Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research

Article *in* Informs Journal on Computing · February 1999

DOI: 10.1287/ijoc.11.1.15 · Source: DBLP

1 author:

Kate Smith-Miles
University of Melbourne
297 PUBLICATIONS **6,617** CITATIONS

INFORMS Journal on Computing
Vol. 11, No. 1, Winter 1999

0899-1499/99/1101-0015 $05.00

Neural Networks for Combinatorial Optimization: A Review of More Than a Decade of Research

KATE A. SMITH / *School of Business Systems, Monash University, Clayton, Victoria, 3168, Australia.*
Email: ksmith@bs.monash.edu.au

(Received: June 1997; revised February 1998, May 1998; accepted: October 1998)

It has been over a decade since neural networks were first applied to solve combinatorial optimization problems. During this period, enthusiasm has been erratic as new approaches are developed and (sometimes years later) their limitations are realized. This article briefly summarizes the work that has been done and presents the current standing of neural networks for combinatorial optimization by considering each of the major classes of combinatorial optimization problems. Areas which have not yet been studied are identified for future research.

In a recent survey of meta-heuristics, Osman and Laporte[130] reported that while neural networks are a very powerful technique for solving problems of prediction, classification and pattern recognition, they "have not been as successful when applied to optimization problems and are not competitive with the best meta-heuristics from the operations research literature, when applied to combinatorial optimization problems." Researchers have been trying for over a decade now to make neural networks competitive with meta-heuristics[145] such as simulated annealing,[1, 2] tabu search,[67, 68] constraint logic programming,[180] and genetic algorithms,[46, 69] and they have experienced varying degrees of success. Almost every type of combinatorial optimization problem (COP) has been tackled by neural networks, and many of the approaches result in solutions that are very competitive with alternative techniques in terms of solution quality. Unfortunately, the reputation of neural networks for solving COPs does not reflect this evidence. The reasons are twofold. First, from within the first few years of research, controversy and debate has surrounded the field, discouraging many researchers: a state from which the field has never truly recovered. Second, researchers are forced to simulate the behavior of neural networks on digital computers while awaiting the development of suitable hardware advances. These simulations are designed to evaluate the potential of neural networks for generating near-optimal solutions to COPs, and naturally result in large CPU times that are uncompetitive with alternative techniques. Further research into the design of suitable hardware is contingent upon demonstrated success of neural network simulations. Unfortunately, the many successful applications of neural networks will not receive full merit until the reputation of neural networks has been salvaged. This article aims to clarify the current standing and potential of neural networks for solving COPs after more than a decade of research.

The idea of using neural networks to provide solutions to difficult NP-hard optimization problems[61, 125] originated in 1985 when Hopfield and Tank demonstrated that the Travelling Salesman Problem (TSP) could be solved using a Hopfield neural network.[87] This work is well-known, and their impressive results for the solution of a TSP generated much excitement in the neural network and operations research communities alike. Their work is also quite controversial because many researchers have been unable to reproduce their results (due perhaps to certain omissions in Hopfield and Tank's article[87] relating to simulation procedure, termination criteria, etc.). Wilson and Pawley[192] first published these findings nearly three years after Hopfield and Tank's original article was published. In doing so, they raised serious doubts as to the validity of the Hopfield-Tank (H-T) approach to solving optimization problems, which seemingly served to dampen the enthusiasm surrounding the field. 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 nature of the energy function that the method utilizes causes infeasible solutions to occur most of the time. A number of penalty parameters need to be fixed before each simulation of the network, yet the values of these parameters that will enable the network to generate valid solutions are unknown. The problem of optimally selecting these parameters is not trivial, and much work has been done to try to facilitate this process.[79, 96, 109] 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.[21, 177] Perhaps the most important breakthrough in the field, however, came from the *valid subspace approach* of Aiyer et al.,[6] and the subsequent work of Gee.[63] By representing all of the constraints by a single term, the feasibility of the Hopfield network can now be essentially guaranteed.

The question of solution quality has also been addressed over the last decade by various methods that attempt to avoid the many local minima of the energy function. Many variations of the Hopfield network have been proposed and can be broadly categorized as either deterministic or stochastic. The stochastic approaches are perhaps more successfully able to improve solution quality because they attempt to embed the principles of simulated annealing into the Hopfield network to escape from local minima. Such attempts have resulted in several alternative approaches including Boltzmann,[1, 4, 82] Cauchy,[90, 167] and Gaussian

Subject classifications: Neural networks, combinatorial optimization
Other key words: Combinatorial optimization, neural networks.

Machines,[8] which are able to compete effectively with other techniques in terms of solution quality.

The other main neural network approach to combinatorial optimization is based on Kohonen's Self-Organizing Feature Map.[101] While much of the Hopfield network literature is focused on the solution of the TSP, a similar focus on the TSP is found in almost all of the literature relating to the use of self-organizing approaches to optimization.[11, 52, 56] In this case, the reason is not simply because of the benchmark status of the TSP, but more because the vast majority of these approaches are based upon the *elastic net method*.[49] Kohonen's principles of self-organization[101] are combined with the concept of an "elastic band" containing a circular ring of neurons that move in the Euclidean plane of the TSP cities, so that the "elastic band" eventually passes through all of the cities and represents the final TSP tour. Such approaches rely upon the fact that the "elastic band" can move in Euclidean space, and that physical distances between the neurons and the cities can be measured in the same space. Self-organizing neural networks have been successfully applied to solve other two-dimensional problems such as vehicle routing[173, 174] and shortest path problems.[164] Recently, a generalization of the self-organizing map has been proposed to solve generalized quadratic assignment problems, relieving the restriction to two-dimensional problems, and has been applied to solve several optimization problems arising from industrial situations.[74, 155, 158]

While further discussion regarding many of these approaches is continued in the following sections of this article, we refer the interested reader to survey articles by Burke and Ignizio,[25] Looi,[120] Potvin,[139] and Sharda[152] for comprehensive reviews and discussions of several of the existing neural network techniques for optimization.

1. Hopfield Neural Network Approaches

In this section, we review Hopfield neural networks and the approach used to solve COPs. We then discuss the criticisms of the technique, and present some of the modifications that have been proposed. Finally, modifications that ensure the feasibility of the final solution, as well as improve the solution quality through hill-climbing, are presented.

1.1 Hopfield Networks

In his seminal paper[85] of 1982, John Hopfield described a new way of modeling a system of neurons capable of performing "computational" tasks. The Hopfield neural network emerged, initially as a means of exhibiting a *content-addressable memory* (CAM). A general CAM must be capable of retrieving a complete item from the system's memory when presented with only sufficient partial information. Hopfield showed that his model was not only capable of correctly yielding an entire memory from any portion of sufficient size, but also included some capacity for generalization, familiarity recognition, categorization, error correction, and time-sequence retention.

The Hopfield network, as described in [85, 86], 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}, which may be positive or negative depending on whether the neurons act in an excitatory or inhibitory manner. The internal state of each neuron u_i is equivalent to the weighted sum of the external states of all connecting neurons. The external state of neuron i is given by v_i, with $0 \leqslant v_i \leqslant 1$. An external input, i_i, to each neuron i is also incorporated. The relationship between the internal state of a neuron and its output level in this continuous Hopfield network is determined by an activation function $g_i(u_i)$, which is bounded below by 0 and above by 1. Commonly, this activation function is given by

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

where T is a parameter used to control the gain (or slope) of the activation function.

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. Thus, the following resistance-capacitance differential equation determines the rate of change of u_i, and hence the time evolution of the continuous Hopfield network:

$$C_i \frac{du_i}{dt} = \sum_j W_{ij} v_j - \frac{u_i}{R_i} + i_i \tag{2}$$

$$u_i = g_i^{-1}(v_i).$$

The same set of equations represents the resistively connected network of electronic amplifiers shown in Figure 1.

The synapse (or weight) between two neurons is now defined by a conductance W_{ij}, which connects one of the two outputs of amplifier j to the input of amplifier i. The connection is made with a resistor of value $R_{ij} = \|W_{ij}\|^{-1}$. $\bar{v}_i$ is the output of the inverted amplifier. Figure 1 also includes an input resistance, r_i, for amplifier i. However, this value can be eliminated from the equation because R_i is a parallel combination of r_i and the R_{ij}:

$$\frac{1}{R_i} = \frac{1}{r_i} + \sum_j \frac{1}{R_{ij}}.$$

For simplicity, each neuron/amplifier is assumed to be identical, so that

$$g_i = g, \quad R_i = R, \quad \text{and} \quad C_i = C.$$

Dividing Eq. 2 by C and redefining W_{ij}/C and i_i/C to be W_{ij} and i_i, respectively, we arrive at the equations of motion:

$$\frac{du_i}{dt} = \sum_j W_{ij} v_j - \frac{u_i}{\tau} + i_i \tag{3}$$

$$u_i = g_i^{-1}(v_i) \quad \text{and} \quad \tau = RC.$$

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 the discrete time simulation of Eq. 3 is considerably smaller than unity. Although this "neural" computational network has been described in terms of an electronic circuit, it has been shown[86] that biological mod-

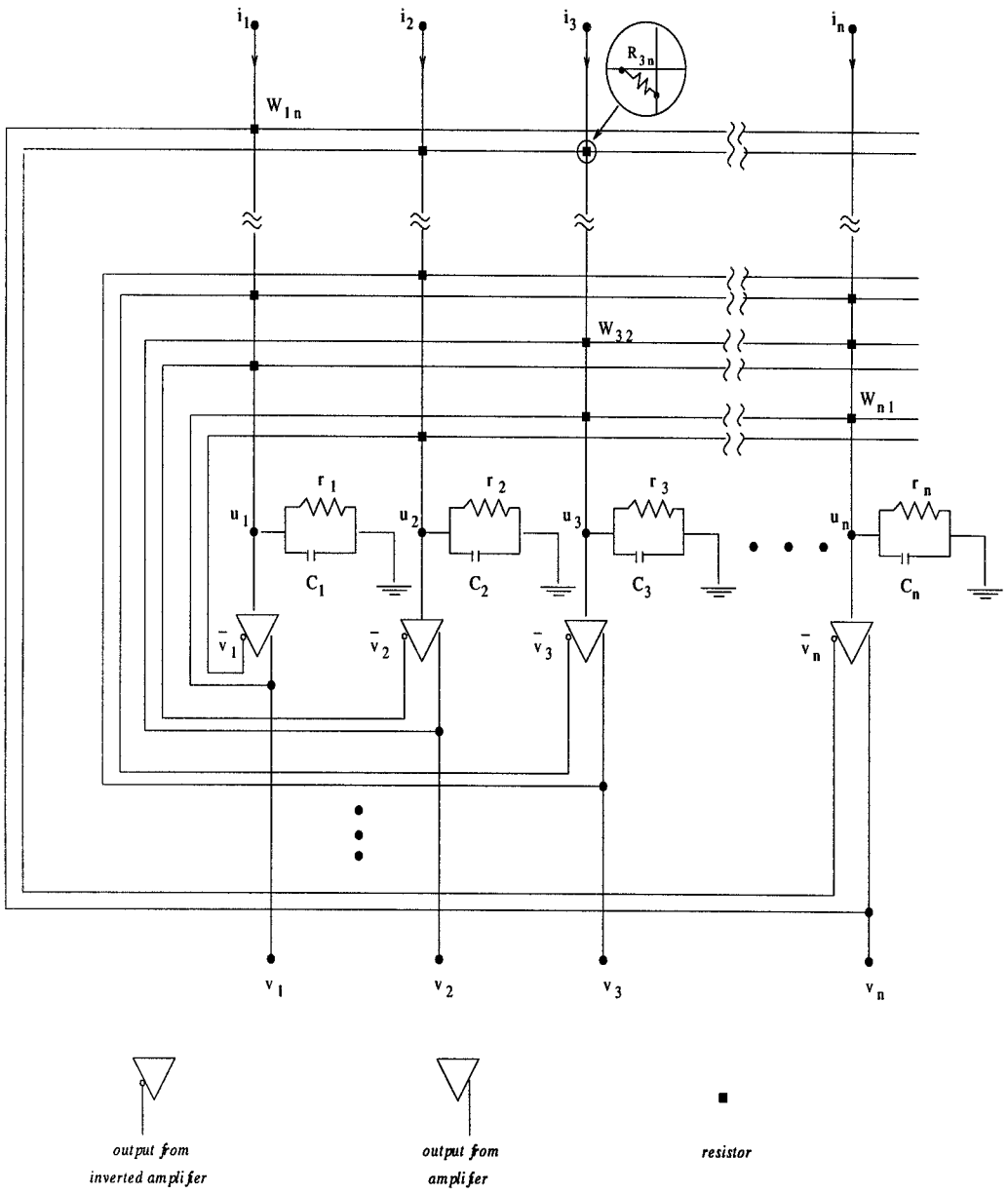

Figure 1. An electronic circuit representation of the continuous Hopfield neural network.

els with action potentials and excitatory and inhibitory synapses can compute in a similar fashion to this electrical hardware.

For the continuous Hopfield network, a Liapunov function can be constructed for the system, which guarantees convergence to stable states. Consider the energy function,

$$\mathscr{E}_c = -\frac{1}{2}\sum_i \sum_j v_i W_{ij} v_j - \sum_i i_i v_i + \int_0^{v_i} g_i^{-1}(\mathbf{v})d\mathbf{v} \quad (4)$$

Provided the matrix of weights **W** is symmetric (although Vidyasagar[183] has shown that convergence is still possible under some asymmetric conditions), the time derivative of $\mathscr{E}_c$ is:

$$\frac{d\mathscr{E}_c}{dt} = -\sum_i \frac{dv_i}{dt}\left(\sum_j W_{ij}v_j - \frac{u_i}{\tau} + i_i\right)$$

$$= -\sum_i \left(\frac{dv_i}{dt}\right)\left(\frac{du_i}{dt}\right)$$

$$= -\sum_i g_i^{-1\prime}(v_i)\left(\frac{dv_i}{dt}\right)^2.$$

Because $g_i^{-1}(v_i)$ is a monotonically increasing function, then

$$\frac{d\mathscr{E}_c}{dt} \leq 0, \quad \text{and} \quad \frac{d\mathscr{E}_c}{dt} = 0 \quad \text{f} \quad \frac{dv_i}{dt} = 0 \quad \forall i. \tag{5}$$

Together with the boundedness of $\mathscr{E}_c$, Eq. 5 shows that under the control of the differential Eq. 2, $\mathscr{E}_c$ decreases and converges to a minimum, at which it stays.

A discrete version of the Hopfield network also exists where v_i is a binary state given by

$$v_i = g_i(u_i) = \begin{cases} 1 & \text{if } u_i > 0 \\ 0 & \text{if } u_i \leq 0 \end{cases} \tag{6}$$

and $g_i(u_i)$ is a discrete threshold function. The discrete Hopfield network has been shown[85] to minimize the energy function

$$\mathscr{E}_d = -\frac{1}{2} \sum_i \sum_j v_i W_{ij} v_j - \sum_i i_i v_i \tag{7}$$

The continuous Hopfield network therefore relates directly to the discrete version in the high-gain limit of the activation function (Eq. 1). In this high-gain limit ($T \approx 0$), $g(u_i)$ approximates the behavior of the discrete threshold function and the local minima of $\mathscr{E}_c$ coincide with the local minima of $\mathscr{E}_d$ and all lie at the vertices of the unit hypercube. Consequently, in the high-gain limit, provided the weight matrix is symmetric and that the inverse function of $g'(u_i)$ (the first derivative of the activation function) exists, the continuous Hopfield network converges to a 0–1 vertex, which minimizes $\mathscr{E}_d$ given by Eq. 7.

1.2 The Hopfield-Tank Approach to Solving Optimization Problems

In 1985, John Hopfield teamed together with David Tank to extend the applications of his model to include solving optimization problems (see [87]). 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 that appropriately represent the function to be minimized and the desired states of the problem. The analog nature of the neurons and the parallel processing 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 that needs to be minimized, while the constraints of the problem are included in the energy function as penalty terms.

Consider the optimization problem:

$$\begin{array}{lll} \text{(P1)} & \text{minimize} & f(\mathbf{v}) \\ & \text{subject to} & [\mathbf{A}]_1\mathbf{v} = b_1 \\ & & [\mathbf{A}]_2\mathbf{v} = b_2 \\ & & \ldots \\ & & [\mathbf{A}]_r\mathbf{v} = b_r \end{array}$$

where $[\mathbf{A}]_i$ (the ith row of the constraint matrix $\mathbf{A}$) and $\mathbf{v}$ are n-dimensional vectors, and r is the number of constraints. Then the H-T energy function is

$$\mathscr{E}(\mathbf{v}) = \alpha f(\mathbf{v}) + \beta_1([\mathbf{A}]_1\mathbf{v} - b_1)^2 + \beta_2([\mathbf{A}]_2\mathbf{v} - b_2)^2 + \cdots + \beta_m([\mathbf{A}]_r\mathbf{v} - b_r)^2.$$

$\alpha, \beta_1, \ldots, \beta_r$ are penalty parameters that are chosen to reflect the relative importance of each term in the energy function. This approach is essentially a Lagrangian relaxation of the constraints, although the nonlinearity of the terms makes determination of the optimal penalty parameters unlikely. Clearly, a constrained minimum of P1 will also optimize the energy function, because the objective function, $f(\mathbf{v})$, will be minimized, and constraint satisfaction implies that the penalty terms will be zero. Once a suitable energy function has been chosen, the network parameters (weights and inputs) can be inferred by comparison with the standard energy function given by Eq. 7. The weights of the continuous Hopfield network, W_{ij}, are the coefficients of the quadratic terms v_iv_j, and the external inputs, i_i, are the coefficients of the linear terms v_i in the chosen energy function. The network can then be initialized by setting the activity level v_i of each neuron to a small random perturbation around 0.5. This places the initial state of the system at approximately the center of the n-dimensional hypercube, and ensures that the initial state is unbiased. Alternative initialization schemes have been investigated by Lai et al.[110] and show that initialization can have a significant effect on solution quality. From its initialized state, asynchronous updating of the network will then allow a minimum energy state to be attained, because the energy level never increases during state transitions. 0–1 solutions of combinatorial problems can be encouraged if desired by setting the parameter T of the activation function to a small enough value that the function approximates the discrete thresholding (step) function given by Eq. 6.

However, these stable states may not necessarily correspond to feasible or good solutions of the optimization problem, and this is one of the major pitfalls of the H-T formulation. Because the energy function comprises several terms (each of which is competing to be minimized), there are many local minima, and a tradeoff exists between which terms will be minimized. An infeasible solution to the problem will arise when at least one of the constraint penalty terms is non-zero. If this occurs, the objective function term is generally quite small, because it has been minimized to the detriment of the constraint terms, thus the solution is "good" but not feasible. Alternatively, all constraints may be satisfied, but a local minimum may be encountered that does not globally minimize the objective function, in which case the solution is feasible but not "good." Certainly, a penalty parameter can be increased to force its associated term to be minimized, but this generally causes other terms to be increased. The solution to this trade-off problem is to find the optimal values of the penalty parameters that balance the terms of the energy function and ensure that each term is minimized with equal priority. Only then will the constraint terms be zero (a feasible solution), and the objective function be also minimized (a "good" solution).

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 (see [168]). It was, however, their results for the combinatorial Travelling Salesman Problem (TSP) that attracted the most attention. Hopfield and Tank[87] simulated a network of 10 cities (100 neurons), chosen at random on the interior of a two-dimensional unit square. Addressing the issue of parameter selection, they claimed that an "anecdotal exploration of parameter values was used to find a good (but not optimized) operating point."[87] They acknowledged the sensitivity of the results to the parameters of the system by stating that "the choice of network parameters which provides good solutions is a compromise between always obtaining legitimate tours . . . and weighting the distances heavily." However, they also state that "an appropriate general size of the parameters was easily found." Unfortunately, Hopfield and Tank omitted certain practical details from their article, including the method they used to simulate the differential Eq. 3 and the termination criteria. 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, because this is more representative of the biological system where, typically, a neuron may be connected to 1000–10,000 others. Because 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 parameters to generate valid tours, and comment 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[115] on the same 30-city problem.

1.3 The Wilson and Pawley Investigation

In 1988, two British researchers, Wilson and Pawley, published a paper[192] that raised doubts as to the reliability and validity of the H-T approach to solving COPs. Their original intention was to imitate the method for increasing city size until restricted by their computational resources. In doing so, they had hoped to find a procedure for scaling to the larger problem sizes of real interest. Starting with a 64-city problem, Wilson and Pawley were unable to find any combination of parameters that would result in a valid solution to the TSP. They then decided to try to reproduce the HT solutions for the 10-city problem in an attempt to find a method for changing the parameters in order to maintain feasibility for larger problems. Simulating the differential Eq. 3 using an Euler approximation, and using the identical parameters specified by Hopfield and Tank[87] for their 10-city problem, Wilson and Pawley found that from 100 random starts, only 15 converged to valid tours, while 45 froze into local minima corresponding to invalid tours, and the remaining 40 did not converge within 1000 iterations. These results were dramatically different from those produced by Hopfield and Tank on the identical problem. Moreover, Wilson and Pawley found that the 15 valid tours were only slightly better than randomly chosen tours. Similarly poor results were found when the 10 cities were randomly generated, with valid tours found only 8% of the time. Certainly, these inconsistent results could be due to differences in the simulation procedure or the termination criteria.

In their article,[192] Wilson and Pawley suggested many modifications to the original H-T formulation to try to correct some of the problems they encountered. These modifications included changes in parameter selection, and increasing the value of the distance from a city to itself in order to deter tours with one city visited twice in succession (a common reason for the invalidity of many of their generated tours). None of these modifications significantly improved the percentage of valid tours obtained, or the quality of the valid tours. Wilson and Pawley concluded that, even for small-sized problems, the original HT formulation "is unreliable and does not offer much scope for improvement." They were unable to discover how the parameters of the model need to change as the size is scaled up because no combination of parameter values (or operating point) could be found that consistently generated valid solutions. However, they also added, "it was felt that it was the method, and not the operating point, which was the root of the problems." As for the irreproducibility of Hopfield and Tank's impressive results, they state, "our simulations indicate that Hopfield and Tank were very fortunate in the limited number of TSP simulations they attempted."

1.4 Parameter Selection and Energy Modifications

Just as Wilson and Pawley felt that the H-T method for solving COPs was unjustified, others have put forward similar comments, questioning the stability of their formulation[95, 96, 170] and the necessity of an energy function with such a high degree of ultrametricity.[118] The solution to the problem, as far as a few of these researchers are concerned, is to abandon the concept of an interconnected system of neurons ever being capable of solving general optimization problems, and turn instead to problem-specific heuristics.[118] Most of these involve starting with a feasible solution and performing variations on swapping-type algorithms to try to minimize the objective function. Under such a scheme, feasibility is never lost, so solutions are guaranteed to be valid.

For many others, however, the fundamental worth of the H-T method is beyond contention. Of those who persist with Hopfield and Tank's original ideas, they are largely divided between two different approaches: rewriting the energy function to eliminate the trade-off problems between valid and good solutions; and accepting the H-T energy function while searching for ways to optimally select the penalty parameters. In 1988, Hegde et al.[79] published the results of their many experiments that studied the interaction between the penalty parameters of the H-T TSP formulation. Plotting in parameter space the region that led to feasible solutions, they found that, as the problem size was slowly increased, the wedge of parameter choices that resulted in feasible tours became narrower. Hedge et al.[79] concluded that their results imply inter-relationships among the parameters, and the nature of these relationships indicate that the H-T formulation for the TSP does not possess good scaling proper-

ties. Theoretical results relating to the nature of the relationships between the parameters of the H-T TSP formulation were published by Kamgar-Parsi et al.[96] They proposed a systematic method for selecting these parameters based on analyzing the dynamic stability of valid solutions. Examining the eigenvalues of the Jacobian matrix of the H-T energy function, they found conditions for the parameters under which a valid tour would be stable. Extensive studies by Davis[45] confirmed their analysis, which showed that there are only a very narrow range of parameter combinations that result in valid and stable solutions to the TSP—explaining the disappointing percentage of valid tours generated by many using the H-T formulation of the TSP. Despite these theoretical results, many researchers continue the search for methods of optimally selecting the penalty parameters. Recent approaches include genetic algorithms, whereby the penalty parameters are genetically bred and improved through subsequent generations to meet the validity requirements of the network (see [109]). Another approach that has been suggested[13] is to treat each penalty parameter as a Lagrange multiplier and attempt to find the optimal parameters exactly.

Efforts to obtain more consistent results by modifying the energy function have been more successful. Most of the approaches have involved rewriting the constraints and reducing the number of terms and parameters in the energy function.[21, 126, 177] Most of these approaches, while generating a greater percentage of feasible solutions, involve modifications that are specific to the TSP. In the following section, we review an important modification to the H-T approach that is suitable for the entire class of problems given by P1 in Section 1.2. We then review ways in which the solution quality of the Hopfield network approaches can be improved.

1.5 Ensuring Feasibility

Consider the general energy function first proposed by Aiyer.[7]

$$\mathscr{E}(\mathbf{v}) = f(\mathbf{v}) + \frac{\gamma}{2} E^{valid}(\mathbf{v}), \tag{8}$$

where $f(\mathbf{v})$ is the objective function expressed in the standard quadratic form $\mathbf{v}^T\mathbf{Q}\mathbf{v}$. The single constraint term E^{valid} consists of the deviation of the vector $\mathbf{v}$ from the constraint plane given by $\mathbf{Av = b}$. It is noted that a similar method was also proposed by Chu.[33] The advantage of this energy function is that only one penalty parameter, γ, needs to be selected. If γ is large enough, then validity of the solution is ensured, because the constraint term will be forced to vanish. Hence, the solution will necessarily lie on the constraint plane. A vector $\mathbf{v}$, when projected onto the solution space of $\mathbf{Av} = \mathbf{b}$ becomes:

$$\mathbf{v} \quad 4 \quad \mathbf{Proj.v} + \mathbf{s} \tag{9}$$

$$\text{where} \quad \mathbf{Proj} = \mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} \tag{10}$$

$$\mathbf{s} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T) - 1\mathbf{b}. \tag{11}$$

and $\mathbf{I}$ is the Identity matrix. Therefore, the deviation of the vector $\mathbf{v}$ from the constraint plane is given by

$$\|\mathbf{v} - (\mathbf{Proj.v} + \mathbf{s})\|^2.$$

So

$$\mathscr{E}(\mathbf{v}) = \mathbf{c}^T\mathbf{v} + \mathbf{v}^T\mathbf{Q}\mathbf{v} + \frac{\gamma}{2}[(\mathbf{Proj.v} + \mathbf{s}) - \mathbf{v}]^2$$

$$= -\frac{1}{2}\mathbf{v}^T[-2\mathbf{Q} + \gamma(\mathbf{Proj} - \mathbf{I})]\mathbf{v}$$

$$- \mathbf{v}^T[\gamma\mathbf{s} - \mathbf{c}] + \frac{\gamma}{2}\mathbf{s}^T\mathbf{s}.$$

Comparing $\mathscr{E}(\mathbf{v})$ to the standard energy function (Eq. 7), we can infer the weights ($\mathbf{W}$) and external inputs ($\mathbf{i}$) of the network from the coefficients of the quadratic and linear terms, respectively. It is noted that the weights are symmetric provided $\mathbf{Q}$ is symmetric, although if $\mathbf{Q}$ is not symmetric, it can be replaced with $(\mathbf{Q} + \mathbf{Q}^T)/2$, which is symmetric, without affecting the cost of the objective function. Additionally, the value of the diagonal terms, W_{ii}, may be positive, negative, or zero, depending upon the nature of the quadratic form $\mathbf{Q}$, the constraint plane, and the value of γ. Because negative diagonal terms may result in a convex energy function and thus convergence to a non-integral feasible solution, we add another term to the energy function to drive the solution trace towards a vertex point (or an alternative annealing strategy can achieve the same result). This term is given by $\delta\mathbf{v}^T(1 - \mathbf{v})$ (where $\mathbf{1}$ is a vector of 1's). The energy function is now given by:

$$\mathscr{E}(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^T[-2\mathbf{Q} + 2\delta\mathbf{I} + \gamma(\mathbf{Proj} - \mathbf{I})]\mathbf{v}$$

$$- \mathbf{v}^T[\gamma\mathbf{s} - \mathbf{c} - \delta\mathbf{1}] + \frac{\gamma}{2}\mathbf{s}^T\mathbf{s}.$$

Thus, the weight matrix and vector of external inputs of this feasible Hopfield network are given by:

$$\mathbf{W} = -2\mathbf{Q} + 2\delta\mathbf{I} + \gamma(\mathbf{Proj} - \mathbf{I}) \tag{12}$$

$$i = \gamma\mathbf{s} - \delta\mathbf{1} - \mathbf{c}. \tag{13}$$

We are still presented with the problem of how to select the parameters γ and δ optimally (if δ is required), so that the solution to which the network converges is both 0–1 and feasible, as well as being near-optimal. This issue can be avoided if we assign to γ a large enough value that the convergence trace is essentially "pinned" to the constraint plane. The parameter δ need only be a small fraction of γ, just large enough to drive the solution trace towards a vertex. Liapunov descent of the energy function will then ensure that the solution trace moves towards vertices that minimize the cost of the solution, while being (necessarily) feasible.

1.6 Improving Solution Quality

Many variations of the Hopfield network have been proposed for improving the solution quality. These approaches can be broadly categorized as either deterministic or sto-

chastic. There have also been developments in hardware implementation that have enabled local minima to be avoided (Lee et al.[112]) and problem-specific theoretical work on basins of attraction that enable the initial states of the network leading to good quality solutions to be calculated.[27] The deterministic approaches include problem-specific enhancements such as the "divide and conquer" method of Foo et al. for solving the TSP,[55] deterministic hill-climbing such as the "rock and roll" perturbation method of Lo,[119] and the use of alternative neuron models within the Hopfield network such as the winner-take-all neurons used by Amartur et al.[9] 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 activation function with a stochastic decision-type activation function,
2. add noise to the weights of the network,
3. add noise to the external inputs (biases) of the network, and
4. any combination of the above methods.

The *Boltzmann machine*[1, 82] utilizes the first method based on a discrete Hopfield model. The inputs are fixed, but the discrete activation function is modified to become probabilistic. Much like simulated annealing, the consequence of modifying the binary activation 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.[8] proposed *Gaussian machines* that combine features of continuous Hopfield networks and the Boltzmann machine. Gaussian machines have continuous outputs with a deterministic activation function like the Hopfield network, but random noise is added to the external input (bias) of each neuron. This noise is normally distributed (or Gaussian) with a mean of zero and a variance controlled by a temperature parameter. However, based upon Szu's fast simulated annealing,[161] which uses Cauchy noise to generate new search states and requires only a $t/log(t)$ cooling schedule, the *Cauchy machine*[160, 167] was proposed as an improvement to solution quality. The Cauchy distribution is said 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.[167] The noise is incorporated into the activation function, while the outputs of the Cauchy machine are binary. In the high-gain limit of the gradient of the stochastic activation function, the Cauchy machine approaches the behavior of the discrete (and deterministic) Hopfield network. Another stochastic approach that has been very successful is *mean-field annealing*,[133, 178, 179] so named because the model computes the mean activation levels of the stochastic binary Boltzmann machine.

Often, however, stochastic neural networks designed to "kick" a solution out of a local minimum suffer from instability.[73] In previous work,[157] we have suggested a modification to the internal dynamics of the modified Hopfield network described in Section 1.5 (with the feasibility guaranteed) that permits escape from local minima through hill-climbing. The rate of change of the neurons is controlled by a multiplier $\alpha(t)$ so that

$$\frac{du_i}{dt} = \alpha(t)\left(\sum_j W_{ij}v_j + i_i \right),$$

which is equivalent to

$$\frac{dv_i}{dt} = -\alpha(t)\frac{\partial f(\mathbf{v})}{\partial v_i}$$

for $\mathbf{v}$ within the unit hypercube and on the constraint plane. Thus, the steepest descent and ascent are achieved when $\alpha(t) = \pm 1$, respectively. The cooling schedule for the parameter $\alpha(t)$ is given by

$$\alpha(t) = random(k(t), 1) \quad \text{where } k(t) = 1 - 2e^{-t/T_k}.$$

Figure 2 shows how the value of k changes with time for $T_k = 40$.

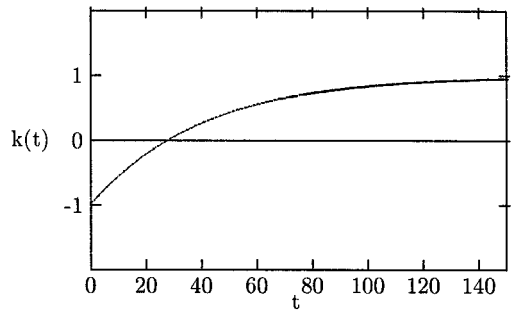

Figure 2. Graph of $k(t) = 1 - 2e^{-t/T_k}$ with $T_k = 40$.

The length of the Markov chain (or the number of random walks permitted in the search space) at each point in time is held constant at a value that depends upon the size of the problem. Initially, $k(0) = -1$, and so $\alpha(0)$ is randomly selected from within the range $[-1, 1]$. Consequently, the energy value (which is equivalent to the objective cost provided the solution trace is confined to the constraint plane through large γ) will often increase initially. As t 3 ∞, however, $k(t)$ 3 1, and so $\alpha(t)$ will also approach unity, which is needed for steepest descent. Thus, this hill-climbing Hopfield network method allows random increases in energy initially, with such increases becoming less likely as time proceeds, until finally the network tends towards a steepest descent (Hopfield) algorithm.

Clearly, there are many approaches to improving the solution quality of the Hopfield network through escape from local minima of the energy function and embedding stochasticity into the dynamics of the network. The valid subspace approach has resulted in a guarantee of feasibility as well. Thus, the initial problems that have plagued the reputation of the Hopfield network have now been resolved.

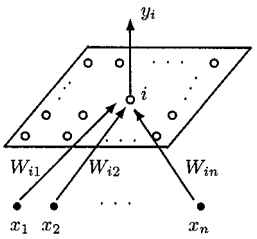

Figure 3. Kohonen's SOFM: mapping features of input vector **x** onto a two dimensional array of neurons.

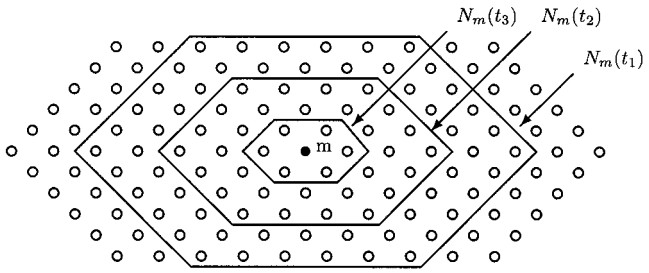

Figure 4. Hexagonal neighborhood function used by Kohonen.

2. Self-Organizing Neural Network Approaches

In this section, an alternative neural network architecture is described: self-organization. The origins of self-organization are presented, followed by a discussion of how this concept was combined with the geometry of the elastic net method to solve planar combinatorial optimization problems like the TSP.

2.1 Self-Organizing Feature Maps

Unlike the Hopfield network, the self-organizing feature map (SOFM)[101] does not contain preset weights that store information about the desired final states. Nor is supervision needed to guide the network externally to such states as with other kinds of neural networks.[18] The SOFM is an unsupervised neural network that simply inspects the input data for regularities and patterns and organizes itself in such a way as to form an ordered description of the data. Kohonen's SOFM converts input patterns of arbitrary dimensionality into the responses of a one- or two-dimensional array of neurons. Learning is achieved through adaptation of the weights connecting the input (pattern) layer to the array of neurons. Figure 3 shows an example of a SOFM with a two-dimensional array of neurons. Each component in the input vector **x** is connected to each of the neurons, and the weight W_{ij} transmits the input x_j toward the ith neuron of the feature array. Input **x** is applied simultaneously to all neurons.

The learning process comprises a competitive stage of identifying the *winning* neuron, which is "closest" (according to some similarity definition) to the input data, and an adaptive stage where the weights of the winning neuron and its neighboring neurons (according to some neighborhood definition) are adjusted in order to approach the presented input data. Identifying the winning neuron requires finding the neuron m such that

$$y_m = \min y_i = f(\mathcal{S}(\mathbf{x}, [\mathbf{W}]_i))$$

for all neurons i in the array, where $\mathcal{S}$ is a function of similarity between the input vector **x** and the ith row of the weight matrix, $[\mathbf{W}]_i$. Selection of the winning neuron also activates its neighboring neurons to react to the same input pattern. The concept of a *neighborhood* is generally regarded to be spatial. That is, neighboring neurons are those that are adjacent in the one- or two-dimensional array of neurons. Figure 4 shows a planar array of neurons with hexagonal neighborhoods, as used by Kohonen.[103] The spatial neighborhood of the winning neuron m is given by the set of neurons N_m. The radius of N_m (given by $r(N_m)$) should decrease as training progresses, with $r(N_m(t_1)) > r(N_m(t_2)) > r(N_m(t_3)) \ldots$, for $t_1 < t_2 < t_3 \ldots$.

The weight adaptation rule for the SOFM is defined by:

$$\Delta[\mathbf{W}]_i(t) = \alpha(N_i, t)[\mathbf{x}(t) - [\mathbf{W}]_i(t)] \quad \forall i \in N_m(t)$$

where $N_m(t)$ denotes the current (spatial) neighborhood of the winning neuron m. For the learning parameter α, Kohonen[103] used the function

$$\alpha(N_i, t) = \alpha(t)e^{-\|\mathbf{r}_i - \mathbf{r}_m\| / \sigma^2(t)}$$

where $\mathbf{r}_m$ and $\mathbf{r}_i$ are the position vectors of the winning neuron and its neighbors, respectively, and $\alpha(t)$ and $\sigma(t)$ are suitably chosen decreasing functions of the learning time t. After all input data have been presented to the network, the size of the neighborhood is reduced and the data presented again. This process continues until the weights connecting the input data to the array of neurons have stabilized. As a result of the weight adaptations, the collective and co-operative learning tunes the network to create localized responses to certain input vectors, and thus to reflect the topological ordering of the input vectors. As such, the SOFM can be considered as a nonlinear projection of the input space onto the neurons in the array that represent certain features.

Kohonen's SOFM has successfully been applied to sensory mapping, robot control, vector quantization pattern recognition and classification, and speech recognition.[102, 146] The principles of self-organization have also been used to solve the TSP, from within the geometry of the elastic net method. We first discuss this method and then present some of the more recent work that merges the two approaches of the elastic net and SOFM.

2.2 The Elastic Net Method

Durbin and Willshaw[49] first proposed the elastic net method in 1987 as a means of solving the TSP. Their approach places the discrete optimization problem into a continuous space framework—the Euclidean space of the TSP cities. The algorithm starts with k points (or nodes) lying on an imaginary "elastic band," where k is greater than the number of cities. The nodes are then moved in the Euclidean space, thus stretching the "elastic band," in such a way that

minimizes the energy function

$$\mathscr{E}(\mathbf{y}, K) = -\alpha K \sum_{i=1}^{k} \ln \sum_{j=1}^{k} e^{-\frac{|\mathbf{x}_i - \mathbf{y}_j|^2}{2K^2}} + \beta \sum_{j=1}^{k} |\mathbf{y}_j - \mathbf{y}_{j+1}|^2$$

where $\mathbf{x}_i$ represents the position of the ith city, $\mathbf{y}_j$ represents the position of the jth node on the "elastic band," and α or β are constants. The first term of the energy function is minimized when every city is covered by a node on the "elastic band," while the second term constitutes the length of the "elastic band," and hence the TSP tour length if the first term is negligible. This is achieved by starting the algorithm with a large value of the parameter K, and then gradually reducing K to keep to a local minimum of $\mathscr{E}$. The path is updated in each time step according to:

$$\Delta \mathbf{y}_j = -\alpha \sum_{i=1}^{k} W_{ij}(\mathbf{x}_i - \mathbf{y}_j) + \beta K(\mathbf{y}_{j+1} + \mathbf{y}_{j-1} - 2\mathbf{y}_j)$$

$$= -K\frac{\partial \mathscr{E}}{\partial \mathbf{y}_j}$$

where

$$W_{ij} = \frac{e^{-|\mathbf{x}_i - \mathbf{y}_j|^2/(2K^2)}}{\sum_k e^{-|\mathbf{x}_i - \mathbf{y}_k|^2/(2K^2)}}.$$

Thus, the elastic net method can be seen as gradient descent of the energy function, and in the limit as $K \to 0$, a local minimum will eventually be reached.

Durbin and Willshaw applied their elastic net method to the 30-city TSP used by Hopfield and Tank,[87] and obtained the best known solution[115] in 1000 iterations (Hopfield and Tank's best solution was 19% worse in terms of solution quality. The authors acknowledged, however, that their method was only suited to Euclidean TSPs and related problems, and not to random TSPs or problems that could not be embedded in the Euclidean plane.

2.3 Self-Organizing Approaches to the TSP

Even before Durbin and Willshaw's work on the elastic net method was published, Fort[56] had been working on the idea of using a self-organizing process to solve the TSP. His approach was to take a one-dimensional circular array of neurons, and map it onto the TSP cities in such a way that two neighboring points on the array are also neighbors in distance. Clearly, this is very similar to the idea behind the elastic net method. Fort acknowledges this by stating "During the time we worked on this paper, Willshaw and Durbin have published a paper . . . in which the basic idea is essentially the same as here and has the same origin."[56] The difference between the two approaches, however, is that Fort's algorithm incorporates stochasticities into the weight adaptations, whereas the elastic net method is completely deterministic in nature. There is also no energy minimization involved with the method of Fort. Testing his approach on the same TSP test sets used by Hopfield and Tank, however, Fort's results were not as good as those obtained using the elastic net method.

Subsequently, researchers began to combine features of both the elastic net and SOFM to arrive at a technique that performs well on the TSP (although results are largely restricted to testing the original TSPs used by Hopfield and Tank, rather than a wide range of TSPs differing in complexity). One of the first such approaches is due to Angéniol et al.[11] In their method, an approximate tour is given by a set of nodes (with coordinates (c_1, c_2)) joined together in a continuous ring. All nodes move freely in the Euclidean plane through an iterative process, until an actual tour has been obtained when every city (with coordinates (x_1, x_2)) has "caught" one node of the ring. This process is similar to the weights converging to the input patterns of the Kohonen SOFM. An iteration step consists of the presentation of one city. The node closest to the city moves towards it and induces its neighbors to move some distance toward the presented city, which is a function $f(\alpha, d)$ of their distance (d) along the ring, and a gain parameter α.

Angéniol et al.[11] also included some node creation and deletion rules. A node is duplicated if it is chosen as the winner for two different cities. The duplicate is inserted onto the ring as a neighbor of the winner, but only permitted to move when the next city is presented. A node is deleted if it has not been selected by a city after three complete presentations of all the N cities. Testing their approach on the same set of TSPs used by Hopfield and Tank,[87] Angéniol et al. found their results to be comparable in quality to the best results of the elastic net method of Durbin and Willshaw.[49] A good average solution (less than 3% greater than the optimum) was obtained in less than 2 seconds on classical hardware, and a solution to a 1000-city problem was found in 20 minutes on a digital computer (they do not mention the machine specifications). The authors concluded that their approach to solving the TSP through a modified SOFM appears to be very promising because the total number of nodes and connections is proportional to N (the number of cities), and thus scales well with problem size (compared to Hopfield's model, which has N^2 nodes and N^4 connections). They also view the single parameter α, which controls the number of iterations as a further advantage.

Burke and Damany[24] suggested an alternative to the approach of Angéniol et al., which eliminates the need for the node creation and deletion rules. They proposed the *guilty net* for solving the TSP, in which the number of nodes is always equal to the number of cities. Using the idea of *conscience* introduced by DeSieno,[47] the guilty net ensures feasibility of the TSP tour as a minimum requirement, with a nearly feasible solution being provided if the network is terminated prematurely. The conscience element is incorporated in a bias term added to each node when the winning node is calculated. This bias term has the effect of inhibiting nodes that are winning too often, and takes the form:

$$bias_j = \frac{win_j}{1 + \sum_k win_k} \tag{14}$$

for each node j, where win_j is the number of times node j has been the winning node thus far. Although the results presented in [24] for TSP problems of size 10, 30, 50, and 100 are

poorer than the elastic net method (on average 9.5% poorer) or simulated annealing (on average 2.2% poorer), the authors point out that the guilty net requires substantially fewer iterations and can terminate before convergence to arrive at an approximate solution.

In more recent work, Burke[23] has extended the guilty net to automate the separation of nodes (rather than needing the bias term and its associated parameter). This new network is called the *vigilant net* and uses a simpler measure inspired by the vigilance of the adaptive resonance networks.[26] Winning nodes are now determined according to a set vigilance level. The effect of the vigilance net is to constrain the ratio between the lengths of the longest and the average links in the TSP tour to be within some desirable interval. Further, the processing time is near linear in the number of cities. 100- and 200-city problems (both uniformly and non-uniformly generated) were tested and compared to heuristics from operations research, including the space-filling curve approach of Platzman and Bartholdi.[136] While the vigilant net compared well with the space-filling curve heuristic, previous neural approaches were not tested on the new data sets.

Favata and Walker[52] also borrowed the idea of a circular array of neurons mapping onto the set of TSP cities. Their use of the SOFM algorithm is much more literal, however, and involves a minimal amount of modification. On the same 30-city TSP used by Hopfield and Tank,[87] Favata and Walker's approach yielded solutions that were on average about 5% more costly than those generated by Kirkpatrick et al.[100] using simulated annealing, but that were obtained consistently faster. Favata and Walker also assessed the scaling characteristics of the algorithm and found that the system converged reliably regardless of the problem size.

Vakhutinsky and Golden[174] considered speeding up the convergence process by employing a hierarchical strategy with the elastic net. The idea of their algorithm is to subdivide the plane into smaller square areas and replace the nodes in each square with the "center of gravity" of those nodes. As the algorithm proceeds, the squares are further divided so that eventually the TSP cities are all represented with no more than one city per square. A speed-up of around 16 times was achieved for a 500-city problem, and the speed-up appears to increase as the problem size is scaled up. This approach is one of global to local perspective, and has also been successfully applied to other neural algorithms[36] to speed-up convergence.

Several other approaches have also emerged over recent years, which are very similar to the aforementioned methods. While Fort[56] suggested some potential applications to graph projection as well as classification of statistical data, other authors have extended these approaches to deal with the Multiple TSP[70] and obstacle avoidance tour planning.[193] All of these approaches are based on a low dimensional deformable template method like the elastic net method. Clearly, the method of merging details of the SOFM algorithm with the elastic net method is very successful for solving TSPs and other optimization problems embedded in the Euclidean plane. However, owing to the fact that the geometry of these methods is based upon the elastic net method (utilizing a circular ring of neurons), they are not readily generalizable to solve any other non-Euclidean problems. In this respect, such approaches are not nearly as useful as the Hopfield-Tank method,[87] which generalizes to solve many practical and diverse optimization problems. Angéniol et al.[11] noted the absence of generalization and stated that they "... plan to generalize this method to other optimization problems by varying the organizing topology."

Recently, such an approach has been proposed,[154, 155] which is not based on the elastic net method and so generalizes to solve a large class of problems with quadratic objective functions and linear constraint sets. The basic idea is to present rows of a permutation matrix (which represents a feasible solution) to a self-organizing feature map whose outputs represent each row number. The outputs compete to gain possession of the row of the permutation matrix under consideration. The competition is based on minimizing the objective function. We refer the interested reader to [155] for more details about this generalization, its implementation, and applications.

3. A Survey of Applications

The majority of neural network research is applied to try to solve the TSP: for historical reasons this problem has become a benchmark. It has been argued, however, that the TSP is perhaps not the best benchmark by which to judge the effectiveness of neural networks for optimization.[156] Almost every class of combinatorial (and non-combinatorial) optimization problem has been tackled by neural networks over the last decade, and many of the results are very competitive with alternative techniques in terms of solution quality, particular those of recent years. Unfortunately, many researchers have not always compared their neural approach with the best performing alternative technique available at the time, and in doing so, their results are often viewed as inconclusive. Recent articles that define appropriate methods for reporting computational experiments and testing heuristics[17, 84] appear to have rectified this tendency.

In the following sections, we briefly review some of the many neural network approaches that have been reported for classical COPs. This survey is not intended to be exhaustive, but provides interested readers with appropriate starting points for considering neural approaches to various classes of problems.

3.1 Assignment Problems

Assignment problems have been well tackled by neural network researchers, no doubt due to the wide variety of practical applications that can be categorized as either general assignment problems (with a linear objective function) and quadratic assignment problems. While there exist a number of reliable and efficient exact algorithms for solving linear assignment problems,[195] neural algorithms are still pursued as a solution technique for their promise of rapid execution through eventual hardware implementation.

3.1.1 General Assignment Problems.

The general assignment problem is best known as the problem of assigning N tasks to N people, where each person may only perform one task. This problem has applications to crew scheduling and manpower planning,[124] as well as allocation applications such weapon target assignments.[185] The constraints are similar to the TSP constraint set, and so solution by neural networks has been common.[188] Cho et al.[31] have used a discrete Hopfield network to solve the general assignment problem using a parallel algorithm that converges in near 0(1) time (less than 100 iterations regardless of the size of the problem). Gong et al.[71] use a continuous Hopfield network to solve the general assignment problem and add quadratic concave equality constraints to the energy function to ensure zero-one solutions. They employ an augmented Lagrangian multiplier method to determine the penalty parameters and show that this helps to alleviate the problems of convergence to local minima. A comparison was not performed with existing solution techniques for linear assignment problems however.

3.1.2 Quadratic Assignment Problems.

Quadratic assignment problems find even more application in the real world, and their quadratic objective function means that they are particularly well suited to solution using energy based networks. Hopfield networks have been used to solve facility location problems,[159] cell placement in VLSI design,[44, 172] and assignment of frequencies in satellite and wireless communication networks.[58, 106, 107, 158] Several authors have also proposed neural networks to solve general quadratic assignment problems, regardless of the application, and have tested the applicability of their techniques using a wide range of applications. These include the self-organizing approach of Smith[154] and the mean-field annealing approach of Fang et al.[51]

3.2 Constraint Satisfaction Problems

There are many well-known constraint satisfaction problems found in the literature: both practical ones and those that could best be described as puzzles. Problems such as the N-Queen problem, or the Stable Marriage Problem, involve constraints that are similar to assignment constraints, and so the solution of these puzzles has applicability to a wide range of applications employing similar constraints.

In 1987, Tagliarini and Page[162] used a Hopfield network to solve the N-Queen problem. This problem involves placing N Queens on a chess board in such a way that there is only one Queen in each row, column and diagonal, so that the N Queens are in a configuration where none are under attack. This problem has no objective function, and so the standard Hopfield-Tank method is able to quickly converge to a feasible solution. Takefuji[165] and others[5, 93] have solved many such puzzles using parallel neural networks including tiling problems and the stable marriage problem.[198] Other types of constraint satisfaction problems, involving constraints such as k-out-of-N constraints (where k elements in a row of N have to be "on") have been addressed.[50, 151]

Research into appropriate representations of certain constraints has resulted in automatic translation techniques[59] and the ability to handle difficult constraints such as logical constraints.[169] Knowledge of the best ways to incorporate certain constraints into a neural network is imperative if practical applications such as timetabling are to be attempted. Timetabling is a complex problem, and includes class scheduling, examination timetabling, rostering, etc. There have been several successful implementations of neural networks for timetabling[66, 104, 123, 134] (mostly based on mean-field annealing neural networks), and these results compare well to heuristic approaches.

3.3 Clustering Problems

The problem of clustering data is equivalent to the partitioning of a set of N patterns in a metric space into K clusters, so that those patterns in a given cluster are more similar to each other than to the rest of the patterns. By minimizing the distance between each point and the centroid of each cluster, the problem can be formulated as 0–1 quadratic programming problem, where the (binary) variables represent the probability that a data point is assigned to a certain cluster. Hopfield networks have been used for clustering[94] and were shown to outperform conventional iterative techniques when the clusters are well defined. For fuzzy clustering, or when the number of clusters is not compatible with the structure of the data, the neural network is unable to find valid solutions easily, indicating that something may be wrong with the problem description. Boltzmann machines have also been investigated for clustering[15] and tested using well-known data sets. The results have been compared with traditional clustering algorithms such as K-means[91] and show that neural network methods are able to outperform traditional techniques in this application area in terms of both solution quality and speed. Self-organizing neural network architectures such as Kohonen's Feature Map[103] and the Counter-propagation network[78] are also suitable for clustering.[30]

3.4 Cutting Stock and Packing Problems

This category of problems includes sub-classes such as bin-packing, knapsack, and cutting problems. Solutions to these problems are important because many of these problems find application in industry. Takada et al. have used a neural network for solving the problem of cutting steel sheets into assorted sizes,[163] and such systems find applicability in glass-cutting and other industries. Industrial packing problems have also been solved using neural approaches. Dai et al.[42] have used common heuristic packing rules to construct a Hopfield network to solve a two-dimensional packing problem, while Bahrami et al.[16] have developed a hybrid intelligent system, comprising a neural network and a rule-based system, to solve industry packing problems.

3.4.1 Knapsack Problems.

Most of the research in the application to packing problems has been focused on the knapsack problem. This is no doubt

due to the inequality constraints that require special treatment when using energy function related approaches.[3, 113] Mean-field theory techniques have managed to solve large-scale knapsack problems using a discrete form of the derivative to compensate for the non-differentiable energy function. Ohlson et al.[129] use a mean-field algorithm that scales like NM for problems with N items and M constraints, and solve problems of size up to 10,000 items. Their results are comparable to those obtained using simulated annealing and an exact formulation, and are better than a greedy heuristic. The multidimensional knapsack problem has also been solved using a Boltzmann machine,[176] although the results were not compared to any other technique. Alternative neural network approaches, including a Hopfield network with asymmetric weights[196] and the Dual-Mode Dynamics Neural Network of Lee and Park,[114] have been proposed to handle inequality constraints, and tested using the knapsack problem. Vinod et al.[184] use the idea of orthogonal projections onto convex sets to solve problems with inequality constraints, including the 0–1 knapsack and multidimensional knapsack problems. Their results are shown to compare quite well with optimal and suboptimal solutions obtained using standard techniques.

3.5 Graph Problems

There are many types of graph problems found in the operations research literature that have also been attempted using neural networks. Ramanujam and Sadayappan[142] provide a very good overview of how several graph problems can be solved using a Hopfield network, including graph partitioning, vertex covering, maximum independent set and maximum clique problems, number partitioning, maximum matching, set covering, and graph coloring problems. Their paper shows suitable energy function representations for each of the problems, and the consequent weights and external inputs of the Hopfield network, but does not report results for any of the formulations. Many other authors have used neural networks to solve various graph problems, including Hérault and Niez,[81] and we now briefly review some of the approaches that have been taken.

3.5.1 Graph Sectioning, Partitioning, and Minimum Cut Problems.

Graph bipartitioning is a well-studied problem that finds application in the design of VLSI circuits.[132] Van den Bout and Miller[179] have used mean-field annealing to solve the bipartitioning problem, and their results extend to solve the (more computationally difficult) problem of partitioning into three or more bins. Such problems find extensions to the bin-packing problem, which in turn can be used to solve resource allocation problems. The results for the mean-field annealing network were found to be superior to those obtained using simulated annealing and the heuristic approach of Kernighan and Lin.[98] In fact, the rate of convergence of mean-field annealing on graph bipartitioning problems was as much as 50 times faster than simulated annealing without degrading the solution quality.[179] Other mean-field annealing and Boltzmann machine solutions have been equally successful.[22, 202] Unsupervised competitive neural networks have also been used to solve the circuit bipartitioning problem[171] (also called the Minimum Cut problem), and their results are shown to be comparable to those found by the ratio cut algorithm of Wei and Cheng.[191] Hameenanttila and Carothers[75] consider the use of a Hopfield network to solve the related problem of partitioning circuits with thermal constraints. This inequality constraint is handled using a slack variable approach, and while the results look promising, comparison with existing heuristics is reserved for future work.

3.5.2 Vertex Covering, Maximum Independent Set, and Maximum Clique Problems.

The vertex covering problem and its related problems of finding the maximum independent set and maximum clique find important applications in project scheduling, cluster analysis, facility location problems, and other problems from operations research.[142] Lai et al.[108] used a Hopfield network to solve these problems, but their energy function representations are different from those used by Ramanujam and Sadayappan[142] and are based on logical functions. Problem sets of various sizes are considered, and the Hopfield network results are shown to outperform a sequential greedy algorithm and perform comparably to two fast gradient descent heuristics in terms of solution quality. Jagota[89] has considered the maximum clique problem alone and presents several energy minimizing dynamics of a Hopfield network, both discrete (deterministic and stochastic) and continuous. Some of these dynamics emulate well-known heuristics for the maximum clique problem, and the results show that mean-field annealing and a stochastic version of the Hopfield network perform the best of the neural networks and are comparable to the best of the traditional heuristics.

3.5.3 Graph Coloring Problems.

The problem of coloring or labeling the vertices of a graph so that no two adjacent vertices are the same color is one that finds application in areas such as frequency assignment problems[58, 158] and computer compiler optimization.[62] Takefuji and Lee[166] have used a discrete Hopfield-type network to solve the problem using four colors, while Berger[19] has considered the general problem using k colors. Gassen and Carothers[62] have extended these models to try to minimize the number of colors required for a coloring of a given graph. While their results were not compared to alternative techniques, studies that use similar formulations for applications based on graph coloring problems have confirmed that Hopfield networks can match the performance of simulated annealing and other heuristics in this area.[158]

3.5.4 Tree Problems.

There are many different types of problem that involve finding a tree structure on a graph, including minimum

spanning trees, degree constrained minimum spanning trees, Steiner trees, and shortest path problems. Many of these problems are important because they find application in areas such as the design of electrical connections on a printed circuit board or traffic routing through computer nodes. Craig et al.[37] have conducted an extensive study where Hopfield neural network solutions to the degree-constrained minimum spanning tree problem are compared to a variety of heuristics including simulated annealing and algorithms of Kruskal[105] and Dijkstra.[48] The Hopfield network consists of two layers: one for minimizing the cost of branches in the graph while satisfying the degree constraints, and a second layer for determining if the current graph is a tree or not. The Hopfield network was found to converge to a feasible solution in only approximately 56% of the test problems, but when a feasible solution was found it was generally very close to the known optimal solution. Pornvalai et al. have also used a Hopfield network to solve the constrained Steiner tree problem.[138] This problem reduces to a shortest path problem when the number of destination nodes is one or to a minimum spanning tree problem when the number of destination nodes equals the number of nodes in the graph. Their results show that the Hopfield network can find near optimal solutions in randomly connected graphs, and the performance is comparable to the best heuristics.

3.5.5 Shortest Path Problems.

The shortest path problem is concerned with finding the shortest path from an origin node to a destination node in a directed graph. It finds significant application in the routing of traffic in telecommunications networks[143, 199] and cell placement in VLSI design to minimize total wire length.[27, 164] Neural network solutions to these problems have been very successful. Wang[189] has used a Hopfield network based on an edge path representation (leading to a linear programming formulation) of the shortest path problem. The results show that the Hopfield network can locate optimal solutions to the example problems considered, but no comparison with other techniques was provided. Hong et al.[83] also use a Hopfield network, but their energy function is extremely complicated due to their formulation of the constraints. Although their results show that minimum cost solutions can be found for small sized problems, the many local minima of their energy function will make it unlikely that a strict descent technique like the standard Hopfield network will be able to find optimal solutions for larger problems. Zhao and Dillon[200] eliminate local minima problems by using a variation on the winner-take-all network, which does not involve any energy function minimization. Their approach is guaranteed to reach optimal solutions and possesses good scaling properties.

3.6 Integer and Mixed-Integer Linear Programming Problems

Neural networks can be used to solve 0–1 linear programming problems, by expressing the 0–1 constraint as a quadratic term $\Sigma x_i(1 - x_i)$ added to the energy function. Aourid et al.[13] have used this technique to solve problems with known optimal solutions and have found optimal solutions using a Hopfield network by determining the lower bound of the penalty parameter that enforces the 0–1 constraint. In the work of Foo and Takefuji,[54] the 0–1 constraints of the integer linear programming problem are satisfied through high gain of the activation function for a Hopfield network (so that the continuous activation function approximates the discrete step function). Continuous linear programming can be achieved by using linear activation functions.[39, 43, 97] Consequently, mixed integer linear programming can be achieved through appropriate choice of the activation function to reflect the desired state of each final decision variable. Lin et al.[117] have proposed an alternative to the Hopfield network for the solution of mixed integer linear programming problems. They employ the traditional branch-and-bound approach,[72, 111] but use backpropagation learning with a multilayered feedforward neural network to learn more efficient branching strategies. The technique is shown to be significantly faster than conventional methods for certain classes of mixed integer linear programming problems, particularly knapsack problems.

3.7 Scheduling Problems

Scheduling problems constitute quite a large class of practical optimization problems from industry. These problems include the scheduling of resources such as machinery (job-shop scheduling), labor (crew scheduling), and timetabling.

3.7.1 Crew Scheduling.

Airline crew scheduling has received much attention from the operations research community, principally because it can be formulated as a set partitioning problem, and an optimal solution would enable many related problems to be solved.[32] Airline crew scheduling has been solved using a Boltzmann machine[38] and tested using both artificially generated and real data sets. The results demonstrate that near-optimal solutions can be obtained, but solutions to large problems are inhibited by excessive computation times. The scheduling of crew in the fast food industry has also been attempted using neural networks.[137] This problem (and other manpower scheduling problems) involves the assignment of crew members to jobs so that the correct number of people are scheduled at the right time, performing the right tasks, and with the right ability. Poliac et al.[137] use a network of parallel distributed elements with inhibitory and excitatory connection to enforce the labor, proficiency and availability constraints. Their results were unreported due to the preliminary nature of their study.

3.7.2 Job-Shop Scheduling.

Most of the literature on using neural networks for scheduling has been focused on job-shop scheduling. This is the problem of scheduling each operation i of each job j to a machine k, where there are m machines. Clearly, job-shop scheduling is a resource allocation problem. Precedence relationships can make the problem even more tightly con-

strained. Foo and Takefuji[53] have proposed a stochastic (hill-climbing) version of the Hopfield network for solving the problem with precedence constraints. Their approach produces near optimal solutions, but requires a large number of neurons and is computationally expensive. Foo and Takefuji[54] have also proposed a more efficient approach based on an integer linear programming formulation of the job shop scheduling problem, which halves the number of neurons required. By defining a continuous variable (starting time for a job on a machine), rather than a 0–1 variable, Zhou et al.[201] were able to reduce further the number of neurons and weights needed to solve the problem. There have been many other successful implementations of neural networks for scheduling and sequencing jobs on machines.[28, 99, 135]

3.7.3 Scheduling Applications.

Ansari et al.[12] have used mean-field annealing to schedule computation tasks onto a multiprocessor. Their experimental results indicate that this technique is very effective for solving such problems. Other authors have also solved multiprocessor task scheduling using neural approaches.[80, 121, 144] A stochastic neural approach has been proposed by Vaithyanathan et al.[175] for resource-constrained scheduling. Project scheduling is an important type of resource-constrained scheduling, and has also been attempted using neural networks.[131] Applications to assembly line scheduling have been considered using both Hopfield networks and self-organizing neural networks,[40, 147, 155, 157] and the results appear to compare well with traditional techniques in terms of solution quality. Obviously for industrial applications, solution speed is an important factor, and suitable hardware advances are needed for these techniques to be truly useful to industry.

3.8 Travelling Salesman Problems

Owing to the fact that Hopfield and Tank[87] originally demonstrated the applicability of their neural network for solving optimization problems by solving the TSP, almost all new neural network techniques for optimization are tested on the same problem. Surveys of many of these approaches for solving the TSP can be found in articles by Burke,[23] Looi,[120] and Potvin.[139] Like most neural network applications to optimization, there are mainly two approaches: Hopfield-type networks[21, 64, 116, 177] (energy minimization) and self-organizing approaches[11, 35, 56, 92, 194] (based on the elastic net method.[49] These approaches have already been discussed to some extent in Sections 1.4 and 2.3, respectively. There have also been studies that use neural networks to solve generalizations of the TSP[10] and variants of the TSP like the Orienteering Problem.[190] Further generalizations of the TSP (the Multiple TSP and Vehicle Routing Problems) are discussed below.

3.8.1 Multiple TSPs.

The multiple TSP involves minimizing the distance travelled by multiple salesmen, where each city must now be visited by exactly one salesman sharing each depot. Extensions of the Hopfield-Tank approach have been considered to solve this generalized version of the TSP.[181, 186, 187] Self-organizing approaches have also been successfully applied to the multiple TSP. Goldstein[70] has extended the elastic net approach, and these results show that self-organizing approaches can be very competitive with simulated annealing and other heuristics.

3.8.2 Vehicle Routing Problems.

Vehicle routing involves the problem of finding a route, for each vehicle, that begins at the depot, visits a subset of the cities, and returns to the depot. The distance travelled must be minimized, while the capacity constraints of each vehicle restrict the choice of cities a vehicle can visit in any single trip. Like the multiple TSP, vehicle routing has been solved successfully using elastic net and self-organizing approaches,[65, 122, 140, 173] and energy minimization approaches.[128]

4. Conclusions

In this article, we have attempted to present the current standing of neural networks for combinatorial optimization. We have traced the history of neural network research from the perspective of the two main approaches: Hopfield networks and self-organizing networks. This history has seen neural networks for combinatorial optimization progress from a field plagued with problems of poor solution quality and infeasibility to the state we find them in today: quite competitive with meta-heuristics such as simulated annealing in terms of solution quality.

Nevertheless, there are still several areas of research needing attention. First, there is the never-ending quest to improve solution quality by novel techniques. In this regard, alternative models of neuron dynamics should be investigated, such as chaotic neural networks, which have recently been shown to help improve the search for global minima through their rich dynamics.[29, 57, 77, 88, 127] Another promising direction lies in the hybridization of neural networks with meta-heuristics such as genetic algorithms and simulated annealing in such a way that the advantages of each of the techniques can be combined to overcome the known limitations.[14, 41, 109, 150, 197]

Second, more practical applications need to be solved using neural networks to demonstrate their potential. To date, most research has been focused on solutions to the TSP, and although a wide range of classical combinatorial (and noncombinatorial) optimization problems have also been tackled using neural networks, extensive studies that evaluate the potential of using neural networks to solve practical optimization problems from industry are scarce. In this article, we have briefly reviewed the research that has been done by considering the most common classes of combinatorial optimization problems and trying to report comparisons with alternative techniques. Such comparisons are not always available because, as Looi noted, “although there is a large collection of operations research based on other methods for solving all of these problems, comparisons be-

tween neural network methods with existing methods have been lacking."[120] This is the third area in which future research must focus. The problem is now being addressed as comparisons with existing techniques is seen to be essential for the results to be considered significant.[17, 84]

An accurate evaluation of the capabilities of neural networks for obtaining near-optimal solutions to optimization problems is necessary if neural networks are to be developed and employed in practical situations where their advantages over existing techniques can truly be exploited. Realizing this potential is the fourth main area for future research. While hardware implementation of a neural network is ideal for industrial situations, where the same problem will need to be solved many times as the environment changes, it is however an unlikely choice for an operations researcher, whose methodology tends to be simulation based, and who typically only needs to solve the problem once. Fortunately, recent work in the area of Field Programmable Gate Arrays (FPGAs)[20] has enabled the speed advantages of hardware implementation to be simulated on a digital computer using reconfigurable hardware with desktop programmability. Such a simulation can easily achieve speeds of several million interconnections per second, making the advantages associated with hardware implementation of neural networks more readily attainable. Certainly, satisfactory hardware implementation is still the topic of much research, and many design challenges lie ahead in this field.[34] Yet there is little doubt that it is only a matter of time before VLSI implementations of large scale neural networks are possible.[76, 149, 153] Already, researchers have been very successful in demonstrating this potential on small to medium-sized neural networks that can solve combinatorial and other optimization problems.[141, 148, 182]

Acknowledgements

The author is grateful to three anonymous referees, an associate editor, Dr. M. Gendreau, and Dr. B. Golden for their helpful comments and suggestions.

References

1. E. H. L. AARTS and J. KORST, 1989. *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons, Essex, U.K.
2. E. H. L. AARTS and P. J. M. LAARHOVEN, 1985. Statistical Cooling: A General Approach to Combinatorial Optimisation Problems, *Philips Journal of Research 40*, 193–226.
3. S. ABE, J. KAWAKAMI, and K. HIRASAWA, 1992. Solving Inequality Constrained Combinatorial Optimization Problems by the Hopfield Neural Networks, *Neural Networks 5*, 663–670.
4. D. H. ACKLEY, G. E. HINTON, and T. J. SEJNOWSKI, 1985. A Learning Algorithm for Boltzmann Machines, *Cognitive Science 9*, 147–169.
5. H. M. ADORF and M. D. JOHNSTON, 1990. A Discrete Stochastic Neural Network Algorithm for Constraint Satisfaction Problems, *Proceedings International Joint Conference on Neural Networks 3*, San Diego, 917–924.
6. S. V. B. AIYER, M. NIRANJAN, and F. FALLSIDE, 1990. A Theoretical Investigation into the Performance of the Hopfield Model, *IEEE Transactions on Neural Networks 1*, 204–215.
7. S. V. B. AIYER, 1991. Solving Combinatorial Optimization Problems Using Neural Networks, Technical Report CUED/F-INFENG/TR 89, Cambridge University Engineering Department, Cambridge, U.K.
8. Y. AKIYAMA, A. YAMASHITA, M. KAJIURA, and H. AISO, 1989. Combinatorial Optimization with Gaussian Machines, *Proceedings IEEE International Joint Conference on Neural Networks 1*, 533–540.
9. S. C. AMARTUR, D. PIRAINO, and Y. TAKEFUJI, 1992. Optimization Neural Networks for the Segmentation of Magnetic Resonance Images, *IEEE Transactions on Medical Imaging 11*, 215–220.
10. R. ANDRESOL, M. GENDREAU, and J.-Y. POTVIN, 1997. A Hopfield-Tank Neural Network Model for the Generalized Traveling Salesman Problem, in *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, S. Voss et al. (eds.), Kluwer Academic Publishers, Boston, 393–402.
11. B. ANGENIOL, G. DE LA CROIX, and J.-Y. LE TEXIER, 1988. Self Organising Feature Maps and the Travelling Salesman Problem, *Neural Networks 1*, 289–293.
12. N. ANSARI, Z. Z. ZHANG, and E. S. H. HOU, 1993. Scheduling Computation Tasks onto a Multiprocessor System by Mean Field Annealing of a Hopfield Neural Network, in *Neural Networks in Design and Manufacturing*, J. Wang and Y. Takefuji (eds.), World Scientific, Singapore, 163–184.
13. S. M. AOURID, X. D. DO, and B. KAMINSKA, 1995. Penalty Formulation for 0–1 Linear Programming Problem: A Neural Network Approach, *Proceedings International Conference on Neural Networks 4*, 1690–1693.
14. J. ARABAS, 1994. A Genetic Approach to the Hopfield Neural Network in the Optimization Problems, *Bulletin Polish Academy of Sciences 42*, 59–66.
15. G. P. BABU and M. N. MURTY, 1994. Connectionist Approach for Clustering, *Proceedings International Conference on Neural Networks 7*, 4661–4666.
16. A. BAHRAMI and C. DAGLI, 1994. Hybrid Intelligent Packing System (HIPS) Through Integration of Artificial Neural Networks, Artificial Intelligence and Mathematical Programming, *Applied Intelligence 4*, 321–336.
17. R. S. BARR, B. L. GOLDEN, J. P. KELLY, M. G. C. RESENDE, and W. R. STEWART, 1995. Designing and Reporting on Computational Experiments with Heuristic Methods, *Journal of Heuristics 1*, 9–32.
18. R. BEALE and T. JACKSON, 1990. *Neural Computing: An Introduction*, IOP Publishing Ltd., Bristol, U.K.
19. M. O. BERGER, 1994. k-Coloring Vertices Using a Neural Network with Convergence to Valid Solutions, *Proceedings International Conference on Neural Networks 7*, 4514–4517.
20. N. BOTROS and M. ABDUL-AZIZ, 1994. Hardware Implementation of an Artificial Neural Network Using Field Programmable Gate Arrays (FPGA's), *IEEE Transactions on Industrial Electronics 41*, 665–667.
21. R. D. BRANDT, Y. WANG, A. J. LAUB, and S. K. MITRA, 1988. Alternative Networks for Solving the Travelling Salesman Problem and the List-Matching Problem, *Proceedings International Conference on Neural Networks 2*, 333–340.
22. T. BULTAN and C. AYKANAT, 1991. Circuit Partitioning Using Parallel Mean Field Annealing Algorithms, *Proceedings 3rd IEEE Symposium on Parallel and Distributed Processing*, 534–541.
23. L. I. BURKE, 1994. Adaptive Neural Networks for the Traveling Salesman Problem: Insights from Operations Research, *Neural Networks 7*, 681–690.
24. L. I. BURKE and P. DAMANY, 1992. The Guilty Net for the Travelling Salesman Problem, *Computers and Operations Research 19*, 255–265.

25. L. I. BURKE and J. P. IGNIZIO, 1992. Neural Networks and Operations Research: An Overview, *Computers and Operations Research 19*, 179–189.
26. G. CARPENTER and S. GROSSBERG, 1987. ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns, *Applied Optics 26*, 4919–4946.
27. D. D. CAVIGLIA, G. M. BISIO, F. CURATELLI, L. GIOVANNACCI and L. RAFFO, 1989. Neural Algorithms for Cell Placement in VLSI Design, *Proceedings IEEE International Joint Conference on Neural Networks 1*, 573–580.
28. S. CHANG and B. H. NAM, 1994. Hopfield-type Neural Networks for Standard Form Linear Programming and Jobshop Scheduling, *Transactions Korean Institute Electrical Engineering 43*, 1361–1369.
29. L. CHEN and K. AIHARA, 1995. Chaotic Simulated Annealing by a Neural Network Model with Transient Chaos, *Neural Networks 8*, 915–930.
30. S. K. CHEN, P. MANGIAMELI, and D. WEST, 1995. The Comparative Ability of Self-Organizing Neural Networks to Define Cluster Structure, *Omega 23*, 271–279.
31. Y. B. CHO, T. KUROKAWA, Y. TAKEFUJI, and H. S. KIM, 1993. An O(1) Approximate Parallel Algorithm for the n-Task n-Person Assignment Problem, *Proceedings International Joint Conference on Neural Networks 2*, Nagoya, 1503–1506.
32. N. CHRISTOFIDES, 1975. *Graph Theory: An Algorithmic Approach*, Academic Press, New York.
33. P. CHU, 1992. A Neural Network for Solving Optimization Problems with Linear Equality Constraints, *Proceedings IEEE International Joint Conference on Neural Networks 2*, 272–277.
34. J. COLLINS and P. A. PENZ, 1989. Considerations for Neural Network Hardware Implementations, *Proceedings IEEE International Symposium on Circuits and Systems*, Portland, 834–847.
35. M. COTTRELL and J. C. FORT, 1986. A Stochastic Model of Retinotopy: A Self-Organizing Process, *Biological Cybernetics 53*, 166–170.
36. S. COY, B. GOLDEN, E. WASIL, and G. RUNGER, 1997. Evaluating the Effectiveness of Fine-Tuned Learning Enhancement to Backpropagation, in *Intelligent Engineering Systems Through Artificial Neural Networks*, C. Dagli et al. (eds.), ASME Press, New York, 105–111.
37. G. CRAIG, M. KRISHNAMOORTHY, and M. PALANISWAMI, 1996. Comparison of Heuristic Algorithms for the Degree Constrained Minimum Spanning Tree, in *Meta-Heuristics: Theory and Applications*, I. H. Osman and J. P. Kelly (eds.), Kluwer Academic Press, Boston, 83–96.
38. I. F. CROALL and J. P. MASON, 1992. *Industrial Applications of Neural Networks*, Springer-Verlag, Luxembourg, 160–218.
39. J. C. CULIOLI, V. PROTOPOPESCU, C. L. BRITTON, and M. N. ERICSON, 1990. Neural Network Models for Linear Programming, *Proceedings of International Joint Conference on Neural Networks*, 293–296.
40. C. DAGLI and S. LAMMERS, 1989. Possible Applications of Neural Networks in Manufacturing, *Proceedings of the IEEE International Joint Conference of Neural Networks 2*, 605 (abstract only).
41. C. H. DAGLI and S. SITTISATHANCHAI, 1993. Genetic Neuro-Scheduler for Job-Shop Scheduling, *Computers and Industrial Engineering 25*, 267–270.
42. Z. DAI, J. CHA, W. GUO, and F. WANG, 1994. A Heuristic-Based Neural Network for Packing Problems, *Proceedings International Conference on Data and Knowledge Systems for Manufacturing and Engineering 2*, 698–703.
43. G. B. DANTZIG, 1963. *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ.
44. H. DATE, M. SEKI, and T. HAYASHI, 1990. LSI Module Placement Using Neural Computation Networks, *Proceedings International Joint Conference on Neural Networks 3*, San Diego, 831–836.
45. G. W. DAVIS, 1989. Sensitivity Analysis in Neural Net Solutions, *IEEE Transactions on Systems, Man and Cybernetics 19*, 1078–1082.
46. L. DAVIS (ed.), 1991. *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York.
47. D. DESIENO, 1988. Adding a Conscience Mechanism to Competitive Learning, *Proceedings IEEE International Conference on Neural Networks 1*, 117–124.
48. E. W. DIJKSTRA, 1959. A Note on Two Problems in Connection with Graphs, *Numerische Mathematik 1*, 269.
49. R. DURBIN and D. WILLSHAW, 1987. An Analogue Approach to the Travelling Salesman Problem Using an Elastic Net Method, *Nature 326*, 689–691.
50. L. FANG and T. LI, 1990. Design of Competition-Based Neural Networks for Combinatorial Optimization, *International Journal of Neural Systems 1*, 221–235.
51. L. FANG, W. H. WILSON, and T. LI, 1990. Mean-Field Annealing Neural Net for Quadratic Assignment, *Proceedings International Conference on Neural Networks*, Paris, 282–286.
52. F. FAVATA and R. WALKER, 1991. A Study of the Application of Kohonen-Type Neural Networks to the Travelling Salesman Problem, *Biological Cybernetics 64*, 463–468.
53. Y. P. S. FOO and Y. TAKEFUJI, 1988. Stochastic Neural Networks for Job-Shop Scheduling: Parts 1 and 2, *Proceedings of the IEEE International Conference on Neural Networks 2*, 275–290.
54. Y. P. S. FOO and Y. TAKEFUJI, 1988. Integer Linear Programming Neural Networks for Job Shop Scheduling, *Proceedings of the IEEE International Conference on Neural Networks 2*, 341–348.
55. Y. P. S. FOO and H. SZU, 1989. Solving Large-Scale Optimization Problems by Divide-and-Conquer Neural Networks, *Proceedings IEEE International Joint Conference on Neural Networks 1*, 507–511.
56. 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*, 33–40.
57. T. FUJITA, K. YASUDA, and R. YOKOYAMA, 1995. Global Optimization Method Using Chaos in Dissipative System, *Electronics and Communications in Japan, Part 3, 78*, 881–889.
58. N. FUNABIKI and Y. TAKEFUJI, 1992. A Neural Network Parallel Algorithm for Channel Assignment Problems in Cellular Radio Networks, *IEEE Transactions on Vehicular Technology 41*, 430–437.
59. C. GASPIN, 1990. Automatic Translation of Constraints for Solving Optimization Problems by Neural Networks, *Proceedings International Joint Conference on Neural Networks*, 857–861.
61. M. R. GAREY and D. S. JOHNSON, 1979. *Computers and Intractability*, W. H. Freeman, New York.
62. D. W. GASSEN and J. D. CAROTHERS, 1993. Graph Color Minimization Using Neural Networks, *Proceedings International Joint Conference on Neural Networks 2*, Nagoya, 1541–1544.
63. A. H. GEE, 1993. Problem Solving with Optimization Networks, Ph.D. thesis, Queen's College, Cambridge University, Cambridge, U. K.
64. A. H. GEE and R. W. PRAGER, 1995. Limitations of Neural Networks for Solving Traveling Salesman Problems, *IEEE Transactions on Neural Networks 6*, 280–282.
65. H. GHAZIRI, 1996. Supervision in the Self-Organizing Feature Map: Application to the Vehicle Routing Problem, in *Metaheu-*

ristics: Theory and Applications, I. H. Osman and J. P. Kelly (eds.), Kluwer, Boston, 651–660.
66. L. GISLEN, C. PETERSON and B. SODERBERG, 1989. Teachers and Classes with Neural Networks, *International Journal of Neural Systems 1*, 167–176.
67. F. GLOVER, 1986. Future Paths for Integer Programming and Links to Artificial Intelligence, *Computers and Operations Research 5*, 533–549.
68. F. GLOVER, 1993. A User's Guide to Tabu Search, *Annals of Operations Research 41*, 3–28.
69. D. E. GOLDBERG, 1989. *Genetic Algorithms in Search Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.
70. M. GOLDSTEIN, 1990. Self-Organizing Feature Maps for the Multiple Traveling Salesman Problem (MTSP), *Proceedings IEEE International Conference on Neural Networks, Paris, 258–261.*
71. D. GONG, M. GEN, G. YAMAZAKI, and W. XU, 1995. Neural Network Approach for General Assignment Problem, *Proceedings International Conference on Neural Networks 4*, Perth, 1861–1866.
72. R. E. GOMORY, 1958. Outline for an Algorithm for Integer Solution to Linear Programs, *Bulletin of the American Mathematical Society 64*, 5.
73. S. GROSSBERG, 1988. Nonlinear Neural Networks: Principles, Mechanisms and Architectures, *Neural Networks 1*, 17–61.
74. F. GUERRERO, S. LOZANO, D. CANCA, and K. SMITH, 1998. Machine Grouping in Cellular Manufacturing: A Self-Organising Neural Network, in *Engineering Benefits from Neural Networks*, A. B. Bulsari et al. (eds.), Systems Engineering Association, Turku, Finland, 374–377.
75. T. HAMEENANTTILA and J. D. CAROTHERS, 1994. A Hopfield Neural Network Solution to the TCM Partitioning Problem, *Proceedings IEEE International Conference on Neural Networks 7*, 4676–4680.
76. D. HAMMERSTROM, 1990. A VLSI Architecture for High Performance, Low Cost, On-Chip Learning, *Proceedings IEEE International Joint Conference on Neural Networks 2*, 537–543.
77. Y. HAYAKAWA, A. MARUMOTOT, and Y. SAWADA, 1995. Effects of the Chaotic Noise on the Performance of a Neural Network Model for Optimization Problems, *Physical Review E 51*, 2693–2700.
78. R. HECHT-NIELSEN, 1988. Applications of Counterpropagation Networks, *Neural Networks 1*, 131–139.
79. S. HEDGE, J. SWEET, and W. LEVY, 1988. Determination of Parameters in a Hopfield/Tank Computational Network, *Proceedings IEEE International Conference on Neural Networks 2*, 291–298.
80. B. J. HELLSTROM and L. N. KANAL, 1992. Asymmetric Mean-Field Neural Networks for Multi-Processor Scheduling, *Neural Networks 5*, 671–686.
81. L. HÉRAULT and J. J. NIEZ, 1991. Neural Network and Combinatorial Optimization. A Study of NP-Complete Graph Problems, in *Neural Networks: Advances and Applications*, E. Gelenbe (ed.), North-Holland, Amsterdam, 165–213.
82. G. E. HINTON, T. J. SEJNOWSKI, and D. H. ACKLEY, 1984. Boltzmann Machines: Constraint Satisfaction Networks that Learn, Carnegie Mellon University Technical Report CMU-CS-84-119.
83. S. G. HONG, S. W. KIM, and J. J. LEE, 1995. The Minimum Cost Path Finding Algorithm Using a Hopfield Type Neural Network, *Proceedings IEEE International Conference on Fuzzy Systems 4*, 1719–1726.
84. J. N. HOOKER, 1995. Testing Heuristics: We Have it All Wrong, *Journal of Heuristics 1*, 33–42.
85. J. J. HOPFIELD, 1982. Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proceedings National Academy of Sciences 79*, 2554–2558.
86. J. J. HOPFIELD, 1984. Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons, *Proceedings National Academy of Sciences 81*, 3088–3092.
87. J. J. HOPFIELD and D. W. TANK, 1985. "Neural" Computation of Decisions in Optimization Problems, *Biological Cybernetics 52*, 141–152.
88. M. INOUE and A. NAGAYOSHI, 1992. Solving an Optimization Problem with a Chaos Neural Network, *Proceedings Theoretical Physics 88*, 769–773.
89. A. JAGOTA, 1995. Approximating Maximum Clique with a Hopfield Network, *IEEE Transactions on Neural Networks 6*, 724–735.
90. H. JEONG and J. H. PARK, 1989. Lower Bounds of Annealing Schedule for Boltzmann and Cauchy Machines, *Proceedings IEEE International Joint Conference on Neural Networks 1*, 581–586.
91. A. K. JAIN and R. C. DUBES, 1988. *Algorithms for Clustering*, Prentice Hall, Englewood, NJ.
92. A. JOPPE, H. R. A. CARDON, and J. C. BIOCH, 1990. A Neural Network for Solving the Traveling Salesman Problem, *Proceedings IEEE International Joint Conference on Neural Networks 3*, 961–964.
93. M. KAJIURA, Y. AKIYAMA, and Y. ANZAI, 1990. Solving Large Scale Puzzles with Neural Networks, *Proceedings Tools for AI Conference*, Fairfax, 562–569.
94. B. KAMGAR-PARSI, J. A. GUALTIERI, J. E. DEVANEY, and B. KAMGAR-PARSI, 1990. Clustering with Neural Networks, *Biological Cybernetics 63*, 201–208.
95. B. KAMGAR-PARSI and B. KAMGAR-PARSI, 1987. An Efficient Model of Neural Networks for Optimization, *Proceedings IEEE International Conference on Neural Networks 3*, 785–790.
96. B. KAMGAR-PARSI and B. KAMGAR-PARSI, 1992. Dynamical Stability and Parameter Selection in Neural Optimization, *Proceedings International Joint Conference on Neural Networks 4*, 566–571.
97. M. KENNEDY and L. CHUA, 1988. Neural Networks for Linear and Nonlinear Programming, *IEEE Transactions on Circuits and Systems 35*, 554–562.
98. B. KERNIGHAN and S. LIN, 1970. An Efficient Heuristic Procedure for Partitioning Graphs, *Bell System Technical Journal 49*, 291–307.
99. S. Y. KIM, Y. H. LEE, and D. AGNIHOTRI, 1995. A Hybrid Approach to Sequencing Jobs Using Heuristic Rules and Neural Networks, *Production Planning and Control 6*, 445–454.
100. S. KIRKPATRICK, C. GELATT, and M. VECCHI, 1983. Optimisation by Simulated Annealing, *Science 220*, 671–680.
101. T. KOHONEN, 1982. Self-Organized Formation of Topologically Correct Feature Maps, *Biological Cybernetics 43*, 59–69.
102. T. KOHONEN, 1984. *Self-Organisation and Associative Memory*, Springer-Verlag, Berlin.
103. T. KOHONEN, 1990. The Self-Organizing Map, *Proceedings of the IEEE 78*, 1464–1480.
104. M. KOVACIC, 1993. Timetable Construction with Markovian Neural Network, *European Journal of Operational Research 69*, 92–96.
105. J. KRUSKAL, 1956. On the Shortest Spanning Subtree of a Graph and the Travelling Salesman Problem, *Proceedings American Mathematical Society 7*, 48.
106. D. KUNZ, 1991. Channel Assignment for Cellular Radio Using Neural Networks, *IEEE Transactions on Vehicular Technology 40*, 188–193.
107. T. KUROKAWA and S. KOZUKA, 1994. Use of Neural Networks

for the Optimum Frequency Assignment Problem, *Electronics and Communications in Japan, Part 1, 77*, 106–116.
108. J. S. LAI, S. Y. KUO, and I. Y. CHEN, 1994. Neural Networks for Optimization Problems in Graph Theory, *Proceedings IEEE International Symposium on Circuits and Systems 6*, 269–272.
109. W. K. LAI and G. G. COGHILL, 1992. Genetic Breeding of Control Parameters for the Hopfield/Tank Neural Net, *Proceedings International Joint Conference on Neural Networks 4*, 618–623.
110. W. K. LAI and G. G. COGHILL, 1994. Initialising the Continuous Hopfield Net, *Proceedings International Conference on Neural Networks 7*, 4640–4644.
111. A. H. LAND and A. G. DOIG, 1960. An Automatic Method of Solving Discrete Programming Problems, *Econometrica 28*, 497–520.
112. B. W. LEE and B. J. SHEU, 1989. Design of a Neural-Based A/D Converter Using Modified Hopfield Network, *IEEE Journal of Solid-State Circuits 24*, 1129–1135.
113. H.-M. LEE and C.-C. HSU, 1990. Neural Network Processing Through Energy Minimization with Learning Ability to the Multiconstraint Zero-One Knapsack Problem, *Proceedings Tools for AI Conference*, Fairfax, Virginia, 548–555.
114. S. LEE and J. PARK, 1993. Dual-Mode Dynamics Neural Network (D2NN) for Knapsack Packing Problem, *Proceedings International Joint Conference on Neural Networks 3*, Nagoya, 2425–2428.
115. S. LIN and B. W. KERNIGHAN, 1973. An Effective Heuristic Algorithm for the Travelling Salesman Problem, *Operations Research 21*, 498–516.
116. W. LIN, J. G. DELGADO-FRIAS, G. G. PECHANEK, and S. VASSILIADIS, 1994. Impact of Energy Function on a Neural Network Model for Optimization Problems, *Proceedings IEEE International Conference on Neural Networks 7*, 4518–4523.
117. Y. LIN, L. M. AUSTIN, and J. R. BURNS, 1992. An Intelligent Algorithm for Mixed-Integer Programming Models, *Computers and Operations Research 19*, 461–468.
118. R. LISTER, 1993. Annealing Networks and Fractal Landscapes, *Proceedings IEEE International Conference on Neural Networks 1*, 257–262.
119. J. T-H. LO, 1992. A New Approach to Global Optimization and its Applications to Neural Networks, *Proceedings IEEE International Joint Conference on Neural Networks 4*, 600–605.
120. C.-K. LOOI, 1992. Neural Network Methods in Combinatorial Optimization, *Computers and Operations Research 19*, 191–208.
121. N. MANSOUR, 1994. Parallel Physical Optimization Algorithms for Allocating Data to Multicomputer Nodes, *Journal of Supercomputing 8*, 53–80.
122. Y. MATSUYAMA, 1991. Self-Organization via Competition, Cooperation and Categorization Applied to Extended Vehicle Routing Problems, *Proceedings International Joint Conference on Neural Networks 1*, 385–390.
123. H. MAUSSER, M. J. MAGAZINE, and J. B. MOORE, 1993. Application of an Annealed Neural Network to a Timetabling Problem, Working Paper, School of Business Administration, University of Colorado at Boulder.
124. A. MEHREZ, Y. YUAN, and A. GAFNI, 1988. Stable Solution vs. Multiplicative Utility Solutions for the Assignment Problem, *Operations Research Letters 7*, 131–139.
125. G. L. NEMHAUSER and L. A. WOLSEY, 1988. *Integer and Combinatorial Optimization*, John Wiley & Sons, New York.
126. H. NONAKA and Y. KOBAYASHI, 1992. Sub-Optimal Solution Screening in Optimization by Neural Networks, *Proceedings International Joint Conference on Neural Networks 4*, 606–611.
127. H. NOZAWA, 1994. Solution of the Optimization Problem Using the Neural Network Model as a Globally Coupled Map, in *Towards the Harnessing of Chaos*, M. Yamaguti (ed.), Elsevier Science B. V., Amsterdam, 99–114.
128. K. E. NYGARD, P. JUELI, and N. KADABA, 1990. Neural Networks for Selecting Vehicle Routing Heuristics, *ORSA Journal of Computing 2*, 353–364.
129. M. OHLSSON, C. PETERSON, and B. SODERBERG, 1993. Neural Networks for Optimization Problems with Inequality Constraints: The Knapsack Problem, *Neural Computation 5*, 331–339.
130. I. H. OSMAN and G. LAPORTE, 1996. Metaheuristics: A Bibliography, *Annals of Operations Research 63*, 513–623.
131. R. PADMAN, 1991. Choosing Solvers in Decision Support Systems. A Neural Network Application in Resource-Constrained Project Scheduling, in *Recent Developments in Decision Support Systems 101*, Springer-Verlag, Berlin, 539–574.
132. C. PETERSON and J. ANDERSON, 1988. Neural Networks and NP-Complete Optimization Problems: A Performance Study on the Graph Bisection Problem, *Complex Systems 2*, 59–89.
133. C. PETERSON and B. SODERBERG, 1989. A New Method for Mapping Optimization Problems onto Neural Networks, *International Journal of Neural Systems 1*, 3–22.
134. C. PETERSON and B. SODERBERG, 1993. Artificial Neural Networks, in *Modern Heuristic Techniques for Combinatorial Problems*, C. R. Reeves (ed.), Blackwell Scientific Publications, Oxford, 197–242.
135. P. R. PHILIPOOM, L. P. REES, and L. WIEGMANN, 1994. Using Neural Networks to Determine Internally Set Due-Date Assignments for Shop Scheduling, *Decision Sciences 26*, 825–851.
136. L. K. PLATZMAN and J. J. BARTHOLDI, 1989. Spacefilling Curves and the Planar Travelling Salesman Problem, *Journal of the ACM 36*, 719–737.
137. M. O. POLIAC, E. B. LEE, J. R. SLAGLE, and M. R. WICK, 1987. A Crew Scheduling Problem, *Proceedings IEEE International Conference on Neural Networks 2*, 779–786.
138. C. PORNAVALAI, G. CHAKRABORTY, and N. SHIRATORI, 1995. Neural Networks for Solving Constrained Steiner Tree Problem, *Proceedings IEEE International Conference on Neural Networks 4*, 1867–1870.
139. J.-Y. POTVIN, 1993. The Traveling Salesman Problem: A Neural Network Perspective, *ORSA Journal on Computing 5*, 328–348.
140. J.-Y. POTVIN and C. ROBILLARD, 1995. Clustering for Vehicle Routing with a Competitive Neural Network, *Neurocomputing 8*, 125–139.
141. P. W. PROTZEL, D. L. PALUMBO, and M. K. ARRAS, 1993. Performance and Fault Tolerance of Neural Networks for Optimization, *IEEE Transactions on Neural Networks 4*, 600–614.
142. J. RAMANUJAM and P. SADAYAPPAN, 1995. Mapping Combinatorial Optimization Problems onto Neural Networks, *Information Sciences 82*, 239–255.
143. H. E. RAUCH and T. WINARSKE, 1988. Neural Networks for Routing Communications Traffic, *IEEE Control Systems Magazine*, April, 26–31.
144. C. P. RAVIKUMAR and N. VEDI, 1995. Heuristic and Neural Algorithms for Mapping Tasks to a Reconfigurable Array, *Microprocessing and Microprogramming 41*, 137–151.
145. C. R. REEVES (ed.), 1993. *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publications, Oxford.
146. H. RITTER and K. SCHULTEN, 1988. Kohonen's Self-Organizing Maps: Exploring their Computational Capabilities, *Proceedings IEEE International Conference on Neural Networks 1*, 109–116.
147. R. ROMANO, O. MAIMON, and M. FURST, 1989. Neural Net Implementation for Assigning a Product to a Production Line, *Proceedings of the IEEE International Joint Conference on Neural Networks 2*, 577 (abstract only).

148. B. ROYSAM and A. K. BHATTACHARJYA, 1992. Hierarchically Structured Unit-Simplex Transformations for Parallel Distributed Optimization Problems, *IEEE Transactions on Neural Networks 3*, 108–124.
149. S. SATYANARAYANA, Y. P. TSIVIDIS, and H. P. GRAF, 1992. A Reconfigurable VLSI Neural Network, *IEEE Journal of Solid-State Circuits 27*, 67–81.
150. J. D. SCHAFFER, D. WHITLEY, and L. J. ESHELMAN, 1992. Combinations of Genetic Algorithms and Neural Networks. A Survey of the State of the Art, *Proceedings of the International Workshops on Combinations of Genetic Algorithms and Neural Networks*, 1–37.
151. H. N. SCHALLER, 1993. Problem Solving by Global Optimization: The Rolling Stone Neural Network, *Proceedings International Joint Conference on Neural Networks 2*, Nagoya, 1481–1484.
152. R. SHARDA, 1994. Neural Networks for the MS/OR Analyst: An Application Bibliography, *Interfaces 24*, 116–130.
153. B. SHEU, E. CHOU, R. TSAI, and D. CHEN, 1995. VLSI Neural Networks: Design Challenges and Opportunities, in *Computational Intelligence*, M. Palaniswami, Y. Attikiouzel, R. J. Marks II, D. Fogel, and T. Fukada (eds.), IEEE Press, New York, 261–271.
154. K. SMITH, 1995. Solving the Generalized Quadratic Assignment Problem using a Self-Organising Process, *Proceedings IEEE International Conference on Neural Networks 4*, Perth, 1876–1879.
155. K. SMITH, M. PALANISWAMI, and M. KRISHNAMOORTHY, 1996. A Hybrid Neural Approach to Combinatorial Optimization, *Computers and Operations Research 23*, 597–610.
156. K. SMITH, 1996. An Argument for Abandoning the Traveling Salesman Problem as a Neural Network Benchmark, *IEEE Transactions of Neural Networks 7*, 1542–1544.
157. K. SMITH, M. PALANISWAMI, and M. KRISHNAMOORTHY, 1996. Traditional Heuristic versus Hopfield Neural Network Approaches to a Car Sequencing Problem, *European Journal of Operational Research 93*, 300–316.
158. K. SMITH and M. PALANISWAMI, 1997. Static and Dynamic Channel Assignment Using Neural Networks, *IEEE Journal on Selected Areas in Communications 15*, 238–249.
159. K. SMITH, M. KRISHNAMOORTHY, and M. PALANISWAMI, 1996. Neural versus Traditional Approaches to the Location of Interacting Hub Facilities, *Location Science 4*, 155–171.
160. H. SZU, 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.
161. H. SZU and R. HARTLEY, 1987. Fast Simulated Annealing, *Physics Letters A 122*, 157–162.
162. G. A. TAGLIARINI and E. W. PAGE, 1987. Solving Constraint Satisfaction Problems with Neural Networks, *Proceedings IEEE International Conference on Neural Networks 3*, 741–747.
163. T. TAKADA, K. SANOU, and S. FUKUMARA, 1995. A Neural-Network Systems for Solving an Assortment Problem in the Steel-Industry, *Annals of Operations Research 57*, 265–281.
164. M. TAKAHASHI, K. KYUMA, and E. FUNADA, 1993. 10000 Cell Placement Optimization using a Self-Organizing Map, *Proceedings International Joint Conference on Neural Networks 3*, 2417–2420.
165. Y. TAKEFUJI, 1992. *Neural Network Parallel Computing*, Kluwer Academic Publishers, Boston, MA.
166. Y. TAKEFUJI and K. C. LEE, 1991. Artificial Neural Networks for Four-Coloring Map Problems and K-Colorability Problems, *IEEE Transactions on Circuits and Systems 38*, 326–333.
167. Y. TAKEFUJI and H. SZU, 1989. Design of Parallel Distributed Cauchy Machines, *Proceedings IEEE International Joint Conference on Neural Networks 1*, 529–532.
168. D. W. TANK and J. J. HOPFIELD, 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.
169. D. A. THOMAE and D. VAN DEN BOUT, 1990. Encoding Logical Constraints into Neural Network Cost Functions, *Proceedings International Joint Conference on Neural Networks 3*, San Diego, 863–868.
170. Y. UESAKA, 1993. Mathematical Basis of Neural Networks for Combinatorial Optimization Problems, *Optoelectronics 8*, 1–9.
171. M. K. UNALTUNA and V. PITCHUMANI, 1994. Unsupervised Competitive Learning Neural Network Algorithms for Circuit Bipartitioning, *Proceedings World Congress on Neural Networks 1*, San Diego, 302–307.
172. K. URAHAMA and H. NISHIYUKI, 1993. Neural Algorithms for Placement Problems, *Proceedings International Joint Conference on Neural Networks 3*, Nagoya, 2421–2424.
173. A. I. VAKHUTINSKY and B. L. GOLDEN, 1994. Solving Vehicle Routing Problems Using Elastic Nets, *Proceedings IEEE International Conference on Neural Networks 7*, 4535–4540.
174. A. I. VAKHUTINSKY and B. L. GOLDEN, 1995. A Hierarchical Strategy for Solving Traveling Salesman Problems Using Elastic Nets, *Journal of Heuristics 1*, 67–76.
175. S. VAITHAYANATHAN and J. IGNIZIO, 1992. A Stochastic Neural Network for Resource Constrained Scheduling, *Computers and Operations Research 19*, 241–254.
176. S. VAITHYANATHAN, H. OGMEN, and J. IGNIZIO, 1994. Generalized Boltzmann Machines for Multidimensional Knapsack Problems, *Intelligent Engineering Systems Through Artificial Neural Networks 4*, ASME Press, New York, 1079–1084.
177. D. E. VAN DEN BOUT and T. K. MILLER III, 1988. A Travelling Salesman Objective Function That Works, *Proceedings IEEE International Conference on Neural Networks 2*, 299–303.
178. D. E. VAN DEN BOUT and T. K. MILLER III, 1989. Improving the Performance of the Hopfield-Tank Neural Network through Normalization and Annealing, *Biological Cybernetics 62*, 129–139.
179. D. E. VAN DEN BOUT and T. K. MILLER III, 1990. Graph Partitioning Using Annealed Neural Networks, *IEEE Transactions on Neural Networks 1*, 192–203.
180. P. VAN HENTENRYCK, 1989. *Constraint Satisfaction in Logic Programming*, MIT Press, Cambridge, MA.
181. R. VAN VLIET and H. CARDON, 1991. Combining a Graph Partitioning and a TSP Neural Network to Solve the MTSP, in *Artificial Neural Networks 2*, T. Kohonen, K. Makisara, O. Simula, and J. Kangas (eds.), North Holland, Amsterdam, 157–162.
182. M. VERLEYSEN and P. JESPERS, 1989. An Analog VLSI Implementation of Hopfield's Neural Network, *IEEE Micro*, December, 46–55.
183. M. VIDYASAGAR, 1993. Location and Stability of the High-Gain Equilibria of Nonlinear Neural Networks, *IEEE Transactions on Neural Networks 4*, 660–672.
184. V. V. VINOD, S. GHOSE, and P. P. CHAKRABARTI, 1996. Resultant Projection Neural Networks for Optimization Under Inequality Constraints, *IEEE Transactions on Systems, Man, and Cybernetics Part B 26*, 509–521.
185. E. WACHOLDER, 1990. A Neural Network-Based Optimization Algorithm for the Static Weapon-Target Assignment Problem, *ORSA Journal on Computing 1*, 232–246.
186. E. WACHOLDER, J. HAN, and R. C. MANN, 1991. An Extension of the Hopfield-Tank Model for Solution of the Multiple TSP,

Proceedings IEEE International Conference on Neural Networks 2, 305–325.
187. E. WACHOLDER, J. HAN, and R. C. MANN, 1989. A Neural Network Algorithm for the Multiple TSP, *Biological Cybernetics 61*, 11–19.
188. J. WANG, 1992. Analogue Neural Networks for Solving the Assignment Problem, *Electronics Letters 28*, 1047–1050.
189. J. WANG, 1994. A Recurrent Neural Network for Solving the Shortest Path Problem, *Proceedings IEEE International Symposium Circuits and Systems 6*, 319–322.
190. Q. WANG, X. SUN, B. GOLDEN, and J. JIA, 1995. Using Artificial Neural Networks to Solve the Orienteering Problem, *Annals of Operations Research 61*, 111–120.
191. Y. C. WEI and C. K. CHENG, 1991. Ratio Cut Partitioning for Hierarchical Designs, *IEEE Transactions on CAD*, July, 911–921.
192. G. V. WILSON and G. S. PAWLEY, 1988. On the Stability of the TSP Algorithm of Hopfield and Tank, *Biological Cybernetics 58*, 63–70.
193. W. S. WONG and C. A. FUNKA-LEA, 1990. An Elastic Net Solution to Obstacle Avoidance Tour Planning, *Proceedings International Joint Conference on Neural Networks 3*, San Diego, 799–804.
194. X. XU and W. T. TSAI, 1991. Effective Neural Algorithms for the Travelling Salesman Problem, *Neural Networks 4*, 193–205.
195. S. YAMADA and T. KASAI, 1990. An Efficient Algorithm for the Linear Assignment Problem, *Electronics and Communications in Japan, Part 3, 73*, 28–36.
196. A. YAMAMOTO, M. OHTA, H. UEDA, A. OGIHARA, and K. FUKUNAGA, 1995. Asymmetric Neural Network and its Application to Knapsack Problem, *IEICE Transactions Fundamentals E78-A*, 300–305.
197. X. YAO, 1993. Evolutionary Artificial Neural Networks, *International Journal of Neural Systems 4*, 203–222.
198. P. YIP and Y. TAKEFUJI, 1994. Constrained Optimization with Use of Two-Dimensional Maximum Neurons, *Proceedings International Conference on Neural Networks 7*, 4667–4671.
199. L. ZHANG and S. C. A. THOMOPOULOS, 1989. Neural Network Implementation of Shortest Path Algorithm for Traffic Routing in Communication Networks, *Proceedings International Joint Conference on Neural Networks 2*, Washington DC, 591 (abstract only).
200. S. ZHAO and T. S. DILLON, 1993. Parallel Distributed Implementation of the Shortest Path Algorithm, *Proceedings International Joint Conference on Neural Networks 2*, Nagoya, 1598–1601.
201. D. N. ZHOU, V. CHERKASSKY, T. R. BALDWIN, and D. W. HONG, 1990. Scaling Neural Networks for Job-Shop Scheduling, *Proceedings International Joint Conference on Neural Networks 3*, San Diego, 889–894.
202. V. ZISSIMOPOLOUS, V. PASCHOS, and F. PEKERGIN, 1991. On the Approximation of NP-Complete Problems by Using the Boltzmann Machine Method: The Case of Some Covering and Packing Problems, *IEEE Transactions on Computers 40*, 1413–1418.