & Sons., Essex.

Aiyer, S.V.B., Niranjan, M. and Fallside, F. (1990) A theoretical investigation into the performance of the Hopfield model. *IEEE Transactions on Neural Networks*, **1**, 204–215.

Akiyama, Y., Yamashita, A., Kajiura, M. and Aiso, H. (1989) Combinatorial optimization with gaussian machines. *Proceedings IEEE International Joint Conference on Neural Networks*, **1**, 533–540.

Amartur, S.C., Piraino, D. and Takefuji, Y. (1992) Optimization neural networks for the segmentation of magnetic resonance Images. *IEEE Transactions on Medical Imaging*, **11**, 215–220.

Angeniol, B., Vaubois, G. and Le Texier, J.Y. (1988) Self-organizing feature maps and the travelling salesman problem. *Neural Networks,* **1**, 289–293.

Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G.C. and Stewart, W.R. (1995) Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics,* **1**, 9–32.

Boeres, M.S.C., de Carvalho, L.A.V. and Barbosa, V.C. (1992) A faster elastic net algorithm for the traveling salesman problem. In: *Proceedings of the International Joint Conference on Neural Networks.* Baltimore, U.S.A., II-215-220.

Brandt, R.D., Wang, Y., Laub, A.J. and Mitra, S.K. (1988) Alternative networks for solving the travelling salesman problem and the list-matching problem. *Proceedings International Conference on Neural Networks* Vol. 2, 333–340.

Bruch, J. (1990) On the convergence properties of the Hopfield model. *Proceedings of the IEEE,* **78**(10), 1579–1585.

Burke, L.I. (1994) Adaptive neural networks for the traveling salesman problem: insights from operations research. *Neural Networks,* **7**, 681–690.

Burke, L.I. and Damany, P. (1992) The guilty net for the traveling salesman problem. *Computers & Operations Research,* **19**, 255–265.

Burke, L.I. and Ignizio, J.P. (1992) Neural networks and operations research: an overview. *Computers & Operations Research,* **19**, 179–189.

Chen, L. and Aihara, K. (1995) Chaotic simulated annealing by a neural network model with transient chaos. *Neural Networks,* **8**(6), 915–930.

Desieno, D. (1988) Adding a conscience mechanism to competitive learning. In: *Proceedings of the IEEE International Conference on Neural Networks,* San Diego, U.S.A., I-117-124.

Durbin, R. and Willshaw, D.J. (1987) An analogue approach to the traveling salesman problem using an elastic net method. *Nature,* **326**, 689–691.

Favata, F. and Walker, R. (1991) A study of the application of Kohonen-type neural networks to the traveling salesman problem. *Biological Cybernetics,* **64**, 463–468.

Foo, Y.P.S. and Szu, H. (1989) Solving large-scale optimization problems by divide-and-conquer neural networks. *Proceedings IEEE International Joint Conference on Neural Networks,* **1**, 507–511.

Garey, M.R. and Johnson, D.S. (1979) *Computers and Intractability.* W.H. Freeman, New York.

Gee, A.H. (1993) *Problem Solving with Optimization Networks,* Ph.D. Dissertation, Queen's College, Cambridge University, U.K.

Ghaziri, H. (1991) Solving routing problems by a self-organizing map. In: T. Kohonen, K. Makisara, O. Simula and J. Kangas (eds.), *Artificial Neural Networks.* North-Holland, Amsterdam, pp. 829–834.

Ghaziri, H. (1996) Supervision in the self-organizing feature map: application to the vehicle routing problem. In: I.H. Osman and J.P. Kelly (eds.), *Meta-Heuristics: Theory & Applications.* Kluwer, Boston, pp. 651–660.

Glover, F. (1994) Optimization by ghost image processes in neural networks. *Computers & Operations Research,* **21**, 801–822.

Goldstein, M. (1990) Self-organizing feature maps for the multiple traveling salesmen problem. In: *Proceedings of the International Neural Network Conference.* Paris, France, pp. 258–261.

Hegde, S., Sweet, J. and Levy, W. (1988) Determination of parameters in a hopfield/tank computational network. *Proceedings IEEE International Conference on Neural Networks,* **2**, 291–298.

Hinton, G.E., Sejnowski, T.J. and Ackley, D.H. (1984) Boltzmann machines: constraint satisfaction networks that learn. Carnegie Mellon University Technical Report CMU-CS-84-119.

Hopfield, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences,* **79**, 2554–2558.

Hopfield, J.J. (1984) Neurons with Graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences,* **81**, 3088–3092.

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

Hooker, J.N. (1995) Testing heuristics: we have it all wrong. *Journal ofHeuristics,* **1**, 33–42.

Johnson, D.S. (1990) Local optimization and the traveling salesman problem. In: G. Goos and J. Hartmanis (eds.), *Automata, Languages and Programming, Lecture Notes in Computer Science 443.* Springer-Verlag, Berlin, pp. 446–461.

Kamgar-Parsi, B. and Kamgar-Parsi, B. (1987) An efficient model of neural networks for optimization. *Proceedings IEEE International Conference on Neural Networks,* **3**, 785–790.

Kamgar-Parsi, B. and Kamgar-Parsi, B, (1992) Dynamical stability and parameter selection in neural optimization. *Proceedings IEEE International Conference on Neural Networks,* **4**, 566–571.

Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983) Optimization by simulated annealing. *Science,* **220**, 671–680.

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

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

Lai, W.K. and Coghill, G.G. (1992) Genetic breeding of control parameters for the Hopfield/Tank neural net. *Proceedings International Joint Conference on Neural Networks,* **4**, 618–623.

Lai, W.K. and Coghill, G.G. (1994) Initialising the continuous Hopfield net. *Proceedings IEEE International Conference on Neural Networks,* **7**, 4640–4644.

Lin, S. and Kernighan, B.W. (1973) An effective heuristic algorithm for the travelling salesman problem. *Operations Research,* **21**, 498–516.

Lo, J.T.-H. (1992) A new approach to global optimization and its applications to neural networks. *Proceedings IEEE International Joint Conference on Neural Networks,* **4**, 600–605.

Looi, C.K. (1992) Neural network methods in combinatorial optimization. *Computers & Operations Research,* **19**, 191–208.

Matsuyama, Y. (1991) Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems. In: *Proceedings of the International Joint Conference on Neural Networks.* Seattle, U.S.A., I-385-390.

McCulloch, W.S. and Pitts, W. (1943) A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics,* **5**, 115–133.

Minsky, M. and Papert, S. (1969) *Perceptrons.* MIT Press, Cambridge, MA.

Naphade, K. and Tuzun, D. (1995) Initializing the Hopfield–Tank network for the TSP using a convex hull: a computational study. *Intelligent Engineering Systems Through Artificial Neural Networks.* vol. 5. ASME Press, New York, pp. 399–404.

Nemhauser, G.L. and Wolsey, L.A. (1988) *Integer and Combinatorial Optimization.* John Wiley & Sons, Canada.

Peterson, C. and Soderberg, B. (1989) A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems,* **1**, 3–22.

Peterson, C. and Soderberg, B. (1993) Artificial neural networks. In: C.R. Reeves (ed.), *Modern Heuristic Techniques for Combinatorial Optimisation,* Chapter 5. Blackwell Scientific Publishers, Oxford, UK.

Potvin, J.Y. (1993) The traveling salesman problem: a neural network perspective. *ORSA Journal on Computing,* **5**, 328–348.

Potvin, J.Y. and Robillard, C. (1995) Clustering for vehicle routing with a competitive neural network model. *Neurocomputing,* **8**, 125–139.

Rosenblatt, F. (1958) The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review,* **65**, 386–408.

Rumelhart, D.E. and McClelland, J.L. (1986) *Parallel distributed processing: explorations in the microstructure of cognition,* I & II. MIT Press, Cambridge, MA.

Schwartz, B.L., Das, P. and Koch, J.B. (1991) Initialization in Hopfield networks. *Neurocomputing,* **3**(3), 135–145.

Simmen, M.W. (1991) Parameter sensitivity of the elastic net approach to the traveling salesman problem. *Neural Computation,* **3**, 363–374.

Smith, K.A. (1995) Solving the generalized quadratic assignment problem using a self-organizing process. In: *Proceedings IEEE International Conference on Neural Networks 4,* Perth, Australia, pp. 1876–1879.

Smith, K.A. (1999) Neural networks for combinatorial optimization: a review of more than a decade of research. *INFORMS Journal on Computing,* **11**, 15–34.

Smith, K.A., Abramson, D. and Duke, D. (1999) Efficient timetabling formulations for Hopfield neural networks. In: C. Dagli et a), (eds.), *Smart Engineering System Design: Neural Networks, Fuzzy Logic, Evolutionary Programming, Data Mining, and Complex Systems,* vol. 9. ASME Press, pp. 1027–1032.

Smith, K.A., Palaniswami, M., Krishnamoorthy, M. (1996) A hybrid neural approach to combinatorial optimization. *Computers & Operations Research,* **23**, 597–610.

Szu, H. and Hartley, R. (1987) Fast simulated annealing. *Physics Letters A,* **122**, 157–162.

Szu, H. (1988) Fast TSP algorithm based on binary neuron output and analog input using zero-diagonal interconnect matrix and necessary and sufficient conditions of the permutation matrix. *Proceedings IEEE International Conference on Neural Networks,* **2**, 259–266.

Takefuji, Y. and Szu, H. (1989) Design of parallel distributed Cauchy machines *Proceedings IEEE International Joint Conference on Neural Networks,* **1**, 529–532.

Tank, D.W. and Hopfield, J.J. (1986) Simple neural optimization networks: an A/D converter, signal decision circuit and a linear programming circuit. *IEEE Transactions on Circuit Systems,* **33**, 533–541.

Vakhutinsky, A.I. and Golden, B.L. (1994) Solving vehicle routing problems using elastic nets. *Proceedings of the IEEE International Conference on Neural Networks,* **7**, 4535–4540.

Vakhutinsky, A.I. and Golden, B.L. (1995) A hierarchical strategy for solving traveling salesman problems using elastic nets. *Journal of Heuristics,* **1**, 67–76.

Van Den Bout, D.E. and Miller, T.K. (1988) A travelling salesman objective function that works. *Proceedings IEEE International Conference on Neural Networks,* **2**, 299–303.

Van Den Bout, D.E. and Miller, T.K. (1989) Improving the performance of the Hopfield–Tank neural network through normalization and annealing. *Biological Cybernetics,* **62**, 129–139.

Van Den Bout, D.E. and Miller, T.K. (1990) Graph partitioning using annealed neural networks. *IEEE Transactions on Neural Networks,* **1**, 192–203.

Willshaw, D.J. and von der Malsburg, C. (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**, 203–243.

Wilson, G.V. and Pawley, G.S. (1988) On the stability of the TSP algorithm of Hopfield and Tank. *Biological Cybernetics,* **58**, 63–70.

Wolfe, W.J. (1999) A fuzzy Hopefield–Tank traveling salesman problem model. *INFORMS Journal on Computing,* **11**(4), 329–344.