Neural Techniques for Combinatorial Optimization with Applications

Kate Smith, *Member, IEEE,* Marimuthu Palaniswami, *Senior Member, IEEE,* and Mohan Krishnamoorthy

Abstract— **After more than a decade of research, there now exist several neural-network techniques for solving NP-hard combinatorial optimization problems. Hopfield networks and self-organizing maps are the two main categories into which most of the approaches can be divided. Criticism of these approaches includes the tendency of the Hopfield network to produce infeasible solutions, and the lack of generalizability of the self-organizing approaches (being only applicable to Euclidean problems). This paper proposes two new techniques which have overcome these pitfalls: a Hopfield network which enables feasibility of the solutions to be ensured and improved solution quality through escape from local minima, and a self-organizing neural network which generalizes to solve a broad class of combinatorial optimization problems. Two sample practical optimization problems from Australian industry are then used to test the performances of the neural techniques against more traditional heuristic solutions.**

Index Terms—**Assembly line, combinatorial optimization, Hopfield networks, hub location, NP-hard, self-organization, sequencing, traveling salesman problem.**

I. Introduction

THE idea of using neural networks to provide solutions to difficult NP-complete optimization problems has been pursued for over a decade. Hopfield and Tank's seminal paper [18] in 1985 demonstrated that the traveling salesman problem (TSP) could be solved using a Hopfield neural network. Yet the technique, which requires minimization of an energy function containing several terms and parameters, was shown to often yield infeasible solutions to the TSP [38]. For the remainder of the decade, researchers tried to either modify the energy function [3], [37] or optimally tune the numerous parameters involved [19], [23] so that the network would converge to a feasible TSP solution. Subsequent efforts to confine the Hopfield network to the feasible constraint plane have resulted in a method which can now ensure the final solution is feasible [6], [13].

Despite this success, however, the reputation of the Hopfield network for solving combinatorial optimization problems does not appear to have been resurrected. Recent results have shown that, unless the TSP is Euclidean, the quality of the solutions found using a Hopfield network is unlikely to be comparable to those obtained using traditional techniques [14]. So while the feasibility issue of Hopfield networks has been essentially eliminated, the question of solution quality still raises some doubts as to the suitability of the technique.

Of concern here is the possibility that Hopfield networks are not being used to solve practical optimization problems which have arisen from industrial situations, simply because the literature appears to be focused on the deficiencies of the technique for solving the TSP. In recent work [33] we have argued that the TSP may not be an appropriate benchmark problem anyway, due to the existence of an alternative linear formulation which makes comparisons unfair and biases the findings against neural and other techniques using a nonlinear formulation. We do not advocate the application of a technique which is known to yield inferior solutions. We are, however, observing that the performance of neural networks for solving practical optimization problems has been relatively untested. For many practical NP-complete problems, heuristic approaches are employed due to the need for rapid solutions. Obtaining the globally optimal solution is not as imperative as arriving at a near-optimal solution quickly. Certainly, one of the principal advantages of neural techniques is the rapid computation power and speed which can be obtained through hardware implementation, and this consideration is even more valuable in industrial situations. The relative scarcity of literature comparing the performances of neural techniques to more traditional methods for practical optimization problems suggests that this advantage is not being realized.

A similar focus on the TSP is found in the literature relating to the use of self-organizing approaches to optimization [2], [10], [12]. 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* [8]. Kohonen's principles of self-organization [21] are combined with the concept of an "elastic band" containing a circular ring of neurons which 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. Any self-organizing approach which uses the elastic net method as its basis will thus be greatly limited in its generalizability.

Recently, we have proposed a new self-organizing approach to combinatorial optimization which generalizes to solve a broad class of "0–1" optimization problems [32]. This self-organizing neural network (SONN) is combinatorial in nature, operating within feasible permutation matrices rather than

Manuscript received January 6, 1996; revised June 10, 1998.

K. Smith is with the School of Business Systems, Monash University, Clayton, Victoria 3168, Australia.

M. Palaniswami is with the Department of Electrical and Electronic Engineering, University of Melbourne, Parkville, Victoria 3052, Australia.

M. Krishnamoorthy is with the CSIRO, Division of Mathematics and Statistics, Clayton, Victoria 3168, Australia.

Publisher Item Identifier S 1045-9227(98)07352-4.

within the Euclidean plane. It is ideally suited to 0–1 sequencing, assignment, and transportation problems, and is thus applicable to a wide range of practical optimization problems. New theoretical results are provided in this paper to demonstrate the convergence properties of our SONN.

In this paper, we are principally concerned, however, with providing an evaluation of the comparative performances of an improved Hopfield network and the SONN against traditional techniques for practical optimization problems. If neural techniques are to be employed by industry to solve practical optimization problems, (where their rapid computational power can best be utilized), we must be able to demonstrate their suitability as a technique which finds near-optimal solutions of practical problems, rather than just the TSP. In Section III, we describe the Hopfield energy function representation which can ensure a feasible solution [13], as well as a method of escaping local minima of the energy function in order to improve solution quality. Details of the SONN approach are provided in Section IV along with new convergence results. The first of the practical applications is considered in Section V. Here, the car sequencing problem (CSP), which involves the optimal sequencing of different car models along an assembly line, is described. Results comparing the performance of the improved Hopfield network, the SONN, simulated annealing, and an exact solution are presented and discussed. A second practical application is considered in Section VI. Here a postal delivery network is described, in which a set of postal districts need to be allocated to mail sorting centers, and the location of the sorting centers needs to be determined in order that the total freight costs of the network are minimized. Comparative results are again presented and discussed. The two practical applications from Australian industry have been chosen as sample NP-hard practical problems. In previous work [34], [35], we have solved other applications with similar results. Conclusions as to the suitability of neural techniques for solving practical optimization problems are drawn in Section VII.

II. A Class of Problems

Consider a 0–1 combinatorial optimization problem with the general form (COP1)

$$\text{minimize } F(\mathbf{X}) = \sum_{k=1}^{N}\sum_{j=1}^{M} X_{k,j} \sum_{i=1}^{N}\sum_{l=1}^{N} \mathcal{Q}(k,j,i,l) X_{i,l} + \sum_{k=1}^{N}\sum_{j=1}^{M} \mathcal{C}(k,j) X_{k,j} \quad (1)$$

$$\text{subject to } \sum_{j} X_{k,j} = 1 \qquad \forall k = 1, \cdots, N \quad (2)$$

$$\sum_{k} X_{k,j} = D_j \qquad \forall j = 1, \cdots, M \quad (3)$$

$$X_{k,j} \in \{0, 1\} \quad (4)$$

where $X_{k,j}$ is the element in the kth row and jth column of the $(N \times M)$-dimensional 0–1 matrix $\mathbf{X}$, the objective function $F(\mathbf{X})$ is a quadratic cost function of the solution matrix $\mathbf{X}$, $\mathcal{C}(k,j)$ is the linear cost associated with having $X_{k,j}$ "on," and $\mathcal{Q}(k,j,i,l)$ is the cost associated with having $X_{k,j}$ and $X_{i,l}$ "on" simultaneously. D_j is an integer representing the total demand for the jth column, so that

$$\sum_{j=1}^{M} D_j = N.$$

The linear constraints specify that there be exactly one matrix element "on" in each row (assignment constraints), and exactly D_j elements "on" in the jth column (transportation constraints). These constraints are commonly encountered in many optimization problems including assignment, sequencing, and resource allocation problems. Clearly, the TSP constraints are represented if $D_j = 1, \forall j = 1, \cdots, M$, and the problem becomes a quadratic assignment problem. As will be shown in subsequent sections of this paper, the proposed techniques generalize to solve problems with other types of linear constraints such as inequalities. For the sake of clarity, however, the techniques will be explained using the general form of (COP1).

An alternative representation of this problem can be derived by replacing the solution matrix $\mathbf{X}$ with a solution vector $\mathbf{x}$. The general form of the 0–1 combinatorial optimization problem then becomes (COP2)

$$\text{minimize } f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \mathbf{x}^T\mathbf{Q}\mathbf{x} \quad (5)$$

$$\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b} \quad (6)$$

$$\text{and } x_i \in \{0, 1\} \qquad \forall i = 1, \cdots, n \quad (7)$$

where $n = NM$ is the length of the solution vector $\mathbf{x}$ obtained by concatenating the rows of the solution matrix $\mathbf{X}$. $\mathbf{c}$, $\mathbf{Q}$, $\mathbf{A}$, and $\mathbf{b}$ are readily derived by converting the objective function and constraints of (COP1) into vector form.

In the following sections of this paper, we will propose an improved Hopfield network, a new self-organizing neural-network approach, and simulated annealing heuristics to solve particular applications which can be formulated in the general forms of (COP1) and (COP2).

III. A Hopfield Network Approach

The Hopfield network [16], [17] comprises a fully interconnected system of n neurons. Neuron i has internal state u_i and output level x_i (bounded by zero and one). The internal state u_i incorporates a bias current (or negative threshold) denoted by i_i, and the weighted sums of outputs from all other neurons. The weights, which determine the strength of the connections from neuron j to i, are given by W_{ij}. The relationship between the internal state of a neuron and its output level is determined by an activation function $g_i(u_i)$, which is bounded below by zero and above by one. Commonly, this activation function is given by

$$x_i = g_i(u_i) = \frac{1}{2}\Big(1 + \tanh\Big(\frac{u_i}{T}\Big)\Big)$$

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

Hopfield [17] showed that the system for hardware implementation is determined by the resistance-capacitance equations

$$\frac{du_i}{dt} = \sum_j W_{ij}x_j - \frac{u_i}{\tau} + i_i \tag{8}$$

$$u_i = g_i^{-1}(x_i) \tag{9}$$

where $\tau = RC$, R is the resistance, and C is the capacitance. For high-gain activation functions ($T \approx 0$), the output values approach either zero or one, and the state space of the network outputs is the set of corners of the n-dimensional hypercube $\{0, 1\}^n$. The final state of the network is, therefore, one of these corners.

Hopfield networks can be used as an approximate method for solving 0–1 optimization problems because, provided the weights are symmetric ($W_{ij} = W_{ji}$), the network converges to a minimum of the energy function

$$\mathcal{E}(\mathbf{x}) = -\tfrac{1}{2}\sum_i \sum_j W_{ij}x_i x_j - \sum_i i_i x_i. \tag{10}$$

The proof of stability of such continuous Hopfield networks relies upon the fact that $\mathcal{E}(\mathbf{x})$ is a Lyapunov function (see [17]), provided that the inverse function of $g'(u_i)$ (the first derivative of the activation function) exists.

Furthermore, if there are no self-connections ($W_{ii} = 0$ for all i and j, although this condition is often relaxed in practical situations), in the high-gain limit of the activation function these minima will be at or near a vertex of $\{0, 1\}^n$. It is noted that negative W_{ii} do not interfere with the Lyapunov descent, but may force the network to converge to an interior local minimum. In this case, annealing techniques are usually employed to drive the solution trace toward the vertices.

Hopfield and Tank [18] showed that if a combinatorial optimization problem can be expressed in terms of a quadratic energy function of the general form given by (10), a Hopfield network can be used to find locally optimal solutions of the energy function, which may translate to local minimum solutions of the optimization problem. Typically, the network energy function is made equivalent to the objective function which is to be minimized, while each of the constraints of the optimization problem are included in the energy function as penalty terms. Clearly, a constrained minimum of the optimization problem will also optimize the energy function, since the objective function term will be minimized and constraint satisfaction implies that the penalty terms will be zero. Unfortunately, a minimum of the energy function does not necessarily correspond to a constrained minimum of the objective function due to the fact that there are likely to be several terms in the energy function which contribute to many local minima. Thus, a tradeoff exists between which terms will be minimized completely, and feasibility of the network is unlikely unless the penalty parameters are chosen carefully. Furthermore, even if the network does manage to converge to a feasible solution, its quality is likely to be poor compared to other techniques, since the Hopfield network is a descent technique and converges to the first local minimum it encounters.

A. An Improved Hopfield Network Approach

Problems of infeasibility and poor solution quality can be essentially eliminated by an appropriate form of energy function and modification of the internal dynamics of the Hopfield network. By expressing all constraints of the problem in a single term, the overall number of terms and parameters in the energy function can be reduced. Consider the general energy function

$$\mathcal{E}(\mathbf{x}) = f(\mathbf{x}) + \frac{\gamma}{2}\|\mathbf{x} - (\mathbf{Proj}.\mathbf{x} + \mathbf{s})\|^2 \tag{11}$$

where

$$\mathbf{Proj} = \mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A} \tag{12}$$

$$\mathbf{s} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}. \tag{13}$$

The first term of the energy function is the objective function, while the second term measures the deviation of the vector $\mathbf{x}$ from the constraint plane given by $\mathbf{Ax} = \mathbf{b}$. 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, since the constraint term will be forced to vanish. Hence, the solution will necessarily lie on the constraint plane. Energy functions of this nature have been suggested by Aiyer [1] and Gee [13].

We now propose to modify the internal dynamics of the Hopfield network to permit temporary increases in this energy function in order to allow escape from local minima. The motivation for this is that the Hopfield network cannot be expected to compete with other hill-climbing heuristics such as simulated annealing while it remains a strict Lyapunov descent algorithm. This improved hill-climbing Hopfield network (which we refer to as HCHN) provides a mechanism for escaping local minima by varying the direction of motion of the neurons in such a way that, while descent of the energy function is always permitted, ascent of the energy function is permitted often initially and is less likely as the algorithm proceeds. Clearly, this is similar to the concept of simulated annealing. The modified differential equation is given by

$$\frac{du_i}{dt} = \alpha(t)\left(\sum_j W_{ij}x_j + i_i\right) \tag{14}$$

$$u_i = g^{-1}(x_i). \tag{15}$$

Here, the decay term $(-u_i/\tau)$ from (8) has been dropped, since it has been shown to inhibit convergence of the network [36]. Furthermore, the sigmoidal activation function has been replaced by the piecewise linear function

$$x_i = g(u_i) = \begin{cases} 0 & u_i \le 0 \\ u_i & 0 < u_i < 1 \\ 1 & u_i \ge 1 \end{cases} \tag{16}$$

so that $u_i = x_i$ within the unit hypercube. The modified differential equation means that the direction of the change in any neuron is now controlled by a new parameter $\alpha(t)$, where

$$\alpha(t) = \mathit{random}[k(t), 1] \quad \text{and} \quad k(t) = 1 - 2e^{-t/\tau}.$$

Fig. 1 shows how the value of k changes with time for $\tau = 40$.

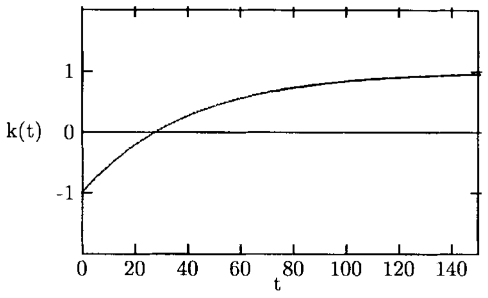


Fig. 1. Graph of $k(t) = 1 - 2e^{-t/\tau}$ with $\tau = 40$.

Now,

$$\frac{du_i}{dt} = \alpha(t)\left(\sum_j W_{ij}x_j + i_i\right) = -\alpha(t)\,\frac{\partial \mathcal{E}(\mathbf{x})}{\partial x_i} = \frac{dx_i}{dt}$$

for $\mathbf{x}$ within the unit hypercube, and

$$\frac{\partial \mathcal{E}(\mathbf{x})}{\partial x_i} = \frac{\partial f(\mathbf{x})}{\partial x_i}$$

for $\mathbf{x}$ confined to the constraint plane by a large value of γ. Thus, steepest descent and ascent of the objective function on the constraint plane are achieved when $\alpha(t) = \pm 1$, respectively. Initially, $k(0) = -1$, and so $\alpha(0)$ is randomly selected from the range $[-1, 1]$. Consequently, the energy value (which is equivalent to the objective cost provided $\mathbf{x}$ lies on the constraint plane) will often increase initially. As $t \to \infty$, however, $k(t) \to 1$, and so $\alpha(t)$ will also approach unity which is needed for strict Lyapunov descent. The length of the Markov chain (or the number of random walks permitted in multidimensional space) at each point in time is held constant at a value which depends upon the size of the problem. Thus, the modified Hopfield network HCHN allows random increases in energy initially, with such increases becoming less likely as time proceeds, until finally the network tends toward a steepest descent algorithm. Provided the value of the parameter γ is large enough, the convergence trace will be forced to lie on the constraint plane, resulting in a feasible solution.

B. Simulation Issues

It has been observed [13] that while this treatment of the energy function is very suitable and promising for an electronic circuit representation of the Hopfield network, simulation of this system on a digital computer is highly impractical. The large value of γ, which is necessary to confine the trace to the constraint plane, results in correspondingly large values for du_i/dt in (8) when u_i strays marginally from the constraint plane. Therefore, a large time-step for the discrete time simulation of (8) is bound to lead to unstable oscillations around the constraint plane. The extremely small size of the time-step which is necessary to avoid such oscillations makes this approach highly impractical to simulate on any digital machine.

The approach can be efficiently simulated, however, if we consider that the Lyapunov descent of the energy function (11) for large γ, is analogous to steepest descent of the objective function $f(\mathbf{x})$ while $\mathbf{x}$ is confined to the constraint plane. Variations of this approach utilizing a strict descent dynamic have been considered independently by Chu [6] and Gee [13].

It should also be noted here that an annealing technique may be necessary in order to drive the convergence trace to a vertex of the hypercube. This will be necessary if the matrix of weights $\mathbf{W}$ is neither positive definite nor positive semidefinite. Since annealing of the standard Hopfield network is usually created by slowly cooling the value of the activation function parameter T, we propose a further modification to the piecewise linear activation function to create an annealing effect. The activation function $g(u_i)$ in (16) is replaced by

$$x_i = g^{ann}(u_i) = \begin{cases} 0 & u_i \le \mathcal{L} \\ \dfrac{u_i - \mathcal{L}}{\mathcal{U} - \mathcal{L}} & \mathcal{L} < u_i < \mathcal{U} \\ 1 & u_i \ge \mathcal{U}. \end{cases} \tag{17}$$

Initially $\mathcal{L} = 0$ and $\mathcal{U} = 1$ so that $g^{ann} = g$. The annealing effect is created by allowing the values of $\mathcal{L}$ and $\mathcal{U}$ to slowly approach one another by iterating

$$\mathcal{L} = \mathcal{L} + \varepsilon, \qquad \mathcal{U} = \mathcal{U} - \varepsilon, \qquad \varepsilon = 10^{-5}$$

after each pass through the clipping function g^{ann}.

We now present an algorithm for the efficient simulation of the modified Hopfield network HCHN for solving problems of the form (COP2).

C. The Algorithm

Step 1) Initialize the parameters of the network as

$$\begin{aligned} \mathbf{W} &= -2\mathbf{Q} + \gamma(\mathbf{Proj} - \mathbf{I}) \\ \mathbf{i} &= \gamma\mathbf{s} - \mathbf{c} \end{aligned}$$

obtained by expanding out (11) and comparing the coefficients to the standard energy function (10), $x_i \approx 0.5$, $\Delta t = 10^{-4}$, $\varepsilon = 10^{-5}$, $\mathcal{L} = 0$, $\mathcal{U} = 1$, $\tau = 40$.

Step 2) Update $k(t) = 1 - 2e^{-t/\tau}$, and generate $\alpha(t)$ randomly from the range $[k(t), 1]$.

Step 3) Update neurons according to

$$x_i \leftarrow x_i - (\Delta t)\left(\alpha(t)\,\frac{\partial f}{dx_i}\right).$$

This will most likely take $\mathbf{x}$ off the constraint plane.

Step 4) Project $\mathbf{x}$ back onto the constraint plane, and within the unit hypercube, according to the iterative procedure shown in Fig. 2. This is the projection and clipping algorithm suggested by Gee [13].

Step 5) $\mathcal{L} \leftarrow \mathcal{L} + \varepsilon$, $\mathcal{U} \leftarrow \mathcal{U} - \varepsilon$. Repeat from Step 3) for one Markov chain length.

Step 6) Increase t and repeat from Step 2, until $k(t) = 1$ and $dx_i/dt = 0$ for all i.

Clearly, this procedure is very similar to the dynamics of the modified Hopfield network HCHN if implemented in hardware with a large value of γ. The network updates itself in a systematic way which performs simulated annealing on the energy function, while the neurons are forced to assume a feasible configuration, just as they would be for large γ in

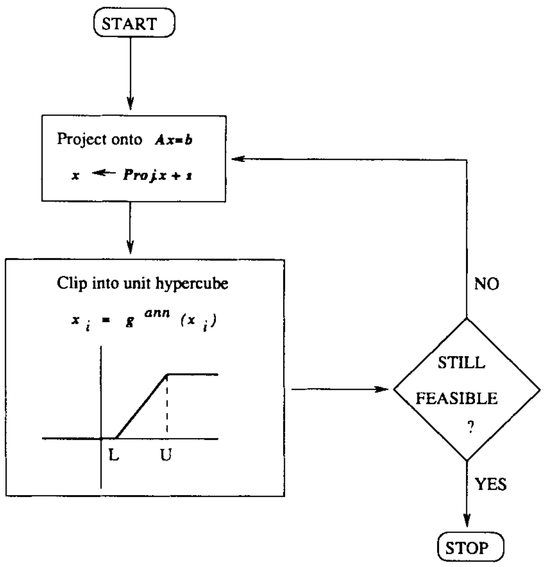


Fig. 2. Flowchart representation of Step 4) of the HCHN algorithm.

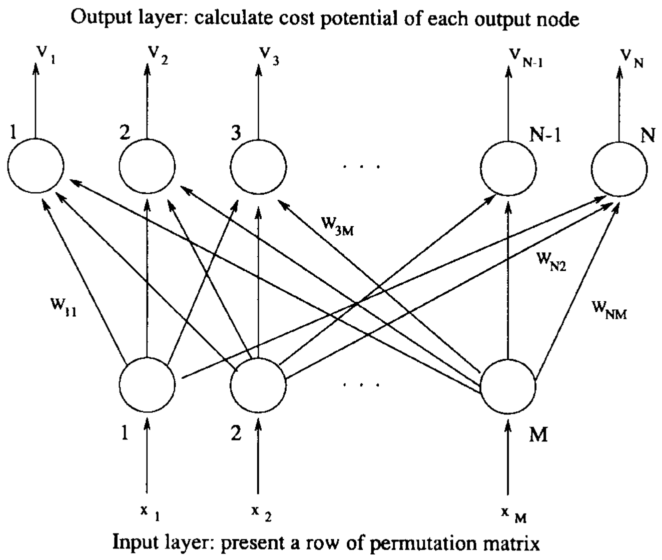

Output layer: calculate cost potential of each output node
V_1
V_2
V_3
V_{N-1}
V_N
1
2
3
N-1
N
W_{3M}
W_{N2}
W_{11}
W_{NM}
1
2
M
x_1
x_2
x_M
Input layer: present a row of permutation matrix

Fig. 3. Architecture of SONN.

the Hopfield network. Thus, the algorithm can be seen as an efficient and convenient simulation approach to the modified Hopfield network with large γ. The feasibility of the final solution can be guaranteed, since the solution trace is confined to the constraint plane, and 0–1 solutions can be encouraged using the annealing function g^{ann} without excessive computation. Furthermore, the network is still implementable in hardware, making the potential for rapid execution speed a further advantage.

IV. A Self-Organizing Neural-Network Approach

In this section we propose a new SONN based upon Kohonen's self-organizing feature map [21], and modified so that the network is able to solve 0–1 optimization problems of the general form (COP1) presented in Section II. We first discuss the ideas behind the technique, and present the network and the algorithm. We then present some new theoretical results which address the issues of convergence and stability of the network.

Consider the general form of the problem (COP1). Any matrix $\mathbf{X}$ which satisfies the constraints of (COP1) will have as its rows a permutation of the set of M-dimensional vectors

$$\begin{aligned} &(1, 0, 0, \cdots, 0) \quad \text{represented } D_1 \text{ times}\\ &(0, 1, 0, \cdots, 0) \quad \text{represented } D_2 \text{ times}\\ &\qquad\vdots\\ &(0, 0, 0, \cdots, 1) \quad \text{represented } D_M \text{ times.}\end{aligned}$$

Such a feasible solution matrix we call a *permutation matrix*. All feasible solutions to (COP1) (and hence all permutation matrices) lie at vertices of the n-dimensional unit hypercube (where $n = NM$) which also intersect the constraint plane. Since D_j is integer valued for all j, the constraint set forms an integral polytope.[1] Suppose we allow a continuous approach to such a vertex from within the unit hypercube, starting from a point on the constraint plane and inside the unit hypercube (a feasible, noninteger solution), and gradually moving along the constraint plane to approach a feasible vertex. Let us denote the continuous variable (matrix element) in the interior of the unit hypercube by $W_{i,j}$, so that $F(\mathbf{W}) = F(\mathbf{X})$ at the vertices. Essentially, $W_{i,j}$ represents the *probability* that the variable in the ith row and jth column of the matrix $\mathbf{X}$ is "on." It is the matrix of probabilities (weights), $\mathbf{W}$, to which we will apply Kohonen's principles of self-organization, modifying existing definitions to enable the process to solve 0–1 optimization problems. The SONN drives $\mathbf{W}$ to the cheapest vertex, while employing Hopfield descent on an energy function to ensure that $\mathbf{W}$ also lies on the feasible constraint plane. As such, this self-organizing neural approach can be seen as an *interior point method*.

[1]The proof of this involves rewriting the constraints in terms of vector variables $\mathbf{x}$ so that the constraints can be expressed as $\mathbf{Ax} = \mathbf{b}$, and showing that the matrix $\mathbf{A}$ is totally unimodular.

A. The Network

The architecture of the SONN (shown in Fig. 3) consists of an input layer of M nodes, and a linear array of N output nodes. The output nodes represent the row indexes of the solution matrix, and the input layer represents the M columns for the given problem. The weight connecting input node j to node k of the output layer is given by $W_{k,j}$.

Unlike other (elastic net based) self-organizing approaches, the nodes do not move in Euclidean space. Rather, they are fixed in this configuration, and the weights of the network are adapted. Rows of the permutation matrix are presented to the network through the input layer, and the nodes of the output layer compete with each other to determine which row of the solution matrix can facilitate the input vector with least cost. The weights are then adapted to reflect this decision using the neighborhood topology.

Suppose we present a row of the permutation matrix (with a "1" in column $j^\star$) to the network, and for each node k of the output layer, we calculate $V_{k,j^\star}$ which is a linear combination of the cost to the objective function of assigning the input vector the row k, and the cost (to convergence) of potentially unsettling the current values of the weight matrix.

Definition 1: The *cost potential*, $V_{k,j^\star}$, of node k for a particular input vector $\mathbf{x}$ ($x_j = 0$, $\forall j \neq j^\star$, $x_{j^\star} = 1$) is

$$\begin{aligned} V_{k,j^\star} =& \mathcal{C}(k, j^\star) + \sum_{i=1}^{N} \sum_{l=1}^{M} \mathcal{Q}(i, l, k, j^\star) W_{i,l} \\ &+ \beta \sum_{p,q \in \mathcal{W}} W_{p,q} \end{aligned} \tag{18}$$

where β is a parameter to be selected, and $\mathcal{W}$ is a subset of the indices whose weight adaptation at that time could affect the convergence and stability of the network. This subset is dependent upon the structure of the problem, and the natural tendencies of the network for a given type of problem. In Sections V and VI, we will utilize two different subsets for $\mathcal{W}$ which make use of the underlying structures of the two applications.

Definition 2: The *winning node*, m_0, of the output layer is the node with minimum cost potential for a particular input vector. That is

$$V_{m_0,j^\star} \leq V_{k,j^\star} \qquad \text{for all other nodes } k \text{ and fixed } j^\star.$$

The last term of (18) is needed so that the winning node is not just the cheapest row in which to assign the vector $\mathbf{x}$ at a particular instance in time (which may cause oscillations if certain rows are too popular), but also considers the current weight matrix and attempts to optimize around its current values. Taking into account the *history* of the weight matrix in this manner has been found to aid convergence.

Definition 3: The *neighborhood* of the winning node, m_0, is the set of nodes $m_1, m_2, \cdots, m_{\eta_{j^\star}}$ (closest neighbor to farthest neighbor) such that

$$V_{m_0,j^\star} \leq V_{m_1,j^\star} \leq V_{m_2,j^\star} \leq \cdots \leq V_{m_{\eta_{j^\star}},j^\star}$$

where $\eta_{j^\star}$ is the size of the neighborhood for column $j^\star$.

Thus, the neighborhood of the winning node is not defined spatially, according to the physical architecture of the network, but is only defined once the cost potential of each node in the output layer has been calculated and ranked for a particular input vector. Therefore, winning nodes and the neighborhood are determined by competition according to the objective function, and the weights are modified according to Kohonen's weight adaptation rule within the winning neighborhood. The size of the winning neighborhood is dependent upon which row of the permutation matrix is currently under consideration.

At the end of the Kohonen weight adaptations, the weight matrix $\mathbf{W}$ has moved in a direction which has most likely taken $\mathbf{W}$ off the constraint plane and so the solution is infeasible. The next stage of the SONN involves the weights of the other nodes then organizing themselves around the modified weights so that the matrix of network weights $\mathbf{W}$ remains a feasible solution to the problem at all times. This can be achieved in hardware[2] via a Hopfield neural network. Transforming the weight matrix $\mathbf{W}$ into a vector $\mathbf{w}$ (which represents the states of the continuous Hopfield network), we perform random and asynchronous updates of $\mathbf{w}$ (excluding the weights within the winning neighborhood) to minimize the energy function

$$\mathcal{E} = \frac{\gamma}{2} \|\mathbf{w} - (\mathbf{P}\mathbf{w} + \mathbf{s})\|^2 \tag{19}$$

where ($\mathbf{P}\mathbf{w} + \mathbf{s}$) represents the projection of $\mathbf{w}$ onto the constraint plane $\mathbf{A}\mathbf{w} = \mathbf{b}$. The Hopfield network here does not need to employ the hill-climbing dynamic introduced in the previous section of this paper, since we only need to arrive at a point on the constraint plane. Once the energy function has reached its minimum (so that $\mathbf{w}$ lies on $\mathbf{A}\mathbf{w} = \mathbf{b}$), we return to the Kohonen updating stage, presenting another randomly selected row of the permutation matrix to the SONN, determining the winning node and its neighbors, and modifying their weights. The entire process is repeated until the network weights stabilize to a feasible 0–1 solution which is a local minimum of the optimization problem.

During convergence, the magnitude of the weight adaptations, and the size of the neighborhoods is gradually decreased. Initially, the size of the neighborhood for each column of $\mathbf{W}$ (given by $\eta = (\eta_1, \eta_2, \cdots, \eta_M)$) is large, but is decreased linearly until $\eta_j = D_j$ (the demand for column j) for all j.

It is worth noting that this self-organizing neural approach is inherently stochastic in nature, since the weight modifications made in the SONN are completely dependent upon the order in which the rows of the permutation matrix are presented. Consequently, the network can be run several times to arrive at different local minima.

B. The Algorithm

Step 1) Initialize weights of the network as

$$W_{k,j} = \frac{D_j}{N}$$

thus giving an initial feasible (noninteger) solution.

Step 2) Randomly select a row from a permutation matrix. Call this vector $\mathbf{x}$ (input vector). Find the column $j^\star$ which is "on," i.e., $x_{j^\star} = 1$.

Step 3) Calculate the *potential* $V_{k,j^\star}$ for each node k in the output layer according to (18).

Step 4) Choose winning node, m_0, (by competition) such that

$$V_{m_0,j^\star} = \min V_{k,j^\star}, \qquad \forall k$$

and identify its neighboring nodes

$$m_1, m_2, \cdots, m_{\eta_{j^\star}}$$

where $\eta_{j^\star} \geq D_{j^\star}$ is the size of the neighborhood for $j^\star$.

Step 5) Update weights in neighborhood of winning node according to

$$\begin{aligned} \Delta W_{k,j^\star} =& \alpha(\eta, t)[1 - W_{k,j^\star}] \\ & \forall k\colon V_{k,j^\star} < V_{m_{\eta_{j^\star}},j^\star} \end{aligned}$$

[2]Efficient simulation of the network on a digital computer can be achieved by replacing the Hopfield network with the algorithm described in Section IV-B with no hill-climbing.

where

$$\alpha(\eta, t) = \frac{\alpha(t)\rho_{j^\star}}{D_{j^\star}} \exp\left(-\frac{|V_{l_0, j^\star} - V_{k, j^\star}|}{\sigma(t)}\right)$$

(a modified version of Kohonen's SOFM updating rule). α and σ are monotonically decreasing and positive functions of time. ρ is a normalized weighting vector used to help the network decide how to break ties for a node. All other weights (not included in neighborhood updating) have $\Delta W_{k,j} = 0$. The modified weights are

$$W_{k,j} \leftarrow W_{k,j} + \Delta W_{k,j}.$$

Step 6) The weights will no longer lie on the constraint plane, so we employ a Hopfield neural network to enforce feasibility. With a large parameter γ, $\mathbf{w}$ is modified around the weight adaptations of the SONN so that $\mathbf{Aw} = \mathbf{b}$.

Step 7) Repeat from Step 2) until all N rows of the permutation matrix have been selected as input vectors. This is one epoch. Repeat for Ω epochs. Decrease α and σ geometrically.

Step 8) Repeat from Step 2) for another permutation matrix until $|\Delta W_{k,j}| \approx 0, \forall k, j$. This represents a stable convergence of the weights for a given neighborhood size. Decrease the neighborhood sizes η_j linearly for all j.

Step 9) Repeat entire process until $\eta_j = D_j, \forall j = 1, \cdots, M$.

C. Convergence Properties

Convergence of Kohonen's SOFM has been proven by several researchers [7], [28], and since the weight adaptations in Stage 1 of the algorithm are an exact implementation of the SOFM (with modification to the criteria for winning node and neighborhood selection), it too converges under similar conditions. Stage 2 of the algorithm (the Hopfield network with no hill-climbing) also converges to a stable solution since the energy function can be shown to be a Lyapunov function, which never increases and is minimized when the states of the Hopfield network are stable [17]. Unfortunately, when these two neural networks are joined together, as in our SONN approach, the convergence of each is potentially disrupted. Consequently, exposition of a formal proof of convergence of the SONN is unlikely. Instead, we put forth some observations which are the foundations of an intuitive explanation of the apparent convergence of the algorithm. Proof of the following remarks and theorems can be found in Appendix A.

Remark 1: If $W_{k,j^\star}$ initially lies in or on the unit hypercube $\{0, 1\}^n$, then $W_{k,j^\star}$ is bounded above by one and below by zero, and will stay within the hypercube provided

$$0 < \alpha(t) \leq \min D_j \text{ and } 0 < \rho_j < 1, \qquad \forall j = 1, \cdots, M.$$

Remark 2: If $\mathbf{W}(t)$ approaches a feasible vertex of $\{0, 1\}^n$ as $t \to \infty$, then $\Delta W_{k,j}(t) \to 0$, and hence, the algorithm converges.

$\mathbf{W}(t)$ is likely to approach a vertex since, if $\alpha(0)$ is chosen to be large (at or near its maximum value of min D_j by Remark 1), then many updated weights will quickly approach 1, and the others will be forced to approach zero in Stage 2 to maintain feasibility of the weight matrix. Furthermore, once a weight is dominant, it is likely to be selected again for weight adaptation (it is generally cheaper than increasing the value of surrounding weights), which will result in the strongest weights getting stronger, and the weakest weights dying off. Thus, $\mathbf{W}(t)$ is able to approach a vertex of the hypercube $\{0, 1\}^n$, and will converge to a stable solution by Remarks 1 and 2.

The last remark assumes that the weights *approach* a vertex in a continuous sense, without any oscillations or jumps. The convergence of the network can be controlled to this effect by suitable choice of β (the larger the value of β the less likely are oscillations), and also by a small step size $\alpha(t)$ in the updating rule. If $\alpha(t)$ is small enough, then Stage 1 will create only a very slight change in the weight matrix. Consequently, the weights will not have moved very far off the constraint plane, and Stage 2 will not need to perturb the weights too far from their previous values in order to restore feasibility. In this way, it can be seen that a small value of $\alpha(t)$ generates a limiting behavior of the SONN which is only a slight perturbation from the behavior of Stage 1 alone. It is under this assumption that we present the following mathematical results.[3]

Suppose that an input vector $\mathbf{x}_l$ has probability of occurrence given by p_l, so that

$$\mathcal{P}(\mathbf{x}) = \sum_{l=1}^{M} p_l \delta(\mathbf{x} - \mathbf{x}_l) \tag{20}$$

where $\delta()$ is the Dirac delta function.

Definition 3: For a given node k of the output layer, the Voronoi set $\mathcal{V}_k$ comprises the set of all input vectors $\mathbf{x}_l$ for which node k is selected in the winning neighborhood.

Definition 4: The three-dimensional neighborhood function $h[(V_{k,j} - V_{m_0,j}), j^\star]$ is defined to be

$$h[(V_{k,j} - V_{m_0,j}), j^\star] = \begin{cases} e^{-((V_{k,j} - V_{m_0,j})/\sigma(t))} & \text{if } j = j^\star \\ 0 & \text{otherwise.} \end{cases}$$

The nature of this neighborhood function is shown in Fig. 4.

Remark 3: Provided the learning step size $\alpha(t)$ is small, there exists a function $\mathcal{F}$ given explicitly by

$$\mathcal{F}(\mathbf{W}) = \tfrac{1}{2} \sum_{m_0} \sum_{k} \sum_{j} h[(V_{k,j} - V_{m_0,j}), j^\star] \cdot \sum_{\mathbf{x}_l \in \mathcal{V}_k} p_l \|[\mathbf{x}_l]_j - W_{k,j}\|^2 \tag{21}$$

[3] Similar results have been derived for the SOFM [25], [29] which have been adapted here to include the optimization process, and the two-stage nature of the SONN.

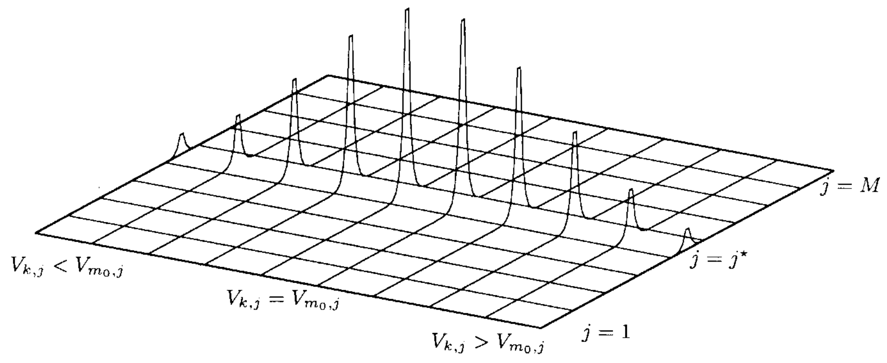


Fig. 4. Three-dimensional neighborhood function for SONN $h[(V_{k,j} - V_{m_0,j}), j^\star]$

whose expected change given a change in the state of $\mathbf{W}$ under a single learning step is given by

$$E(\Delta\mathcal{F}) = -\alpha(t) \sum_k \sum_j \|\nabla_{W_{k,j}}\mathcal{F}\|^2. \tag{22}$$

This result tells us that, on average, $\mathcal{F}$ decreases and the learning process tries to find states which minimize $\mathcal{F}$. Any individual learning step can lead to an increase in $\mathcal{F}$, but the overall trend is for the network to decrease $\mathcal{F}$. Clearly, this is very similar to the concept of simulated annealing, where the learning step size $\alpha(t)$ plays the role of temperature. $\mathcal{F}$ is continuous, but only piecewise differentiable. Furthermore, all local extrema (where $\mathcal{F}$ is differentiable) are local minima, and the nature of the updating provides an opportunity to escape from local minima.

Thus, for a small learning step size $\alpha(t)$, the SONN can be seen to converge approximately like the function $\mathcal{F}$ since a small step off the constraint plane during Stage 1 of the SONN requires only a small step during Stage 2 in order that feasibility be restored. Under this premise, Stage 2 of the SONN can be seen to have a negligible effect on the convergence trace of the network if $\alpha(t)$ is small enough. Remark 3 demonstrates that this system will then converge to the minimum of the function $\mathcal{F}$.

Addressing the issue now of whether or not this convergence minimizes the objective function, we have the following remarks.

Remark 4: Since $\Delta W_{k,j} \geq 0$ then any weight modifications in Stage 1 of the SONN will increase the objective function.

This is only a temporary increase in the objective function, since during Stage 2 of the SONN, other weights will be decreased in order to restore feasibility, and this will cause a decrease in the objective function.

Remark 5: Since

$$F(\mathbf{W}) = \sum_{k=1}^{N} \sum_{j=1}^{M} W_{k,j} V_{k,j}$$

assuming that the network is in a state of convergence, then the partial derivative is such that

$$\frac{\partial F(\mathbf{W})}{\partial W_{k,j}} = 2V_{k,j} - \mathcal{C}(k, j) \qquad \forall k \text{ and } j.$$

It can be seen from this last remark that selecting the winning node by locating the minimum $V_{k,j}$ is akin to selecting the node k which will minimize the (temporary) increase in $F(\mathbf{W})$ when $W_{k,j}$ is increased.

Thus the SONN has been shown to converge provided the learning step size $\alpha(t)$ is small, in which case the two-stage process can be viewed as a single stage (Stage 1) under a slight perturbation. Stage 1 attempts to move in the direction which will minimize the temporary increase in the objective function. Oscillations can be controlled or prevented by a suitable choice of β.

We have now presented two types of neural approaches to solving combinatorial optimization problems: a Hopfield neural network which guarantees feasibility, and allows escape from local minima; and a self-organizing neural network which generalizes to solve a broad class of assignment-type problems. These techniques are now demonstrated using two practical optimization problems which have arisen from Australian industry.

V. Application 1: The Car Sequencing Problem

We consider the problem of sequencing M different car models along an assembly line. The demands for each of the models, and some *contiguity constraints* are assumed to be known ahead of the scheduled manufacturing time. The contiguity constraints take the form of separation rules, dictating the minimum distance with which cars of the same model should ideally follow each other. These separation rules are a form of load balancing constraint. They are required because the times taken to assemble each of the models is different, and like-modeled cars must be spaced accordingly, in order that the workload be evenly spread. These distances are chosen with the current workforce and the average time taken to assemble each model in mind.

The goal of the CSP is then to identify an optimal sequence of N cars on the assembly line such that the demands for each of the M models are met (hard constraints), and the separation rules are satisfied as closely as possible (soft constraints). A similar problem has been studied by Parretto *et al.* [27] using automated reasoning, and many assembly line balancing problems have been solved using various approximate algorithms [15], [24]. To the best of our knowledge, there is no literature relating to the use of nonlinear optimization or neural-network techniques to solve such a problem.

A. Mathematical Formulation

We define a set of binary variables

$$X_{k,j} = \begin{cases} 1 & \text{if the } k\text{th car in the sequence is of type } j \\ 0 & \text{otherwise} \end{cases}$$

for $k = 1, \cdots, N$ and $j = 1, \cdots, M$. Also, let D_j be the required demand and ξ_j be the ideal minimum separation distance for cars of model $j = 1, \cdots, M$.

The CSP can be formulated as the following 0–1 quadratic programming problem:

(CSP)

$$\text{minimize} \quad \sum_{k=1}^{N}\sum_{j=1}^{M} X_{k,j} \sum_{i=1}^{N} P_{|k-i|,\,j} X_{i,j} \tag{23}$$

$$\text{subject to} \quad \sum_{j=1}^{M} X_{k,j} = 1, \qquad \forall k = 1, \cdots, N, \tag{24}$$

$$\sum_{k=1}^{N} X_{k,j} = D_j, \quad \forall j = 1, \cdots, M, \tag{25}$$

$$X_{k,j} \in \{0, 1\}. \tag{26}$$

Constraint (24) ensures that no more than one car is assigned to the same position in the sequence. Constraint (25) ensures that the demand is satisfied (the sum of each column of $\mathbf{X}$ equals the demand for the car model represented by the column). The objective function (23) enables the soft constraints relating to ideal separation distances to be satisfied as closely as possible. The cost matrix $\mathbf{P}$ assigns a penalty to the objective function which depends upon the distance between like-modeled cars. Sequences of cars are not only penalised for violating separation rules, but the importance of each of the rules, and the severity of the violations are also taken into account when assigning a penalty. This is achieved through the choice of the cost matrix $\mathbf{P}$. The particular matrix $\mathbf{P}$ which we use in this paper is

$$\mathbf{P} = \begin{pmatrix} 4 & 9 & 22 & 7 \\ 2 & 7 & 20 & 5 \\ 0 & 5 & 18 & 3 \\ 0 & 4 & 16 & 0 \\ 0 & 3 & 14 & 0 \\ 0 & 0 & 12 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 8 & 0 \end{pmatrix}. \tag{27}$$

Null rows are then added to $\mathbf{P}$ to produce N rows. $P_{0,j}$ is defined to be zero for all j. Clearly, the relative weightings of each column in $\mathbf{P}$ indicate that (in this example) it is considered most important to attempt to satisfy separation rule 3, followed by separation rules 2, 4, and 1. The number of nonzero elements in each column j of $\mathbf{P}$ is exactly equal to the ideal separation distance ξ_j. Separating like-modeled cars by a distance greater than ξ_j incurs no penalty. Decreasing the cost of violations as two like-modeled cars get further apart (as indicated by the row number), provides a means of measuring the severity of separation rule violations.

Further details of the problem and its formulation can be found in [31]. It is worth noting that several other practical optimization problems have been expressed in a similar form, including the airline-gate assignment [30] and intermodal trailer assignment problems [11].

Expressing (23) in standard quadratic programming form $\mathbf{x}^T\mathbf{Q}\mathbf{x}$, and examining the eigenvalues of sample $\mathbf{Q}$ matrices based on $\mathbf{P}$ as in (27), reveals that the objective function is an indefinite quadratic form. Consequently, any optimization technique which requires at least a positive semidefinite form will be unable to locate the global minimum of CSP. Such techniques include the commercial optimization package GAMS using the nonlinear solver MINOS-5 [4]. In Section V-C, GAMS/MINOS-5 is used as a local optimization technique when comparing its performance on various instances of the CSP with the improved Hopfield network, the SONN, and a simulated annealing heuristic.

B. Heuristic and Neural-Network Approaches to the CSP

In this section, we briefly describe the simulated annealing heuristic, and the choice of parameters used for the neural techniques.

1) Simulated Annealing: The form of simulated annealing (SA) which we employ for the CSP involves interchanging rows of feasible solution matrices, which retains the feasibility of the matrix. We start with a random initial (feasible) solution matrix $\mathbf{X}$, and walk through feasible solution space by randomly selecting a row j of $\mathbf{X}$ to interchange with a randomly chosen row i under consideration. We then calculate the energy of the system as follows.

- Let dx be the partial cost due to cars in positions i and j of the sequence given by $\mathbf{X}$.
- Swap rows i and j of $\mathbf{X}$ to give a new (feasible) solution matrix $\mathbf{Y}$.
- Let dy be the partial cost due to cars in positions i and j of the sequence given by $\mathbf{Y}$.
- Let $F(\mathbf{X})$ be the total cost of solution matrix $\mathbf{X}$. Clearly

$$F(\mathbf{Y}) = F(\mathbf{X}) - (dx - dy).$$

Let $\triangle F = dx - dy$. By the theory of simulated annealing, Y becomes the new feasible solution matrix if either

$$\triangle F < 0 \quad \text{or} \quad \exp(-\triangle F/kT) > \textit{random}[0, 1]$$

where k = Boltzmann's constant, T = temperature, and *random*$[0, 1]$ returns a uniformly distributed random number between zero and one. Using this analogy, it is clear that $\triangle F$ represents the change in energy of the system.

A *cooling schedule* is completely specified by the initial and final temperatures (T_0 and T_η), a temperature decrement rule (usually geometric), and the length of the Markov chain, L_n, (the number of random walks to be allowed at each temperature stage). For the car sequencing problem with N cars and M models, we adopt the following cooling scheme:

$$T_{n+1} = 0.95T_n, \quad T_0 = 64, \quad T_\eta = 0.5, \quad L_n = 2MN.$$

The values for T_0 and T_η have been selected using the acceptance ratio technique. This involves calculating values of χ (the probability that an uphill move is accepted) by the method proposed by Kirkpatrick *et al.* [20] for a selection of problems (i.e., varying M and N). T_0 is then the smallest temperature which gives a value of χ greater than 0.8, for all selected problems. Similarly, T_η is the largest temperature which gives a value of χ less than 0.2 for all selected problems. We have selected four different sized problems on which our cooling scheme results are based: the number of models is fixed at $M = 4$, but the length of the sequence is varied for $N = 20$, 40, 60, and 80.

2) Hopfield Network Parameters for the CSP: For the efficient simulation technique of the Hopfield network, the solution matrix $\mathbf{X}$ of the CSP is first replaced with a solution vector $\mathbf{x}$ (obtained by concatenating the rows of the matrix). The corresponding objective function and linear constraints can be readily derived using this representation [31]. The value of the time-step for the steepest descent is selected to be $\Delta t = 0.0001$. The neurons are initialized to small random perturbations around the center of the hypercube. The use of annealing to drive the solution toward a vertex is found to be unnecessary, since it can be shown that no constrained interior point of the CSP objective function can ever be a local minimum. Results are presented for both the hill-climbing dynamic, and the same network with no hill-climbing ($\alpha(t) = 1$ always). While the former network is referred to as HCHN, the latter will be denoted by HN. For the hill climbing network HCHN the length of the Markov chain is given by $20(MN)$, which varies with the size of the problem. The value of τ is set to 40.

3) SONN Parameters for the CSP: The SONN algorithm as applied to the CSP can best be understood in terms of the N output layer nodes of the SONN (representing rows of $\mathbf{X}$, and positions in the sequence) each competing for possession of the car model presented through the input vector $\mathbf{x}$. By comparing the objective function of the CSP, (23), with the objective function of (COP1), we see that

$$\mathcal{Q}(k, j^\star, i, l) = P_{|k-i|, j^\star}$$

if $l = j^\star$, and is zero otherwise, and $\mathcal{C}(k, j) = 0$ for all k and j. Without the final term of the cost potential function (18), the natural tendency of the SONN for the CSP is for each car model to try to be positioned either first or last in the sequence, since the potential for violations of the separation rules is then reduced. Oscillations will therefore occur since only one car model can occupy a position in the sequence. The final term of the cost potential function (18) can help to avoid such oscillations by detering a node from dominating and "overwriting" an existing decision unless it is very much cheaper to allow it. For the CSP, the subset of indices $\mathcal{W}$ is chosen so that the final term sums over the weight elements in the row k, except for the element $W_{k, j^\star}$. Thus

$$\mathcal{W} = \{(p, q): p = k, q \in \{1, \cdots, M\}, q \neq j^\star\}.$$

In this way, the suitability of each row as the winner is not just determined by its cost, but also by the current state of the row. If the final term of (18) is large, we are permitting an oscillation if we select node k as a winning node. If this term is negligible, then the only strong weight in row k is $W_{k, j^\star}$ and we are not disrupting convergence. The magnitude of the parameter β is chosen in such a way that oscillations are only permitted if they will significantly improve the cost of the solution. For the cost matrix $\mathbf{P}$ as specified by (27), $\beta = 9 + 2(D_2 - 1)$ is found to achieve the desired balance between avoiding unnecessary oscillations, and permitting changes to the current weight matrix. This value is derived analytically in Appendix B.

Another way of avoiding oscillations is to use the normalized weighting vector ρ to help the network decide which of the input vectors competing for a certain node is the ultimate winner. These relative weightings reflect the cost and difficulty associated with assigning the specified demand of cars within the sequence according to the separation rules. One measure of this difficulty is

$$\rho_j = \frac{\xi_j D_j}{N}. \tag{28}$$

If $\rho_j \le 1$, then the required number of cars of model j (D_j) could be assigned to positions in the sequence without violating the separation rule given by ξ_j (if the assignments in the other columns permit it). If $\rho_j > 1$ however, separation rule violations are inevitable. The vector ρ acts as an *encouragement factor* since the SONN is unlikely to choose such assignments often. Each element ρ_j is then normalized before the algorithm is run, so that $\rho_j \le 1$ for all j which is a necessary condition for stability of the network as discussed in Section IV-C.

For each particular instance of the CSP considered in Section V-C we select the following values for the SONN parameters:

$$\begin{aligned} \alpha(0) &= \min D_j, & \alpha(t+1) &= 0.95\alpha(t) \\ \sigma(0) &= 9, & \sigma(t+1) &= 0.95\sigma(t) \\ \eta_j(0) &= D_j + N/10, & \eta_j(t+1) &= \eta_j(t) - 1 \end{aligned}$$

$\Omega = 5$ and ρ is defined according to (28). It is the vector ρ which appears to affect the results most significantly. Most notably, ρ as defined by (28) produces considerably better results than those obtained using a uniform ρ vector of "1's". The values of σ and Ω were determined experimentally, while the initial value of α was chosen so that the weights are always bounded above by one.

For the Hopfield network section of the SONN, we use the efficient simulation technique described in Section IV-B with no hill-climbing ($\alpha = 1$). Since there is no objective function for this section of the SONN (only constraint satisfaction terms), the Hopfield network simply consists of the "projection and clipping" algorithm respresented by Fig. 2.

C. Results for the CSP

The results which follow are based on five problem classes for four model types ($M = 4$) as described in Table I. The demands are expressed as a proportion of the sequence length N. The cost matrix $\mathbf{P}$ is that of (27). In practical situations,

TABLE I
PROBLEM CLASS DESCRIPTIONS FOR THE CSP

Problem Class	ξ_j	D_j/N
1	(2,5,8,3)	(0.2,0.3,0.2,0.3)
2	(4,4,4,4)	(0.1,0.3,0.2,0.4)
3	(2,2,6,3)	(0.4,0.1,0.1,0.4)
4	(1,7,5,2)	(0.3,0.2,0.1,0.4)
5	(2,3,3,4)	(0.3,0.1,0.5,0.1)

demands will often fluctuate, and the number of workers on the assembly line may not remain constant. Since the number of workers at different stations along the assembly line determines the ideal separation distances (and so ξ_j), it is necessary to ensure that the methods used can handle variations in D_j and ξ_j.

Results for each problem class are presented in Table II, for $N = 20, 40, 60$, and 80 cars in the sequence. For all heuristics, the algorithms are run from ten different random starting points. In Table II, "AvMin" represents the average value of those ten final solutions, while "BestMin" is the cheapest cost found during the ten runs.

The first result to note from Table II is that the hill-climbing techniques (SA and HCHN) considerably outperform the techniques which only permit descent (GAMS/MINOS-5 and HN), and the SONN which falls into neither of these categories. This is to be expected since the CSP objective function is known to be highly convoluted, and plagued by an extremely large number of local minima [31]. On the particular instances of the CSP represented in Table II, HCHN marginally outperforms the SA heuristic, as evidenced by better average values of the local minima as the problem size increases. A comparison of these two hill-climbing approaches is shown in Fig. 5, where the best minimum results for both SA and HCHN are plotted as a percentage deviation from the best found solution. It is noted that for all problem classes of all problems sizes, either SA or HCHN finds this best solution.

The pure descent version of the improved Hopfield network, HN also outperforms both the SONN and GAMS/MINOS-5, while the SONN consistently locates better local minima than GAMS/MINOS-5. This relationship can be seen in Fig. 6. Of interest in this graph, is the fact that as the problem size increases, the performance of the three nonhill climbing techniques becomes more even. This is in keeping with the asymptotic properties of quadratic assignment problems (of which the CSP is a generalized form), which state that the ratio between the worst and best solutions of quadratic assignment problems approaches unity as the problem size increases. We refer the reader to [5] for a detailed exposition and proof of this property.

A final observation from the graphs in Figs. 5 and 6 is that the scale of the percentage deviation axis is considerably larger in Fig. 6 than in Fig. 5. While the majority of the deviations of the hill-climbing techniques are within 3% of the best found solution, the deviations from the nonhill climbing techniques are mostly within 20%.

While it is unfair to compare hill-climbing techniques with those utilizing strict descent, it is clear from each of the graphs that the neural techniques perform well compared to the more traditional techniques in the same category. The SONN, although falling into neither a hill-climbing or strict descent category, could be further improved by embedding the ability for foresight into the algorithm. Currently, the algorithm is *greedy* in the sense that the winning node is the one with least cost potential. If the algorithm sometimes permitted nodes to win which were not necessarily the cheapest at that point in time, much like simulated annealing, then the SONN may be able to escape local minima too. Permutations of the winning neighborhood might be another way to achieve this effect. Further work on the choice of appropriate β values is also needed to ensure that these parameters are optimized.

VI. APPLICATION 2: A POSTAL DELIVERY NETWORK

In this section we describe a practical optimization problem which has arisen from the postal services industry. The problem considered here is an instance of the p-hub location-allocation problem and is formulated using the quadratic integer model of O'Kelly [26].

The PDN consists of N postal districts, each of which has a certain volume of mail which needs to be delivered to the other districts. Each postal district is represented by a single node in the plane. In this paper, for the purposes of simplicity, we refer to this representative node as a *postoffice*. The (x, y) coordinates of each postoffice are known. Mail can only be transfered from one postoffice to another via sorting centers. There are p such sorting centers which need to be located at existing postoffices, acting as hubs in the PDN at which incoming mail is processed and transfered to other sorting centers for distribution to the destination postoffices. Consider any two postoffices j and i, and two sorting centers located at postoffices k and l, in the planar domain. If j is allocated to a sorting center at postoffice k and i is allocated to a sorting center at postoffice l, then all mail originating from postoffice j and intended for delivery to postoffice i must first be collected from postoffice j and sent to the sorting center at k, transfered to the sorting center at l, and then delivered to postoffice i. The freight costs involved in such a route are due to the individual costs of collection, transfer, and delivery per unit distance per unit of volume of mail, multiplied by the volume of mail from j to i and the distance travelled in delivering this mail.

The problem is then to determine which of the N postoffices should be designated as the p sorting centers, and to which sorting center each of the postoffices should be allocated, so that the total freight costs of the postal delivery network (PDN) are minimized. This practical optimization problem is a particular instance of the *hub location problem* studied by O'Kelly [26]. A few simplifying assumptions have been made so that we can use the same model for the PDN: we assume that the distances between postoffices is Euclidean (although road-map distances could just as easily be used as input data); there are no existing sorting centers and no capacity restrictions on the p sorting centers; and finally, the major cost contributor to the PDN is assumed to be freight cost, so that other fixed costs are not taken into consideration. We do not assume that the matrix of costs is symmetric, since

TABLE II
RESULTS OF CSP TEST PROBLEMS FOR GAMS/MINOS-5, SA, HN, HCHN, AND SONN

		GAMS/MINOS-5	SA		HN		HCHN		SONN	
N	Class	BestMin	AvMin	BestMin	AvMin	BestMin	AvMin	BestMin	AvMin	BestMin
20	1	63	60.55	58	61.4	60	60.6	59	62.5	60
	2	49	41.45	40	49.6	48	43.1	42	50.0	46
	3	34	30.56	30	33.9	31	30.3	29	33.0	32
	4	21	10.6	10	19.0	14	13.6	11	19.4	16
	5	150	150	150	150	150	150	150	156.6	152
40	1	168	150.25	146	162.4	155	151.3	146	164.6	158
	2	123	100.2	94	125.3	122	100.4	95	126.0	121
	3	73	70.25	67	73.2	70	69.4	66	75.4	74
	4	43	35.3	33	39.9	36	35.0	34	45.0	41
	5	376	360.2	354	363.4	361	359.0	352	369.2	363
60	1	260	243.55	238	256.1	248	239.6	235	258.3	250
	2	172	168.2	155	189.7	175	165.5	152	185.6	173
	3	115	111.65	108	113.4	109	108.2	105	117.2	110
	4	65	63	60	65.2	59	60.8	58	65.1	61
	5	610	570.8	566	571.1	564	565.2	562	608.3	575
80	1	349	342.05	331	341.3	336	336.6	330	340.4	341
	2	248	230.6	215	253.4	245	233.2	220	256.1	245
	3	151	154.15	150	152.8	148	149.6	146	154.6	147
	4	89	88.9	86	88.9	83	85.4	82	90.3	85
	5	795	781.4	772	788.5	777	777.8	774	799.4	780

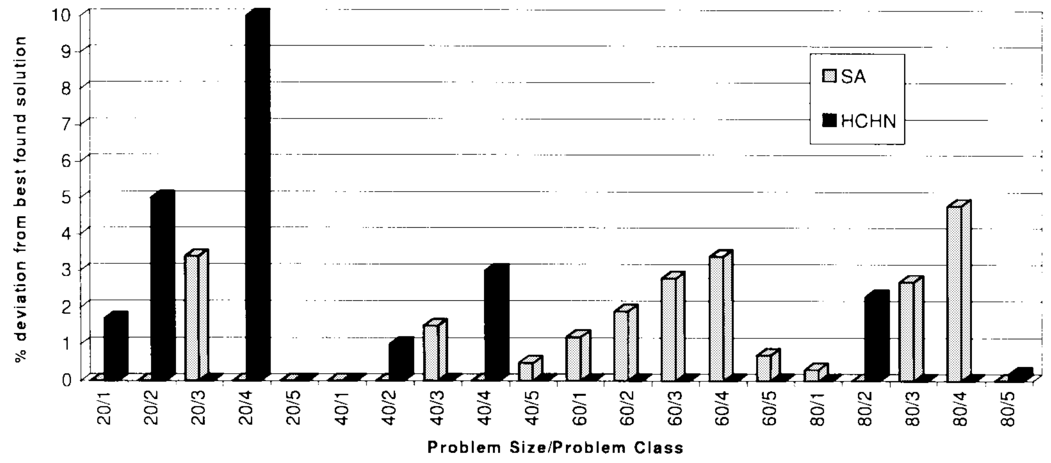


Fig. 5. Comparison of hill-climbing techniques for the CSP test problems.

the three components to the freight costs—collection, transfer, and distribution—could be unequal.

A. Mathematical Formulation

We define a set of binary variables

$$X_{jk} = \begin{cases} 1 & \text{if postoffice } j \text{ is assigned to a sorting center located at postoffice } k \\ 0 & \text{otherwise} \end{cases}$$

and

$$X_{kk} = \begin{cases} 1 & \text{if postoffice } k \text{ is a sorting center} \\ 0 & \text{otherwise.} \end{cases}$$

Let D_{jk} be the distance between postoffices j and k, V_{ji} is the mail volume from j to i, and γ^c, γ^d, and γ^t are the unit costs of collection, distribution, and transfer, respectively. The PDN can then be formulated as the following 0–1 quadratic programming problem:

(PDN)

$$\text{minimize} \quad \sum_j (\gamma^c o_j + \gamma^d d_j) \sum_k X_{jk} D_{jk} + \gamma^t \left(\sum_j \sum_i V_{ji} \sum_k \sum_l X_{jk} X_{il} D_{kl} \right) \tag{29}$$

$$\text{subject to} \quad \sum_k X_{jk} = 1 \quad \text{for all } j \tag{30}$$

$$\sum_k X_{kk} = p \tag{31}$$

$$X_{jk} \le X_{kk} \quad \text{for all } j \text{ and } k \tag{32}$$

$$X_{jk} \in \{0, 1\} \quad \text{for all } j \text{ and } k. \tag{33}$$

The first term of the objective function (29) represents the linear costs of collection and delivery, while the second term is

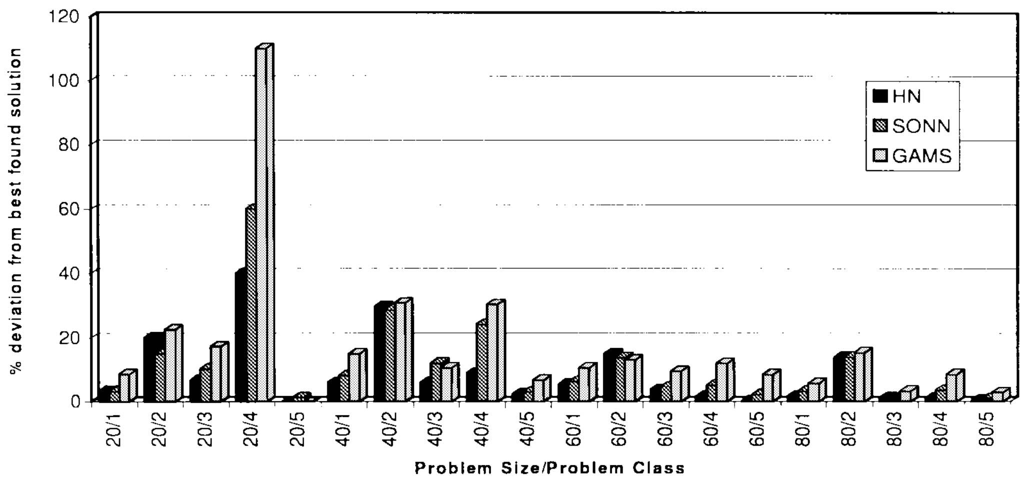


Fig. 6. Comparison of non-hill-climbing techniques for the CSP test problems.

the quadratic cost of transfering mail between sorting centers. Constraint (30) ensures each postoffice is allocated to exactly one sorting center, while constraint (31) makes sure exactly p sorting centers are located. Constraint (32) ensures that a postoffice is only allocated to a sorting center, and not to another postoffice which is not a sorting center.

Converting the objective function (29) into the standard quadratic form $\mathbf{c}^T\mathbf{x}+\mathbf{x}^T\mathbf{Q}\mathbf{x}$ reveals $\mathbf{Q}=\mathbf{C}^T\otimes\mathbf{V}$, where $\otimes$ denotes the Kronecker product of the two matrices, $\mathbf{c}^T$ is the vector obtained by concatenating the rows of $c_{ij}=(o_i+d_i)C_{ji}$ and

$$C_{jk}=\gamma^c D_{jk},\ C_{kj}=\gamma^d D_{kj},\quad \text{and}\quad C_{kl}=\gamma^t D_{kl}$$

for postoffices j and i, allocated to sorting centers k and l, respectively. The constraints (30)–(32) can readily be converted into the vector form $\mathbf{Ax}\le\mathbf{b}$, and can further be transformed into $\mathbf{A}^+\mathbf{x}^+=\mathbf{b}^+$ by employing slack variables. While the vector $\mathbf{x}$ is of dimension N^2, the inequality constraints (32) result in a further N^2 slack variables, and so the length of the augmented vector $\mathbf{x}^+$ is $2N^2$. This augmented vector form has no effect of the objective function other than to add null rows and columns to correspond to the slack variables (which do not effect the cost). Although the linear constraints are slightly different from those represented in (COP1), it will be shown that the neural techniques can still be applied.

The PDN can now be expressed in the standard quadratic integer form with linear constraints which is the required form for the neural networks. In the following sections, we drop the augmented notation for simplicity and assume the vector $\mathbf{x}$ to already contain slack variables. The nature of the quadratic form is indefinite due to the fact that the matrix $\mathbf{C}$ contains zero diagonals, since it is related to the distance matrix. Consequently, in Section VI-C, the use of the commercial optimization package GAMS/MINOS-5 cannot be expected to obtain the globally optimal solution, since it requires a positive definite form to guarantee globally optimal solutions. We use the package merely as another local optimization technique with which to compare the results of simulated annealing, the Hopfield network, and the SONN. The use of the hill-climbing Hopfield network (HCHN) was not found necessary for the PDN, due to the fact that there are relatively fewer local minima in the PDN as compared to the CSP.

B. Heuristic and Neural-Network Approaches to the PDN

In this section, we briefly outline the form of simulated annealing used for the results in Section VI-C. We also specify the parameters which were chosen for the neural techniques.

1) Simulated Annealing: The simulated annealing heuristic which we apply to solve the PDN solves the location-allocation problem by using centroids to determine the location of the sorting centers and a simple exchange approach for the allocation of the postoffices to those sorting centers. An initial feasible solution is obtained by locating the first two sorting centers at the postoffices which are the furthest apart in terms of distance weighted by the mail volumes. The next sorting center is located at the postoffice which is furthest from the first two sorting centers in the same fashion. The process is continued until p sorting centers have been located. Initial allocations are then made which minimize the distance weighted by volume from each postoffice to a sorting center. The initial feasible solution is completed by relocating the sorting centers to lie at the postoffice which is closest in distance to the centroid of each cluster.

The simulated annealing algorithm then proceeds by selecting postoffices at random and reallocating them to another sorting center if the cost of doing so decreases, or if the Boltzmann probability factor satisfies the requirements of the simulated annealing algorithm [20] allowing a temporary increase in cost (this enables escape from local minima). The centroids of each cluster are recalculated after each transition, and the sorting centers are relocated to the postoffices nearest each centroid.

2) Hopfield Network Parameters for the PDN: For the efficient Hopfield network simulation technique, HN [with $\alpha(t)=1$ always], the value of the time-step for the steepest descent is selected to be $\Delta t=0.0001$. The neurons are again initialized to small random perturbations around the center of the hypercube. Unlike the CSP, the PDN requires the use of annealing to drive the solution toward a vertex, and the value

ε used to generate the parameters $\mathcal{L}$ and $\mathcal{U}$ in the clipping function (17) is 10^{-5}.

The HCHN is not used for the PDN, since the HN yields good quality solutions without the need for a continued search for better quality local minima.

3) SONN Parameters for the PDN: When mapping the PDN onto the self-organizing framework described in Section IV, we first observe that the objective function and constraints of the PDN are not in exactly the same form as specified by the general problem class identified in Section II. The form of the demand constraint has changed from a sum down the columns in the general form, to a sum down the diagonals in the PDN. These modifications can be easily incorporated into the algorithm. For the SONN as applied to the PDN, a postoffice is presented to the network, and the other postoffices compete to determine which of them will be the hub for this postoffice. The first two terms of the cost potential function (18) are simply the cost of assigning a postoffice $j^\star$ to a hub at k. This is the first derivative of the PDN objective function (29)

$$(\gamma^c o_{j^\star} + \gamma^d d_{j^\star}) D_{j^\star k} + \gamma^t \left(\sum_i \sum_l V_{j^\star i} W_{il} D_{kl} \right). \quad (34)$$

Thus

$$\begin{aligned} \mathcal{C}(j^\star, k) &= (\gamma^c o_{j^\star} + \gamma^d d_{j^\star}) D_{j^\star k} \\ \mathcal{Q}(j^\star, k, i, l) &= \gamma^t V_{j^\star i} D_{kl} \end{aligned}$$

and $M = N$. Clearly, the natural tendency of the network without the final term of the cost potential function will be for all postoffices to try to assign themselves to their own hub, since the first term of (34) will vanish. Since only p of the postoffices can be hubs, however, oscillations will occur. For the PDN, the form of the final term of the cost potential function needs to take into account the hubs which have already formed, and permit a new hub to be formed only if it is very much cheaper to allow it. The subset of indexes $\mathcal{W}$ is only nonempty if $k = j^\star$, and then consists of the diagonal indexes of the weight matrix $\mathbf{W}$, except for the element $W_{j^\star, k}$, i.e., $\mathcal{W} = \{(p, q)\colon p = q, (p, q) \neq (j^\star, j^\star)\}$. For the case where the mail volumes are uniform, and the problem becomes one of minimum distance only, the value of β is chosen according to the formula

$$\beta \approx \min D_{kj} \frac{(\gamma^c o_j + \gamma^d d_j)}{p} \quad (35)$$

for all k and j, which is derived in Appendix B. For nonuniform mail volumes, a value of $\beta = 0$ does not inhibit convergence to a feasible solution, and is the value used.

For the PDN, the vector ρ is chosen to be uniform since it should be no more difficult to assign one postoffice than another. The remaining SONN parameters are selected as follows:

$$\begin{aligned} \alpha(0) &= 0.9, & \alpha(t+1) &= 0.95\alpha(t) \\ \sigma(0) &= 3.0, & \sigma(t+1) &= 0.95\sigma(t) \\ \eta_j(0) &= p + N/5, & \eta_j(t+1) &= \eta_j(t) - 1. \end{aligned}$$

The size of the neighborhood is identical for all postoffices, and is decreased until $\eta = 1$. Ω is selected to be equal to N.

For the Hopfield network section of the SONN, the "projection and clipping" algorithm utilizes the annealing function (17) with a value of $\varepsilon = 10^{-5}$.

C. Results for the PDN

A particular instance of the PDN is completely specified by the number of postoffices in the region (N), the required number of sorting centers (p), the (x, y) coordinates of each postoffice in the Euclidean plane (from which the distances between postoffices can be calculated), the volume of mail between each of the N postoffices ($\mathbf{V}$), and finally, the unit costs of collection, distribution, and transfer of mail. The results presented in Table III consider variations in all of these parameters (except the unit costs which are held constant at $\gamma^c = 3.2$, $\gamma^d = 0.5$, and $\gamma^t = 0.4$).

For $N = 10$, two different configurations of ten postoffices were randomly generated in the Euclidean plane (named configurations $10A$ and $10B$). For each of these configurations, two types of volume matrices are considered: a uniform matrix (named $10U$—all elements are equal to unity), and a nonuniform volume matrix whose elements were randomly generated (named $10NU$). Results are presented for each of these combinations of parameters for $p = 2$ and $p = 3$. Similarly, for $N = 15$, an extra five postoffices were added to the configurations $10A$ and $10B$ to generate the configurations $15A$ and $15B$, while five extra rows and columns (randomly generated in the nonuniform case) were added to the volume matrices. The procedure was repeated for $N = 20$ where another five postoffices and corresponding mail volumes were added to the combinations for $N = 15$. Again, for $N = 15$ and $N = 20$ each of the instances of the PDN were solved for $p = 2$ and $p = 3$. The results for each of these instances are presented in Table III for GAMS/MINOS-5, simulated annealing (SA), HN, and the SONN. The column labeled "Optimal" provides the exact global minimum found using an alternative mixed integer linear programming formulation[4] [9], while the remaining columns indicate the percentage deviation from this optimal solution.

It is clear from Table III that the modified Hopfield neural network is particularly well suited to this type of problem, outperforming all of the other techniques consistently. In fact, HN locates the optimal solution in all but two instances ($N = 20$, $p = 2$, and configuration B). It should be noted here that the simulated annealing heuristic has not been optimized, and has been run using the same cooling schedule as the authors of the code originally specified for this problem [22]. The simulated annealing results could no doubt be improved if the cooling schedule was optimized. The SONN appears to perform competitively with SA and the GAMS/MINOS-5 solver, but seems more suited to solving the PDN with

[4]While the quadratic formulation uses N^2 binary variables and $(1 + N + N^2)$ linear constraints, this linear formulation uses $(N^3 + N^2)$ variables of which N^2 are binary, and $(1 + N + 2N^2)$ linear constraints. For small sized problems this difference has little effect on computation time, but as the problem size is increased, the linear formulation will quickly become intractable.

TABLE III
RESULTS OF PDN TEST PROBLEMS FOR GAMS/MINOS-5, SA, HN, AND SONN

N	p	V	configuration	Optimal	GAMS	SA	HN	SONN
10	2	U	A	981.92	0.0%	6.4%	0.0%	0.1%
			B	1542.99	5.1%	0.0%	0.0%	1.2%
		NU	A	53300.0	5.5%	3.3%	0.0%	3.3%
			B	66519.19	0.0%	0.0%	0.0%	0.0%
	3	U	A	769.49	0.0%	20.1%	0.0%	0.0%
			B	1255.16	0.2%	0.0%	0.0%	0.0%
		NU	A	40857.80	0.0%	18.8%	0.0%	14.5%
			B	49408.13	0.0%	13.8%	0.0%	12.2%
15	2	U	A	2244.08	0.9%	0.0%	0.0%	0.0%
			B	4560.90	0.1%	0.0%	0.0%	0.1%
		NU	A	130514.62	0.0%	2.3%	0.0%	1.6%
			B	226894.42	0.0%	0.0%	0.0%	0.7%
	3	U	A	1820.01	0.0%	0.0%	0.0%	5.6%
			B	3822.23	0.2%	7.7%	0.0%	0.0%
		NU	A	107587.85	0.1%	0.6%	0.0%	5.3%
			B	177406.10	0.0%	0.0%	0.0%	0.5%
20	2	U	A	3958.67	1.8%	0.0%	0.0%	0.0%
			B	9170.87	0.4%	0.0%	0.1%	0.1%
		NU	A	222029.80	3.4%	1.4%	0.0%	1.4%
			B	467362.41	0.1%	4.9%	0.1%	0.1%
	3	U	A	3331.02	1.6%	5.9%	0.0%	1.6%
			B	7156.63	0.01%	14.8%	0.0%	0.0%
		NU	A	184236.23	1.5%	3.3%	0.0%	0.0%
			B	355666.89	0.01%	0.1%	0.0%	0.1%

a uniform volume matrix, rather than the nonuniform case. The difference that the nature of the volume matrix makes to the complexity of the problem is equivalent to the difference in complexity between a TSP whose distances are either Euclidean or random. An explanation for the slightly poorer performance of the SONN in the nonuniform problems might be that the choice of β needs improvement. While β was chosen according to (35) for the problems with uniform volume, a value of $\beta = 0$ was used for the nonuniform problems, since convergence problems do not seem to arise when the volume matrix is nonuniform. Nevertheless, the performance of the SONN is still very competitive with the SA heuristic and GAMS/MINOS-5.

For each of the techniques GAMS/MINOS-5, SA, HN and SONN, the average percentage deviation from the optimal solution over all the test problems is 0.87, 4.3, 0.008, and 2.02%, respectively. While most of the results presented in Table III are within 5% of the optimal solution, the difference in the nature of the solution is quite subtle. Typically a large variation in the percentage deviation from the best found solution only corresponds to a difference of a single location or allocation. Suboptimally assigning just one postoffice to a sorting center can account for a significantly poorer solution (as in the case where $N = 10$, $p = 3$ and the volume matrix is nonuniform).

The results also confirm the expectation that network costs can be considerably reduced by allowing more sorting centers (although the initial setup costs of constructing additional sorting centers is not included in the objective function). By including such set-up costs, it should be possible to determine the critical value of p at which a minimum cost for the PDN can be attained.

VII. CONCLUSIONS

In this paper, we have demonstrated that neural-network techniques *can* compete effectively with more traditional heuristic solutions to practical combinatorial optimization problems. We have seen how the Hopfield network has evolved to the stage where is can now be guaranteed to find a feasible solution to the problem, and we have extended the theory to enable the quality of those solutions to be improved via a hill-climbing modification to the internal dynamics. We have also presented an SONN which, unlike existing self-organizing approaches to optimization, is able to generalize to solve a broad class of 0–1 optimization problems, rather than just the TSP or related Euclidean problems. A new set of theoretical remarks has been proposed to address the convergence of the SONN.

While it is important to test techniques on theoretical problems such as the TSP, it is equally important not to let the results of such tests dominate the future direction of a field of research, or inhibit the application of a potentially powerful technique to solve real-world problems. To date, the accepted wisdom on the suitability of neural networks for solving combinatorial optimization problems has been fairly mixed. Furthermore, there is a relative scarcity of literature which uses practical optimization problems as the benchmarks by which to test a neural approach. This paper has attempted to address this issue by solving two sample practical optimization problems which have arisen from industry. In previous work [34], [35], we have solved other practical optimization problems and reached the same conclusions. Comparative results between the Hopfield and SONN approaches against simulated annealing and the commercial optimization package GAMS/MINOS-

5 have been presented and discussed. These results show quite clearly that the neural approaches can match the performance of simulated annealing, and can even outperform it in many instances. Combining this knowledge with the fact that neural networks have the potential for rapid computational power and speed through hardware implementation, it is clear that neural-network techniques are immensely useful for solving optimization problems of practical significance.

Appendix A
Proofs

Proof of Remark 1:

$$\begin{aligned}W_{k,j^\star}(t+1) &= W_{k,j^\star}(t)+\Delta W_{k,j^\star}(t)\\ &= W_{k,j^\star}(t)(1-\alpha(\eta_{j^\star},t))+\alpha(\eta_{j^\star},t).\end{aligned}$$

Therefore, if

$$0 \le W_{k,j^\star}(t) \le 1$$

then

$$\alpha(\eta_{j^\star},t) \le W_{k,j^\star}(t+1) \le 1.$$

Since

$$\alpha(\eta_{j^\star},t) = \frac{\alpha(t)\rho_{j^\star}}{D_{j^\star}} \exp\left(-\frac{|V_{m_0}-V_k|}{\sigma(t)}\right)$$

then $0 \le W_{k,j^\star}(t+1) \le 1$ provided $0 \le \alpha(t) \le \min D_j$ and $0 < \rho_j < 1$. □

Proof of Remark 2: If β is large enough, oscillations will be dampened as $t \to \infty$, so that $[\mathbf{W}]_k \to \mathbf{x}$ if k is in the winning neighborhood. Thus $W_{k,j^\star} \to x_{j^\star} = 1$ and $W_{k,j} \to x_j = 0$ (for $j \neq j^\star$).

Near a vertex then,

$$(1-W_{k,j^\star}(t)) \to 0$$

so

$$\Delta W_{k,j^\star}(t) \to 0 \qquad \text{as } t \to \infty$$

if $\mathbf{W}(t)$ approaches a vertex as $t \to \infty$. □

Proof of Theorem 3: Let $h[(V_{k,j}-V_{m_0,j}),j^\star]$ be denoted by $h(k,j,j^\star)$. Using

$$W_{k,j} \leftarrow W_{k,j}+\alpha(t)h(k,j,j^\star)(x_j-W_{k,j})$$

the change in $\mathcal{F}$ due to an update of the weight $W_{k,j}$ is

$$\begin{aligned}\Delta\mathcal{F} &= \tfrac{1}{2}\sum_{m_0}\sum_{k}\sum_{j} h^2(k,j,j^\star)\\ &\quad\cdot \sum_{\mathbf{x}_l\in\mathcal{V}_k} p_l\{-2\alpha(t)(x_j-W_{k,j})[\mathbf{x}_l]_j\\ &\quad + 2\alpha h(k,j,j^\star)(x_j-W_{k,j})W_{k,j}\\ &\quad + \alpha^2(t)h^2(k,j,j^\star)(x_j-W_{k,j})^2\}\\ &= -\sum_{m_0}\sum_{k}\sum_{j} h^2(k,j,j^\star)\\ &\quad\cdot \sum_{\mathbf{x}_l\in\mathcal{V}_k} p_l\{\alpha(t)(x_j-W_{k,j})([\mathbf{x}_l]_j-W_{k,j})\\ &\quad + o(\alpha^2(t))\}.\end{aligned}$$

Omitting the higher order terms which vanish for limiting $\alpha(t)$, we arrive at an expression for $\Delta\mathcal{F}$ as

$$\begin{aligned}\Delta\mathcal{F} = &-\alpha(t)\sum_{m_0}\sum_{k}\sum_{j} h^2(k,j,j^\star)\\ &\cdot \sum_{\mathbf{x}_l\in\mathcal{V}_k} p_l([\mathbf{x}_l]_j-W_{k,j})^2.\end{aligned}$$

So the expected value of $\Delta\mathcal{F}$ is

$$\begin{aligned}E(\Delta\mathcal{F}) &= \sum_{\mathbf{x}_l\in\mathcal{V}_k} p_l(\Delta\mathcal{F})\\ &= -\alpha(t)\sum_{m_0}\sum_{k}\sum_{j} h^2(k,j,j^\star)\\ &\quad\cdot \sum_{\mathbf{x}_l\in\mathcal{V}_k} p_l^2([\mathbf{x}_l]_j-W_{k,j})^2\\ &= -\alpha(t)\|\nabla_{W_{k,j}}\mathcal{F}\|^2.\end{aligned}$$

□

Appendix B
Derivation of β Values for SONN

For the CSP: β needs to be chosen so that $V_{1,j^\star}$ or $V_{N,j^\star}$ are not always going to be the least expensive cost potentials. Since

$$\sum_{q\neq j^\star} W_{k,q} \le 1$$

(the weight matrix is feasible each time the cost potential is calculated), then as a worst case, the first or last rows will always win provided

$$V_{1,j^\star}+\beta \le V_{k,j^\star} \quad \text{or} \quad V_{n,j^\star}+\beta \le V_{k,j^\star}$$

for $k \neq 1$ or N. We need to find where $\beta > (\min V_{k,j^\star}) - V_{1,j^\star}$ for $k \neq 1$ and then the first row will not always win, and the oscillations caused by all car models trying to win the first row can be controlled. Now

$$\begin{aligned}\beta &> V_{k,j^\star}-V_{1,j^\star}\\ &= \sum_{i=1}^{N} P_{|k-i|,j^\star}W_{i,j^\star} - \sum_{i=1}^{N} P_{|1-i|,j^\star}W_{i,j^\star}.\end{aligned}$$

Suppose the next cheapest position to sequence car model $j^\star$ (after the first and last positions) is row $k=2$. Then

$$\begin{aligned}\beta > &P_{1,j^\star}(W_{1,j^\star}-W_{2,j^\star}+W_{3,j^\star})\\ &+P_{2,j^\star}(W_{4,j^\star}-W_{3,j^\star})+P_{3,j^\star}(W_{5,j^\star}-W_{4,j^\star})\\ &+\cdots+P_{N-2,j^\star}(W_{N,j^\star}-W_{N-1,j^\star})\\ &-P_{N-1,j^\star}W_{N,j^\star}.\end{aligned}$$

Due to the special structure of the matrix $\mathbf{P}$, i.e., $P_{i+1,j^\star} = P_{i,j^\star}-2$, this condition can be further simplified to

$$\begin{aligned}\beta &> P_{i,j^\star}(W_{1,j^\star}-W_{2,j^\star})+2(W_{3,j^\star}+\cdots+W_{N,j^\star})\\ &> P_{1,j^\star}+2(D_{j^\star}-1).\end{aligned}$$

If $j^\star = 3$ it will be cheapest if the first row (and/or the last row) is the winner, since it is relatively expensive for any separation rule violations. But if $j^\star = 2$ (the next most expensive car model), it would be preferable if the first row did not always win, to enable a model 3 car to occupy the first position in the sequence. We therefore select $\beta > P_{1,2} + 2(D_2 - 1)$.

For the PDN: For the PDN, we need to find the critical value of β at which all N diagonal element of the weight matrix no longer share the p sorting centers evenly (as is the natural tendency of the network with $\beta = 0$). When a postoffice $j^\star$ is presented to the network, it is cheapest to allocate it to a sorting center also at postoffice $j^\star$, since there is no linear contribution to the cost. However, if all the postoffices are selected as sorting centers, the network will converge to a stable nonintegral solution with values of p/N along the diagonal elements of the weight matrix $\mathbf{W}$.

For $k = j^\star$ to be the winning node when presented with postoffice $j^\star$, we require

$$V_{j^\star, j^\star} + \beta \sum_{i \neq j^\star} W_{i,i} < V_{k, j^\star}$$

for all nodes $k \neq j^\star$. Substituting into (34), this condition becomes

$$\gamma^t \left(\sum_i \sum_l V_{j^\star, i} W_{i,l} D_{j^\star, l} \right) + \beta \sum_{i \neq j^\star} W_{i,i} < (\gamma^c o_{j^\star} + \gamma^d d_{j^\star}) D_{j^\star, k} + \gamma^t \left(\sum_i \sum_l V_{j^\star, l} W_{i,l} D_{k,l} \right).$$

If the matrix of volumes is uniform, and the problem becomes one of minimum distance only, then we can make the simplifying assumption that $j^\star$ is near k in Euclidean space, since $j^\star$ should only be considering sorting centers which are nearest in distance. From this assumption, $D_{j^\star, l} \approx D_{k,l}$ and the inequality reduces to

$$\beta \sum_{i \neq j^\star} W_{i,i} < (\gamma^c o_{j^\star} + \gamma^d d_{j^\star}) D_{j^\star, k}$$

for all $k \neq j^\star$. The condition for $k = j^\star$ *not* to necessarily be the winner is therefore

$$\beta \sum_{i \neq j^\star} W_{i,i} \geq (\gamma^c o_{j^\star} + \gamma^d d_{j^\star}) D_{j^\star, k}$$

for some $k \neq j^\star$. Since $p - 1 \leq \sum_{i \neq j^\star} W_{i,i} \leq p$, then $k = j^\star$ will not always be the winner provided

$$\beta > \approx \frac{(\gamma^c o_{j^\star} + \gamma^d d_{j^\star})}{p}$$

for some $k \neq j^\star$.

REFERENCES

[1] S. V. B. Aiyer, "Solving combinatorial optimization problems using neural networks," Cambridge Univ. Eng. Dept., Tech. Rep. CUED/F-INFENG/TR 89, 1991.

[2] B. Angéniol, G. De La Croix, and J.-Y. Le Texier, "Self organizing feature maps and the travelling salesman problem," *Neural Networks,* vol. 1, pp. 289–293, 1988.

[3] R. D. Brandt, Y. Wang, A. J. Laub, and S. K. Mitra, "Alternative networks for solving the travelling salesman problem and the list—Matching problem," in *Proc. Int. Conf. Neural Networks,* 1988, vol. II, pp. 333–340.

[4] A. Brooke, D. Kendrick, and A. Meeraus, *GAMS—A User's Guide.* California: Scientific, 1990.

[5] R. Burkard and U. Fincke, "Probabilistic asymptotic properties of some combinatorial optimization problems," *Discrete Appl. Math.,* vol. 12, pp. 21–29, 1985.

[6] P. Chu, "A neural network for solving optimization problems with linear equality constraints," in *Proc. Int. Joint Conf. Neural Networks,* 1992, vol. II, pp. 272–277.

[7] M. Cottrell and J. C. Fort, "A stochastic model of retinotopy: A self-organizing process," *Biol. Cybern.,* vol. 53, pp. 166–170, 1986.

[8] R. Durbin and D. Willshaw, "An analogue approach to the travelling salesman problem using an elastic net method," *Nature,* vol. 326, pp. 689–691, 1987.

[9] A. Ernst and M. Krishnamoorthy, "Efficient algorithms for the uncapacitated single allocation p-hub median problem," *Location Sci.,* vol. 4, no. 3, pp. 139–154, 1995.

[10] F. Favata and R. Walker, "A study of the application of Kohonen-type neural networks to the travelling salesman problem," *Biol. Cybern.,* vol. 64, pp. 463–468, 1991.

[11] T. A. Feo and J. Gonzalez-Verlarde, "The intermodal trailer assignment problem," Operations Res. Group, Univ. Texas, Austin, Tech. Paper, 1992.

[12] J. C. Fort, "Solving a combinatorial problem via self-organizing process: An application of the Kohonen algorithm to the traveling salesman problem," *Biol. Cybern.,* vol. 59, pp. 33–40, 1988.

[13] A. H. Gee, "Problem solving with optimization networks," Ph.D. dissertation, Queen's College, Cambridge, U.K., 1993.

[14] A. H. Gee and R. W. Prager, "Limitations of neural networks for solving traveling salesman problems," *IEEE Trans. Neural Networks,* vol. 6, pp. 280–282, 1995.

[15] S. T. Hackman, M. Magazine, and T. Wee, "Fast, effective algorithms for simple assembly line balancing problems," *Operations Res.,* vol. 37, no. 6, pp. 916–924, 1989.

[16] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in *Proc. Nat. Academy Sci.,* 1982, vol. 79, pp. 2554–2558.

[17] ——, "Neurons with graded response have collective computational properties like those of two-state neurons," in *Proc. Nat. Academy Sci.,* 1984, vol. 81, pp. 3088–3092.

[18] J. J. Hopfield and D. W. Tank, "'Neural computation of decisions in optimization problems," *Biol. Cybern.,* vol. 52, pp. 141–152, 1985.

[19] B. Kamgar-Parsi and B. Kamgar-Parsi, "Dynamical stability and parameter selection in neural optimization," in *Proc. Int. Joint Conf. Neural Networks,* 1992, vol. IV, pp. 566–571.

[20] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science,* vol. 220, pp. 671–680, 1983.

[21] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.,* vol. 43, pp. 59–69, 1982.

[22] M. Krishnamoorthy, G. Mills, and D. Sier, "Strategic configuration of the mail processing network: Location-allocation modeling stage-1," CSIRO Tech. Rep. DMS-C94/9, 1994.

[23] W. K. Lai and G. G. Coghill, "Genetic breeding of control parameters for the Hopfield/Tank neural net," in *Proc. Int. Joint Conf. Neural Networks,* 1992, vol. IV, pp. 618–623.

[24] S. T. McCormick, M. Pinedo, S. Shenker, and B. Wolf, "Sequencing in an assembly line with blocking to minimize cycle time," *Operations Res.,* vol. 37, no. 6, pp. 925–935, 1989.

[25] E. Oja and J. Lampinen, "Unsupervised learning for feature extraction," in *Computational Intelligence: Imitating Life,* J. Zurada, R. J. Marks, II, and C. Robinson, Eds. Piscataway, NJ: IEEE Press, 1994, pp. 13–22.

[26] M. E. O'Kelly, "A quadratic integer program for the location of interacting hub facilities," *European J. Operational Res.,* vol. 32, pp. 393–404, 1987.

[27] B. D. Parretto, W. Kabat, and L. Wos, "Jobshop scheduling using automated reasoning: A case study of the car sequencing problem," *J. Automated Reasoning,* vol. 2, pp. 1–42, 1986.

[28] H. Ritter and K. Schulten, "Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability, and dimension selection," *Biol. Cybern.,* vol. 60, pp. 59–71, 1988.

[29] ——, "Kohonen's self-organizing maps: Exploring their computational capabilities," in *Proc. Int. Conf. Neural Networks,* 1988, vol. 1, pp. 109–116.

[30] H. D. Sherali and E. L. Brown, "A quadratic partial assignment and packing model and algorithm for the airline gate assignment problem," in *Quadratic Assignment and Related Problems,* P. M. Pardalos and H. Wolkowicz, Eds. Providence, RI: American Math. Soc., 1993.

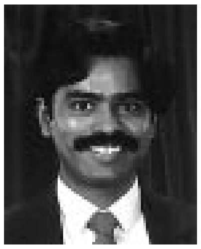
[31] K. Smith, M. Palaniswami, and M. Krishnamoorthy, "Traditional heuristic versus Hopfield neural-network approaches to a car sequencing problem," *European J. Operational Res.,* vol. 93, no. 2, pp. 300–316, 1996.

[32] ——, "A hybrid neural approach to combinatorial optimization," *Comput. Operations Res.*, vol. 23, no. 6, pp. 597–610, 1996.
[33] K. Smith, "An argument for abandoning the traveling salesman problem as a neural-network benchmark," *IEEE Trans. Neural Networks*, vol. 7, pp. 1542–1544, 1996.
[34] K. Smith and M. Palaniswami, "Static and dynamic channel assignment using neural networks," *IEEE J. Select. Areas Commun.*, vol. 15, no. 2, pp. 238–249, 1997.
[35] K. Smith, M. Krishnamoorthy, and M. Palaniswami, "Neural versus traditional approaches to the location of interacting hub facilities," *Location Sci.*, vol. 4, no. 3, pp. 155–171, 1996.
[36] Y. Takefuji, *Neural-Network Parallel Computing.* Boston, MA: Kluwers, 1992.
[37] D. E. Van den Bout and T. K. Miller III, "A travelling salesman objective function that works," in *Proc. Int. Conf. Neural Networks*, 1988, vol. II, pp. 299-303.
[38] G. V. Wilson and G. S. Pawley, "On the stability of the TSP algorithm of Hopfield and Tank," *Biol. Cybern.*, vol. 58, pp. 63–70, 1988.

Kate Smith (S'93–M'96) received the Bachelor of Science (Honors) and Ph.D. degrees from the University of Melbourne, Victoria, Australia, in 1993 and 1996, respectively.

She is currently a Lecturer in the Department of Business Systems, Monash University, Australia. Her research interests include neural networks, combinatorial optimization, communications systems, data mining, and applications of new techniques to business and industry problems. She has published more than 30 papers in international journals and conference proceedings.

Dr. Smith is on the organizing committee of several international conferences on neural networks. She is a guest editor of a special issue of *Computers and Operations Research* on neural networks in business, to be published in 1999.



Marimuthu Palaniswami (S'84–M'85–SM'94) received the B.E. (Hons) degree from the University of Madras, India, the M.E. degree from the Indian Institute of Science, and the Ph.D. degree from the University of Newcastle, Australia.

He is an Associate Professor at the University of Melbourne, Australia. His research interests are in the fields of computational intelligence, nonlinear dynamics, computer vision, intelligent control and bio-medical engineering. He has published more than 150 conference and journal papers on these topics. He has completed several industry-sponsored projects for National Australia Bank, Broken Hill Propriety Limited, Defence Science and Technology Organization, Integrated Control Systems Pty Ltd., and Signal Processing Associates Pty Ltd. He has a number of collaborative projects with international research institutions such as Curtin University of Technology, Florida International University, Indian Institute of Science, CSIRO, Nanyang Technological University and the Communications Research Laboratory, Japan.

Dr. Palaniswami was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and is on the editorial board of two other computing journals. He served as a Technical Program Cochair for the IEEE International Conference on Neural Networks, 1995 and was on the program committees of a number of conferences including several IEEE Conferences on Neural Networks, IEEE Workshop on Emerging Technologies and Factory Automation, Australian Conferences on Neural Networks, and IEEE Australia–New Zealand Conferences on Intelligent Information Processing Systems. He also received several ARC's, APA(I)s, ATERB's, DITARD, and DIST grants for both fundamental and applied projcts. He was also the recipient of Foreign Specialist Award from the Ministry of Education, Japan.

Mohan Krishnamoorthy received the M.Sc. degree in operations research from Delhi University, India, the M.Sc. degree in management science from Imperial College, London, U.K., and the Ph.D. degree in operations research from Imperial College, London, U.K. He also received the Diploma degree from Imperial College (DIC).

He taught Operations Research for a year at the University of Kent, Canterbury, U.K. He joined the OR group of CSIRO in January 1992. He has published in Australian and international journals as well as in conference proceedings. At CSIRO, he undertakes tactical research into operations research (OR) problems faced by industrial clients. He also carries out strategic research into generic OR problems and their solution methodologies. He has carried out several consultancy projects for industrial clients including Dampier Salt (Western Australia), Dynamic Transport Management (Melbourne), Aurora Vehicles (Ford), RACV (Melbourne), Australia Post (Sydney) and The Operations Research Group (Sydney), Cathay Pacific (Hong Kong), The Preston Group (Melbourne), The Department of Corrective Services (Sydney, NSW), Dalrymple Bay Coal Terminal and The National Rail Corporation. His research interests lie mainly in the design, development, analysis, and computational testing of (exact, heuristic, and novel solution) algorithms for graph, network, and combinatorial optimization problems, as well as complex routing and scheduling problems, man power scheduling, rostering, the location of spatially interacting facilities, vehicle dispatch problems, constrained spanning trees, neural networks, and a variety of operations research problems faced by the airline industry.