A Theoretical Investigation into the Performance of the Hopfield Model

SREERAM V. B. AIYER, MAHESAN NIRANJAN, AND FRANK FALLSIDE

Abstract—**This paper analyzes the behavior of the Hopfield model as a content addressable memory (CAM) and as a method of solving the traveling salesman problem (TSP). The analysis is based on the geometry of the subspace set up by the degenerate eigenvalues of the connection matrix. The dynamic equation is shown to be equivalent to a projection of the input vector onto this subspace. In the case of content addressable memory, it is shown that spurious fixed points can occur at any corner of the hypercube that is on, or near, the subspace spanned by the memory vectors. There then is an analysis of why the network can frequently converge to an invalid solution when applied to the traveling salesman problem. From this geometric view, analytic expressions are derived for the constant terms used in constructing the traveling salesman problem energy function. With these expressions, the network can be made robust, and can reliably solve the traveling salesman problem with tour sizes of 50 cities or more.**

I. Introduction

THE autoassociative memory model proposed by Hopfield [2] has attracted considerable interest both as a content addressable memory (CAM) and, more interestingly, as a method of solving difficult optimization problems. The model is a network of simple multi-input, nonlinear, analog processing elements. The claim by Hopfield and subsequent researchers is that with proper programming of the parameters, the dynamics of such a network can have useful computational properties.

Many researchers have studied the properties of Hopfield networks both as CAM's [1], [4]–[11] and for solving optimization problems [3], [9], [12]–[15]. In both cases it has generally been found that although the networks do produce some useful results, they suffer from significant drawbacks. These include a large number of spurious stable points in the case of CAM, and a high percentage of invalid solutions for the case of the traveling salesman problem (TSP).

This paper analyzes the degenerate eigenvalues of the model's connection matrix and the geometry of the corresponding subspaces. The dynamics of the Hopfield network are explained and it is shown how it is possible to account for the problems encountered by the researchers mentioned above, as well as demonstrating (where possible) how to overcome them. Lastly, the knowledge of the network dynamics is used in order to optimize the key parameters used in the design of Hopfield networks both as CAM's and as a method of solving the traveling salesman problem.

Manuscript received September 15, 1989; revised February 19, 1990.

The authors are with the Cambridge University Engineering Department, Cambridge CB2 1PZ, England.

IEEE Log Number 9035246.

II. The Hopfield Model

The Hopfield network is constructed by connecting a large number of simple processing elements to each other. In general, the ith processing element or neuron is described by two variables: its current state or "mean soma potential" [2] denoted by u_i, and its output denoted by v_i. The output is usually related to the state by a simple nondecreasing monotonic output function $g(u_i)$. This function is normally designed to limit the possible values of v_i to the range -1 to $+1$: hence the function will be nonlinear. For simplicity, $g(u_i)$ is frequently a step function or a hyperbolic tangent.

The output of the ith neuron is fed to the input of the jth neuron by a connection of strength T_{ij}. In addition, each neuron has an offset bias of i_i^b fed to its input. The state of the ith neuron u_i is updated by a function of the net total input to the neuron, the exact nature of this function varying according to whether a continuous or discrete model is being used. In the rest of this paper, the states of the neurons will be collectively denoted by the vector $\boldsymbol{u}$, the outputs by the vector $\boldsymbol{v}$, the connection strengths by the matrix $\boldsymbol{T}$, and the offset biases by the vector $\boldsymbol{i}^b$.

There are two variants of the Hopfield model:

Version (a): The discrete version where the value of the state vector at time step $n + 1$ is related the neuron output vector at time step n by the equation

$$\boldsymbol{u}^{(n+1)} = \boldsymbol{T}\boldsymbol{v}^{(n)} + \boldsymbol{i}^b \tag{1}$$

and the output function $g(u_i)$ is usually a step function of the form

$$g(u_i^{(n+1)}) = \text{sign}\,(u_i^{(n+1)}).$$

Version (b): The continuous version where the behavior of the network is described by the differential equation

$$\dot{\boldsymbol{u}} = -\frac{\boldsymbol{u}}{\tau} + \boldsymbol{T}\boldsymbol{v} + \boldsymbol{i}^b \tag{2}$$

and the output function $g(u_i)$ is a hyperbolic tangent

$$g(u_i) = \tanh\left(\frac{u_i}{u_0}\right). \tag{3}$$

If the elements of $\boldsymbol{u}$ are updated asynchronously (the element to be updated being selected randomly) and the connection matrix is symmetric, then Hopfield [3] shows that version (a) has a Liapunov function of the form

$$E = -\tfrac{1}{2}\boldsymbol{v}^t \boldsymbol{T}\boldsymbol{v} - (\boldsymbol{i}^b)^t \boldsymbol{v}. \tag{4}$$

For version (b) he showed with either synchronous or asynchronous update, the following Liapunov function exists:

$$E = -\frac{1}{2}\boldsymbol{v}^t \boldsymbol{T}\boldsymbol{v} - (\boldsymbol{i}^b)^t \boldsymbol{v} + \frac{1}{\tau}\sum_{i=1}^{N}\int_0^{v_i} g^{-1}(x)\,dx. \tag{5}$$

Further, he argues that for the continuous, version (b), nonlinearity $g(x)$, then in the high-gain limit (i.e., $u_0 \to 0$ and $g(x)$ asymptotically approaches a step function), the Liapunov function of version (b) is approximately the same as that of (a).

A. Analysis of Hopfield Liapunov Functions

The basis of the analysis in the rest of this paper is an understanding of the relationship between the fixed points of the network, the Hopfield Liapunov functions, and the eigenvalues of the connection matrix $\boldsymbol{T}$.

For simplicity, consider the Liapunov function (4) and the discrete update equation (1) without the $\boldsymbol{i}^b$ term:

$$E = -\tfrac{1}{2}\boldsymbol{v}^t \boldsymbol{T}\boldsymbol{v} \quad \text{and } \boldsymbol{u} = \boldsymbol{T}\boldsymbol{v} \text{ with } v_i^{(n+1)} = \text{sign}\,(u_i^{(n)}).$$

Since $\boldsymbol{T}$ is symmetric (in order for the Liapunov function to be valid), it can be completely characterized by its eigenvalues and corresponding *orthogonal* eigenvectors. Let these be denoted

$$\lambda_1 \cdots \lambda_M \quad \text{and } \boldsymbol{e}^1 \cdots \boldsymbol{e}^M.$$

The eigenvalues may be degenerate in which case instead of a corresponding eigenvector there is a corresponding subspace. Further $\boldsymbol{T}$ may have a degenerate eigenvalue of zero with a corresponding subspace termed the null subspace. The vector $\boldsymbol{v}$ can be written in terms of its component $\boldsymbol{v}^i$ in the direction of the eigenvector $\boldsymbol{e}^i$ plus its component $\boldsymbol{q}$ in the null subspace as follows:

$$\boldsymbol{v} = \sum_{i=1}^{M} \boldsymbol{v}^i + \boldsymbol{q} \tag{6}$$

where

$$\boldsymbol{v}.\boldsymbol{e}^i = \boldsymbol{v}^i = \gamma_i \boldsymbol{e}^i. \tag{7}$$

Similarly, the connection matrix, Liapunov function, and dynamic update equation can be expressed as follows:

$$\boldsymbol{T} = \sum_{i=1}^{M} \lambda_i \boldsymbol{e}^i \boldsymbol{e}^{i^t} \tag{8}$$

$$E = -\tfrac{1}{2}\sum_{i=1}^{M} \lambda_i |\boldsymbol{v}^i|^2 = -\tfrac{1}{2}\sum_{i=1}^{M} \lambda_i \gamma_i^2 \tag{9}$$

$$\boldsymbol{u} = \boldsymbol{T}\boldsymbol{v} = \sum_{i=1}^{M} \lambda_i \boldsymbol{v}^i. \tag{10}$$

It can be now seen that to minimize E the network must move $\boldsymbol{v}$ so as to: 1) reduce to zero magnitude all $\boldsymbol{v}^i$'s where $\lambda_i < 0$; and 2) increase the magnitude of all $\boldsymbol{v}^j$'s where $\lambda_j > 0$.

With no bounds placed on the magnitude of $\boldsymbol{v}$, the network would indefinitely increase $\boldsymbol{v}$'s magnitude in the direction of the positive λ_j's, gradually favoring the larger positive λ_j's. However, as in the Hopfield model, $\boldsymbol{v}$ is usually limited to the unit hypercube. In this case E is minimized when $\boldsymbol{v} = \gamma_{\max}\boldsymbol{e}^{\max}$ where $\boldsymbol{e}^{\max}$ is the eigenvector corresponding to the largest positive eigenvalue $\lambda_{\max}$. As will be seen later, if there is only one positive eigenvalue which is multiply degenerate, then E is minimized when $\boldsymbol{v}$ lies wholly in the corresponding subspace. Hence, the whole subspace must be considered as the location of the minimum of E.

III. Content Addressable Memory

A CAM is defined as a memory in which the stored patterns are recalled by presenting the memory with a partial form of the stored patterns. Hopfield [3] sets up an N neuron model in which the stored patterns are a series of M memory vectors $\boldsymbol{m}^\alpha$ of dimensionality N. The memory vectors represent a binary pattern, so each element of the $\boldsymbol{m}^\alpha$ takes the value ± 1. The capacity of the model (i.e., the relationship between the number of storable memories and the size of the network) has been widely studied both experimentally [1] and from a probabilistic/information theoretic point of view [4]–[6], and a number of useful expressions for the worst and average case capacities with various formulations of $\boldsymbol{T}$ have been derived. All these studies have reported the existence of spurious stable states which act as a severe limitation on the capacity and error correcting abilities of the model. The intention of this part on CAM is not so much to derive new expressions for capacity, as to use the Liapunov function analysis given above as a basis for understanding the nature of the spurious points and dynamics of the network.

A. Relationship between CAM Learning Rule and Eigenvalues of $\boldsymbol{T}$

The learning rule specifies how the network parameters, primarily the connection matrix, are set according to the memory vectors which are to be stored. The essential feature of any such learning rule is that the original stored memory vectors must be stable states of the network. A further desirable property is that these memory stable states should have a radius of attraction. Initial states of the network which lie within this radius should then be corrected to the corresponding stored memory vector. If $\boldsymbol{v}$ is a memory vector and if the first condition that it must be a stable state is to be satisfied, then from (10) the following must be true

$$\begin{aligned} v_k &= \text{sign}\,(u_k), \qquad k = 1 \cdots N \\ &= \text{sign}\left(\sum_{i=1}^{M} \lambda_i v_k^i\right) \end{aligned}$$

$$\text{using (6)} \Rightarrow \sum_{i=1}^{M} v_k^i + q_k = \text{sign}\left(\sum_{i=1}^{M} \lambda_i v_k^i\right). \tag{11}$$

The only way that (11) can be guaranteed for any set of memory vectors is if

$$\boldsymbol{q} = 0, \quad \lambda_i = \lambda \quad \text{for } i = 1 \cdots M \text{ with } \lambda > 0.$$

In other words:

1) To ensure $\boldsymbol{q} = 0$ the null subspace must be orthogonal to all the memory vectors.

2) As a result, all the memory vectors must be completely specified by $\Sigma_{i=1}^{M} \boldsymbol{v}^i$, hence the eigenvectors of $\boldsymbol{T}$ must at least span the subspace spanned by the memory vectors. This will automatically ensure that 1) is true.

3) So that $\lambda_i = \lambda$ for $i = 1 \cdots M$, the connection matrix must have a single positive degenerate eigenvalue corresponding to the memory vector subspace.

The simplest learning rule that satisfies these conditions is the outer product rule. Let $\boldsymbol{T}^M$ be the matrix formed by this rule. Thus

$$\boldsymbol{T}^M = \sum_{\alpha=1}^{M} \boldsymbol{m}^\alpha \boldsymbol{m}^{\alpha'}. \tag{12}$$

Let $\mathfrak{M}$ denote the subspace spanned by the memory vectors. Then the complementary subspace to this will be the null subspace of $\boldsymbol{T}^M$. Let this be denoted $\mathfrak{Q}$. The vector $\boldsymbol{v}$ can be written in terms of its component in $\mathfrak{M}$, denoted $\tilde{\boldsymbol{m}}$, and its component in $\mathfrak{Q}$, $\boldsymbol{q}$

$$\boldsymbol{v} = \tilde{\boldsymbol{m}} + \boldsymbol{q} \qquad \text{where } \tilde{\boldsymbol{m}}.\boldsymbol{q} = 0.$$

If the $\boldsymbol{m}^\alpha$'s are assumed to be linearly independent, then $\tilde{\boldsymbol{m}}$ can written as a linear combination of them

$$\tilde{\boldsymbol{m}} = \sum_{\alpha=1}^{M} \gamma_\alpha \boldsymbol{m}^\alpha.$$

Consider now the value of $\boldsymbol{T}^M\boldsymbol{v}$

$$\begin{aligned}
\boldsymbol{T}^M\boldsymbol{v} &= \boldsymbol{T}^M(\tilde{\boldsymbol{m}} + \boldsymbol{q})\\
&= \left(\sum_{\alpha=1}^{M} \boldsymbol{m}^\alpha \boldsymbol{m}^{\alpha'}\right)\left(\sum_{\beta=1}^{M} \gamma_\beta \boldsymbol{m}^\beta + \boldsymbol{q}\right)\\
&= \sum_{\alpha=1}^{M} \boldsymbol{m}^\alpha \boldsymbol{m}^{\alpha'} \gamma_\alpha \boldsymbol{m}^\alpha + \sum_{\alpha=1}^{M} \sum_{\beta \neq \alpha}^{M} \boldsymbol{m}^\alpha \boldsymbol{m}^{\alpha'} \gamma_\beta \boldsymbol{m}^\beta\\
&= N \sum_{\alpha=1}^{M} \gamma_\alpha \boldsymbol{m}^\alpha + \sum_{\alpha=1}^{M} \sum_{\beta \neq \alpha}^{M} \boldsymbol{m}^\alpha \boldsymbol{m}^{\alpha'} \gamma_\beta \boldsymbol{m}^\beta\\
&= N\tilde{\boldsymbol{m}} + \boldsymbol{m}^{\text{noise}}.
\end{aligned} \tag{13}$$

Clearly for $\boldsymbol{T}^M$ to have a single degenerate eigenvalue of N in the $\mathfrak{M}$ subspace, $\boldsymbol{m}^{\text{noise}}$ must be zero. This is only true if the $\boldsymbol{m}^\alpha$'s are orthogonal. Hence the simple outer product summation will only satisfy the above conditions 1)-3) with orthogonal memory vectors. Several researchers [4]-[6] have argued that if N is much larger than M and the $\boldsymbol{m}^\alpha$'s are chosen randomly, then there is a high probability that the $\boldsymbol{m}^\alpha$'s will be nearly orthogonal. In this case the values of the λ_i's will be similar enough for conditions 1)-3) to hold (in a probabilistic sense).

Alternatively, a more complex learning rule can be chosen which ensures that $\boldsymbol{T}^M$ has a single degenerate eigenvalue regardless of whether the memory vectors are orthogonal. Such "spectral schemes" have been explored by Venkatesh and Psaltis [5] and Dembo [6]. As an example consider

$$\boldsymbol{T}^M = \boldsymbol{Q}(\boldsymbol{Q}'\boldsymbol{Q})^{-1}\boldsymbol{Q}' \qquad \text{where } \boldsymbol{Q} = [\boldsymbol{m}^1, \boldsymbol{m}^2 \cdots \boldsymbol{m}^M].$$

With this rule, $\boldsymbol{T}^M\tilde{\boldsymbol{m}} = \tilde{\boldsymbol{m}}$ as long as the $\boldsymbol{m}^\alpha$'s are linearly independent. Unfortunately, this is achieved at the expense of a huge increase in computational complexity, a problem which plagues all the alternatives to the outer product rule.

B. Content Addressability and Spurious Stable States

For the Hopfield model to function as a CAM it must be able to recover an original memory vector when presented with a probe vector close to it (usually in terms of Hamming distance). If the probe vector is considered as a corrupted version of the original memory vector, then it is possible to view the network's operation as a form of error correction. In the analysis of the Liapunov functions presented before, it was shown that the dynamics of network will reduce to zero the components of $\boldsymbol{v}$ that lie in subspaces spanned by the eigenvectors of $\boldsymbol{T}$ that have a corresponding negative eigenvalue. With the formulation of $\boldsymbol{T}^M$ given previously $\mathfrak{Q}$ corresponds to the null subspace of $\boldsymbol{T}^M$ and hence has a corresponding eigenvalue of zero. Thus if a negative eigenvalue is introduced into this subspace, the network will be able to perform a limited type of error correction by removing the $\mathfrak{Q}$ component from any initial probe vector. Hence, errors which result in the corruption of a memory vector by the addition of a small $\mathfrak{Q}$ component should be correctable by this technique.

The easiest way to introduce a negative eigenvalue for this subspace is simply to subtract a multiple of the identity matrix $\boldsymbol{I}$ from $\boldsymbol{T}^M$. If the outer product rule has been used to form $\boldsymbol{T}^M$ then the value of the elements on the leading diagonal of $\boldsymbol{T}^M$ will be equal to the number of memory vectors stored (M). Therefore it is usual to subtract $M\boldsymbol{I}$ from $\boldsymbol{T}^M$ in order to set the leading diagonal to zero, which is considered an advantage because it corresponds to the network having no self connections. So now

$$\boldsymbol{T} = \boldsymbol{T}^M - M\boldsymbol{I}$$

$$(13) \Rightarrow \boldsymbol{T}\boldsymbol{v} = (N - M)\tilde{\boldsymbol{m}} + \boldsymbol{m}^{\text{noise}} - M\boldsymbol{q} \tag{14}$$

and the Liapunov function becomes

$$\begin{aligned}
E &= -\tfrac{1}{2}\boldsymbol{v}'\boldsymbol{T}\boldsymbol{v}\\
&= -\tfrac{1}{2}(\tilde{\boldsymbol{m}} + \boldsymbol{q})((N - M)\tilde{\boldsymbol{m}} + \boldsymbol{m}^{\text{noise}} - M\boldsymbol{q})\\
&= -\tfrac{1}{2}(N - M)|\tilde{\boldsymbol{m}}|^2 - \tfrac{1}{2}\tilde{\boldsymbol{m}}'\boldsymbol{m}^{\text{noise}} + \tfrac{1}{2}M|\boldsymbol{q}|^2.
\end{aligned} \tag{15}$$

Since an error correcting ability implies that at minimum an original memory will be corrected to itself, it must be assumed that the memory vectors are stable states. As mentioned before, this can be achieved by either choosing orthogonal memory vectors or using one of the spectral schemes for setting $\boldsymbol{T}$. Either way, the effect is to ensure that the $\boldsymbol{m}^{\text{noise}}$ term is negligible. Hence, with this assumption (15) simplifies to

$$E = -\tfrac{1}{2}(N - M)|\tilde{\boldsymbol{m}}|^2 + \tfrac{1}{2}M|\boldsymbol{q}|^2.$$

Clearly E is minimized if $|\boldsymbol{q}| = 0$ and $|\tilde{\boldsymbol{m}}| \rightarrow \infty$. The effect of the nonlinear output function is to counteract the tendency to send $\tilde{\boldsymbol{m}}$ to ∞ and force the network to stabilize at a hypercube corner. Thus allowing for the nonlinear effects of forcing $\boldsymbol{v}$ to point at a hypercube corner, the Hopfield network can essentially be viewed as a vector projector, i.e., it projects it probe vector into $\mathfrak{M}$ and then moves the resulting projected vector to the nearest hypercube corner.

It is now possible to account for the majority of spurious stable points. They are simply: a) hypercube corners, other than the original memories, that lie in $\mathfrak{M}$. For such points $|\boldsymbol{q}| << |\tilde{\boldsymbol{m}}|$ and since by (14) sign $(\Sigma_l T_{kl}\tilde{m}_l)$ = sign $(\tilde{m}_l)$, they will be stable. b) Hypercube corners, which although not exactly in $\mathfrak{M}$ are close enough for sign (v_l) = sign $(\tilde{m}_l)$, $l = 1 \cdots N$. In other words, the projection of hypercube corner $\boldsymbol{v}$ into $\mathfrak{M}$ does not alter the signs of the elements.

Although the number of points satisfying the strict condition a) above will be quite small, the number satisfying the less strict condition b) increases very rapidly with the number of stored memory vectors (i.e., as the dimensionality of $\mathfrak{M}$ increases).

C. Conclusions on the Use of a Hopfield Network as a CAM

Starting from the assumption that the connection matrix $\boldsymbol{T}$ is symmetric, we have shown that there is an inherent tradeoff between the need to preserve the memory vectors as stable points and the number of spurious stable points. The need for stable memory vectors necessitates that $\boldsymbol{T}$ has a degenerate eigenvalue corresponding to the subspace spanned by the memory vectors $\mathfrak{M}$. In turn this causes all the other hypercube corners that lie in or near $\mathfrak{M}$ to be spurious stable points. Further, so that the network has some error correction ability, $\boldsymbol{T}$ needs a negative eigenvalue in the subspace orthogonal to $\mathfrak{M}$. This is introduced through setting the elements of the leading diagonal of $\boldsymbol{T}$ to zero. These factors together imply that the Hopfield network essentially functions by vector projection, i.e., it projects its probe vector into $\mathfrak{M}$ and then moves the resulting projected vector to the nearest hypercube corner. As other researchers [1], [4], [7] have pointed out (though for differing reasons) the relative number of spurious points decreases as the dimensionality of the memory vectors increases with respect to the number of stored vectors. Eventually, a point is reached where there are relatively so few within a certain Hamming radius of each original memory vector that it becomes valid to consider each memory as having a fixed radius of convergence. For this reason we conclude that the Hopfield network can act as a CAM, but only in the limit where there are so few spurious states that its operation as a CAM is indistinguishable from its true operation as a vector projector.

IV. Traveling Salesman Problem

This part is an analysis of the Hopfield model as a method of solving the traveling salesman problem. This topic has been the subject of much research, with the main thrust being to overcome a key problem: that even though such networks can produce valid solutions to the TSP, on most occasions the final stable state of the network is invalid (Wilson and Pawley [12]). Kahng [13] gives a figure of 85% invalid output for 10 city tours based on random city positions in a unit square. Abe [9] completely reformulates the energy function, while Tagliarini and Page [14] use extra terms in it to try and force valid solutions. Others have proposed a range of heuristics such as renormalization to eliminate any asymmetry in the distribution of city locations (Van den Bout and Miller [15]) and "nearest neighbor skewing" (Kahng [13]).

Following in the lines of the Liapunov analysis given at the beginning of this paper, we first completely determine the eigenvalues of the connection matrix. (Abe [9] also uses an eigenvalue based approach in his analysis of the Hopfield network. However, he is mainly concerned with convergence per se, rather than convergence to a valid or invalid solution, which is the primary concern of this work.) We use the knowledge of the eigenvalues to gain an understanding of the relationship between the network parameters and the type of solution the network reaches. Using this understanding, we explain many of the difficulties encountered by the above researchers, and also show how to overcome them, while keeping Hopfield and Tank's [3] formulation of the TSP basically intact. To substantiate these claims we will show analytically why Hopfield and Tank's [3] experimentally derived values for the arbitrary constants in their energy function allow the network to produce valid solutions. Finally, we will extend the argument to demonstrate how to compute the values for these arbitrary constants in a way which our results indicate can ensure that the network produces 100% valid solutions even for 50 city TSP's.

A. A Brief Description of the Traveling Salesman Problem

The traveling salesman problem consists of finding the shortest closed path by which every city out of a set of N cities is visited once and only once. For example, in a four city problem, if the cities are labelled A, B, C, and D, then a possible tour starting at D would be $DACBD$. If the distance between two cities X and Y is d_{XY}, then the total path length of the tour would be $d_{DA} + d_{AC} + d_{CB} + d_{BD}$. This problem is NP complete (Hopcroft and Ullman [16, p. 341]) and the only known certain method of finding the shortest path is to search through every single possible path, of which there are $N!/2N$. It is possible to use some heuristics to make this number more tractable. For example, if the cities all lie on a plane then the optimum tour nearly always connects a city to one of its closest four neighbors [3, sec. VI]. Nevertheless, the number of computations still remains formidable, and a heuristic relying on the cities being on a plane is of no help in the general case.

B. Formulating TSP as a Problem Solvable by a Hopfield Network

Hopfield and Tank use N^2 neurons to represent a tour taken to visit N cities. The N^2 neurons are grouped into N groups of N neurons. Each group of N neurons is used to represent the position in the tour of a particular city. In the four city tour given earlier (*DACBD*), the neuron outputs would be represented by the vector

$$\boldsymbol{v} = (0\ 1\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 0\ 0\ 0)$$

$$\text{or in 2D form } \begin{array}{c} \uparrow \\ x \\ \downarrow \end{array} \overset{\leftarrow\, i\, \rightarrow}{\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}}.$$

The first group of four neurons corresponds to city A and 0 1 0 0 means A is the second city to be visited. The second group corresponds to city B and 0 0 0 1 means B is the fourth city to be visited. In general, if $\boldsymbol{v}$ is the output vector, then v_{xi} picks out the ith element of the xth group. This element will be 1 (i.e., the neuron is on) if city x comes in position i in the tour, otherwise it is 0 (i.e., the neuron is off).

N.B This notation can be confusing because v_{xi} represents the element of a vector and not a matrix. To be strictly correct, v_{xi} should be written v_{Nx+i}. However, in order to be consistent with Hopfield's [3] notation, v_{xi}, u_{xi}, and $T_{xi,yj}$ will be used to represent the elements of the vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ and the matrix $\boldsymbol{T}$.

Using this representation it is easy to see that the conditions on the N^2 neurons that are needed to ensure that $\boldsymbol{v}$ corresponds to a valid tour are: C1) There must be one and only one neuron on in each row of v_{xi}; and C2) there must be one and only one neuron on in each column of v_{xi}.

As with CAM, the operation of the Hopfield network is given by the differential equation

$$\dot{\boldsymbol{u}} = -\frac{\boldsymbol{u}}{\tau} + \boldsymbol{Tv} + \boldsymbol{i}^b \qquad \text{where } v_{xi} = g(u_{xi}) \quad (16)$$

and $g(u_{xi})$ has the form $g(u_{xi}) = \frac{1}{2}(1 + \tanh(u_{xi}/u_0))$.

Again, as with CAM the object is to find a $\boldsymbol{T}$ such that the location ($\boldsymbol{v}_{\min}$) of the minima of the energy equation

$$E = -\frac{1}{2}\boldsymbol{vTv} - (\boldsymbol{i}^b)^t\boldsymbol{v} + \frac{1}{\tau}\sum_{i=1}^{N}\int_0^{v_{xi}} g^{-1}(\alpha_{xi})\, d\alpha_{xi} \tag{17}$$

correspond to valid tours of the TSP, i.e., $\boldsymbol{v}_{\min}$ satisfies C1) and C2). In addition, $\boldsymbol{T}$ must also be constructed so that the $\boldsymbol{v}_{\min}$ are not only valid solutions but also good ones in the sense that they represent optimal path lengths.

The energy equation that Hopfield and Tank give to satisfy all these constraints is:

$$\begin{aligned} E = {} & \frac{A}{2}\sum_x\sum_i\sum_{j\neq i} v_{xi}v_{xj} + \frac{B}{2}\sum_i\sum_x\sum_{y\neq x} v_{xi}v_{yi} \\ & + \frac{C}{2}\left(\sum_x\sum_i v_{xi} - N\right)^2 \\ & + \frac{D}{2}\sum_x\sum_{y\neq x}\sum_i d_{xy}v_{xi}(v_{y,i+1} + v_{y,i-1}) \end{aligned}$$

The $i + 1$ and $i - 1$ subscripts are given modulo N. (18)

- The A term equals 0 if C1) is satisfied.
- The B term equals 0 if C2) is satisfied.
- The C term equals 0 if $\Sigma_x \Sigma_i v_{xi} = N$. In the case that $\boldsymbol{v}$ is at the corner of the hypercube, this corresponds to there being exactly N neurons on.
- The D term measures the total distance of the tour to which $\boldsymbol{v}$ corresponds (assuming $\boldsymbol{v}$ is a valid tour).

In order to obtain the above energy equation, the elements of the connection matrix $\boldsymbol{T}$ and the excitation $\boldsymbol{i}^b$ are given by

$$\begin{aligned} T_{xi,yj} = {} & -A\delta_{xy}(1 - \delta_{ij}) - B\delta_{ij}(1 - \delta_{xy}) \\ & -C \\ & -D\, d_{xy}(\delta_{j,i+1} + \delta_{j,i-1}) \\ & [\delta_{ij} = 1 \quad \text{if } i = j \text{ and is 0 otherwise}] \end{aligned} \tag{19}$$

and for the excitation $\boldsymbol{i}^b$

$$i_{xi}^b = +CN. \tag{20}$$

The C term is required because the A and B terms equal 0 not only if there is just a single neuron on per row/column, but also if there are no neurons on. A trivial case is where $\boldsymbol{v} = 0$, in which case both the A and B terms are 0. To compensate for this effect the C term is used to ensure that the network settles down with exactly N neurons being on.

C. Determination of the Eigenvalues of the Connection Matrix

In order to describe the dynamics of the network, the first step is to determine the eigenvalues and corresponding eigenvectors/subspaces of the connection matrix. Initially, the eigenvalues of the connection matrix formed without the D term will be calculated, because the first three terms of (18) are responsible for ensuring that the network settles down on a valid solution. It will also be assumed that $A = B$ so that the constraints C1) and C2) have equal weight.

Assuming $A = B$ and setting $D = 0$ reduces $\boldsymbol{T}$ to

$$T_{xi,yj} = -A[\delta_{xy}(1 - \delta_{ij})] - A[\delta_{ij}(1 - \delta_{xy})] - C. \tag{21}$$

This connection matrix can be shown to have three distinct eigenvalues, as follows:

$\lambda_1 = -CN^2 - 2A(N - 1)$
 This has a corresponding eigenvector in the direction $\boldsymbol{e}^1 = (1, 1, \cdots 1)'$. Let the normalized eigenvector be $\hat{\boldsymbol{e}}^1$ where

$$\hat{\boldsymbol{e}}^1 = \frac{1}{N}(1, 1, \cdots 1)'.$$

$\lambda_2 = 2A$
 This is degenerate and corresponds to a subspace which we call the valid subspace. Each valid solution to the TSP (i.e., the valid forms of $\boldsymbol{v}$) can be decomposed into a component in this subspace and a component in the direction of $\boldsymbol{e}^1$.

$\lambda_3 = -A(N - 2)$
 This is also degenerate and corresponds to a subspace orthogonal to both $\boldsymbol{e}^1$ and the valid subspace. We call this the invalid subspace.

The derivations for the eigenvalues λ_1 and λ_2 are fairly straightforward and are given below, however, the determination of λ_3 is much more involved, and this is given in Appendix A.

1) Determination of λ_1: This will be done by showing that $\boldsymbol{T}\boldsymbol{e}^1 = \lambda_1\boldsymbol{e}^1$.

We now have $u_{xi} = \sum_y \sum_j T_{xi,yj} v_{yj}$.

Also, $\boldsymbol{v} = \boldsymbol{e}^1$

so that $v_{yj} = 1$ for all y and j. (22)

Substituting (21) gives

$$u_{xi} = -A \sum_{y=1}^{N} \sum_{j=1}^{N} [\delta_{xy}(1 - \delta_{ij})] - A \sum_{y=1}^{N} \sum_{j=1}^{N} [\delta_{ij}(1 - \delta_{xy})] - CN^2$$

but $\delta_{xy} = 1$ and $\delta_{ij} = 1$ only when $x = y$ and $i = j$. Therefore

$$u_{xi} = -A \sum_{j=1}^{N} (1 - \delta_{ij}) - A \sum_{y=1}^{N} (1 - \delta_{xy}) - CN^2 = -2A(N - 1) - CN^2.$$

The eigenvalue for the eigenvector $\boldsymbol{e}^1 = (1\ 1 \cdots 1)$ is $\lambda_1 = -2A(N - 1) - CN^2$.

2) Determination of λ_2: We derive this by showing that if $\boldsymbol{v}$ is a valid solution, there exists a vector $\boldsymbol{v}'$ where

$$\boldsymbol{v}' = \boldsymbol{v} - (\boldsymbol{v}.\hat{\boldsymbol{e}}^1)\hat{\boldsymbol{e}}^1$$

and $\boldsymbol{T}\boldsymbol{v}' = \lambda_2\boldsymbol{v}'$.

From the description of λ_2 given previously, it can be seen that the space spanned by $\boldsymbol{v}'$ corresponds exactly to what we termed the valid subspace.

For the purposes of illustration let $\boldsymbol{v}$ be the output of a network designed to solve a three city problem (which, of course, has a trivial solution because there is only one possible path, but can still be used to illustrate the effect of multiplying by the $\boldsymbol{T}$ matrix).

Let $\boldsymbol{v}$ represent the path $BACB$, i.e., $\boldsymbol{v}$ is given by:

$$\boldsymbol{v} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad \text{or} \quad \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ in 2D form.}$$

In general, if v_{yi} is a valid form (i.e., corresponds to a legitimate tour) then for each instance of y, the value of i for which $v_{yi} = 1$ is different. In addition, since there are exactly N cities in a valid tour $\Sigma_y \Sigma_j v_{yj} = N$.

Thus

$$\boldsymbol{v}' = \boldsymbol{v} - (\boldsymbol{v}.\hat{\boldsymbol{e}}^1)\hat{\boldsymbol{e}}^1 = \boldsymbol{v} - \left(\frac{1}{N}\sum_y \sum_j v_{yj}\right)\hat{\boldsymbol{e}}^1 = \boldsymbol{v} - \hat{\boldsymbol{e}}^1.$$

If

$$\boldsymbol{u}' = \boldsymbol{T}\boldsymbol{v}' \text{ and } \boldsymbol{u} = \boldsymbol{T}\boldsymbol{v}$$

then

$$\boldsymbol{u}' = \boldsymbol{T}\boldsymbol{v} - \boldsymbol{T}\hat{\boldsymbol{e}}^1 = \boldsymbol{T}\boldsymbol{v} - \lambda_1\hat{\boldsymbol{e}}^1 = \boldsymbol{u} - \lambda_1\hat{\boldsymbol{e}}^1. \tag{23}$$

The next step is to evaluate $\boldsymbol{u}$.

From (22) and (21)

$$u_{xi} = -A \sum_y \sum_j \delta_{xy}(1 - \delta_{ij})v_{yj} - A \sum_y \sum_j \delta_{ij}(1 - \delta_{xy})v_{yj} - C \sum_y \sum_j v_{yj}$$
$$= -A \sum_j (1 - \delta_{ij})v_{xj} - A \sum_y (1 - \delta_{xy})v_{yi} - CN$$
$$= -A \sum_{j \neq i} v_{xj} - A \sum_{y \neq x} v_{yi} - CN.$$

Since $\boldsymbol{v}$ represents a valid tour

$$\sum_j v_{xj} = 1 \Rightarrow \sum_{j \neq i} v_{xj} = 1 - v_{xi}$$
$$\sum_y v_{yj} = 1 \Rightarrow \sum_{y \neq x} v_{yi} = 1 - v_{xi}.$$

Thus

$$\begin{aligned} \boldsymbol{u} &= -A[\boldsymbol{e}^1 - \boldsymbol{v}] - A[\boldsymbol{e}^1 - \boldsymbol{v}] - CN\boldsymbol{e}^1 \\ &= -(2A + CN)\boldsymbol{e}^1 + 2A\boldsymbol{v}. \end{aligned}$$

From (23) and substituting for λ_1 by $-2A(N-1) + CN^2$

$$\begin{aligned} \boldsymbol{u}' &= -(2A + CN)N\hat{\boldsymbol{e}}^1 + 2A\boldsymbol{v} \\ &\quad + (2A(N-1) - CN^2)\hat{\boldsymbol{e}}^1 \\ &= 2A(\boldsymbol{v} - \hat{\boldsymbol{e}}^1) \\ &= 2A\boldsymbol{v}' \end{aligned}$$

$$\boldsymbol{u}' = \boldsymbol{T}\boldsymbol{v}' = \lambda_2 \boldsymbol{v}' \text{ where } \lambda_2 = 2A.$$

D. Analysis of the Dynamics of the TSP Network

The analysis of the dynamics will be based on showing how the eigenvalues and eigenvectors/subspaces of $\boldsymbol{T}$ together with the offset bias $\boldsymbol{i}^b$ determine the direction in which $\dot{\boldsymbol{u}}$ moves the state vector $\boldsymbol{u}$. Using this, the path of $\boldsymbol{v}$ from a point in the interior of the hypercube to a final solution can be described. It will then be possible to predict under what conditions the network will reach a valid solution. Hence we can optimize the constants A, B, C, and D to ensure that the network always converges to a valid solution.

The change in $\boldsymbol{u}$, i.e., $\dot{\boldsymbol{u}}$, is given by (16)

$$\dot{\boldsymbol{u}} = -\frac{\boldsymbol{u}}{\tau} + \boldsymbol{T}\boldsymbol{v} + \boldsymbol{i}^b \qquad \text{where } v_{xi} = g(u_{xi}).$$

Unfortunately, $\dot{\boldsymbol{u}}$ is a function of $\boldsymbol{v}$, not $\boldsymbol{u}$. Moreover, $\boldsymbol{v}$ is a nonlinear function of $\boldsymbol{u}$, and the dynamic equation contains a $-(\boldsymbol{u}/\tau)$ term. It turns out that although these factors are important, they do not alter the fundamental behavior of the network, which is dictated by the eigenvalues of $\boldsymbol{T}$. Therefore, to simplify the analysis the initial description will be based on the following dynamic equation:

$$\dot{\boldsymbol{u}} = \boldsymbol{T}\boldsymbol{u} + \boldsymbol{i}^b. \tag{24}$$

Later this description will be extended to take into account the effects of using the complete dynamic equation (16).

1) Description of the Dynamics of the Simplified Equation: Firstly, let $\boldsymbol{u}$ be decomposed into its valid and invalid subspace components plus its component in the direction of $\boldsymbol{e}^1$. Let these be $\boldsymbol{u}^{\text{val}}$, $\boldsymbol{u}^{\text{inv}}$, and $\boldsymbol{u}^1$, respectively. Thus

$$\boldsymbol{u} = \boldsymbol{u}^1 + \boldsymbol{u}^{\text{val}} + \boldsymbol{u}^{\text{inv}} \tag{25}$$

with

$$\boldsymbol{u}^1 = (\boldsymbol{u}.\hat{\boldsymbol{e}}^1)\hat{\boldsymbol{e}}^1.$$

These components are all mutually orthogonal since the valid and invalid subspaces are orthogonal to each other as well as to $\boldsymbol{e}^1$. Also, the offset bias which is given by $i^b_{xi} = CN$ (see (20)), can be written in vector notation as

$$\boldsymbol{i}^b = CN\boldsymbol{e}^1 = CN^2\hat{\boldsymbol{e}}^1.$$

The product $\boldsymbol{T}\boldsymbol{u}$ becomes

$$\boldsymbol{T}\boldsymbol{u} = \lambda_1\boldsymbol{u}^1 + \lambda_2\boldsymbol{u}^{\text{val}} + \lambda_3\boldsymbol{u}^{\text{inv}}.$$

The simplified dynamic equation (24) can be written in terms of the eigenvalues of $\boldsymbol{T}$ as follows:

$$\begin{aligned} \dot{\boldsymbol{u}} &= \boldsymbol{T}\boldsymbol{u} + CN\boldsymbol{e}^1 \\ &= \lambda_1\boldsymbol{u}^1 + \lambda_2\boldsymbol{u}^{\text{val}} + \lambda_3\boldsymbol{u}^{\text{inv}} + CN^2\hat{\boldsymbol{e}}^1 \\ &= [CN^2 + \lambda_1(\boldsymbol{u}.\hat{\boldsymbol{e}}^1)]\hat{\boldsymbol{e}}^1 + \lambda_2\boldsymbol{u}^{\text{val}} + \lambda_3\boldsymbol{u}^{\text{inv}} \\ &= [CN^2 + \lambda_1(\boldsymbol{u}.\hat{\boldsymbol{e}}^1)]\hat{\boldsymbol{e}}^1 + 2A\boldsymbol{u}^{\text{val}} - A(N-2)\boldsymbol{u}^{\text{inv}}. \end{aligned} \tag{26}$$

The corresponding Liapunov function for (24) is

$$\begin{aligned} E &= -\frac{1}{2}\boldsymbol{u}^t\boldsymbol{T}\boldsymbol{u} - (\boldsymbol{i}^b)^t\boldsymbol{u} \\ \Rightarrow 2E &= -\lambda_1|\boldsymbol{u}^1|^2 - \lambda_2|\boldsymbol{u}^{\text{val}}|^2 - \lambda_3|\boldsymbol{u}^{\text{inv}}|^2 \\ &\quad - 2CN^2(\hat{\boldsymbol{e}}^1.\boldsymbol{u}) \\ &= -\lambda_1|\boldsymbol{u}^1|^2 - 2CN^2|\boldsymbol{u}^1| - 2A|\boldsymbol{u}^{\text{val}}|^2 \\ &\quad + A(N-2)|\boldsymbol{u}^{\text{inv}}|^2 \\ &= (CN^2 + 2A(N-1))\left(|\boldsymbol{u}^1| + \frac{CN^2}{\lambda_1}\right)^2 \\ &\quad - 2A|\boldsymbol{u}^{\text{val}}|^2 + A(N-2)|\boldsymbol{u}^{\text{inv}}|^2 + \frac{(CN^2)^2}{\lambda_1}. \end{aligned} \tag{27}$$

Since A, C, and N are positive, to minimize E in (27) what is required is that

$$|\boldsymbol{u}^1| = -\frac{CN^2}{\lambda_1} = \frac{CN^2}{CN^2 + 2A(N-1)} \tag{28}$$

$$|\boldsymbol{u}^{\text{val}}| \to \infty \tag{29}$$

$$|\boldsymbol{u}^{\text{inv}}| \to 0. \tag{30}$$

By examining the dynamic equation (26) it is possible to see how the network dynamics bring about (28)–(30):

a) The net component of $\dot{\boldsymbol{u}}$ in the direction of $\boldsymbol{e}^1$, will be

$$\begin{aligned} &CN\boldsymbol{e}^1 + \lambda_1(\boldsymbol{u}.\hat{\boldsymbol{e}}^1)\hat{\boldsymbol{e}}^1 \\ &\quad = [CN^2 - (CN^2 + 2A(N-1))(\boldsymbol{u}.\hat{\boldsymbol{e}}^1)]\hat{\boldsymbol{e}}^1. \end{aligned}$$

This means that $\dot{\boldsymbol{u}}$ will contain a component in the direction of $\boldsymbol{e}^1$ that will move $\boldsymbol{u}$ towards the hyperplane where

$$[CN^2 - (CN^2 + 2A(N-1))(\boldsymbol{u}.\hat{\boldsymbol{e}}^1)] = 0. \tag{31}$$

b) The positive eigenvalue, $\lambda_2 = 2A$, for the valid subspace, means that $\dot{\boldsymbol{u}}$ will move $\boldsymbol{u}$ in the direction of its component in the valid subspace. Hence this component will increase in magnitude.

c) The negative eigenvalue, $\lambda_3 = -A(N-2)$, for the invalid subspace, will cause $\dot{\boldsymbol{u}}$ to move $\boldsymbol{u}$ in the opposite direction of its component in the invalid subspace. Therefore this component will decrease in magnitude.

Fig. 1 is a 3D illustration of factors a) and b). The small arrows in the figure show the components of $\dot{\boldsymbol{u}}$ if $\boldsymbol{v}$ lies in the valid subspace. In other words, they show how λ_2 ensures that $\boldsymbol{v}$ is pushed through the valid subspace to the corners of the hypercube. The large arrows indicate the direction of the component of $\boldsymbol{e}^1$ in $\dot{\boldsymbol{u}}$, depending on which side of the valid subspace $\boldsymbol{v}$ lies. In effect these show the confining effect of the λ_1 and $\boldsymbol{i}^b$. The shaded area represents the valid subspace in which $\boldsymbol{v}.\boldsymbol{e}^1 = 1$.

2) Implications of Using the Complete Dynamic Equation (16): The main implication is that $\dot{\boldsymbol{u}}$ is a function of $\boldsymbol{Tv}$, where $\boldsymbol{v}$ is related to $\boldsymbol{u}$ by the nonlinear function

$$g(u_{xi}) = \frac{1}{2}\left(1 + \tanh\left(\frac{u_{xi}}{u_0}\right)\right).$$

Fig. 2 depicts the output nonlinearity.

Differentiating v_{xi} it can be shown

$$\frac{dv_{xi}}{du_{xi}} = \frac{1}{u_0}(1 - v_{xi})v_{xi}$$

$$\frac{d^2v_{xi}}{du_{xi}^2} = \frac{1}{u_0^2}(1 - 2v_{xi})(1 - v_{xi})v_{xi}.$$

When $u_{xi} \approx 0$ then $v_{xi} \approx 0.5$ and the second derivative of $v_{xi} \approx 0$. Therefore, $g(u_{xi})$ is nearly linear for small values of u_{xi}, with a slope $\approx (0.25/u_0)$. So if $|\boldsymbol{u}|$ is small the nonlinearity can be approximated by

$$v_{xi} = 0.5 + \frac{1}{4u_0}u_{xi}. \tag{32}$$

In this case

$$\dot{\boldsymbol{u}} = 4u_0\dot{\boldsymbol{v}}.$$

If this is substituted into Hopfield's actual dynamic equation (16), then an equation very similar to the simplified dynamic equation (24) is obtained

$$4u_0\dot{\boldsymbol{v}} = -\frac{\boldsymbol{u}}{\tau} + \boldsymbol{Tv} + \boldsymbol{i}^b. \tag{33}$$

Apart from the substitution of $\boldsymbol{u}$ by $\boldsymbol{v}$, the only significant differences from the simplified equation are the scalar term $4u_0$, which can be ignored since it will not affect the path $\boldsymbol{v}$ takes, and the term $-(\boldsymbol{u}/\tau)$.

Fortunately, the effect of this term is negligible if the values of A and C are large, because the $\boldsymbol{Tv} + \boldsymbol{i}^b$ part of (33) will be very much larger than $\boldsymbol{u}/\tau$. Due to the nature of the nonlinearity, $|u_{xi}|$ can get very large as $v_{xi} \to 1$ or 0, but as Hopfield [2] points out, all this means is that the network will stabilize slightly in from the exact corners of the hypercube.

Overall, (33) shows that the analysis of the dynamics using the simplified equation is valid for the complete dynamic equation, as long as $\boldsymbol{u}$ is within the region where $g(u_{xi})$ is approximately linear. Clearly, as $|\boldsymbol{u}|$ gets larger this assumption breaks down, but it is very probable that

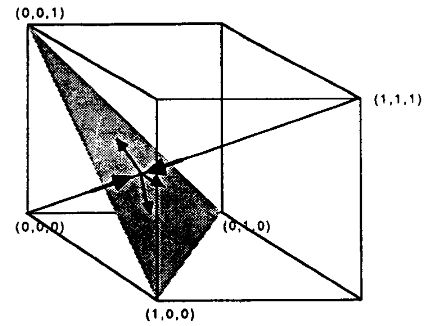


Fig. 1. A 3D illustration of factors a) and b).

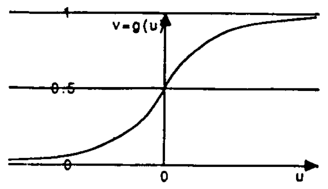


Fig. 2. The output nonlinearity.

before this happens $\boldsymbol{u}$ will already be pointing in the rough direction of a hypercube corner that is a valid solution. This can be ensured by always starting with a $\boldsymbol{v}$ that has a sufficiently small value of $\boldsymbol{v}^{\text{val}}$. Hence the main contribution of the nonlinearity will be simply to force the network to stabilize exactly at, or very near, the hypercube corner, depending on the precise effect of the $\boldsymbol{u}/\tau$ term.

3) Why Does the Network Frequently Produce Invalid Solutions?: The reason that Hopfield and Tank [3] give for introducing the C term is to ensure that the network reaches final solutions with exactly N neurons on. However, because the A and B terms of the connection matrix have an eigenvector in the direction of $\boldsymbol{i}^b$, i.e., $\boldsymbol{e}^1$, this is not actually what happens. To see this, recall (31) which specifies the hyperplane in which the network will try to confine $\boldsymbol{u}$

$$\left[CN^2 - (CN^2 + 2A(N-1))(\boldsymbol{u}.\hat{\boldsymbol{e}})\right] = 0.$$

This equation must be changed by substituting $\boldsymbol{v}$ for $\boldsymbol{u}$, as the equation was derived from the simplified dynamic equation rather than the complete one. Also, for generality the offset bias can be written as $i_{xi}^b = CN_e$ with N_e normally set to N. Now the hyperplane which $\boldsymbol{v}$ will be confined to, and hence the one in which the network will reach its final stable state, is given by

$$\begin{aligned} 0 &= CN_eN - (CN^2 + 2A(N-1))(\boldsymbol{v}.\hat{\boldsymbol{e}}^1) \\ &= CN_e - \left(C + \frac{2A(N-1)}{N^2}\right)\boldsymbol{v}.\boldsymbol{e}^1 \end{aligned}$$

$$\Rightarrow \boldsymbol{v}.\boldsymbol{e}^1 = \frac{N_e}{1 + \dfrac{2A(N-1)}{CN^2}}.$$

Writing $\boldsymbol{v}.\boldsymbol{e}^1$ as $\Sigma_y \Sigma_j v_{yj}$ produces the equation

$$\sum_y \sum_j v_{yj} = \frac{N_e}{1 + \frac{2A(N-1)}{CN^2}}. \tag{34}$$

Since A and $C > 0$, the RHS of (34) is less than N_e. The constant N_e would normally be set to N, and so the network will confine $\boldsymbol{v}$ to a hyperplane where $\Sigma_y \Sigma_j v_{yj} < N$. As a result in its final stable state, the network will tend to have less than N neurons on, and hence $\boldsymbol{v}$ will be an invalid solution. This is the main reason why the network frequently does not converge to a valid solution and can be used to explain Kahng's [13] poor convergence results for the ten city TSP.

Another reason why the network may produce invalid solutions is because the confining effects of λ_1, $\boldsymbol{i}^b$, and λ_3 are not well balanced with each other or the expansion effect of λ_2. Although not crucial for the ten city problem, it becomes very important for TSP's with a large number of cities (i.e., 30) that these effects are correctly balanced. This factor and ways of dealing with it are discussed in detail in the following sections.

4) How Does Hopfield and Tank's Choice of A, B, and C in the Ten City TSP Ensure that Valid Solutions are Produced?: In their 1985 paper Hopfield and Tank [3] claim to get successful results with ten city TSP's using the following constant values:

$$A = B = 500, \quad C = 200, \quad D = 500,$$
$$u_0 = 0.05, \quad N_e = 15.$$

For the ten city problem the RHS of (34) should be 10 if the final stable value of $\boldsymbol{v}$ is to have exactly 10 neurons on. It can be seen by substituting the values given above into the RHS of (34), that this is very nearly the case because

$$\frac{N_e}{1 + \frac{2A(N-1)}{CN^2}} = \frac{15}{1 + \frac{2.500.9}{200.100}} = \frac{15}{1.45} = 10.34.$$

However, this is achieved at the expense of abandoning the theory behind the C term, which would require N_e to be equal to N, i.e., 10. Instead N_e is changed from 10 to 15. Hopfield and Tank justify this as follows:

> The parameter N was not fixed as 10, but was used to adjust the neutral position of the amplifiers which would otherwise also need an adjustable offset parameter in their gain functions.

It is also apparent that the values for A, B, C, and D are very large. This ties in with the analysis in the subsection on the implications of using the complete dynamic equation, which required large values for these constants in order to minimize the effect of the $-(\boldsymbol{u}/\tau)$ term.

E. An Analytic Expression for A, B, C, and D

Hopfield and Tank's approach to finding appropriate values for A, B, C, and D is purely experimental, and has been shown to work fairly well for the ten city problem. However, in the light of the previous analysis a more systematic approach can now be developed.

What is required is to isolate the components in $\boldsymbol{T}$ and $\boldsymbol{i}^b$ that are responsible for the three main factors that determine the network dynamics. These are controlled by the eigenvalues λ_1, λ_2, and λ_3 which with the connection matrix specified by Hopfield (19) (assuming $D = 0$) have the following values and effects:

$\lambda_1 = -CN^2 - 2A(N-1)$

This should work in conjunction with $\boldsymbol{i}^b$ to confine $\boldsymbol{v}$ to the subspace where $\boldsymbol{v}.\boldsymbol{e}^1 = N$. However, as discussed previously, this is not the case because of the extra $-2A(n-1)$.

$\lambda_2 = 2A.$

Moves $\boldsymbol{u}$ and $\boldsymbol{v}$ out to the edges of the hypercube through the "valid subspace."

$\lambda_3 = -A(N-2)$

Confines $\boldsymbol{u}$ and $\boldsymbol{v}$ to the valid subspace, i.e., it "repells" them from the invalid subspace.

Once the effects have been isolated, it should be possible to adjust their relative magnitudes in order to achieve optimal convergence to valid solutions.

The first step is to ensure that the eigenvalue of $\boldsymbol{T}$ in the direction of $\boldsymbol{e}^1$ (i.e., λ_1) is exactly $-CN^2$ rather than $-CN^2 - 2A(N-1)$. This is easily achieved by adding $2A(N-1)/N^2$ to every element of $\boldsymbol{T}$.

Now that the C term has been isolated, the next step is to allow the ratio between λ_2 and λ_3 to be varied, so that with appropriate values of A and C it should be possible to arbitrarily set the ratios of the three eigenvalues. To achieve this extra degree of freedom, an extra constant, say $2A_1$, should be subtracted from the leading diagonal of $\boldsymbol{T}$. This has the effect of subtracting $2A_1$ from every eigenvalue of $\boldsymbol{T}$. Thus λ_2 becomes $2(A - A_1)$ and so its relative magnitude can be varied. Unfortunately, λ_1 becomes $-CN^2 - 2A_1$ and so a further $2A_1/N^2$ must be added to every element of $\boldsymbol{T}$ in order to ensure that the value of λ_1 is exactly $-CN^2$.

The final form of $\boldsymbol{T}$ is therefore

$$\begin{aligned} T_{xi,yj} = &-A\delta_{xy}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{xy}) \\ &-2A_1\delta_{xy}\delta_{ij} \\ &-C + \frac{2(AN - A + A_1)}{N^2} \\ &-D\, d_{xy}(\delta_{j,i+1} + \delta_{j,i-1}) \end{aligned} \tag{35}$$

with i^b_{xi} always set to CN where N is exactly the number of cities. In addition, B is always set equal to A.

The corresponding eigenvalues are now

$$\lambda_1 = -CN^2$$
$$\lambda_2 = 2(A - A_1)$$
$$\lambda_3 = -AN + 2(A - A_1).$$

Abe [9, Sec. I, p. 563] has also proposed a reformulation of $\boldsymbol{T}$ for the TSP. For him $\boldsymbol{T}$ should be specified by

$$T_{xi,yj} = -A\delta_{xy}(1 - \delta_{ij}) - A\delta_{ij}(1 - \delta_{xy}) - D\, d_{xy}(\delta_{j,i+1} + \delta_{j,i-1}).$$

N.B The signs have all been reversed when compared with his version.

Which is equivalent to our version to setting

$$A_1 = 0 \qquad C = \frac{2(AN - A)}{N^2}.$$

However, according to the assumptions upon which our version is based

$$i_{xi}^b = CN = \frac{2(AN - A)}{N} = 2A - \frac{2A}{N}$$

whereas Abe sets $\boldsymbol{i}^b$ according to

$$i_{xi}^b = A.$$

F. Simulation Results

To test our formulation of $\boldsymbol{T}$ we ran simulations with 10, 30, 40, and 50 city unit square TSP's. We used an update differential equation (16) without the $-(\boldsymbol{u}/\tau)$ term since it has a detrimental effect and can anyway be compensated for by having large values for A, A_1, C, and D. Further, instead of using a hyperbolic tangent output function we employed the following piecewise linear function:

$$g(u_i) = \begin{cases} 0 & \text{if } u_i <= -0.5 \\ u_i + 0.5 & \text{if } -0.5 < u_i < 0.5 \\ 1 & \text{if } u_i >= +0.5. \end{cases}$$

With this function we found that the network achieved better "quality" solutions (in terms of tour length) as well as taking far fewer iterations to reach them. This is possibly because the linearity assumption is valid over a larger range than with the hyperbolic tangent.

As long as the following rules were broadly adhered to the network always converged to a valid solution in about 700-3000 iterations:

a) The ratio of the confining effects of λ_1 and λ_3 should be about 160 times the magnitude of the expansion effect of λ_2.

For the ten city problem this is achieved by setting $A_1 = (31/32)A$, $C = (A/10)$.

For the 30 city problem this is achieved by setting $A_1 = (31/32)A$, $C = (A/30)$.

b) The D term should be approximately equal to $\lambda_3/80 = AN/80$.

c) Since there is tradeoff between the size of the eigenvalues of $\boldsymbol{T}$ and the time step size used in the integration of (16), if large values of A, A_1, etc., are used then a similarly small time step should be employed. (We have found experimentally that if the change in any one time step to $\boldsymbol{u}$ is greater than 5% then the network can become unstable).

d) The initial value of $\boldsymbol{v}$, although random, should be such that $|\boldsymbol{v}^{\text{val}}|$ is very small and $\boldsymbol{v}.\boldsymbol{e}^1 \approx N$. If the latter condition is not satisfied or the time step size is initially too large, then λ_1 can rapidly cause the network to become unstable.

For the actual values used for the various parameters in the 10 and 50 city problems see Appendix B.

1) Contribution of the D Term and Quality of Solutions: The value of D has so far not been discussed in this paper. Without it the TSP network is useless, since there will be no constraint forcing it to produce good as well as valid solutions. At first sight the value of D should not affect the solutions since confining effects which ensure a valid solution act independently of the D term. However, this term will introduce components in the dynamic equation that will tend to move $\boldsymbol{v}$ away from the valid subspace, and so D should be set at a level which does not offset the confining effects of λ_1, λ_3, and $\boldsymbol{i}^b$. Experimentally we have found for a random 10 city unit square TSP, that the largest positive eigenvalue of $\boldsymbol{T}$ with $A = A_1 = C = 0$; $D = 1$, is approximately 4. Again by experiment we found that rule d), given above, should be used to set the value of D.

On average, the solution quality of the our results from the 10 and 30 city TSP's was at least as good as those produced by the nearest neighbor algorithm (see Lawler *et al.* [17] for a detailed discussion of the algorithm). Considering that on many occasions the nearest neighbor algorithm produces the optimal solution for the 10 city TSP, and according to [17] will provide good solutions for larger sized TSP's, we feel the results are more than satisfactory evidence that the network is functioning correctly.

G. Conclusions on TSP

By analyzing the eigenvalues and corresponding subspaces of the connection matrix, we have given a comprehensive description of the way in which a Hopfield network solves the TSP. We have computed the eigenvalues of the connection matrix and shown how to relate these to analytic expressions for the constants in the energy function. With these expressions it is possible to solve TSP's of any size.

V. General Conclusions

The Hopfield network functions by projecting its input vector into a subspace. The nature of this projection and the subspace can be determined through evaluating the eigenvalues of the connection matrix.

In the case of the content addressable memory, to ensure that the original memory vectors are stable, this subspace must correspond to a single degenerate eigenvalue. Thus, every corner of the subspace is projected onto itself, and they will all be spurious stable points.

In the case of the traveling salesman problem, it has been shown that Hopfield's formulation of the connection matrix sets up a subspace which contains only the valid solutions of the problem. Hence, by carefully selecting the network parameters, it is possible to ensure that the network converges to hypercube corners in this subspace. To confirm this, analytic expressions for these parameters were derived, and an experiment using the network to solve 10, 30, 40, and 50 city TSP's was run. This experiment produced 100% valid solutions.

Appendix A
Determination of λ_3

This is based on first deriving a matrix $\boldsymbol{S}$ which is formed from the sum of outer products of all the valid forms of $\boldsymbol{v}$. By definition, a point in the invalid subspace cannot be expressed as a linear combination of the valid forms of $\boldsymbol{v}$. Therefore, the eigenvalue of S in the invalid subspace must be zero. It will be shown that $\boldsymbol{T}$ can be expressed in terms of $\boldsymbol{S}$ in such a way that it is possible to work out the eigenvalues of $\boldsymbol{T}$ from the eigenvalues of $\boldsymbol{S}$. This will be used to determine the eigenvalue of $\boldsymbol{T}$ in the invalid subspace i.e., λ_3.

First we derive a general form for $\boldsymbol{S}$.

Let N be the number of cities, and $\boldsymbol{v}^{(a)}$ be a valid form of $\boldsymbol{v}$.

Since S is formed from the sum of outer products of all valid solutions, $\boldsymbol{S}$ is given by

$$S = \sum_a \boldsymbol{v}^{(a)} \boldsymbol{v}^{(a)'}$$

$$\Rightarrow S_{xi,yj} = \sum_a v_{xi}^{(a)} v_{yj}^{(a)}.$$

Now, consider the following cases:

- If $i = j$ and $x \neq y$.
  By the definition of a valid solution it is not possible for $v_{xi} = 1$ and $v_{yi} = 1$ if $x \neq y$. Therefore all the elements of S where $i = j$ and $x \neq y$ are zero.
- If $i = j$ and $x = y$.
  In this case, we ask how many possible permutations exist for a fixed position of one city. This is $(N - 1)!$. Therefore
  $$S_{xi,xi} = (N-1)!$$
- If $i \neq j$ and $x = y$.
  Here, again it is not possible for $v_{xi} = 1$ and $v_{xj} = 1$ if $i \neq j$. Therefore, all the elements of S where $x = y$ and $i \neq j$ are zero.
- If $i \neq j$ and $x \neq y$.
  In this case, the position of two cities in the tour are fixed and the number of possible permutations of the others is $(N - 2)!$. Hence
  $$S_{xi,yj} = (N-2)! \qquad \text{when } x \neq y \text{ and } i \neq j.$$

$\boldsymbol{T}$ can now be expressed in terms of $\boldsymbol{S}$ as follows:

$$\boldsymbol{T} = \frac{A}{(N-2)!}\boldsymbol{S} - A\boldsymbol{e}^1\boldsymbol{e}^{1'} - A(N-2)\boldsymbol{I} - C\boldsymbol{e}^1\boldsymbol{e}^{1'}$$
$$= \frac{A}{(N-2)!}\boldsymbol{S} - A(N-2)\boldsymbol{I} - (C + A)\boldsymbol{e}^1\boldsymbol{e}^{1'}.$$

The multiplication of $\boldsymbol{S}$ by $A/(N-2)!$ just scales the eigenvalues of $\boldsymbol{S}$, and the subtraction of $(A + C)\boldsymbol{e}^1\boldsymbol{e}^{1'}$ only affects the eigenvalue of the eigenvector in the direction of $\boldsymbol{e}^1$. Since $\boldsymbol{S}$ has an eigenvalue of zero in the invalid subspace, the subtraction of $A(N-2)\boldsymbol{I}$ will introduce an eigenvalue of $-A(N-2)$ for the invalid subspace of $\boldsymbol{T}$.

Therefore $\lambda_3 = -A(N-2)$.

Appendix B

Parameters values used in 10 city TSP experiments:

$$A = 8; \quad A_1 = 7.75; \quad C = 0.8; \quad D = 1;$$

time step = 0.02 s.

Parameters values used in 50 city TSP experiments:

$$A = 8; \quad A_1 = 7.75; \quad C = 0.16; \quad D = 5;$$

time step = 0.005 s.

Acknowledgment

The authors wish to thank M. Plumbley, D. Lam, and the rest of the Cambridge University Engineering Department Speech and Vision group for their many valuable comments. The helpful comments of the anonymous referees were also greatly appreciated.

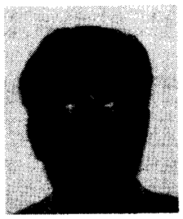
References

[1] J. J. Hopfield, "Unlearning has a stabilizing effect in collective memories," *Nature*, vol. 304, p. 158, 1983.
[2] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat. Acad. Sci. U.S.*, vol. 81, pp. 3088-3092, 1984.
[3] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biolog. Cybern.*, vol. 52, pp. 1-25, 1985.
[4] R. J. McEliece, C. E. Posner, R. R. Rodemich and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. IT-33, vol. 4, pp. 461-482, 1987.
[5] S. S. Venkatesh and D. Psaltis, "Linear and logarithmic capacities in associative neural networks," *IEEE Trans. Inform. Theory*, vol. IT-35, no. 3, pp. 558-568, 1989.
[6] A. Dembo, "On the capacity of associative memories with linear threshold functions," *IEEE Trans. Inform. Theory*, vol. IT-35, no. 4, pp. 709-720, 1989.
[7] J. D. Keeler, "Comparison between sparsely distributed memory and Hopfield-type neural network models," Phys. Dep. and Institute for Nonlinear Sci., Univ. of California, San Diego, CA, B-019, 1986.
[8] T. Kohonen, *Self-Organization and Associative Memory* (Springer Series on Information Sciences). Springer-Verlag, 1989.
[9] S. Abe, "Theories on the Hopfield neural networks," in *Proc. IJCNN 89*, vol. I, 1989, pp. 557-564.
[10] A. Lapedes and R. Farber, "Programming a massively parallel, computation universal system: static behavior," Amer. Inst. Phys., 1986.
[11] Y. S. Abu-Mustafa and J. St. Jacques, "Information capacity of the Hopfield model," *IEEE Trans. Inform. Theory*, vol. IT-31, pp. 461-464, 1985.
[12] V. Wilson and G. S. Pawley, "On the stability of the TSP problem algorithm of Hopfield and Tank," *Biolog. Cybern.*, vol. 58, pp. 63-70, 1988.
[13] A. Kahng, "Traveling salesman heuristics and embedding dimension in the Hopfield model," in *Proc. IJCNN 89*, vol. I, 1989, pp. 513-520.
[14] G. A. Tagliarini and E. W. Page, "Solving constraint satisfaction problems with neural networks," in *Proc. ICNN 87*, vol. III, 1987, pp. 741-747.
[15] D. E. Van den Bout and T. K. Miller, "A traveling salesman objective function that works," in *Proc. ICNN 88*, vol. II, 1988, pp. 299-303.

[16] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory Languages and Computation*. Reading, MA: Addison-Wesley, 1979.
[17] E. L. Lawler, J. K. Lenstra, Kan Rinnooy, and P. B. Shmoys, *The Traveling Salesman Problem*. New York: Wiley, 1985.

*

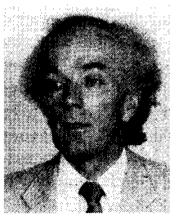
Sreeram V. B. Aiyer was born in London, England, on April 21, 1966. He received the B.A.Honours degree in electrical engineering and information science in 1988, the M.Phil. degree in computer speech and language processing in 1989, both from the University of Cambridge, England. Currently he is studying towards the Ph.D. degree with the Speech and Vision Group at Cambridge University Engineering Department. His main area of research is the study of feedback neural networks and their applications to speech recognition.



Mahesan Niranjan was born in Jaffna, Sri Lanka, on November 28, 1959. He received the B.Sc. degree from the University of Peradeniya, Sri Lanka, in 1982 and the M.E.E. degree from the Netherlands Universities Foundation in 1985.

He is currently a Research Associate with Cambridge University Engineering Department, working on speech recognition in noisy environments. His research interests include neural networks, pattern recognition, speech coding and segmentation.

*



Frank Fallside received the B.Sc. degree in electrical engineering from the University of Edinburgh in 1953 and the Ph.D. degree in control engineering from University College Swansea in 1958.

After a period with English Electric Stevenage he moved to Cambridge University. He is currently Professor of Information Engineering and Head of that division. He leads a research team in speech processing—coding, recognition, and synthesis, currently with particular emphasis on the use of artificial neural networks. He has also published widely in the subject. He is a joint Editor of *Computer Speech & Language*.

Dr. Fallside is a member of the Institute of Electrical Engineers.