Neural Networks and Combinatorial Optimization Problems — The Key to a Successful Mapping

(CUED/F-INFENG/TR 77)

Andrew H. Gee, Sreeram V. B. Aiyer and Richard W. Prager [1]

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

July 1991

Abstract

For several years now there has been much research interest in the use of Hopfield networks to solve combinatorial optimization problems. Although initial results were disappointing, it has since been demonstrated how modified network dynamics and better problem mapping can greatly improve the solution quality. The aim of this paper is to build on this progress by presenting a new analytical framework in which problem mappings can be evaluated without recourse to purely experimental means. A linearized analysis of the Hopfield network's dynamics forms the main theory of the paper, followed by a series of experiments in which some problem mappings are investigated in the context of these dynamics. In all cases the experimental results are compatible with the linearized theory, and observed weaknesses in the mappings are fully explained within the framework. What emerges is a largely analytical technique for evaluating candidate problem mappings, without having to resort to the more usual trial and error.

1 Introduction and Outline

Ever since the Hopfield network was first proposed as a means of approximately solving NP-complete combinatorial optimization problems [9], there has been considerable research effort to improve the network's performance, which at first was fairly disappointing [18, 10]. Most of this effort has been directed towards improving the reliability with which the network finds valid solutions, while at the same time developing various annealing schedules to help prevent entrapment in local minima of the objective function and force eventual convergence to a hypercube corner. Perhaps the most successful such modification has been the Mean Field Annealing algorithm with neuron normalization [13, 16], which has proved to be very successful with the classic benchmark Travelling Salesman and Graph Partitioning problems [13, 12, 17], as well as with some more practical scheduling problems [6]. The considerable improvement in performance is at the expense of analogue hardware implementability: the Mean Field Annealing dynamics employ a recursive update rule, as well as a neuron normalization division operation, which cannot be performed by the analogue circuit implementation of the original Hopfield network [8]. However, the Mean Field Annealing algorithm is still highly parallel and could greatly benefit from implementation in parallel digital hardware; even when run on conventional serial machines, it remains highly competitive with the other contending optimization techniques [13, 17].

In addition, some researchers have persevered with the original network dynamics, demonstrating that better problem *mapping* can greatly improve the overall quality of solutions. One key step towards this improvement lay in the realization that, for a broad class of problems, all the points in the network's output space corresponding to valid solutions lie on a particular affine subspace [1]. Using this fact, the validity constraints can be grouped together into a single penalty term which

---

[1] Email: ahg/svb10/rwp @eng.cam.ac.uk

no longer frustrates the optimization objective while the network state vector lies on this 'valid subspace'. The new mapping has been applied with considerable success to the Travelling Salesman problem [1, 2], a point matching problem in pattern recognition [5] and a novel implementation of the Viterbi algorithm for Hidden Markov Models [3].

In this paper, we use the structure of the valid subspace to shed new light on the relationship between the network's dynamics and the problem mapping. In Section 2 we introduce the matrix notation relevant to the rest of the paper. In particular we make extensive use of Kronecker (or Tensor) products [7], which simplify the mapping mathematics considerably. In Section 3 we review the mathematical background of the subject in hand, introducing the form of the valid subspace along the way. Section 4 contains a linearized analysis of the network's dynamics in the context of the valid subspace. We identify a matrix $\mathbf{A}$, the eigenvectors of which are most influential in the convergence of the network's state vector towards a hypercube corner. We also identify a vector $\mathbf{b}$ which is solely responsible for the initial direction of the state vector. In Sections 5 and 6 we test this theory against two different classes of optimization problem, namely the Euclidean Hamilton path problem and the point matching problem. For each case we associate $\mathbf{b}$ with a linear optimization problem, which we call the 'auxiliary linear problem': the relationship between the auxiliary linear problem and the parent optimization problem has considerable bearing on the eventual quality of solutions obtained. In both cases the network's behaviour is found to be as predicted by the theory, and observed shortcomings of the problem mappings are fully explained within this theoretical framework. Section 6 also describes an illustrative experiment in invariant pattern recognition which demonstrates the success of the Hopfield network in solving the point matching problem. Finally, Section 7 presents a brief discussion of the results of the paper, as well as some general conclusions. The Appendices contain ancillary information of relevance to the central material of the paper.

2 Notation and Definitions

2.1 Kronecker Product Notation and Matrix Identities

Let $\mathbf{A}^T$ denote the transpose of $\mathbf{A}$.
Let $[\mathbf{A}]_{ij}$ refer to the element in the i^{th} row and j^{th} column of the matrix $\mathbf{A}$.
Similarly, let $[\mathbf{a}]_i$ refer to the i^{th} element of the vector $\mathbf{a}$.
Sometimes, where the above notation would appear clumsy, and there is no danger of ambiguity, the same elements will be alternatively denoted A_{ij} and a_i.
Let the modulus of an $n \times m$ matrix $\mathbf{A}$ be defined as follows:

$$|\mathbf{A}|^2 = \left[\sum_{i=1}^{n}\sum_{j=1}^{m}[\mathbf{A}]_{ij}^2\right] = \text{trace}\left(\mathbf{A}^T\mathbf{A}\right) \tag{1}$$

Let $\mathbf{A} \otimes \mathbf{B}$ denote the Kronecker product of two matrices. If $\mathbf{A}$ is an $n \times n$ matrix, and $\mathbf{B}$ is an $m \times m$ matrix, then $\mathbf{A} \otimes \mathbf{B}$ is an $nm \times nm$ matrix given by:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} A_{11}\mathbf{B} & A_{12}\mathbf{B} & \dots & A_{1n}\mathbf{B} \\ A_{21}\mathbf{B} & A_{22}\mathbf{B} & \dots & A_{2n}\mathbf{B} \\ \dots & \dots & \dots & \dots \\ A_{n1}\mathbf{B} & A_{n2}\mathbf{B} & \dots & A_{nn}\mathbf{B} \end{bmatrix} \tag{2}$$

Let vec($\mathbf{A}$) be the function which maps the $n \times m$ matrix $\mathbf{A}$ onto the nm-element vector $\mathbf{a}$. This function is defined by:

$$\mathbf{a} = \text{vec}(\mathbf{A}) = [A_{11}, A_{21}, \dots, A_{n1}, A_{12}, A_{22}, \dots, A_{n2}, \dots, A_{1m}, A_{2m}, \dots, A_{nm}]^T \tag{3}$$

Throughout this paper we make use of the following identities (see [7] for proofs):

$$\text{trace}(\mathbf{AB}) \quad = \quad \text{trace}(\mathbf{BA}) \quad \text{(for } \mathbf{AB} \text{ and } \mathbf{BA} \text{ both square)} \tag{4}$$

$$\begin{aligned}(\mathbf{A}\otimes\mathbf{B})^T &= \mathbf{A}^T\otimes\mathbf{B}^T & (5)\\ (\mathbf{A}\otimes\mathbf{B})(\mathbf{X}\otimes\mathbf{Y}) &= (\mathbf{AX}\otimes\mathbf{BY}) & (6)\\ \mathrm{trace}(\mathbf{AB}) &= \left[\mathrm{vec}(\mathbf{A}^T)\right]^T \mathrm{vec}(\mathbf{B}) \quad (\text{for } \mathbf{A} \text{ and } \mathbf{B} \text{ both } n\times n) & (7)\\ \mathrm{vec}(\mathbf{AYB}) &= (\mathbf{B}^T\otimes\mathbf{A})\mathrm{vec}(\mathbf{Y}) & (8)\end{aligned}$$

If $\mathbf{w}$ and $\mathbf{x}$ are n-element column vectors, and $\mathbf{h}$ and $\mathbf{g}$ are m-element column vectors, then

$$(\mathbf{w}\otimes\mathbf{h})^T(\mathbf{x}\otimes\mathbf{g}) = (\mathbf{w}^T\mathbf{x})(\mathbf{h}^T\mathbf{g}) \tag{9}$$

If the $n\times n$ matrix $\mathbf{A}$ has eigenvectors $\{\mathbf{w}_1\ldots\mathbf{w}_n\}$ with corresponding eigenvalues $\{\gamma_1\ldots\gamma_n\}$, and the $m\times m$ matrix $\mathbf{B}$ has eigenvectors $\{\mathbf{h}_1\ldots\mathbf{h}_m\}$ with corresponding eigenvalues $\{\mu_1\ldots\mu_m\}$, then the $nm\times nm$ matrix $\mathbf{A}\otimes\mathbf{B}$ has eigenvectors $\{\mathbf{w}_i\otimes\mathbf{h}_j\}$ with corresponding eigenvalues $\{\gamma_i\mu_j\}$ $(i\in\{1\ldots n\}, j\in\{1\ldots m\})$.

2.2 Other Notation and Definitions

Let $\mathbf{I}^n$ be the $n\times n$ identity matrix.
Let $\mathbf{o}^n$ be the n-element column vector of ones:

$$[\mathbf{o}^n]_i = 1 \quad (i\in\{1,\ldots,n\}) \tag{10}$$

Let $\mathbf{O}^n$ be the $n\times n$ matrix of ones:

$$[\mathbf{O}^n]_{ij} = 1 \quad (i,j\in\{1,\ldots,n\}) \tag{11}$$

Let δ_{ij} be the Kronecker impulse function:

$$\delta_{ij} = \begin{cases} 1 & \text{if} \quad i=j\\ 0 & \text{if} \quad i\neq j\end{cases} \tag{12}$$

Let $\mathbf{R}^n$ be the $n\times n$ matrix given by

$$\mathbf{R}^n = \mathbf{I}^n - \tfrac{1}{n}\mathbf{O}^n \tag{13}$$

Multiplication by $\mathbf{R}^n$ has the effect of setting the column sums of a matrix to zero:

$$\begin{aligned}\sum_{i=1}^{n}[\mathbf{R}^n\mathbf{a}]_i &= \sum_{i=1}^{n}[\mathbf{a}-\tfrac{1}{n}\mathbf{O}^n\mathbf{a}]_i = \mathbf{o}^{nT}[\mathbf{a}-\tfrac{1}{n}\mathbf{O}^n\mathbf{a}]\\ &= \mathbf{o}^{nT}\mathbf{a}-\tfrac{1}{n}(\mathbf{o}^{nT}\mathbf{o}^n\mathbf{o}^{nT})\mathbf{a} = \mathbf{o}^{nT}\mathbf{a}-\mathbf{o}^{nT}\mathbf{a} = 0 & (14)\end{aligned}$$

Another way of considering $\mathbf{R}^n$ is as a projection matrix which removes the $\mathbf{o}^n$ component from any vector it pre-multiplies, since

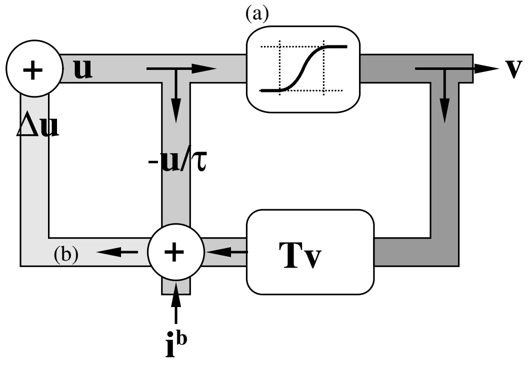
$$\mathbf{R}^n\mathbf{o}^n = (\mathbf{I}^n-\tfrac{1}{n}\mathbf{O}^n)\mathbf{o}^n = \mathbf{o}^n-\tfrac{1}{n}n\mathbf{o}^n = \mathbf{0} \tag{15}$$

Also note that since $\mathbf{R}^n$ is a projection matrix, $\mathbf{R}^n\mathbf{R}^n = \mathbf{R}^n$.

3 Background and Introduction to the Valid Subspace

A schematic diagram of the continuous Hopfield network [8] is shown in Figure 1. Neuron i has input $[\mathbf{u}]_i$, output $[\mathbf{v}]_i$, and is connected to neuron j with a weight $[\mathbf{T}]_{ij}$. Associated with each neuron is also an input bias term $[\mathbf{i}^{\mathrm{b}}]_i$. The dynamics of the network are governed by the following equations:

$$\begin{aligned}\dot{\mathbf{u}} &= -\tfrac{1}{\tau}\mathbf{u}+\mathbf{T}\mathbf{v}+\mathbf{i}^{\mathrm{b}} & (16)\\ [\mathbf{v}]_i &= g([\mathbf{u}]_i) & (17)\end{aligned}$$



(a) Non-linear threshold functions constraining $\mathbf{v}$ to the unit hypercube.

(b) The change in $\mathbf{u}$ is specified by the differential equation (16).

Figure 1: Schematic diagram of the continuous Hopfield network.

$[\mathbf{v}]_i$ is a continuous variable in the interval 0 to 1, and $g([\mathbf{u}]_i)$ is a continuous function which constrains $[\mathbf{v}]_i$ to this interval, usually a hyperbolic tangent of the form

$$g([\mathbf{u}]_i) = \frac{1}{1 + \exp(-[\mathbf{u}]_i/T)} \tag{18}$$

The network has stable states which are local minima of the Liapunov function [8]

$$E = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}\mathbf{v} - (\mathbf{i}^{\mathrm{b}})^T\mathbf{v} + \tfrac{1}{\tau}\sum \int_0^{[\mathbf{v}]_i} g^{-1}(V)\, dV \tag{19}$$

The network was subsequently proposed as a means of solving combinatorial optimization problems which can somehow be expressed as the constrained minimization of

$$E^{\mathrm{op}} = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{\mathrm{op}}\mathbf{v} - (\mathbf{i}^{\mathrm{op}})^T\mathbf{v} \quad ([\mathbf{v}]_i \in \{0,1\}) \tag{20}$$

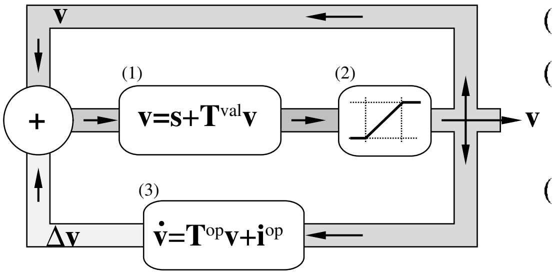
The idea is that the network's Liapunov function, invariably with $\tau \to \infty$, is associated with the cost function to be minimized in the combinatorial optimization problem. The network is then run and allowed to converge to a hypercube corner, which is subsequently interpreted as the solution of the problem. Hopfield and Tank showed in [9] how the network output can be used to represent a solution to the Travelling Salesman problem, and how the interconnection weights and input biases can be programmed appropriately for that problem: this process has subsequently been termed 'mapping' the problem onto the network. The same authors later showed how a variety of other problems can be mapped onto the same network, and reported on the results of using analogue hardware implementations to solve the problems [15].

The difficulties with mapping problems onto the Hopfield network lie with the satisfaction of hard constraints. The network acts to minimize a single Liapunov function, and yet the typical combinatorial optimization problem requires the minimization of a function *subject to* a number of constraints: if any of these constraints are violated then the solution is termed 'invalid'. The early mapping techniques coded the validity constraints as terms in the Liapunov function which were minimized when the constraints were satisfied:

$$E = E^{\mathrm{op}} + c_1 {E^{\mathrm{cns}}}_1 + c_2 {E^{\mathrm{cns}}}_2 + \ldots \tag{21}$$

The c_i parameters in equation (21) are constant weightings given to the various energy terms. The multiplicity of terms in the Liapunov function tend to frustrate one another, and the success of the network is highly sensitive to the relative values of the c_i parameters; it is not surprising, therefore, that the network frequently found invalid solutions, let alone high quality ones [10, 18].

In [1] an eigenvector and subspace analysis of the network's behaviour revealed how the E^{op} and E^{cns} terms in equation (21) can be effectively decoupled into different subspaces so that they no



(1) Projection of $\mathbf{v}$ onto valid subspace

(2) Nonlinear 'symmetric ramp' threshold functions constraining $\mathbf{v}$ to the unit hypercube.

(3) Change in $\mathbf{v}$ given by the gradient of the optimization energy term E^{op}, with $\Delta\mathbf{v} = \Delta t\dot{\mathbf{v}}$

Figure 2: Schematic diagram of the modified network implementation.

longer frustrate one another. For a wide variety of problems, it was realized that all the hypercube corners corresponding to valid solutions lie on a particular affine subspace with equation

$$\mathbf{v} = \mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s} \tag{22}$$

where $\mathbf{T}^{\mathrm{val}}$ is a projection matrix (ie. $\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{val}} = \mathbf{T}^{\mathrm{val}}$) and $\mathbf{T}^{\mathrm{val}}\mathbf{s} = \mathbf{0}$. This subspace was termed the **valid subspace**. The constrained optimization can now be re-expressed using only a single constraint term as follows [2]:

$$E = E^{\mathrm{op}} + \tfrac{1}{2}c_0\left|\mathbf{v} - (\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s})\right|^2 \tag{23}$$

Expanding equation (23) we obtain, to within a constant

$$E = E^{\mathrm{op}} - c_0\left(\tfrac{1}{2}\mathbf{v}^T(\mathbf{T}^{\mathrm{val}} - \mathbf{I})\mathbf{v} + \mathbf{s}^T\mathbf{v}\right) \tag{24}$$

from which we see that the Hopfield network parameters must be set as follows:

$$\mathbf{T} = \mathbf{T}^{\mathrm{op}} + c_0(\mathbf{T}^{\mathrm{val}} - \mathbf{I}) \tag{25}$$

$$\mathbf{i}^{\mathrm{b}} = \mathbf{i}^{\mathrm{op}} + c_0\mathbf{s} \tag{26}$$

In the limit of large c_0, $\mathbf{v}$ will be pinned to the valid subspace throughout convergence, and the network dynamics will minimize $E = E^{\mathrm{op}}$ as required. The resulting system can be simulated with increased efficiency on a serial machine using a modified network [2] which directly enforces the validity constraints — see Figure 2.

The top loop contains a projection operator (1) which directly confines $\mathbf{v}$ to the valid subspace. The nonlinear operator (2) ensures that $\mathbf{v}$ stays within the unit hypercube, by applying a 'symmetric ramp' threshold function to each of the elements of $\mathbf{v}$:

$$[\mathbf{v}]_i \rightarrow g([\mathbf{v}]_i) \tag{27}$$

$$\text{where } g([\mathbf{v}]_i) = \begin{cases} 1 & \text{if } [\mathbf{v}]_i > 1 \\ [\mathbf{v}]_i & \text{if } 0 \leq [\mathbf{v}]_i \leq 1 \\ 0 & \text{if } [\mathbf{v}]_i < 0 \end{cases} \tag{28}$$

Operation (3) in the bottom loop updates $\mathbf{v}$ using the dynamic equation

$$\Delta\mathbf{v} = \dot{\mathbf{v}}\Delta t = (\mathbf{T}^{\mathrm{op}}\mathbf{v} + \mathbf{i}^{\mathrm{op}})\Delta t \tag{29}$$

The dynamics expressed in (29) minimize E^{op} by steepest descent. Precise implementation details relating to the modified network can be found in [2]. Broadly speaking, however, the mode of operation is that a single traversal of the bottom loop, (3), is followed by several traversals of the top loop, (1) & (2); the whole cycle is repeated continually until $\mathbf{v}$ has converged to a hypercube corner.

4 Linearized Analysis of the Network Dynamics

In this section we carefully analyze the operation of the network in the early stages of convergence. In particular, we consider the operating region in which $\mathbf{v}$ is contained within the bounds of the unit hypercube, and has not yet come to any of the hypercube's faces. We start by deriving an expression for $\dot{\mathbf{v}}^{\mathrm{val}}$, the component of $\dot{\mathbf{v}}$ (as defined in equation (29)) which lies in the valid subspace. Since $\mathbf{v}$ is constantly confined to the valid subspace (ie. $\mathbf{v} = \mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s}$), any component of $\dot{\mathbf{v}}$ orthogonal to $\dot{\mathbf{v}}^{\mathrm{val}}$ is continually suppressed: hence it is $\dot{\mathbf{v}}^{\mathrm{val}}$, not $\dot{\mathbf{v}}$, which best characterizes the network's overall dynamics.

$$\begin{aligned} \dot{\mathbf{v}}^{\mathrm{val}} &= \mathbf{T}^{\mathrm{val}}\dot{\mathbf{v}} = \mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}}\mathbf{v} + \mathbf{i}^{\mathrm{op}}) \\ &= \mathbf{T}^{\mathrm{val}}\left(\mathbf{T}^{\mathrm{op}}(\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s}) + \mathbf{i}^{\mathrm{op}}\right) \\ &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}}\mathbf{s} + \mathbf{i}^{\mathrm{op}}) \end{aligned} \tag{30}$$

We see that $\dot{\mathbf{v}}^{\mathrm{val}}$ is made up of two parts, a constant term, $\mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}}\mathbf{s} + \mathbf{i}^{\mathrm{op}})$, and a term which depends on $\mathbf{v}$, $\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}}\mathbf{v}$. Let us simplify these expressions be writing

$$\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}} = \mathbf{A} \tag{31}$$

$$\mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}}\mathbf{s} + \mathbf{i}^{\mathrm{op}}) = \mathbf{b} \tag{32}$$

which gives

$$\dot{\mathbf{v}}^{\mathrm{val}} = \mathbf{A}\mathbf{v} + \mathbf{b} = \mathbf{A}\mathbf{v}^{\mathrm{val}} + \mathbf{b} \quad \text{where} \quad \mathbf{v}^{\mathrm{val}} = \mathbf{T}^{\mathrm{val}}\mathbf{v} \tag{33}$$

In Section 5.4 we will note that with $\mathbf{v}$ confined to the valid subspace, E^{op} may be expressed (to within a constant) as

$$E^{\mathrm{op}} = -\tfrac{1}{2}\mathbf{v}^T\mathbf{A}\mathbf{v} - \mathbf{b}^T\mathbf{v} \tag{34}$$

It is apparent, therefore, that the dynamics $\dot{\mathbf{v}} = \dot{\mathbf{v}}^{\mathrm{val}} = \mathbf{A}\mathbf{v} + \mathbf{b}$ simply perform steepest descent on E^{op} within the valid subspace, which is consistent with our goal of finding a valid solution which minimizes E^{op}. The general solution of (33) can be expressed by means of the matrix exponential (see [14]):

$$\mathbf{v}^{\mathrm{val}}(t) = e^{\mathbf{A}t}\mathbf{v}_{\mathrm{o}}^{\mathrm{val}} + \int_0^t e^{\mathbf{A}(t-\tau)}\mathbf{b}\,d\tau \tag{35}$$

where $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ is the value of $\mathbf{v}^{\mathrm{val}}$ at time $t = 0$, usually set to be a small, random vector. Rewriting the matrix exponentials as power series gives

$$\begin{aligned} \mathbf{v}^{\mathrm{val}}(t) &= \sum_{k=0}^{\infty}\frac{t^k}{k!}\mathbf{A}^k\mathbf{v}_{\mathrm{o}}^{\mathrm{val}} + \int_0^t \sum_{k=0}^{\infty}\frac{(t-\tau)^k}{k!}\mathbf{A}^k\mathbf{b}\,d\tau \\ &= \sum_{k=0}^{\infty}\frac{t^k}{k!}\mathbf{A}^k\mathbf{v}_{\mathrm{o}}^{\mathrm{val}} + \sum_{k=0}^{\infty}\frac{\mathbf{A}^k\mathbf{b}}{k!}\int_0^t (t-\tau)^k\,d\tau \\ &= \sum_{k=0}^{\infty}\frac{t^k}{k!}\mathbf{A}^k\mathbf{v}_{\mathrm{o}}^{\mathrm{val}} + \sum_{k=0}^{\infty}\frac{t^{k+1}}{(k+1)!}\mathbf{A}^k\mathbf{b} \end{aligned}$$

Suppose $\mathbf{A}$ has eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$, with associated normalized eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$. We will need to distinguish between the zero and non-zero eigenvalues of $\mathbf{A}$, so define the set $\mathcal{Z}$ such that $\lambda_i = 0$ for $i \in \mathcal{Z}$ and $\lambda_i \neq 0$ for $i \notin \mathcal{Z}$. Let $\mathbf{v}^{\mathrm{val}}$, $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ and $\mathbf{b}$ be decomposed along $\mathbf{A}$'s eigenvectors as follows:

$$\mathbf{v}^{\mathrm{val}} = \sum_{i=1}^{N} v_i\mathbf{u}_i\ , \quad \mathbf{v}_{\mathrm{o}}^{\mathrm{val}} = \sum_{i=1}^{N} o_i\mathbf{u}_i\ , \quad \mathbf{b} = \sum_{i=1}^{N} b_i\mathbf{u}_i \tag{36}$$

Then we have

$$\mathbf{A}^k\mathbf{v}_{\mathrm{o}}^{\mathrm{val}} = \sum_{i=1}^{N} o_i\lambda_i^k\mathbf{u}_i\ , \quad \mathbf{A}^k\mathbf{b} = \sum_{i=1}^{N} b_i\lambda_i^k\mathbf{u}_i$$

giving

$$
\begin{aligned}
\mathbf{v}^{\mathrm{val}}(t) &= \sum_{k=0}^{\infty} \frac{t^k}{k!} \sum_{i=1}^{N} o_i \lambda_i^k \mathbf{u}_i + \sum_{k=0}^{\infty} \frac{t^{k+1}}{(k+1)!} \sum_{i=1}^{N} b_i \lambda_i^k \mathbf{u}_i \\
&= \sum_{i=1}^{N} o_i \mathbf{u}_i \sum_{k=0}^{\infty} \frac{t^k \lambda_i^k}{k!} + \sum_{i \notin \mathcal{Z}} \frac{b_i \mathbf{u}_i}{\lambda_i} \sum_{k=0}^{\infty} \frac{t^{k+1} \lambda_i^{k+1}}{(k+1)!} + \sum_{i \in \mathcal{Z}} b_i \mathbf{u}_i \sum_{k=0}^{\infty} \frac{t^{k+1} 0^k}{(k+1)!} \\
&= \sum_{i=1}^{N} e^{\lambda_i t} o_i \mathbf{u}_i + \sum_{i \notin \mathcal{Z}} \frac{b_i \mathbf{u}_i}{\lambda_i} \left(\sum_{k=0}^{\infty} \frac{t^k \lambda_i^k}{k!} - 1 \right) + \sum_{i \in \mathcal{Z}} b_i \mathbf{u}_i t \\
&= \sum_{i=1}^{N} e^{\lambda_i t} o_i \mathbf{u}_i + \sum_{i \notin \mathcal{Z}} \frac{b_i \mathbf{u}_i}{\lambda_i} \left(e^{\lambda_i t} - 1 \right) + \sum_{i \in \mathcal{Z}} b_i \mathbf{u}_i t \qquad (37)
\end{aligned}
$$

Equation (37) is completely general in that it holds for arbitrary $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$. However, we can simplify the expression if we define $\mathbf{A}$ and $\mathbf{b}$ as in equations (31) and (32). In this case the eigenvectors of $\mathbf{A}$ with corresponding zero eigenvalues are confined to spanning the invalid subspace (unless $\mathbf{T}^{\mathrm{op}}\mathbf{v}^{\mathrm{val}} = \mathbf{0}$ for some $\mathbf{v}^{\mathrm{val}}$ in the valid subspace, which does not happen for any of the $\mathbf{T}^{\mathrm{op}}$ considered in this paper), whilst $\mathbf{b}$ lies wholly within the valid subspace, so $b_i = 0$ for $i \in \mathcal{Z}$. Equation (37) subsequently becomes

$$
\mathbf{v}^{\mathrm{val}}(t) = \sum_{i=1}^{N} e^{\lambda_i t} o_i \mathbf{u}_i + \sum_{i \notin \mathcal{Z}} \frac{b_i \mathbf{u}_i}{\lambda_i} \left(e^{\lambda_i t} - 1 \right) \qquad (38)
$$

It will be revealing to examine the expression for $\mathbf{v}^{\mathrm{val}}(t)$ given by equation (37) in the limits of small and large t. For small t, we make the approximation

$$
e^{\lambda_i t} \approx 1 + \lambda_i t
$$

which gives

$$
\mathbf{v}^{\mathrm{val}}(t) \approx \sum_{i=1}^{N} \left[o_i (1 + \lambda_i t) + b_i t \right] \mathbf{u}_i
$$

Further noting that for a small, random $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ the o_i are often small in comparison with the b_i, we get

$$
\mathbf{v}^{\mathrm{val}}(t) \approx t \sum_{i=1}^{N} b_i \mathbf{u}_i = t\mathbf{b} \qquad (39)
$$

So it is apparent that $\mathbf{v}^{\mathrm{val}}$ initially takes off in the direction of the vector $\mathbf{b}$. In the limit of large t, equation (37) indicates that $\mathbf{v}^{\mathrm{val}}$ will tend towards the eigenvector of $\mathbf{A}$ corresponding to the largest positive eigenvalue (let us call this eigenvector $\mathbf{u}_{\mathrm{max}}$, the corresponding eigenvalue λ_{max}, and the various weighting terms in this direction v_{max}, o_{max} and b_{max}). It should be noted at this point that the dominance of $\mathbf{u}_{\mathrm{max}}$ is enhanced by the action of Matrix Graduated Non-Convexity, a scheme proposed in [2] to prevent entrapment in local minima and force eventual convergence to a hypercube corner (see Appendix B). Of course, before the large t limit is approached $\mathbf{v}$ would have impeded on one of the hypercube faces, and the linear analysis above would no longer be valid. However, it is reasonable to assume that the vectors $\mathbf{b}$ and $\mathbf{u}_{\mathrm{max}}$ are the main influences on the initial evolution of $\mathbf{v}^{\mathrm{val}}$, an assumption which will be justified experimentally in later sections. We would also expect that when $\mathbf{v}$ has successfully converged to a valid hypercube corner, the associated $\mathbf{v}^{\mathrm{val}}$ will invariably contain a significant component in the direction of $\mathbf{u}_{\mathrm{max}}$; this expectation will also be experimentally verified in later sections of this paper.

5 The Euclidean Hamilton Path Problem

5.1 Problem Formulation

The Euclidean Hamilton Path problem can be expressed as follows:

> Given n cities, all of which have to be visited once only on a single journey, find the best order in which to visit them such that the total journey length is minimized.

As such it is very similar to the Euclidean Travelling Salesman problem, except that in the latter a closed tour is specified. It is therefore not surprising that the mathematical problem formulation and the mapping onto the Hopfield network are also very similar [2].

The solution to the Hamilton Path problem may be expressed by means of an n-element vector $\mathbf{p}$: suppose the cities are originally presented in an ordered list of Cartesian coordinates, $\mathbf{C}$ say. $\mathbf{C}$ is an $n \times 2$ matrix, in which $[\mathbf{C}]_{j1}$ is the x-coordinate of city j, and $[\mathbf{C}]_{j2}$ is the y-coordinate of the same city. Let $[\mathbf{p}]_i = j$ if the city initially in list position j is to be placed in position i of the journey. We must also enforce

$$[\mathbf{p}]_i \neq [\mathbf{p}]_j \quad (i, j \in \{1, \ldots, n\}, i \neq j) \tag{40}$$

if all the cities in the original list are to be present in the final journey. The reordering may be achieved directly by means of an $n \times n$ matrix operator $\mathbf{V}(\mathbf{p})$, which is constructed as follows:

$$[\mathbf{V}(\mathbf{p})]_{ij} = \begin{cases} 1 & \text{if } [\mathbf{p}]_i = j \\ 0 & \text{otherwise} \end{cases} \tag{41}$$

With the above definition of $\mathbf{V}(\mathbf{p})$, the $n \times 2$ matrix $\mathbf{C}' = \mathbf{V}(\mathbf{p})\mathbf{C}$ contains the same city coordinates as $\mathbf{C}$, but reordered in the manner defined by $\mathbf{p}$. Note that $\mathbf{V}(\mathbf{p})$ is a permutation matrix: its elements are all either 1's or 0's, and each row and column contains only a single 1. Expressed mathematically, this means:

$$[\mathbf{V}(\mathbf{p})]_{ij} \in \{1, 0\} \tag{42}$$

$$\sum_{i=1}^{n} [\mathbf{V}(\mathbf{p})]_{ij} = \sum_{j=1}^{n} [\mathbf{V}(\mathbf{p})]_{ij} = 1 \quad (i, j \in \{1, \ldots, n\}) \tag{43}$$

By making use of the property of the matrix $\mathbf{R}^n$ given in equation (14), these conditions may be more neatly expressed as [2]:

$$[\mathbf{V}(\mathbf{p})]_{ij} \in \{1, 0\} \tag{44}$$

$$\mathbf{V}(\mathbf{p}) = \mathbf{R}^n \mathbf{V}(\mathbf{p}) \mathbf{R}^n + \mathbf{S} \tag{45}$$

$$\text{where } \mathbf{S} = \tfrac{1}{n}\mathbf{O}^n = \tfrac{1}{n}\mathbf{o}^n \mathbf{o}^{nT} \tag{46}$$

We must now find an expression (involving $\mathbf{V}(\mathbf{p})$) for the total path length of a journey specified by ordering $\mathbf{p}$. Let $\mathbf{P}$ be the $n \times n$ matrix of negated inter-city distances given by

$$[\mathbf{P}]_{ij} = - \text{ (distance between cities } i \text{ and } j) \tag{47}$$

Let $\mathbf{Q}$ be the $n \times n$ matrix given by

$$[\mathbf{Q}]_{ij} = \delta_{j-1,i} + \delta_{j+1,i} \quad (i, j \in \{1 \ldots n\}) \tag{48}$$

Following this specification, for $n = 5$, $\mathbf{Q}$ would be

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Then the path length corresponding to ordering $\mathbf{p}$ is given by

$$d(\mathbf{p}) = -\tfrac{1}{2}\,\text{trace}\left[\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T\mathbf{Q}\right] \tag{49}$$

and the optimization problem may be concisely expressed as

$$\min_{\mathbf{p}} d(\mathbf{p}) \quad \text{where} \quad d(\mathbf{p}) = -\tfrac{1}{2}\,\text{trace}\left[\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T\mathbf{Q}\right] \tag{50}$$

5.2 Mapping the Problem onto the Hopfield Network

Having established the mathematical problem formulation, it is now necessary to show how the problem may be mapped onto the Hopfield network for solution. The converged state vector of this network, call it $\mathbf{v}(\mathbf{p})$, is used to represent the desired permutation matrix, $\mathbf{V}(\mathbf{p})$, by way of the vec function:

$$\mathbf{v}(\mathbf{p}) = \text{vec}(\mathbf{V}(\mathbf{p})) \tag{51}$$

Hence, if there are n cities in the problem, a network with $N = n^2$ units is required to find the reordering $\mathbf{p}$. We must also define a matrix, $\mathbf{V}$, which is the matrix equivalent of the unconverged state vector $\mathbf{v}$:

$$\mathbf{v} = \text{vec}(\mathbf{V}) \tag{52}$$

Consider what happens if we confine $\mathbf{V}$ so that

$$\mathbf{V} = \mathbf{R}^n\mathbf{V}\mathbf{R}^n + \mathbf{S} \tag{53}$$

If we subsequently force $\mathbf{v}$ into a hypercube corner, ie. if we enforce $[\mathbf{V}]_{ij} \in \{1, 0\}$, then it is clear that $\mathbf{V}$ will satisfy the conditions for $\mathbf{V}(\mathbf{p})$ in equations (44) and (45), and will therefore represent a valid solution to the combinatorial optimization problem.

Equation (53) corresponds to a valid subspace of the form given in equation (22). The parameters $\mathbf{T}^{\text{val}}$ and $\mathbf{s}$ are derived as follows:

$$\begin{array}{rlll} & \mathbf{V} & = & \mathbf{R}^n\mathbf{V}\mathbf{R}^n + \mathbf{S} & \\ \Leftrightarrow & \text{vec}(\mathbf{V}) & = & \text{vec}(\mathbf{R}^n\mathbf{V}\mathbf{R}^n) + \text{vec}(\mathbf{S}) & \\ \Leftrightarrow & \text{vec}(\mathbf{V}) & = & (\mathbf{R}^n \otimes \mathbf{R}^n)\text{vec}(\mathbf{V}) + \text{vec}(\mathbf{S}) & \text{(using (8))} \\ \Leftrightarrow & \mathbf{v} & = & (\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{v} + \text{vec}(\mathbf{S}) & \text{(using (52))} \end{array}$$

Hence, by comparison with (22), we have

$$\mathbf{T}^{\text{val}} = \mathbf{R}^n \otimes \mathbf{R}^n \tag{54}$$

$$\mathbf{s} = \text{vec}(\mathbf{S}) = \tfrac{1}{n}(\mathbf{o}^n \otimes \mathbf{o}^n) \tag{55}$$

Note that $\mathbf{T}^{\text{val}}$ is a projection matrix, since

$$\begin{aligned} \mathbf{T}^{\text{val}}\mathbf{T}^{\text{val}} &= (\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{R}^n \otimes \mathbf{R}^n) = (\mathbf{R}^n\mathbf{R}^n \otimes \mathbf{R}^n\mathbf{R}^n) \\ &= (\mathbf{R}^n \otimes \mathbf{R}^n) = \mathbf{T}^{\text{val}} \end{aligned} \tag{56}$$

In this problem we are attempting to minimize $d(\mathbf{p})$, the path length, with respect to the point ordering $\mathbf{p}$. The Hopfield network minimizes E^{op}, so if we set $E^{\text{op}} = d(\mathbf{p})$, then the Hopfield network will carry out the desired optimization. We are now in a position to derive the form of the matrix $\mathbf{T}^{\text{op}}$:

$$\begin{array}{rlll} E^{\text{op}} & = & d(\mathbf{p}) & \\ & = & -\tfrac{1}{2}\,\text{trace}\left[\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T\mathbf{Q}\right] & \\ & = & -\tfrac{1}{2}\,\text{trace}\left[\mathbf{V}(\mathbf{p})^T\mathbf{Q}\mathbf{V}(\mathbf{p})\mathbf{P}\right] & \text{(using (4))} \\ & = & -\tfrac{1}{2}\,[\text{vec}(\mathbf{V}(\mathbf{p}))]^T\,\text{vec}(\mathbf{Q}\mathbf{V}(\mathbf{p})\mathbf{P}) & \text{(using (7))} \\ & = & -\tfrac{1}{2}\,[\text{vec}(\mathbf{V}(\mathbf{p}))]^T\,(\mathbf{P} \otimes \mathbf{Q})\text{vec}(\mathbf{V}(\mathbf{p})) & \text{(using (8))} \\ & = & -\tfrac{1}{2}\,\mathbf{v}(\mathbf{p})^T(\mathbf{P} \otimes \mathbf{Q})\mathbf{v}(\mathbf{p}) & \text{(using (51))} \end{array}$$

By comparison with (20), we see that

$$\begin{aligned} \mathbf{T}^{\mathrm{op}} &= (\mathbf{P} \otimes \mathbf{Q}) && (57) \\ \mathbf{i}^{\mathrm{op}} &= \mathbf{0} && (58) \end{aligned}$$

5.3 Applying the Linearized Analysis of the Network Dynamics

If we are now to analyse the initial evolution of the vector $\mathbf{v}$, then we need to derive expressions for the vectors $\mathbf{b}$ and $\mathbf{u}_{\max}$, which we have identified as the main influences on the initial direction of $\mathbf{v}$. Now,

$$\begin{aligned} \mathbf{b} &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s} + \mathbf{T}^{\mathrm{val}}\mathbf{i}^{\mathrm{op}} \\ &= \tfrac{1}{n}(\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{P} \otimes \mathbf{Q})(\mathbf{o}^n \otimes \mathbf{o}^n) \\ &= \tfrac{1}{n}(\mathbf{R}^n\mathbf{P}\mathbf{o}^n \otimes \mathbf{R}^n\mathbf{Q}\mathbf{o}^n) \quad \text{(using (6))} && (59) \end{aligned}$$

$$\begin{aligned} \mathbf{A} &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}} \\ &= (\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{P} \otimes \mathbf{Q})(\mathbf{R}^n \otimes \mathbf{R}^n) \\ &= (\mathbf{R}^n\mathbf{P}\mathbf{R}^n \otimes \mathbf{R}^n\mathbf{Q}\mathbf{R}^n) \quad \text{(using (6))} && (60) \end{aligned}$$

As we saw in the earlier section on Kronecker products, we can relate the eigenvectors of $\mathbf{A}$ to those of $\mathbf{R}^n\mathbf{P}\mathbf{R}^n$ and $\mathbf{R}^n\mathbf{Q}\mathbf{R}^n$. We shall find it useful to start by looking at the eigenvectors of $\mathbf{Q}$, which are given by

$$[\mathbf{h}_k]_i = \sin\left(\frac{k\pi i}{n+1}\right) \qquad (i, k \in \{1 \ldots n\}) \tag{61}$$

with corresponding eigenvalues

$$\mu_k = 2\cos\left(\frac{k\pi}{n+1}\right) \qquad (k \in \{1 \ldots n\}) \tag{62}$$

The proof is as follows:

$$\begin{aligned} [\mathbf{Q}\mathbf{h}_k]_i &= \sin\left(\frac{k\pi(i+1)}{n+1}\right) + \sin\left(\frac{k\pi(i-1)}{n+1}\right) \\ &= 2\cos\left(\frac{k\pi}{n+1}\right)\sin\left(\frac{k\pi i}{n+1}\right) \\ &= 2\cos\left(\frac{k\pi}{n+1}\right)[\mathbf{h}_k]_i \end{aligned}$$

The first line of the proof holds even for $i = 1$ and $i = n$, since $\sin(0) = \sin(k\pi) = 0$. Now, those $\mathbf{h}_k$ with even k (ie. with $k = 2j$) exhibit the property $\sum_{i=1}^{n} [\mathbf{h}_k]_i = 0$, and so

$$\begin{aligned} \mathbf{h}_{2j}^T\mathbf{o}^n &= 0 && (63) \\ \text{and } \mathbf{R}^n\mathbf{h}_{2j} &= \mathbf{h}_{2j} && (64) \end{aligned}$$

It can readily be shown that $\mathbf{h}_{2j}$ is also an eigenvector of $\mathbf{R}^n\mathbf{Q}\mathbf{R}^n$, with the same eigenvalue μ_{2j}, since

$$\begin{aligned} \mathbf{R}^n\mathbf{Q}\mathbf{R}^n\mathbf{h}_{2j} &= \mathbf{R}^n\mathbf{Q}\mathbf{h}_{2j} \quad \text{(using (64))} \\ &= \mu_{2j}\mathbf{R}^n\mathbf{h}_{2j} \\ &= \mu_{2j}\mathbf{h}_{2j} \quad \text{(using (64))} \end{aligned}$$

This accounts for about half of the eigenvectors of $\mathbf{R}^n\mathbf{Q}\mathbf{R}^n$. The others seem to be more difficult to derive analytically, while experiments indicate that they are almost, but not exactly, sampled sinusoids in form. The experiments also reveal that the eigenvector of $\mathbf{R}^n\mathbf{Q}\mathbf{R}^n$ corresponding to the largest positive eigenvalue is invariably one of those which is also an eigenvector of $\mathbf{Q}$: let us call this eigenvector $\mathbf{h}_{\max}$ and its corresponding eigenvalue $\mu_{\max}$. Another interesting fact brought to light by the experiments is that all the eigenvalues of $\mathbf{R}^n\mathbf{P}\mathbf{R}^n$ are nonnegative; this seems to hold for all negated distance matrices $\mathbf{P}$ under a Euclidean distance metric. We can now use the properties of Kronecker products to state that

$$\mathbf{u}_{\max} = \theta(\mathbf{w}_{\max} \otimes \mathbf{h}_{\max}) \tag{65}$$

where $\mathbf{w}_{\max}$ is the eigenvector of $\mathbf{R}^n\mathbf{P}\mathbf{R}^n$ corresponding to the largest positive eigenvalue, and θ is a normalizing constant. It transpires that, for this problem, $\mathbf{b}$ and $\mathbf{u}_{\max}$ are mutually orthogonal, since

$$\begin{aligned}
{\mathbf{u}_{\max}}^T\mathbf{b} &= \tfrac{1}{n}(\mathbf{w}_{\max} \otimes \mathbf{h}_{\max})^T(\mathbf{R}^n\mathbf{P}\mathbf{o}^n \otimes \mathbf{R}^n\mathbf{Q}\mathbf{o}^n) & \\
&= \tfrac{1}{n}({\mathbf{w}_{\max}}^T\mathbf{R}^n\mathbf{P}\mathbf{o}^n)({\mathbf{h}_{\max}}^T\mathbf{R}^n\mathbf{Q}\mathbf{o}^n) & \text{(using (9))} \\
&= \tfrac{1}{n}({\mathbf{w}_{\max}}^T\mathbf{R}^n\mathbf{P}\mathbf{o}^n)({\mathbf{h}_{\max}}^T\mathbf{Q}\mathbf{o}^n) & \text{(using (64))} \\
&= \tfrac{\mu_{\max}}{n}({\mathbf{w}_{\max}}^T\mathbf{R}^n\mathbf{P}\mathbf{o}^n)({\mathbf{h}_{\max}}^T\mathbf{o}^n) & \\
&= 0 & \text{(using (63))}
\end{aligned}$$

Referring back to equation (37), this means that $b_{\max} = 0$, and so the evolution of $\mathbf{v}^{\text{val}}$ in the direction of $\mathbf{u}_{\max}$ is governed solely by $o_{\max}$, which has random sign. We have already established that $\mathbf{v}^{\text{val}}$ evolves with a very significant component in the direction of $\mathbf{u}_{\max}$, and now we see that the sign of this component depends wholly on the random $o_{\max}$, and so we conclude that $\mathbf{v}^{\text{val}}$ can follow one of two very different paths, depending on the random starting vector $\mathbf{v}_{\text{o}}^{\text{val}}$. It is hoped that the two paths will lead to the two optimal solutions, corresponding to traversing the shortest path in opposite directions. To verify this we examine computer simulations of the network being used to solve a specific 10-city problem.

The ten cities were placed randomly within the unit square, and then the shortest Hamilton path was found by exhaustive search. The cities were then rearranged in the order of the shortest path before being passed to the modified Hopfield network. In this way we know that the two optimal solutions $\mathbf{V}(\mathbf{p})$ are the identity and reversed identity matrix. Figure 3 shows the ten cities in their optimal ordering, as well as the shortest Hamilton path through them. The modified Hopfield network was initialized with a small random vector $\mathbf{v}_{\text{o}}^{\text{val}}$, and then allowed to converge to a hypercube corner. Figure 4 shows the network output at several stages of convergence. In the decomposition of $\mathbf{v}^{\text{val}}$, the leftmost 100 bars represent the v_i of equation (36). The v_i are ordered such that those with associated negative eigenvalues are to the left and those with positive eigenvalues to the right. Hence the component along the dominant eigenvector $\mathbf{u}_{\max}$ appears in position 100. The bar to the right of this, at position 110, indicates the component of $\mathbf{v}^{\text{val}}$ along $\mathbf{b}$ — ie. it measures $(\mathbf{b}^T\mathbf{v}^{\text{val}})/|\mathbf{b}|$. The plots confirm what the linear theory predicted: as early as iteration 10 it is clear that $\mathbf{v}^{\text{val}}$ has taken off in the direction of $\mathbf{b}$, by iteration 100 $\mathbf{v}^{\text{val}}$ contains a dominant component in the direction of $-\mathbf{u}_{\max}$, and when fully converged there are significant components along both $\mathbf{b}$ and $\mathbf{u}_{\max}$. We note in passing that the network has converged to the identity matrix, and has therefore found the optimal solution. Figure 5 shows the same problem being solved again, only this time initializing the network with $-\mathbf{v}_{\text{o}}^{\text{val}}$. As predicted, this time the component along $\mathbf{u}_{\max}$ comes in with positive sign, leading to the other optimal solution.

5.4 The Auxiliary Linear Problem

Having seen that the linear analysis of the earlier sections does indeed reflect the initial convergence of the network, we are now in a position to investigate the likely extent of the network's success on the Hamilton path problem. Clearly, the network may fail to find the optimal solution if either $\mathbf{b}$ or $\mathbf{u}_{\max}$ takes $\mathbf{v}^{\text{val}}$ into a completely different region of space from the optimal hypercube corner. In order to gain a better understanding of what $\mathbf{b}$ does, we can associate $\mathbf{b}$ with an **auxiliary linear problem**. First we rewrite the problem's objective function (20) for $\mathbf{v}$ lying on the valid subspace (22):

$$E^{\text{op}} = -\tfrac{1}{2}(\mathbf{T}^{\text{val}}\mathbf{v} + \mathbf{s})^T\mathbf{T}^{\text{op}}(\mathbf{T}^{\text{val}}\mathbf{v} + \mathbf{s}) - (\mathbf{i}^{\text{op}})^T(\mathbf{T}^{\text{val}}\mathbf{v} + \mathbf{s}) \tag{66}$$

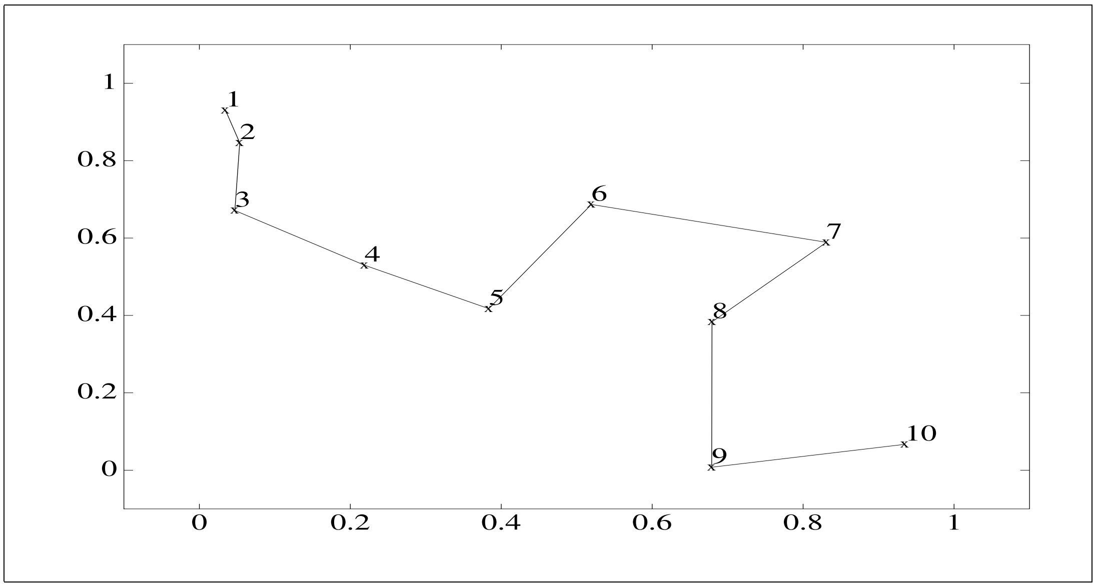


Figure 3: The 10 city positions with their shortest Hamilton path.

Simplifying equation (66), we obtain to within a constant

$$E^{\text{op}} = -\tfrac{1}{2}\mathbf{v}^T\mathbf{A}\mathbf{v} - \mathbf{b}^T\mathbf{v} \tag{67}$$

So we can associate the action of $\mathbf{b}$ with the minimization of an auxiliary linear problem with objective function

$$E^{\text{alp}} = -\mathbf{b}^T\mathbf{v} \tag{68}$$

The overall network dynamics, rewritten in terms of $\mathbf{A}$ and $\mathbf{b}$, are

$$\dot{\mathbf{v}} = \dot{\mathbf{v}}^{\text{val}} = -\nabla E^{\text{op}} = \mathbf{A}\mathbf{v} + \mathbf{b} \tag{69}$$

At the start of convergence, when $\mathbf{v}$ is near the centre of the valid subspace (ie. $\mathbf{v} \approx \mathbf{s}$), we have $\mathbf{A}\mathbf{v} \approx \mathbf{0}$, and so $\dot{\mathbf{v}} \approx \mathbf{b}$. But $\mathbf{b} = -\nabla E^{\text{alp}}$, so we conclude that the initial behaviour of the network is concerned largely with minimizing E^{alp}. For the Hamilton path problem, substituting for $\mathbf{b}$ gives

$$\begin{aligned}
E^{\text{alp}} &= -(\mathbf{s}^T\mathbf{T}^{\text{op}} + \mathbf{i}^{\text{op}T})\mathbf{T}^{\text{val}}\mathbf{v} \\
&= -\tfrac{1}{n}(\mathbf{o}^{nT} \otimes \mathbf{o}^{nT})(\mathbf{P} \otimes \mathbf{Q})(\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{v} \\
&= -\tfrac{1}{n}(\mathbf{o}^{nT}\mathbf{P}\mathbf{R}^n \otimes \mathbf{o}^{nT}\mathbf{Q}\mathbf{R})\operatorname{vec}(\mathbf{V}) && \text{(using (6), (52))} \\
&= -\tfrac{1}{n}\operatorname{vec}(\mathbf{o}^{nT}\mathbf{Q}\mathbf{R}^n\mathbf{V}\mathbf{R}^n\mathbf{P}\mathbf{o}^n) && \text{(using (8))} \\
&= -\tfrac{1}{n}\operatorname{trace}(\mathbf{o}^{nT}\mathbf{Q}\mathbf{R}^n\mathbf{V}\mathbf{R}^n\mathbf{P}\mathbf{o}^n) && \text{(since the argument is a scalar)} \\
&= -\operatorname{trace}(\mathbf{R}^n\mathbf{V}\mathbf{R}^n\mathbf{P}\mathbf{S}\mathbf{Q}) && \text{(using (4), (46))}
\end{aligned}$$

Adding a constant term, we obtain

$$\begin{aligned}
E^{\text{alp}} &= -\operatorname{trace}(\mathbf{R}^n\mathbf{V}\mathbf{R}^n\mathbf{P}\mathbf{S}\mathbf{Q} + \mathbf{S}\mathbf{P}\mathbf{S}\mathbf{Q}) \\
&= -\operatorname{trace}((\mathbf{R}^n\mathbf{V}\mathbf{R}^n + \mathbf{S})\mathbf{P}\mathbf{S}\mathbf{Q}) \\
&= -\operatorname{trace}(\mathbf{V}\mathbf{P}\mathbf{S}\mathbf{Q}) \quad \text{(for } \mathbf{V} \text{ satisfying (53))} \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\left[(\mathbf{V}(-\mathbf{P}))^T\right]_{ij}[\mathbf{S}\mathbf{Q}]_{ij} \qquad (70)
\end{aligned}$$

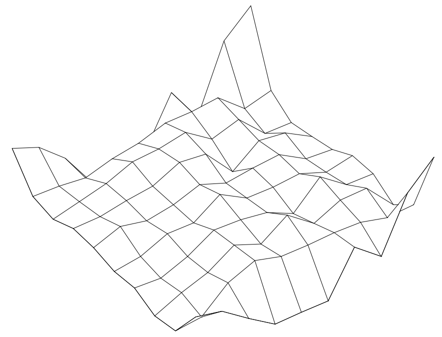
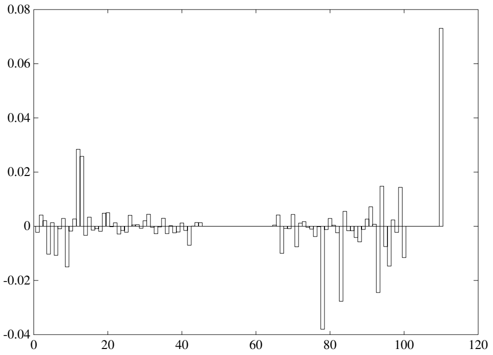
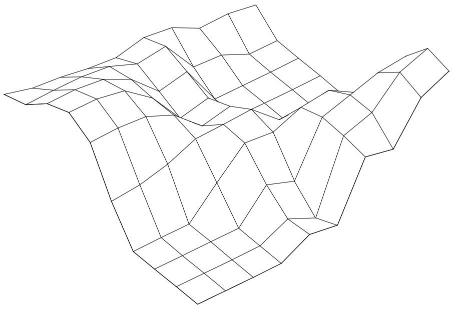
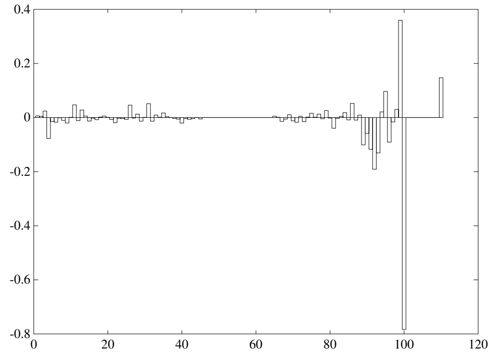
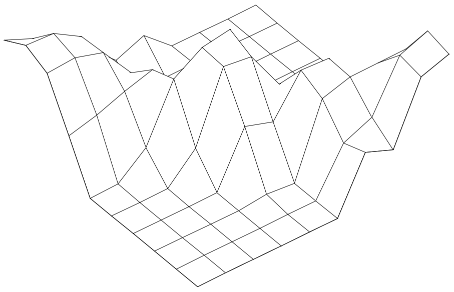
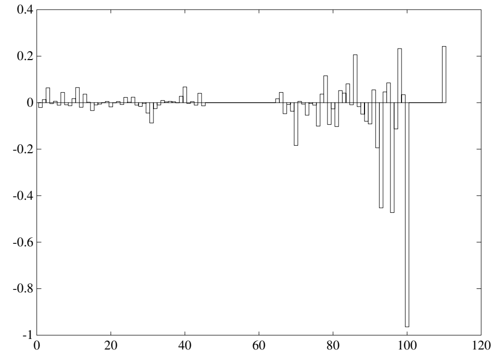
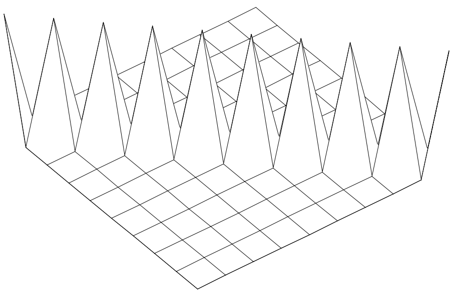
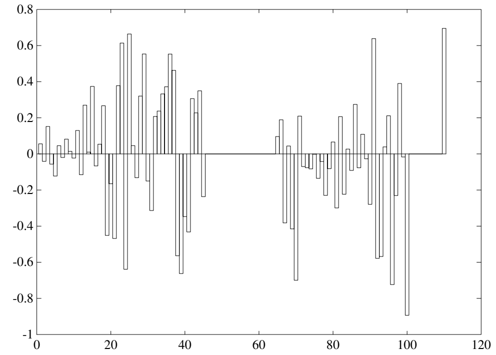
Mesh plot of $\mathbf{V}$	Decomposition of $\mathbf{v}^{\mathrm{val}}$
 Iteration 10	 Iteration 10
 Iteration 100	 Iteration 100
 Iteration 200	 Iteration 200
 Fully Converged	 Fully Converged

Figure 4: 10 city Hamilton path problem at various stages of convergence.

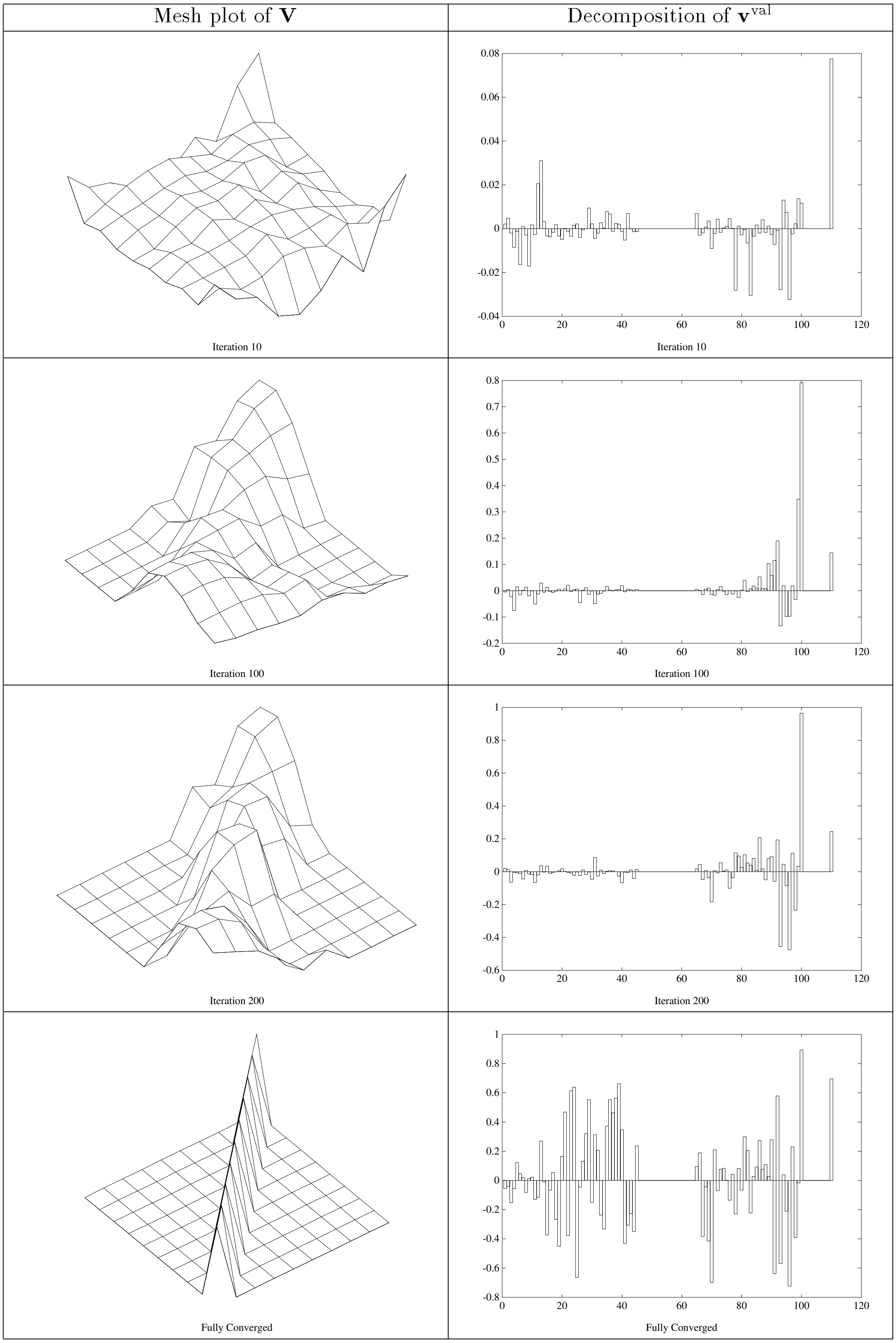


Figure 5: 10 city Hamilton path problem with negated starting vector.

Now, the matrix $\mathbf{SQ}$ is given as follows

$$[\mathbf{SQ}]_{ij} = \begin{cases} 1/n & \text{if} \quad j \in \{1, n\} \\ 2/n & \text{if} \quad 2 \leq j \leq (n-1) \end{cases} \tag{71}$$

For $n = 5$, $\mathbf{SQ}$ would be

$$\begin{bmatrix} 0.2 & 0.4 & 0.4 & 0.4 & 0.2 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.2 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.2 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.2 \\ 0.2 & 0.4 & 0.4 & 0.4 & 0.2 \end{bmatrix}$$

To minimize E^{alp}, $\mathbf{V}$ must reorder the rows of $(-\mathbf{P})$ such that the two rows with the largest summed elements are moved to the top and bottom of the matrix $\mathbf{V}(-\mathbf{P})$. This is equivalent to ensuring that the two cities with the largest summed distances from all the other cities are placed first and last in the journey. If this is done, then the contribution of these large distances to the cost of E^{alp} is halved compared with what would have been the case had the cities not been placed first and last. In this way, we can associate with the auxiliary linear problem a certain strategy, that being to place the two most remote cities first and last in the tour; the network follows this strategy in the early stages of convergence.

Armed with this new insight, we can now address the question of how well the network can tackle the general Euclidean Hamilton path problem. The auxiliary linear problem suggests that the network may fail for problems in which the optimal tour deviates significantly from starting and finishing at the most remote cities. Such a problem is illustrated in Figure 6. Here we see sixteen cities arranged in a spiral pattern, and numbered so that the optimal solutions $\mathbf{V}$ are the identity and reverse identity matrices. The optimal path length is 54.0 units. Figure 7 shows that the network behaves as predicted by the linear analysis, putting in large $\mathbf{b}$ and $\mathbf{u}_{\max}$ components from the start, only this time a sub-optimal solution is reached (path length 56.2 units), with the journey starting and ending at outlying cities — see Figure 8. Clearly this is a result of the auxiliary linear problem being poorly related to the parent Hamilton path problem in this case. Indeed, Figure 9 shows that the optimal solution's $\mathbf{v}^{\text{val}}$ contains a small *negative* component in the direction of $\mathbf{b}$. While the network is sometimes capable of completely overturning an initial bias towards the vector $\mathbf{b}$, this capability is highly dependent on the random starting vector $\mathbf{v}_{\text{o}}^{\text{val}}$, and more often than not poor solutions like the one in Figure 8 are obtained.

6 The Point Matching Problem

6.1 Problem Formulation and Mapping

The point matching problem as presented in this section forms part of an invariant pattern recognition system. The aim is to compare a set of unknown points to a set of template points and compute some measure of similarity between the two sets. However, before the similarity can be computed, each of the unknown points has to be matched with a particular template point. Suppose the template and unknown point sets contain the same number of points n: let the template set be called $\mathcal{Q}$ and the unknown point set be called $\mathcal{P}$. Let us also define distance matrices for these two sets, $\mathbf{P}$ and $\mathbf{Q}$, where $[\mathbf{P}]_{ij}$ is the distance between unknown points i and j, and $[\mathbf{Q}]_{kl}$ is the distance between template points k and l. Then we can match unknown points with template points using a permutation $\mathbf{p}$, where $[\mathbf{p}]_i = j$ if unknown point j is to be matched with template point i. A distance measure which is invariant to translation and rotation of the point sets is then

$$d(\mathbf{p}) = \left| \mathbf{V}(\mathbf{p}) \mathbf{P} \mathbf{V}(\mathbf{p})^T - \mathbf{Q} \right|^2 \tag{72}$$

where $\mathbf{V}(\mathbf{p})$ is the permutation matrix associated with $\mathbf{p}$. The objective of the point matching problem is to find the permutation $\mathbf{p}$ which minimizes the distance $d(\mathbf{p})$ between the two sets — ie.

$$\min_{\mathbf{p}} \; d(\mathbf{p}) \quad \text{where} \quad d(\mathbf{p}) = \left| \mathbf{V}(\mathbf{p}) \mathbf{P} \mathbf{V}(\mathbf{p})^T - \mathbf{Q} \right|^2 \tag{73}$$

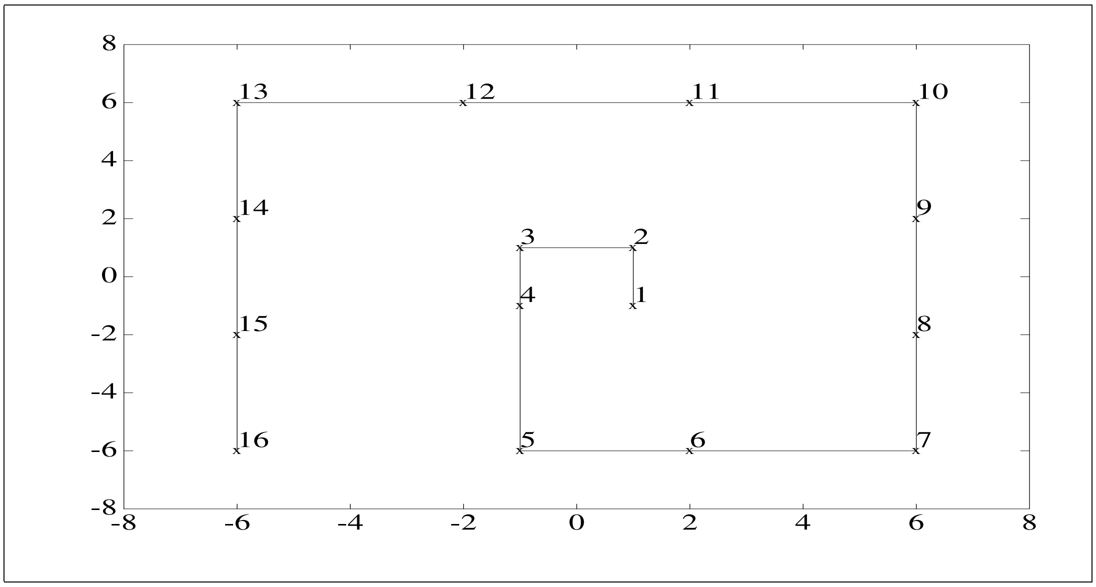


Figure 6: The 16 city positions with their shortest Hamilton path.

This problem is similar to the optimization problems found in the graph matching approaches to invariant pattern recognition [4, 11], with the added constraint that the two sets contain the same number of points. Note that invariance to scaling can be added by normalizing the moduli of the two distance matrices being compared, such that $|\mathbf{P}| = |\mathbf{Q}|$.

We now show how this problem may be mapped onto the Hopfield network. The valid subspace is exactly the same as in the Hamilton path problem, because in each case we are trying to find a permutation matrix $\mathbf{V}(\mathbf{p})$, so $\mathbf{T}^{\mathrm{val}}$ and $\mathbf{s}$ are as given in equations (54), (55) respectively. It is now necessary to derive expressions for $\mathbf{T}^{\mathrm{op}}$ and $\mathbf{i}^{\mathrm{op}}$. We start by expanding the expression for $d(\mathbf{p})$:

$$\begin{aligned} d(\mathbf{p}) &= \left|\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T - \mathbf{Q}\right|^2 \\ &= \mathrm{trace}\left[(\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T - \mathbf{Q})^T(\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T - \mathbf{Q})\right] \\ &= \mathrm{trace}\left[\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T - 2\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T\mathbf{Q} + \mathbf{Q}^T\mathbf{Q}\right] \\ &= |\mathbf{P}|^2 + |\mathbf{Q}|^2 - 2\ \mathrm{trace}\left[\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T\mathbf{Q}\right] \qquad (74) \end{aligned}$$

Eliminating the constants from the above expression, we can set E^{op}as

$$E^{\mathrm{op}} = -\tfrac{1}{2}\ \mathrm{trace}\left[\mathbf{V}(\mathbf{p})\mathbf{P}\mathbf{V}(\mathbf{p})^T\mathbf{Q}\right] \qquad (75)$$

This is of exactly the same form as the expression for E^{op} in the Hamilton path problem, and so we conclude that

$$\begin{aligned} \mathbf{T}^{\mathrm{op}} &= (\mathbf{P} \otimes \mathbf{Q}) \qquad (76) \\ \mathbf{i}^{\mathrm{op}} &= \mathbf{0} \qquad (77) \end{aligned}$$

6.2 The Auxiliary Linear Problem

Since both $\mathbf{P}$ and $\mathbf{Q}$ are problem dependent, it is impossible to say anything definite about their eigenvectors, and hence we cannot derive any interesting relationship between $\mathbf{u}_{\max}$ and $\mathbf{b}$ as we did for the Hamilton path problem. However, we are in a position to examine the auxiliary linear

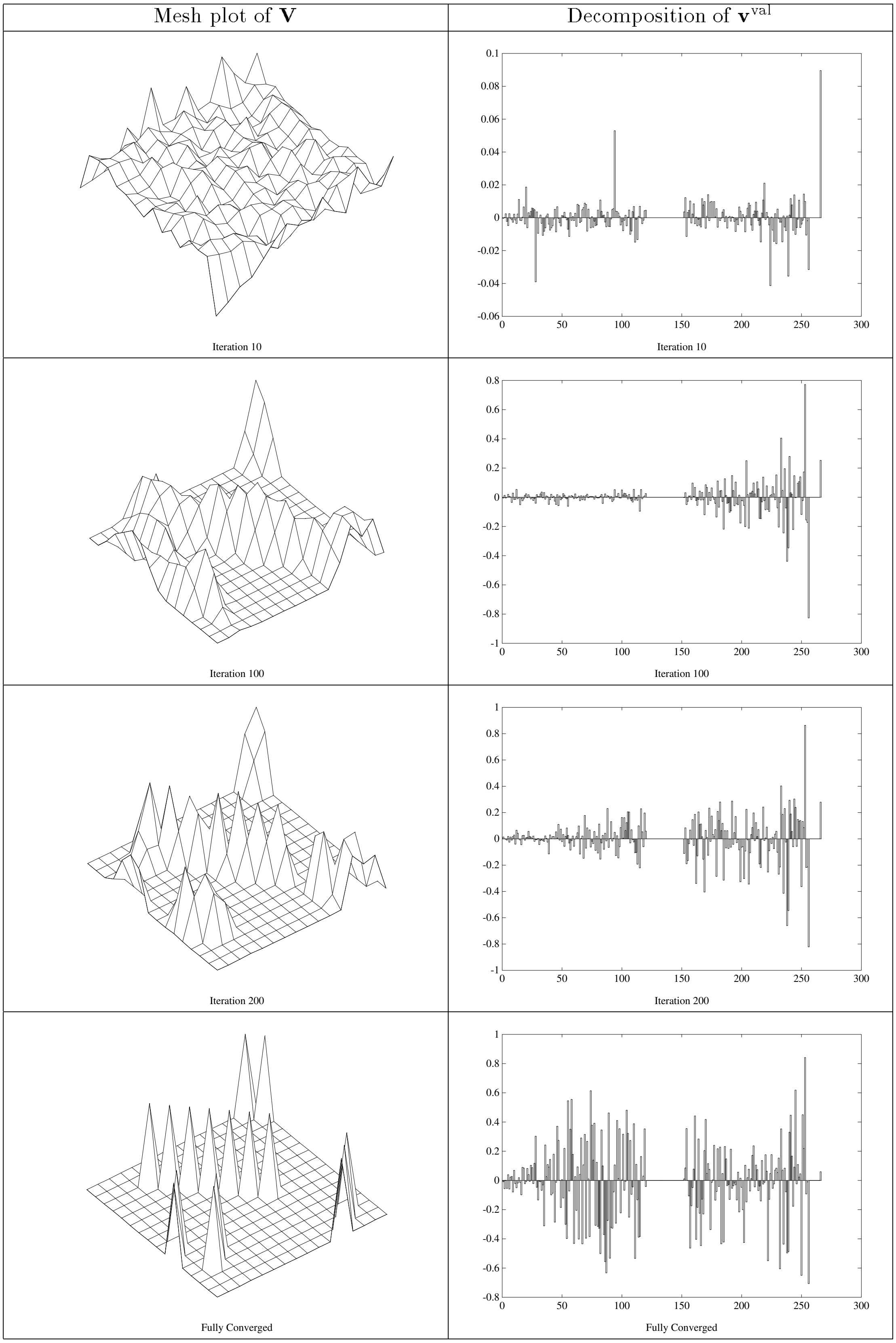


Figure 7: 16 city Hamilton path problem at various stages of convergence.

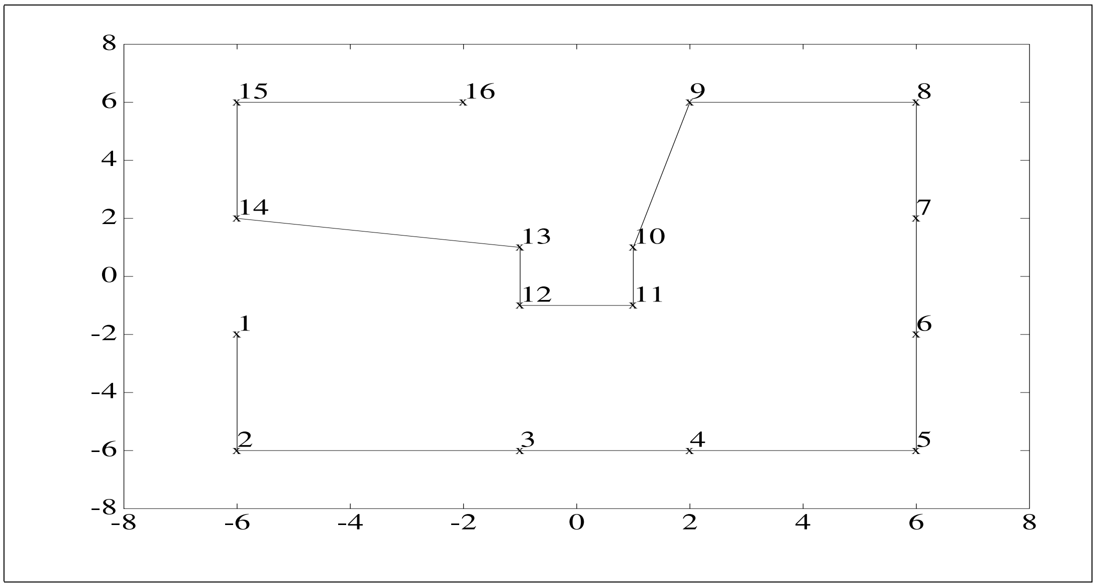
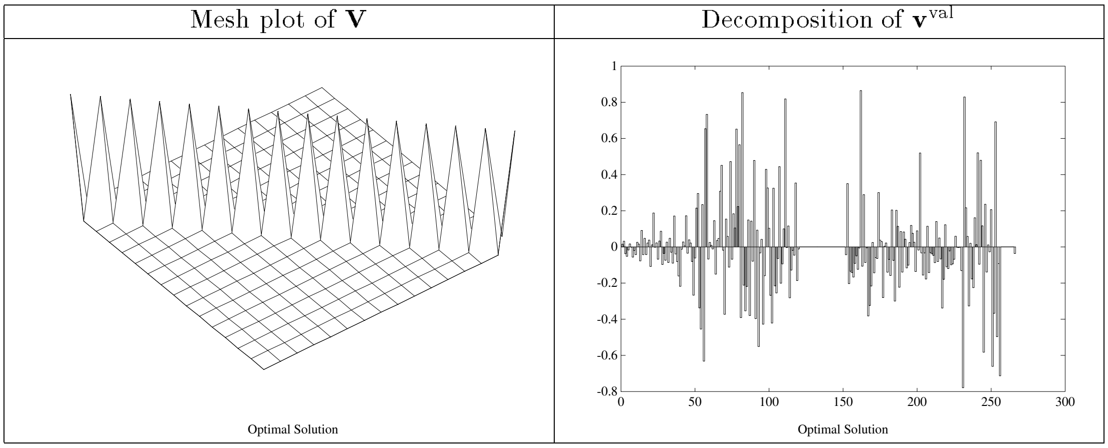


Figure 8: The 16 city positions with the solution found by the modified Hopfield network.


Mesh plot of $\mathbf{V}$

Decomposition of $\mathbf{v}^{\mathrm{val}}$

Figure 9: Optimal solution to the 16 city Hamilton path problem.

problem. Making use of the congruence between this mapping and that of the Hamilton path problem, we can jump straight to equation (70):

$$\begin{aligned} E^{\mathrm{alp}} &= -\operatorname{trace}[\mathbf{VPSQ}] && (\text{for } \mathbf{V} \text{ satisfying (53)}) \\ &= -\tfrac{1}{n}\operatorname{trace}\left[\mathbf{VPo}^n\mathbf{o}^{nT}\mathbf{Q}\right] \\ &= -\tfrac{1}{n}\operatorname{trace}\left[(\mathbf{Qo}^n)^T\mathbf{V}(\mathbf{Po}^n)\right] && (\text{using (4)}) \end{aligned}$$

Now, the vectors $\mathbf{Qo}^n$ and $\mathbf{Po}^n$ simply represent the summed distances of all the other points from each point in the template or unknown, since

$$[\mathbf{Qo}^n]_i = \sum_{j=1}^{n}[\mathbf{Q}]_{ij}\ , \quad [\mathbf{Po}^n]_i = \sum_{j=1}^{n}[\mathbf{P}]_{ij}$$

In order to minimize E^{alp}, $\mathbf{V}$ must reorder the elements of $\mathbf{Po}^n$ so that large elements appear in the same rows as the large elements of $\mathbf{Qo}^n$, whilst small elements appear in the same rows as the small elements of $\mathbf{Qo}^n$. In other words, the auxiliary linear problem matches outlying points in $\mathcal{P}$ with outlying points in $\mathcal{Q}$, and central points in $\mathcal{P}$ with central points in $\mathcal{Q}$.

As with the Hamilton path problem, we must now address whether the auxiliary linear problem is in sympathy with the parent point matching problem. To do this, we study several typical 20-point matching problems for which good solutions have been found — see Figure 10. For each problem, Figure 11 shows the decomposition of the solution's $\mathbf{v}^{\mathrm{val}}$ along $\mathbf{b}$ and the eigenvectors of $\mathbf{A}$; also shown is the decomposition of $\mathbf{b}$ along the same vectors. For clarity, only those components corresponding to the largest positive eigenvalues are displayed, as well as the component along $\mathbf{b}$. It seems that all the good solutions contain significant components along $\mathbf{u}_{\mathrm{max}}$, a strong prerequisite for a successful mapping. It is also apparent that the solutions contain significant components along $\mathbf{b}$, indicating a good degree of sympathy between the auxiliary linear problem and the parent problem. However, the Figure also shows that for some problems the vectors $\mathbf{b}$ and $\mathbf{u}_{\mathrm{max}}$ are nearly mutually orthogonal. This means, as in the case of the Hamilton path problem, that the evolution of $\mathbf{v}^{\mathrm{val}}$ in the direction of $\mathbf{u}_{\mathrm{max}}$ is governed solely by o_{max}, which has random sign. However, unlike the Hamilton path problem, in which each of the two possible routes led to an optimal solution, the point matching problem has only a single optimal solution, and so one of the routes is bound to lead to a sub-optimal hypercube corner. This hypothesis is confirmed in Figure 12, which shows the result of solving the '4' problem of Figure 10 twice on a modified Hopfield network, starting once with $\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$ and then again with $-\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$. As expected, the component of $\mathbf{v}^{\mathrm{val}}$ along $\mathbf{u}_{\mathrm{max}}$ develops different signs in the two runs, leading to two different solutions, one with $d(\mathbf{p}) = 0.157$, the other with $d(\mathbf{p}) = 0.346$. The unknown point labellings corresponding to these solutions are shown in Figure 13; these should be compared with the template in Figure 10.

It is reassuring to observe in Figure 11 that in none of the problems does the vector $\mathbf{b}$ contain a significant component along $\mathbf{u}_{\mathrm{max}}$ with the opposite sign to the good solution's component in that direction. If this were the case, then the action of $\mathbf{b}$ would invariably introduce the $\mathbf{u}_{\mathrm{max}}$ component into $\mathbf{v}^{\mathrm{val}}$ with the wrong sign, leading most probably to poor solutions. So, unlike the Hamilton path problem, we cannot say here that the auxiliary linear problem is antagonizing the parent problem. Instead, the auxiliary linear problem simply doesn't contain enough information to steer $\mathbf{v}^{\mathrm{val}}$ in the right direction from the outset. This is not such a serious shortcoming as in the Hamilton path problem, in that this time we can be sure of obtaining one good solution if we run the network twice, negating the random $\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$ between runs. This may not be a particularly elegant solution, but it is certainly very effective.

6.3 Simulation Results

As a small illustrative experiment, the network was used in a pattern recognition system to recognize instances of the ten handwritten digits. The template and unknown digits used in the experiment are shown in Figure 14. A point positioning algorithm was used to place 20 points around the outlines of the digits; each of the ten unknowns were then compared with each of the

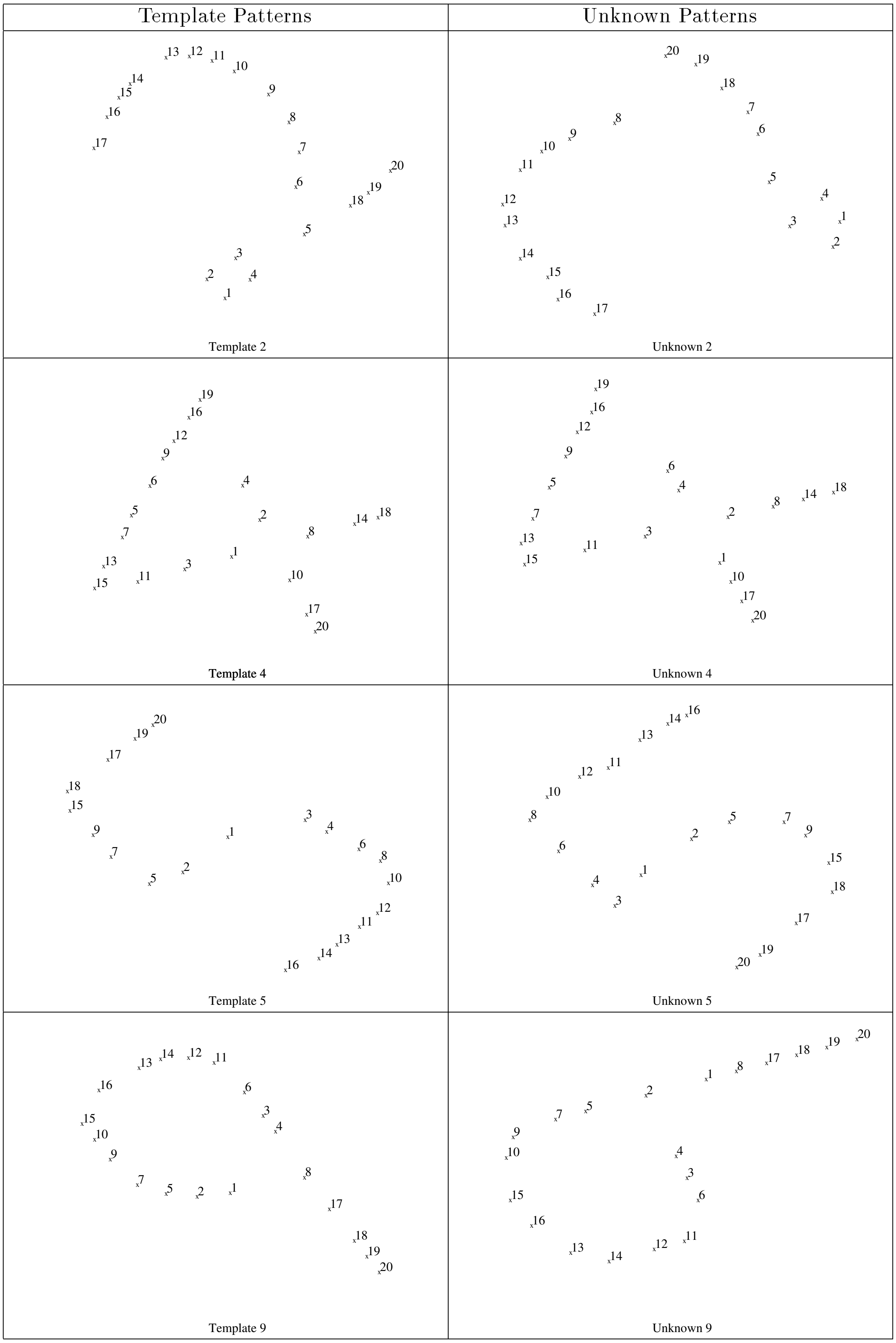


Figure 10: Template and unknown patterns with labelling for a good match.

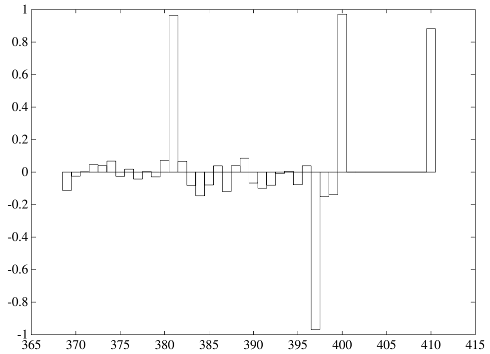
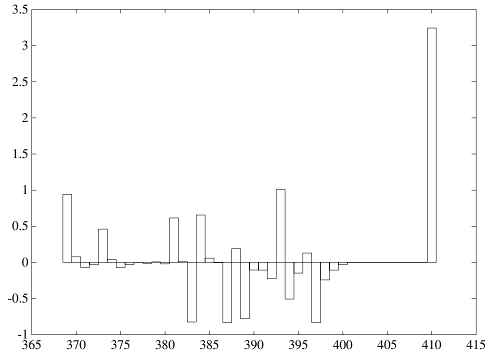
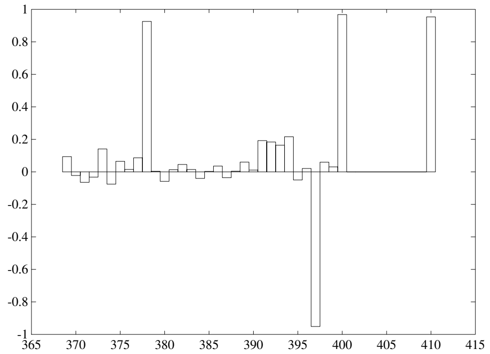
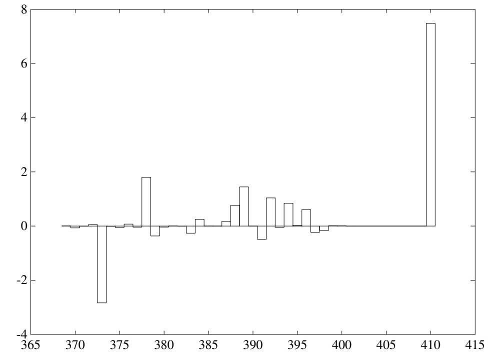
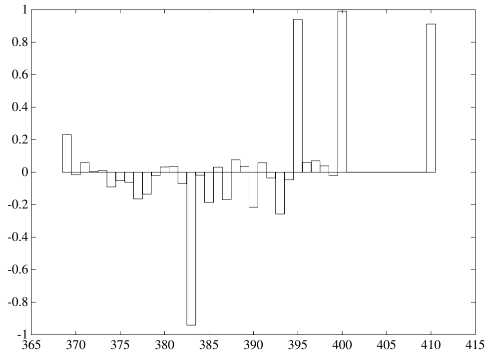
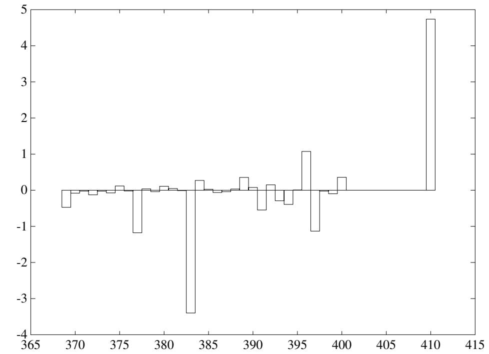
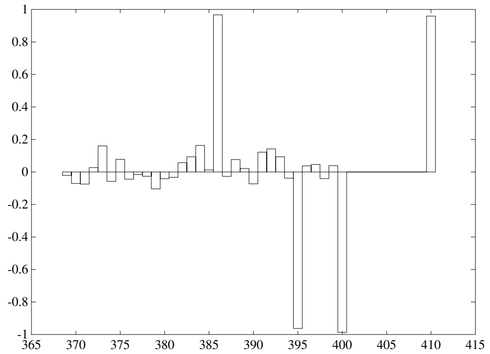
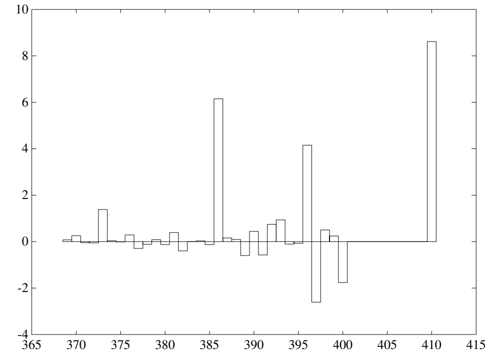
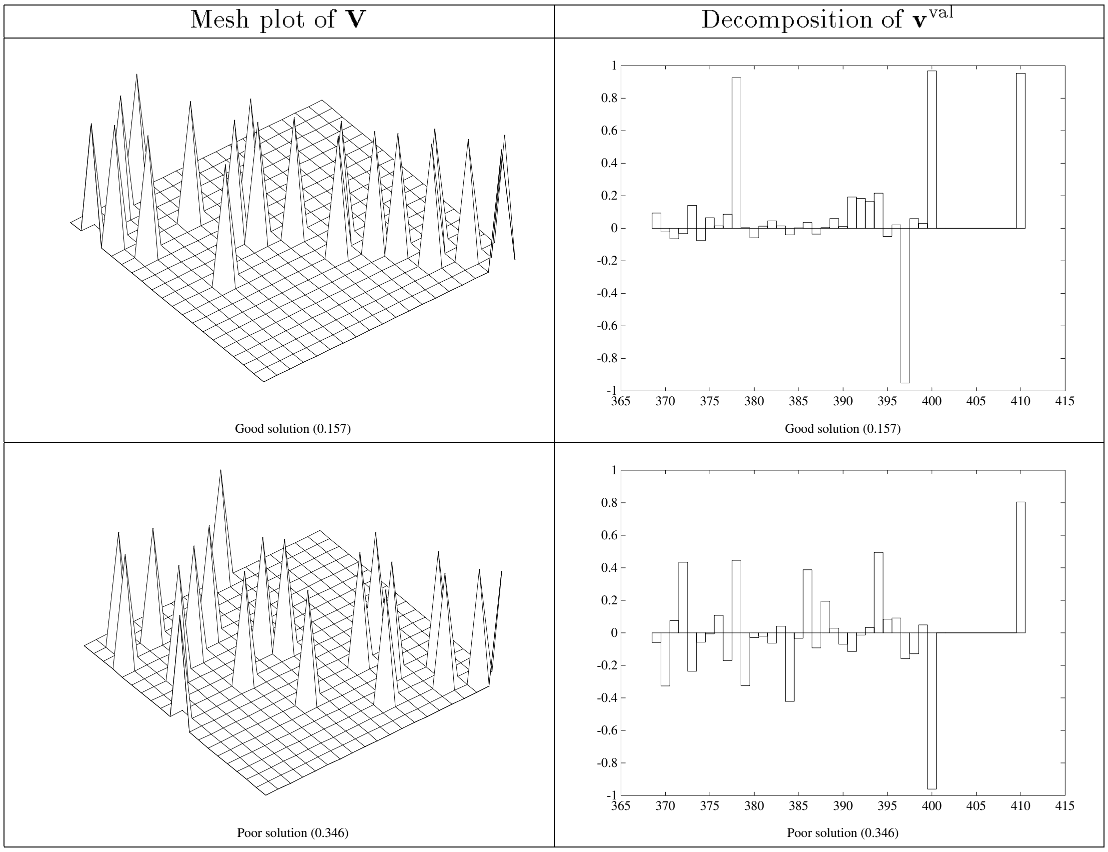
Solution's $\mathbf{v}^{\text{val}}$ Decomposition	**b** Decomposition
 Solution Decomposition for 2 problem	 b Decomposition for 2 problem
 Solution Decomposition for 4 problem	 b Decomposition for 4 problem
 Solution Decomposition for 5 problem	 b Decomposition for 5 problem
 Solution Decomposition for 9 problem	 b Decomposition for 9 problem

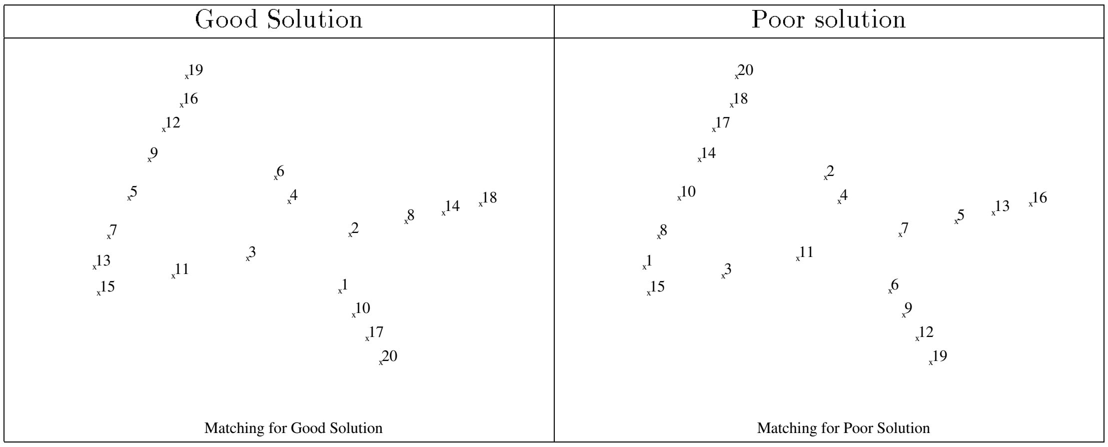
Figure 11: Decompositions for the point matching problems.



Mesh plot of $\mathbf{V}$

Decomposition of $\mathbf{v}^{\mathrm{val}}$

Figure 12: Mesh plots and decompositions for the two solutions to the ‘4’ problem.



Good Solution

Poor solution

Figure 13: Point labellings for good and poor solutions to the ‘4’ problem.

ten templates, using the modified Hopfield network to solve the point matching problems. The network was run twice on each comparison to ensure a good solution was found at least once. The results are displayed in Table 1. The only misclassification was that of the '6', which was mistaken for a '9'; this is clearly a result of the rotational invariance of the recognition scheme.

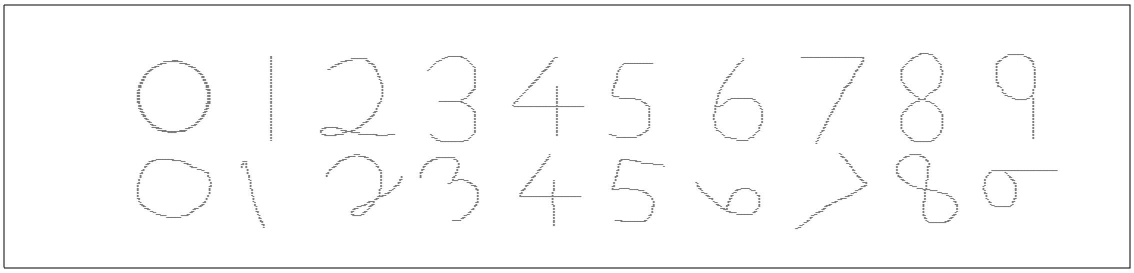


Figure 14: Template (top) and unknown (bottom) digits used in the experiment.

Unknown Digit	**Template Digit**									
	0	1	2	3	4	5	6	7	8	9
0	**0.98**	3.82	2.13	2.16	2.76	2.30	2.12	3.08	2.25	2.81
1	4.49	**0.55**	2.67	2.31	3.33	1.97	3.00	2.51	2.30	3.06
2	2.07	3.20	**1.44**	2.46	2.28	2.16	2.56	1.99	2.96	2.87
3	3.92	1.95	1.85	**1.17**	2.61	1.46	2.22	1.89	2.18	2.51
4	2.88	3.11	2.30	2.30	**1.57**	2.25	2.35	2.63	2.17	2.86
5	2.44	2.95	2.16	2.14	1.95	**1.43**	2.32	2.24	1.93	2.33
6	3.51	2.50	3.06	2.42	2.84	1.97	1.56	2.24	2.16	**1.39**
7	4.36	1.63	2.60	2.24	2.67	1.75	2.55	**1.23**	2.43	2.30
8	3.19	2.18	2.33	1.67	2.87	1.81	2.34	2.73	**1.33**	2.54
9	3.13	2.74	2.92	1.99	2.68	2.19	1.33	2.02	2.07	**1.20**

Table 1: Distance measures for all templates against all unknowns; shortest distances are shown in bold type.

7 Discussion and Conclusions

There have been many papers in the past reporting the discovery of neat mappings of combinatorial optimization problems onto the Hopfield network for solution, but few have reported in-depth results of the network's performance. Indeed, it is often the case that the solutions obtained from the network are not as good as might be expected given the integrity of the mapping, though there has been little attempt to explain this in the past. In this paper we have introduced a new level of understanding into the field: we have demonstrated how it is possible to estimate the likely efficacy of a particular mapping by way of an eigenvector analysis of the network's dynamics and an investigation of the structure of 'good' solutions relative to this eigenvector basis.

Although not observed in experiments reported in this paper, it seems likely that a major cause of mapping failure would be that a good solution's $\mathbf{v}^{\mathrm{val}}$ contains an insignificant component along the dominant eigenvector $\mathbf{u}_{\mathrm{max}}$. The other principle mode of failure arises through inappropriate action of the vector $\mathbf{b}$. In the course of our argument, we have introduced the concept of an 'auxiliary linear problem', which we have found useful in gaining a greater understanding of the 'strategy' associated with $\mathbf{b}$. It remains to be pointed out that not all problems have an auxiliary linear problem: for example, the classic mappings of the Travelling Salesman and Graph Partitioning problems both exhibit the property $\mathbf{b} = \mathbf{0}$. The absence of an auxiliary linear problem is to be associated with a local maximum of the objective function at the centre of the valid subspace, and so the initial direction of $\mathbf{v}$ from this point is entirely random. For the aforementioned two problems, this is consistent with the distribution of the degenerate optimal hypercube corners, which are positioned evenly around the perimeter of the valid subspace. For problems which lack this degeneracy of solution, it is indeed *essential* that the mapping yields an auxiliary linear problem, else the network output is most likely to head away from the optimal hypercube corner from the outset.

However, the presence of an auxiliary linear problem where it is required is not enough to guarantee the success of a mapping. It is also necessary that the auxiliary linear problem be *in sympathy* with the corresponding parent problem. By 'in sympathy', we mean that the $\mathbf{v}^{\mathrm{val}}$ corresponding to good problem solutions must not contain large negative components along $\mathbf{b}$. Furthermore, for problems with a unique optimal solution (eg. the point matching problem), it is necessary that the auxiliary linear problem contain correct information as to the direction of a good solution's $\mathbf{v}^{\mathrm{val}}$ component along $\mathbf{u}_{\mathrm{max}}$. If this information is incorrect, or lacking, then the $\mathbf{u}_{\mathrm{max}}$ component may be introduced with the wrong sign, most probably leading to a poor quality solution. For cases in which the $\mathbf{u}_{\mathrm{max}}$ information is lacking, as with some point matching problems, a good solution can be found by running the network twice, starting once with $\mathbf{v}^{\mathrm{val}} = \mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ and then again with $\mathbf{v}^{\mathrm{val}} = -\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$.

To conclude, the results of this paper demonstrate how it is possible to predict the likely success of a problem mapping without having to resort to trial and error. Weaknesses in the mapping are highlighted by investigating the decomposition of good solutions along the eigenvectors of the network's matrix $\mathbf{A}$ and the auxiliary linear problem vector $\mathbf{b}$. This eigenvector analysis sheds new light on the problem mapping by placing it in the context of the network's dynamics. There is often a certain degree of flexibility in the way a particular problem is mapped onto the network for solution; the analysis presented in this paper seeks to go one step further towards successfully exploiting that flexibility.

References

[1] Aiyer, S. V. B., Niranjan, M. & Fallside, F. *A Theoretical Investigation into the Performance of the Hopfield Model*, IEEE Transactions on Neural Networks, Vol. 1, No. 2, June 1990.

[2] Aiyer, S. V. B. & Fallside, F. *A Subspace Approach to Solving Combinatorial Optimization Problems with Hopfield Networks*, Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR 55, December 1990.

[3] Aiyer, S. V. B. & Fallside, F. *A Hopfield Network Implementation of the Viterbi Algorithm for Hidden Markov Models*, Proc. IJCNN Seattle 1991, Vol. 2, pp. 827–832, July 1991.

[4] Bienenstock, E. & Doursat, R. *Elastic Matching and Pattern Recognition in Neural Networks*, Neural Networks: From Models to Applications, L. Personnaz and G. Dreyfus eds., IDSET, Paris, 1989.

[5] Gee, A. H., Aiyer, S. V. B. & Prager, R. W. *A Subspace Approach to Invariant Pattern Recognition using Hopfield Networks*, Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR 62, 1991.

[6] Gislen, L., Peterson, C. & Söderberg B. *"Teachers and Classes" with Neural Networks*, International Journal of Neural Systems, Vol. 1, No. 2, 1989.

[7] Graham, A. *Kronecker Products and Matrix Calculus: with Applications*, Ellis Horwood Ltd, Chichester 1981.

[8] Hopfield J. J. *Neurons with graded response have collective computational properties like those of two-state neurons*, Proc. Natl. Acad. USA. Vol. 81, pp. 3088–3092, May 1984.

[9] Hopfield, J. J. & Tank, D. W. *"Neural" Computation of Decisions in Optimization Problems*, Biological Cybernetics, Vol. 52, pp. 141–152, 1985.

[10] Kamgar-Parsi, B. & Kamgar-Parsi, B. *On Problem Solving with Hopfield Neural Networks*, Biological Cybernetics, Vol. 62, pp. 415–423, 1990.

[11] Li, W. & Nasrabadi, N. M. *Object Recognition based on Graph Matching Implemented by a Hopfield-Style Neural Network*, Proc. IJCNN Washington DC, 1989.

[12] Peterson, C. & Anderson, J. R. *Neural Networks and NP-complete Optimization Problems; A Performance Study on the Graph Bisection Problem*, Complex Systems 2:1, 1988.

[13] Peterson, C. & Söderberg, B. *A new method for mapping optimization problems onto neural networks*, International Journal of Neural Systems, Vol. 1, No. 1, 1989.

[14] Rosenbrook, H. H. & Storey, C. *Mathematics of Dynamical Systems*, Thomas Nelson and Sons, London 1970.

[15] Tank, D. W. & Hopfield, J. J. *Simple "Neural" Optimization Networks: An A/D Converter, Signal Decision Circuit, and a Linear Programming Circuit*, IEEE Transactions on Circuits and Systems, Vol. CAS-33, No. 5, May 1986.

[16] Van den Bout, D. E. & Miller III, T. K. *Improving the Performance of the Hopfield-Tank Neural Network Through Normalization and Annealing*, Biological Cybernetics, Vol. 62, pp. 129–139, 1989.

[17] Van den Bout, D. E. & Miller III, T. K. *Graph Partitioning using Annealed Neural Networks*, IEEE Transactions on Neural Networks, Vol. 1, No. 2, June 1990.

[18] Wilson, V. & Pawley, G. S. *On them Stability of the TSP Problem Algorithm of Hopfield and Tank*, Biological Cybernetics, Vol. 58, pp. 63–70, 1988.

Appendix A

Invariance of Results to Different Problem Instances

In this Appendix we examine how the decomposition of a particular solution's $\mathbf{v}^{\mathrm{val}}$ along $\mathbf{b}$ and the eigenvectors of $\mathbf{A}$ is affected by different instances of the same problem. Any of the problems studied in this paper may be expressed in many different ways: for example, an n-city Hamilton path problem may be expressed in $n!$ ways, each one corresponding to a different ordering of the cities in the matrix $\mathbf{P}$. For the point matching problem, both the template and unknown point orderings may vary. For the results of this paper to have any significance, it is necessary to prove that the decomposition of a particular solution's $\mathbf{v}^{\mathrm{val}}$ along $\mathbf{b}$ and the eigenvectors of $\mathbf{A}$ is independent of problem instance.

We shall start by looking at the point matching problem. Consider a particular problem instance given by

$$\mathbf{P} = \mathbf{P}_{\mathrm{A}}\,, \quad \mathbf{Q} = \mathbf{Q}_{\mathrm{A}}$$

Call this instance A: henceforth all quantities relating to this instance will be denoted by the subscript $_{\mathrm{A}}$. Then any other instance B of the same problem may be expressed as

$$\begin{aligned} \mathbf{P}_{\mathrm{B}} &= \mathbf{V}_{\mathrm{p}}\mathbf{P}_{\mathrm{A}}\mathbf{V}_{\mathrm{p}}{}^{T} \\ \mathbf{Q}_{\mathrm{B}} &= \mathbf{V}_{\mathrm{q}}\mathbf{Q}_{\mathrm{A}}\mathbf{V}_{\mathrm{q}}{}^{T} \end{aligned}$$

where $\mathbf{V}_{\mathrm{p}}$ and $\mathbf{V}_{\mathrm{q}}$ are arbitrary permutation matrices. Any network output $\mathbf{V}_{\mathrm{A}}$ for instance A may be interpreted as a partial solution to the problem being solved. The same partial solution for instance B is then

$$\mathbf{V}_{\mathrm{B}} = \mathbf{V}_{\mathrm{q}}\mathbf{V}_{\mathrm{A}}\mathbf{V}_{\mathrm{p}}{}^{T} \tag{78}$$

This analysis can also be used for the Hamilton path problem with the added constraint that $\mathbf{V}_{\mathrm{q}} = \mathbf{I}^n$, since $\mathbf{Q}$ is the same for all problem instances. Before going any further, it is necessary to obtain two results concerning the effect of an arbitrary permutation matrix $\mathbf{V}(\mathbf{p})$ on the projection matrix $\mathbf{R}^n$:

$$\begin{aligned} \mathbf{V}(\mathbf{p})^T\mathbf{R}^n\mathbf{V}(\mathbf{p}) &= \mathbf{V}(\mathbf{p})^T\left(\mathbf{I}^n - \tfrac{1}{n}\mathbf{O}^n\right)\mathbf{V}(\mathbf{p}) \\ &= \mathbf{V}(\mathbf{p})^T\mathbf{I}^n\mathbf{V}(\mathbf{p}) - \tfrac{1}{n}\mathbf{V}(\mathbf{p})^T\mathbf{O}^n\mathbf{V}(\mathbf{p}) \\ &= \mathbf{I}^n - \tfrac{1}{n}\mathbf{O}^n = \mathbf{R}^n \end{aligned} \tag{79}$$

$$\begin{aligned} \mathbf{R}^n\mathbf{V}(\mathbf{p}) &= \left(\mathbf{I}^n - \tfrac{1}{n}\mathbf{O}^n\right)\mathbf{V}(\mathbf{p}) \\ &= \mathbf{I}^n\mathbf{V}(\mathbf{p}) - \tfrac{1}{n}\mathbf{O}^n\mathbf{V}(\mathbf{p}) \\ &= \mathbf{V}(\mathbf{p}) - \tfrac{1}{n}\mathbf{O}^n = \mathbf{V}(\mathbf{p})\mathbf{I}^n - \tfrac{1}{n}\mathbf{V}(\mathbf{p})\mathbf{O}^n \\ &= \mathbf{V}(\mathbf{p})\left(\mathbf{I}^n - \tfrac{1}{n}\mathbf{O}^n\right) = \mathbf{V}(\mathbf{p})\mathbf{R}^n \end{aligned} \tag{80}$$

We shall start by relating the vector $\mathbf{v}_{\mathrm{B}}^{\mathrm{val}}$ for instance B to instance A quantities:

$$\begin{aligned} \mathbf{v}_{\mathrm{B}}^{\mathrm{val}} &= \mathbf{T}^{\mathrm{val}}\mathbf{v}_{\mathrm{B}} \\ &= \mathbf{T}^{\mathrm{val}}\mathrm{vec}(\mathbf{V}_{\mathrm{B}}) \\ &= \mathbf{T}^{\mathrm{val}}\mathrm{vec}(\mathbf{V}_{\mathrm{q}}\mathbf{V}_{\mathrm{A}}{\mathbf{V}_{\mathrm{p}}}^T) \\ &= \mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathrm{vec}(\mathbf{V}_{\mathrm{A}}) \quad \text{(using (8))} \\ &= \mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{v}_{\mathrm{A}} \end{aligned}$$

We now go on to obtain an expression for the instance B vector $\mathbf{b}_{\mathrm{B}}$ in terms of instance A quantities:

$$\begin{aligned} \mathbf{b}_{\mathrm{B}} &= \mathbf{T}^{\mathrm{val}}\mathbf{T}_{\mathrm{B}}^{\mathrm{op}}\mathbf{s} \\ &= \tfrac{1}{n}\mathbf{T}^{\mathrm{val}}(\mathbf{P}_{\mathrm{B}} \otimes \mathbf{Q}_{\mathrm{B}})(\mathbf{o}^n \otimes \mathbf{o}^n) \\ &= \tfrac{1}{n}\mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}}\mathbf{P}_{\mathrm{A}}{\mathbf{V}_{\mathrm{p}}}^T \otimes \mathbf{V}_{\mathrm{q}}\mathbf{Q}_{\mathrm{A}}{\mathbf{V}_{\mathrm{q}}}^T)(\mathbf{o}^n \otimes \mathbf{o}^n) \\ &= \tfrac{1}{n}\mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})(\mathbf{P}_{\mathrm{A}} \otimes \mathbf{Q}_{\mathrm{A}})({\mathbf{V}_{\mathrm{p}}}^T\mathbf{o}^n \otimes {\mathbf{V}_{\mathrm{q}}}^T\mathbf{o}^n) \quad \text{(using (6))} \\ &= \tfrac{1}{n}\mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}(\mathbf{o}^n \otimes \mathbf{o}^n) \\ &= \mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}\mathbf{s} \end{aligned}$$

Hence

$$\begin{aligned} \mathbf{b}_{\mathrm{B}}^T\mathbf{v}_{\mathrm{B}}^{\mathrm{val}} &= (\mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}\mathbf{s})^T\mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{v}_{\mathrm{A}} \\ &= \mathbf{s}^T\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})^T\mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{v}_{\mathrm{A}} && \text{(using (56))} \\ &= \mathbf{s}^T\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}({\mathbf{V}_{\mathrm{p}}}^T \otimes {\mathbf{V}_{\mathrm{q}}}^T)(\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{v}_{\mathrm{A}} \\ &= \mathbf{s}^T\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}({\mathbf{V}_{\mathrm{p}}}^T\mathbf{R}^n\mathbf{V}_{\mathrm{p}} \otimes {\mathbf{V}_{\mathrm{q}}}^T\mathbf{R}^n\mathbf{V}_{\mathrm{q}})\mathbf{v}_{\mathrm{A}} && \text{(using (6))} \\ &= \mathbf{s}^T\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}(\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{v}_{\mathrm{A}} && \text{(using (79))} \\ &= \mathbf{s}^T\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}}\mathbf{v}_{\mathrm{A}} \\ &= \mathbf{s}^T\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{val}}\mathbf{v}_{\mathrm{A}} && \text{(using (56))} \\ &= (\mathbf{T}^{\mathrm{val}}\mathbf{T}_{\mathrm{A}}^{\mathrm{op}}\mathbf{s})^T\mathbf{v}_{\mathrm{A}}^{\mathrm{val}} \\ &= \mathbf{b}_{\mathrm{A}}^T\mathbf{v}_{\mathrm{A}}^{\mathrm{val}} \end{aligned}$$

This proves that the component of a particular solution's $\mathbf{v}^{\mathrm{val}}$ along $\mathbf{b}$ is independent of problem instance. Next, we relate the eigenvectors of $\mathbf{A}_{\mathrm{B}}$ (let a general eigenvector be called $\mathbf{u}_{\mathrm{B}}$) to those of $\mathbf{A}_{\mathrm{A}}$ (general eigenvector $\mathbf{u}_{\mathrm{A}}$). We state that the eigenvectors of $\mathbf{A}_{\mathrm{B}}$ are given by

$$\mathbf{u}_{\mathrm{B}} = (\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{u}_{\mathrm{A}} \tag{81}$$

and that they have the same eigenvalues as those of $\mathbf{A}_{\mathrm{A}}$, ie. $\lambda_{\mathrm{B}} = \lambda_{\mathrm{A}}$. The proof is as follows:

$$\begin{aligned}
\mathbf{A}_{\mathrm{B}}\mathbf{u}_{\mathrm{B}} &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}_{\mathrm{B}}\mathbf{T}^{\mathrm{val}}\mathbf{u}_{\mathrm{B}} \\
&= (\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{P}_{\mathrm{B}} \otimes \mathbf{Q}_{\mathrm{B}})(\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{u}_{\mathrm{B}} \\
&= (\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{V}_{\mathrm{p}}\mathbf{P}_{\mathrm{A}}{\mathbf{V}_{\mathrm{p}}}^T \otimes \mathbf{V}_{\mathrm{q}}\mathbf{Q}_{\mathrm{A}}{\mathbf{V}_{\mathrm{q}}}^T)(\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{u}_{\mathrm{A}} && \text{(using (81))} \\
&= (\mathbf{R}^n\mathbf{V}_{\mathrm{p}} \otimes \mathbf{R}^n\mathbf{V}_{\mathrm{q}})(\mathbf{P}_{\mathrm{A}} \otimes \mathbf{Q}_{\mathrm{A}})({\mathbf{V}_{\mathrm{p}}}^T\mathbf{R}^n\mathbf{V}_{\mathrm{p}} \otimes {\mathbf{V}_{\mathrm{q}}}^T\mathbf{R}^n\mathbf{V}_{\mathrm{q}})\mathbf{u}_{\mathrm{A}} && \text{(using (6))} \\
&= (\mathbf{R}^n\mathbf{V}_{\mathrm{p}} \otimes \mathbf{R}^n\mathbf{V}_{\mathrm{q}})\mathbf{T}^{\mathrm{op}}_{\mathrm{A}}(\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{u}_{\mathrm{A}} && \text{(using (79))} \\
&= (\mathbf{V}_{\mathrm{p}}\mathbf{R}^n \otimes \mathbf{V}_{\mathrm{q}}\mathbf{R}^n)\mathbf{T}^{\mathrm{op}}_{\mathrm{A}}(\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{u}_{\mathrm{A}} && \text{(using (80))} \\
&= (\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})(\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{T}^{\mathrm{op}}_{\mathrm{A}}(\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{u}_{\mathrm{A}} && \text{(using (6))} \\
&= (\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}_{\mathrm{A}}\mathbf{T}^{\mathrm{val}}\mathbf{u}_{\mathrm{A}} \\
&= (\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{A}_{\mathrm{A}}\mathbf{u}_{\mathrm{A}} \\
&= \lambda_{\mathrm{A}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{u}_{\mathrm{A}} \\
&= \lambda_{\mathrm{A}}\mathbf{u}_{\mathrm{B}} && \text{(using (81))}
\end{aligned}$$

Hence, for any eigenvector of $\mathbf{A}_{\mathrm{B}}$:

$$\begin{aligned}
\mathbf{u}_{\mathrm{B}}^T\mathbf{v}_{\mathrm{B}}^{\mathrm{val}} &= ((\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{u}_{\mathrm{A}})^T\mathbf{T}^{\mathrm{val}}(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{v}_{\mathrm{A}} \\
&= \mathbf{u}_{\mathrm{A}}^T({\mathbf{V}_{\mathrm{p}}}^T \otimes {\mathbf{V}_{\mathrm{q}}}^T)(\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{V}_{\mathrm{p}} \otimes \mathbf{V}_{\mathrm{q}})\mathbf{v}_{\mathrm{A}} \\
&= \mathbf{u}_{\mathrm{A}}^T({\mathbf{V}_{\mathrm{p}}}^T\mathbf{R}^n\mathbf{V}_{\mathrm{p}} \otimes {\mathbf{V}_{\mathrm{q}}}^T\mathbf{R}^n\mathbf{V}_{\mathrm{q}})\mathbf{v}_{\mathrm{A}} && \text{(using (6))} \\
&= \mathbf{u}_{\mathrm{A}}^T(\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{v}_{\mathrm{A}} && \text{(using (79))} \\
&= \mathbf{u}_{\mathrm{A}}^T\mathbf{T}^{\mathrm{val}}\mathbf{v}_{\mathrm{A}} \\
&= \mathbf{u}_{\mathrm{A}}^T\mathbf{v}_{\mathrm{A}}^{\mathrm{val}}
\end{aligned}$$

This proves that the component of a particular solution's $\mathbf{v}^{\mathrm{val}}$ along any eigenvector of $\mathbf{A}$ is independent of problem instance.

Appendix B

Matrix Graduated Non-Convexity

In [2] a scheme is proposed, dubbed Matrix Graduated Non-Convexity, which both helps avoid entrapment of the network state vector $\mathbf{v}$ in sub-optimal local minima and also forces eventual convergence of $\mathbf{v}$ to a hypercube corner. Its action is similar to that of the annealing procedures proposed for networks of this kind. In this Appendix we show that the results of this paper are not affected by the use of Matrix Graduated Non-Convexity.

Matrix Graduated Non-Convexity works by adding a multiple of the identity matrix to $\mathbf{T}^{\mathrm{op}}$:

$$\mathbf{T}^{\mathrm{op}} \rightarrow \mathbf{T}^{\mathrm{op}} + \beta\mathbf{I} \tag{82}$$

The network is started running with a negative value of β, and then β is gradually increased through to positive values until convergence to a hypercube corner is achieved. First let us see what effect β has on the vector $\mathbf{b}$:

$$\begin{aligned}
\mathbf{b} \rightarrow \mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}} + \beta\mathbf{I})\mathbf{s} &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s} + \beta\mathbf{T}^{\mathrm{val}}\mathbf{s} \\
&= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s} + \beta(\mathbf{R}^n \otimes \mathbf{R}^n)(\mathbf{o}^n \otimes \mathbf{o}^n) \\
&= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s} + \beta(\mathbf{R}^n\mathbf{o}^n \otimes \mathbf{R}^n\mathbf{o}^n) \\
&= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s}
\end{aligned}$$

Hence $\mathbf{b}$ is unchanged by β. The eigenvectors $\mathbf{u}_i$ of $\mathbf{A}$ are also unchanged by β, though the eigenvalues are affected, since

$$\begin{aligned}
\mathbf{A}\mathbf{u}_i \rightarrow \mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}} + \beta\mathbf{I})\mathbf{T}^{\mathrm{val}}\mathbf{u}_i &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}}\mathbf{u}_i + \beta\mathbf{T}^{\mathrm{val}}\mathbf{u}_i \\
&= \begin{cases} \mathbf{0} & \text{for } \mathbf{u}_i \text{ in the invalid subspace.} \\ (\lambda_i + \beta)\mathbf{u}_i & \text{for } \mathbf{u}_i \text{ in the valid subspace.} \end{cases}
\end{aligned}$$

So we see that the eigenvalues in the valid subspace are all shifted by an amount β. However, the relative values of the valid subspace eigenvalues are unchanged, and so the result that the dominant eigenvector $\mathbf{u}_{\max}$ should be a very significant influence on the evolution of $\mathbf{v}^{\mathrm{val}}$ still holds. In fact, Matrix Graduated Non-Convexity enhances the dominance of $\mathbf{u}_{\max}$ in the early stages of convergence, since β is usually started sufficiently negative to ensure that only $\lambda_{\max}$, out of all the eigenvalues, is greater than zero. Referring back to equation (37) this means that, neglecting the action of $\mathbf{b}$, only the component along $\mathbf{u}_{\max}$ is developed, the others being suppressed. In the simulations reported in this paper, we have used Matrix Graduated Non-Convexity to ensure that the best possible performance is extracted from the network.