A SUBSPACE APPROACH TO SOLVING COMBINATORIAL OPTIMIZATION PROBLEMS WITH HOPFIELD NETWORKS

S.V.B.Aiyer & F.Fallside

CUED/F-INFENG/TR 55

January 4, 1991

Cambridge University Engineering Department

Trumpington Street

Cambridge CB2 1PZ

England

Email: svb10 / fallside @dsl.eng.cam.ac.uk

Submitted to IEEE Transactions on Neural Networks

Abstract

This paper extends and generalizes the subspace analysis of the Hopfield Network presented in our earlier work [1]. In [1] it was shown that the ability of the Hopfield Network to confine a vector within a particular subspace was essential to the network's ability to reach valid solutions to the Travelling Salesman problem. Through the use of Kronecker (Tensor) products, this paper shows how an analogous subspace can be constructed for a much larger class of combinatorial optimization problems. By using the form of this subspace as the basis for determining the elements of the network connection matrix, convergence to a valid solution can be guaranteed. Further, the quality of the final solution is significantly improved. This is confirmed by benchmark experiments using 30 and 50 city Travelling Salesman problems as an illustrative example. These indicate that the network can reliably and efficiently achieve solutions within 2% of the global optimum.

Introduction

The Hopfield Network employs one of the simplest possible neural network feedback architectures (see Fig 1). Yet Hopfield and Tank demonstrated in their 1985 paper [4] that the network had significant emergent computational power: specifically it could solve NP-complete, Travelling Salesman type combinatorial optimization problems. In earlier work [2][3] Hopfield had proved that the network operated so as to minimise a Liapunov function which was a quadratic function of the network output. The precise form of this function depended upon the free parameters of the network i.e the matrix of connection strengths and vector of input biases. Their approach for the Travelling Salesman problem relied upon specifying two quadratic energy functions of the network output. The first was minimised when the network output represented a valid problem solution. The second was proportional to the 'cost' of this solution. The sum of both of these formed a problem specific quadratic energy function. They then showed it was possible to set the free parameters of the network so that the network Liapunov function was identical to the problem specific energy function. Hence the network would operate so as to minimise the problem specific energy function and so potentially find the minimum 'cost' solution.

This method of formulating the network can be characterised as a top down approach, the reason being that at first the problem specific energy function is formulated, and then this is used to derive the free parameters of the network. Unfortunately although this approach gives the network the potential to solve the TSP, it does not guarantee that this will be the outcome. This is confirmed by the fact that many researchers have found the network highly unreliable. Hopfield and Tank in [4] themselves found difficulty in using the network to solve 30-city problems, whilst Wilson and Pawley [6] reported problems even replicating the 10-city results. The consensus seems to be that the network rarely reaches a valid solution, and even when it does the solution is of a poor 'quality'.

In our previous paper [1] an intermediate approach to formulating the free parameters of the network was developed. This consisted of first proving the existence of a subspace (strictly speaking a linear manifold) which contains only those hypercube corners which represent valid solutions to the Travelling Salesman problem. It was then demonstrated that with appropriately set parameters, the network would confine its output to this subspace, thus ensuring convergence to a valid solution. The approach is novel in the sense that instead of the first step being the specification of an energy function, the first step is to analyse the form of the subspace within which the network must confine its output. Once this is done, both the energy function and the free parameters of the network can be easily derived.

This paper seeks to take this approach one step further, by generalising the concept of the confinement subspace within which the network output must lie. Through an analysis based on Kronecker products, it is shown that such a subspace can be constructed for a much larger class of combinatorial optimization problems. This is used to develop a general framework for deriving the free parameters of the network, in such a way that convergence to a valid solution is guaranteed. Two other useful results also emerge from this analysis. Firstly it allows a modified network to be proposed, which has significant speed and reliability advantages over the original Hopfield network. Secondly it allows a detailed analysis of the conditions necessary to ensure strict convergence to a hypercube corner. Finally, confirmation of the theoretical claims made in this paper is achieved in experiments with 30 and 50-city Travelling salesman problems.

Outline

Initially there is a brief summary of the Hopfield network and how its output can be used to represent the solutions to certain types of combinatorial optimization problems. The next section then introduces the key properties of Kronecker products, together with the notation conventions and standard matrices employed in the rest of this paper. This is followed by a section which, starting from a simple 3-dimensional example, uses the Kronecker product notation to derive the general expression for the subspace within which the network must confine its output. Using this expression, the next section shows how the free parameters of the network can be derived in such a way that the resulting network Liapunov function will ensure confinement to the correct subspace. The section also demonstrates how the 'cost' term of the problem specific energy function can be cast into the Kronecker product notation. This has the benefit of reducing the computational complexity of simulating the network. Further improvements to the network are proposed in the following section, which develops a modified network with much greater speed and reliability then Hopfield network. However it is shown that the network retains both key functional properties of the original Hopfield network together with its ease of implementation on analogue hardware. In addition a detailed analysis is performed of how the modified network guarantees both convergence to the correct subspace as well as eventual convergence to a hypercube corner. By demonstrating an equivalence with quadratic programming problems, the next section shows that there is a trade-off between the optimality of the final solution and the need to enforce convergence to a hypercube corner. An algorithm for dealing with the adverse effects of this trade-off is then proposed. Finally to confirm the theory developed in this paper, experiments with the modified network are performed using Hopfield & Tank's original 30-city [4] and Durbin & Wilshaw's 50-city Travelling Salesman problems as illustrative examples.

Summary of the Hopfield network

The Hopfield network is constructed by connecting a large number of simple processing elements to each other. In general the i^{th} processing element, or neuron, is described by two variables: its current state u_i, and its output v_i. The output is related to the state by a simple non-decreasing monotonic output function such that

$$v_i = g(u_i)$$

This function operates as a threshold function to limit the output of each neuron to the interval 0 to 1, in order to ensure that the final state of the network corresponds to a corner of the unit hypercube.

The output of the i^{th} neuron is fed to the input of the j^{th} 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 i^{th} neuron, u_i, is updated by a function of the total input to the neuron. The exact nature of the form of $g(u_i)$ and the update procedure depends upon whether the continuous or discrete Hopfield network is being used. In the discrete model $g(u_i)$ is a step function of the form,

$$g(u_i) = \begin{cases} 0 & u_i < 0.5 \\ 1 & u_i \geq 0.5 \end{cases}$$

and consequently v_i is a discrete variable with a value of 0 or 1. In addition u_i is updated in discrete time steps by replacement with a value given by a function of its total input. In the continuous model, v_i is a continuous variable in the interval 0 to 1 and $g(u_i)$ is a continuous function, usually a hyperbolic tangent of the form,

$$g(u_i) = \tfrac{1}{2} + \tfrac{1}{2}\tanh(\beta u_i)$$

In this case, so that stability can be guaranteed, u_i is updated continosly by evaluating $\frac{d}{dt}(u_i)$ as a function of the total input to the i^{th} neuron.

This paper only uses the continuous network with synchronous update. The reason for these choices is that the continuous network is much better at avoiding sub-optimal local minima than the discrete network and synchronous update is much faster and more reliable than asynchronous update. Also in this paper, the states of the neurons will be collectively denoted by the vector $\mathbf{u}$, the outputs by the vector $\mathbf{v}$, the connection strengths by the matrix $\mathbf{T}$ and the offset biases by the vector $\mathbf{i}^b$.

A schematic diagram of the continuous Hopfield Network and the operation of its dynamic equation is shown in Fig 1.

The operation of the continuous Hopfield Network as formulated in [4] is governed by the following dynamic update/differential equation:

$$\dot{\mathbf{u}} = -\frac{\mathbf{u}}{\tau} + \mathbf{T}\mathbf{v} + \mathbf{i}^b \quad \mathit{where}\ v = g(u) \qquad (1)$$

Hopfield proves stability for this network by showing that with this dynamic equation the network output $\mathbf{v}$ evolves so as to minimise the Liapunov function:

$$E = -\frac{1}{2}\mathbf{v}^T\mathbf{T}\mathbf{v} - (\mathbf{i}^b)^T\mathbf{v} + \frac{1}{\tau}\sum\int_0^v g^{-1}(\alpha)d\alpha \quad (2)$$

N.B. For the rest of this paper the $-\frac{\mathbf{u}}{\tau}$ *and* $\frac{1}{\tau}\sum\int_0^v g^{-1}(\alpha)d\alpha$ *terms will be ignored, since by making* $|\mathbf{T}|$ *or* τ *arbitrarily large they can be made negligible.*

Mapping combinatorial optimization problems

A key property of combinatorial optimization problems is that there is a finite set of solutions, hence it is possible to represent each solution by an integer or even a set of integers. Consequently if the output vector of a Hopfield network is to represent such a solution, it must be discrete. The usual way of imposing this constraint is to confine the output of the network to the unit hypercube, and to ensure that the network dynamics will eventually force the output into a corner of the unit hypercube. At these corners the network output vector, $\mathbf{v}$, will have elements which are either 0 or 1. The problem of mapping a particular combinatorial optimization problem therefore reduces to,

(i) Finding a way of using a vector of zeros and ones to represent the solutions of the problem.

(ii) Finding a way of measuring the 'cost' of the solution, in such a way that minimising the quadratic Liapunov function (2) will select a minimum 'cost' solution. Ultimately it is this feature that allows the Hopfield network to solve combinatorial optimization problems.

This paper is concerned with combinatorial optimization problems which have solutions that can be represented by a fixed length vector of integers, each of which is within a certain range. Although this condition is quite restrictive, there are a large class of problems that satisfy it. This class includes the Travelling Salesman problem (TSP), the Graph Partitioning problem (GPP) [1] [2] and even certain types of Dynamic Programming problems.

Let the vector which represents a solution be denoted by $\mathbf{p}$ and be of length N. Further let its elements be integers in the range $\{1, \cdots, M\}$. Thus,

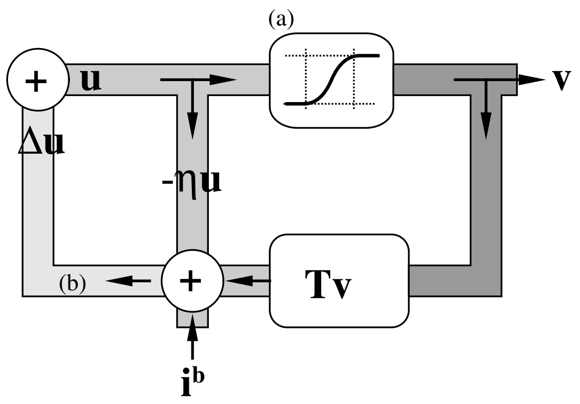
$$p_m \in \{1, \cdots, M\} \quad \text{where} \quad m \in \{1, \cdots, N\} \qquad (3)$$

As a specific example consider the Travelling Salesman problem. For a N-city problem the final solution must represent a possible tour which visits each city once and only once. This can be done by letting p_m be the position of city m in the tour. For example, in a 4-city TSP if the final tour order is:

city 4; city 1; city 3; city 2; city 4

then the vector representing this would be given by,

$$\mathbf{p}^T = [2, 4, 3, 1]$$



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

(b) The change in $\mathbf{u}$ is specified by the differential equation (1). However in a computer simulation a discrete approximation has to be used, where time is split up into a series of discrete time steps Δt and the change in $\mathbf{u}$ is given by,

$$\Delta\mathbf{u} = \Delta t \tfrac{d}{dt}(\mathbf{u}) = \Delta t(-\frac{\mathbf{u}}{\tau} + \mathbf{T}\mathbf{v} + \mathbf{i}^b)$$

Figure 1: Schematic diagram of continuous Hopfield network

[1] A GPP for a graph with N nodes and M partitions, is concerned with partitioning the N nodes into M partitions of equal size, in such a way that the number of edges linking nodes in different partitions is minimised.

[2] Where possible, the analysis developed in this paper will be illustrated with TSP and GPP examples. This does not mean that the analysis cannot be applied to other types of combinatorial problems: it is just that for conciseness other examples are not included.

For a Graph Partitioning problem with N nodes and M partitions, p_m represents the partition to which node m belongs to. Note that in the case of Travelling Salesman problems there are two extra constraints on $\mathbf{p}$. These are that $N = M$ and that the elements of $\mathbf{p}$ must be unique, since each city can only be visited once.

The next task is to find a way of representing $\mathbf{p}$ by a vector $\mathbf{v}$ which is the output of the network and which lies at a hypercube corner: i.e. it is a vector of ones and zeros. This can be done as follows:

Let $\boldsymbol{\delta}^p(N)$ be the N dimensional co-ordinate vector given by,

$$\begin{aligned}[\boldsymbol{\delta}^p(N)]_i &= \begin{cases} 1 & \text{if} \quad i = p \\ 0 & \text{if} \quad i \neq p \end{cases} \quad p \in \{1, \cdots, N\} \\ &= \delta_{pi} \end{aligned} \tag{4}$$

In effect $\boldsymbol{\delta}^p(N)$ is the p^{th} column of the $N \times N$ identity matrix.

N.B. The (N) part of the vector $\boldsymbol{\delta}^p(N)$, specifies the number of elements of this vector and the range of p. If this is obvious from the context of use, then it will be omitted, leaving just $\boldsymbol{\delta}^p$.

Let $\mathbf{v}(\mathbf{p})$ be the NM dimensional vector representing the form of the final network output vector, $\mathbf{v}$, which corresponds to the problem solution denoted by $\mathbf{p}$. This is defined as follows:

$$\mathbf{v}(\mathbf{p}) = \begin{bmatrix} \boldsymbol{\delta}^{p_1}(M) \\ \boldsymbol{\delta}^{p_2}(M) \\ \cdots \\ \boldsymbol{\delta}^{p_N}(M) \end{bmatrix} \tag{5}$$

An alternative way of representing $\mathbf{p}$ is by a $N \times M$ matrix. Let such a matrix be denoted $\mathbf{V}(\mathbf{p})$, and be defined by:

$$\mathbf{V}(\mathbf{p}) = \begin{bmatrix} \boldsymbol{\delta}^{p_1}(M)^T \\ \boldsymbol{\delta}^{p_2}(M)^T \\ \cdots \\ \boldsymbol{\delta}^{p_N}(M)^T \end{bmatrix} \tag{6}$$

or in component form

$$[\mathbf{V}(\mathbf{p})]_{ij} = [\boldsymbol{\delta}^{p_i}(M)]_j$$

For the TSP tour:

city 4; city 1; city 3; city 2; city 4

given above, where $\mathbf{p}^T = [2, 4, 3, 1]$, the vector $\mathbf{v}(\mathbf{p})$ and matrix $\mathbf{V}(\mathbf{p})$ would expand as follows,

$$\mathbf{v}(\mathbf{p})^T = \begin{matrix} [0\,1\,0\,0 & 0\,0\,0\,1 & 0\,0\,1\,0 & 1\,0\,0\,0] \\ [\boldsymbol{\delta}^{p_1 T} & \boldsymbol{\delta}^{p_2 T} & \boldsymbol{\delta}^{p_3 T} & \boldsymbol{\delta}^{p_4 T}] \end{matrix}$$

$$\mathbf{V}(\mathbf{p}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Kronecker product notation and other definitions

This section introduces some alternative notation, based on the use of Kronecker products [9] (also known as tensor products). By using these products it is possible to avoid the double indexing and triple summations common with the notation employed by Hopfield and Tank [4]. In addition definitions are given for all other vectors and matrices which are used later in this paper.

Let $\mathbf{u}^T$ denote the transpose of $\mathbf{u}$.

Let $(\mathbf{P} \otimes \mathbf{Q})$ denote the Kronecker product of two matrices. Thus if $\mathbf{P}$ is an $N \times N$ matrix, and $\mathbf{Q}$ is an $M \times M$ matrix, then $(\mathbf{P} \otimes \mathbf{Q})$ is a $NM \times NM$ matrix given by,

$$(\mathbf{P} \otimes \mathbf{Q}) = \begin{bmatrix} P_{11}\mathbf{Q} & P_{12}\mathbf{Q} & \cdots & P_{1N}\mathbf{Q} \\ P_{21}\mathbf{Q} & P_{22}\mathbf{Q} & \cdots & P_{2N}\mathbf{Q} \\ \cdots & \cdots & \cdots & \cdots \\ P_{N1}\mathbf{Q} & P_{N2}\mathbf{Q} & \cdots & P_{NN}\mathbf{Q} \end{bmatrix} \tag{7}$$

If $p = M(x-1)+i$ and $q = M(y-1)+j$ where,

$$\begin{aligned} x, y &\in \{1, \cdots, N\} \\ i, j &\in \{1, \cdots, M\} \end{aligned}$$

then (7) can be written in component form as,

$$[(\mathbf{P} \otimes \mathbf{Q})]_{pq} = P_{xy} Q_{ij} \tag{8}$$

In Hopfield's notation $[(\mathbf{P} \otimes \mathbf{Q})]_{pq}$ would be written as $[(\mathbf{P} \otimes \mathbf{Q})]_{xi,yj}$.

Similarly if $\mathbf{w}$ is a N-element vector and $\mathbf{h}$ a M-element vector, then $(\mathbf{w} \otimes \mathbf{h})$ is a NM-element vector given by,

$$(\mathbf{w} \otimes \mathbf{h}) = \begin{bmatrix} w_1\mathbf{h} \\ w_2\mathbf{h} \\ \cdots \\ w_N\mathbf{h} \end{bmatrix} \tag{9}$$

The following properties of the Kronecker products $(\mathbf{P} \otimes \mathbf{Q})$ and $(\mathbf{w} \otimes \mathbf{h})$ are utilised later (for the proofs refer to [9]).

$$(\lambda\mathbf{w}\otimes\gamma\mathbf{h}) = \lambda\gamma(\mathbf{w}\otimes\mathbf{h}) \quad (10)$$
$$(\mathbf{w}\otimes\mathbf{h})^T(\mathbf{x}\otimes\mathbf{g}) = (\mathbf{w}^T\mathbf{x})(\mathbf{h}^T\mathbf{g}) \quad (11)$$
$$(\mathbf{P}\otimes\mathbf{Q})(\mathbf{w}\otimes\mathbf{h}) = (\mathbf{P}\mathbf{w}\otimes\mathbf{Q}\mathbf{h}) \quad (12)$$
$$(\mathbf{P}\otimes\mathbf{Q})(\mathbf{E}\otimes\mathbf{F}) = (\mathbf{P}\mathbf{E}\otimes\mathbf{Q}\mathbf{F}) \quad (13)$$

Let $\mathbf{I}^n$ and $\mathbf{I}^{nm}$ be respectively the $N\times N$ and $NM\times NM$ identity matrices, i.e,

$$\begin{aligned} I^n_{ij} &= \delta_{ij} \quad i,j\in\{1,\cdots,N\} \\ \text{and } \mathbf{I}^{nm} &= (\mathbf{I}^n\otimes\mathbf{I}^m) \end{aligned} \quad (14)$$

Let $\mathbf{o}^n$ and $\mathbf{o}^{nm}$ be respectively the N element and NM element vectors of ones, i.e,

$$\begin{aligned} o^n_p &= 1 \quad p\in\{1,\cdots,N\} \\ \text{and } \mathbf{o}^{nm} &= (\mathbf{o}^n\otimes\mathbf{o}^m) \end{aligned} \quad (15)$$

Let $\mathbf{O}^n$ and $\mathbf{O}^{nm}$ be respectively the $N\times N$ and $NM\times NM$ matrices of ones, i.e,

$$\begin{aligned} O^n_{ij} &= 1 \quad i,j\in\{1,\cdots,N\} \\ \text{and } \mathbf{O}^{nm} &= (\mathbf{O}^n\otimes\mathbf{O}^m) \end{aligned} \quad (16)$$

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

$$\mathbf{R}^n = \mathbf{I}^n - \tfrac{1}{N}\mathbf{o}^n\mathbf{o}^{nT} = \mathbf{I}^n - \tfrac{1}{N}\mathbf{O}^n \quad (17)$$

or in component form,

$$R^n_{ij} = \delta_{ij} - \tfrac{1}{N} \quad (18)$$

If $\mathbf{x}$ is an arbitrary N dimensional vector, then

$$\mathbf{R}^n\mathbf{x} = \begin{cases} \mathbf{x} & \text{if } \mathbf{o}^{nT}\mathbf{x}=0 \\ \mathbf{0} & \text{if } \mathbf{x}=\mathbf{o}^n \\ \mathbf{x}-\tfrac{1}{N}\mathbf{O}^n\mathbf{x} & \text{otherwise} \end{cases} \quad (19)$$

Also,

$$\begin{aligned} \mathbf{R}^n\mathbf{R}^n &= \mathbf{I}^n\mathbf{I}^n - \tfrac{1}{N}\mathbf{I}^n\mathbf{O}^n \\ &\quad -\tfrac{1}{N}\mathbf{O}^n\mathbf{I}^n + \tfrac{1}{N^2}\mathbf{O}^n\mathbf{O}^n \\ &= \mathbf{I}^n - \tfrac{1}{N}\mathbf{O}^n \\ \Rightarrow \mathbf{R}^n\mathbf{R}^n &= \mathbf{R}^n \end{aligned} \quad (20)$$

Thus $\mathbf{R}$ is a projection matrix which removes the $\mathbf{o}^n$ component from $\mathbf{x}$, so that if $\mathbf{x}$ is orthogonal to $\mathbf{o}^n$ it is left unchanged. Also note that if $\mathbf{y}=\mathbf{R}^n\mathbf{x}$ then

$$\begin{aligned} \sum_{k=1}^N y_k &= \sum_{k=1}^N[\mathbf{x}-\tfrac{1}{N}\mathbf{O}^n\mathbf{x}]_k \\ &= \mathbf{o}^{nT}[\mathbf{x}-\tfrac{1}{N}\mathbf{O}^n\mathbf{x}] \\ &= \mathbf{o}^{nT}\mathbf{x}-\tfrac{1}{N}(\mathbf{o}^{nT}\mathbf{o}^n\mathbf{o}^{nT})\mathbf{x} \\ &= \mathbf{o}^{nT}\mathbf{x}-\mathbf{o}^{nT}\mathbf{x} \\ \Rightarrow \sum_{k=1}^N y_k &= \sum_k[\mathbf{R}^n\mathbf{x}]_k = 0 \end{aligned} \quad (21)$$

In terms of eigenvalues this corresponds to $\mathbf{R}^n$ having a degenerate eigenvalue of 1 for all vectors orthogonal to $\mathbf{o}^n$ and an eigenvalue of 0 for $\mathbf{o}^n$.

Let $\mathbf{U}_{.q}$ be a M-element vector corresponding to the q^{th} column of the $M\times N$ matrix $\mathbf{U}$. i.e,

$$\mathbf{U}_{.q} = \begin{bmatrix} U_{1q} \\ U_{2q} \\ \cdots \\ U_{Nq} \end{bmatrix} \quad (22)$$

Let $\mathrm{vec}(\mathbf{U})$ be the function which maps the $M\times N$ matrix $\mathbf{U}$ to the NM-element vector $\mathbf{v}$. This function is defined by,

$$\mathbf{v} = \mathrm{vec}(\mathbf{U}) = \begin{bmatrix} \mathbf{U}_{.1} \\ \mathbf{U}_{.2} \\ \cdots \\ \mathbf{U}_{.N} \end{bmatrix} \quad (23)$$

Thus,

$$\begin{aligned} \mathbf{v} &= \sum_{x=1}^N\sum_{i=1}^M U_{xi}(\boldsymbol{\delta}^i\otimes\boldsymbol{\delta}^x) \\ \Rightarrow U_{xi} &= (\boldsymbol{\delta}^i\otimes\boldsymbol{\delta}^x)^T\mathbf{v} \end{aligned} \quad (24)$$

Note also that if $\mathbf{v}$ is a valid solution then

$$\mathbf{v} = \mathbf{v}(\mathbf{p}) = \mathrm{vec}(\mathbf{V}(\mathbf{p})^T) \quad (25)$$

Convergence to Valid Solutions

A valid solution is a form of the network output, $\mathbf{v}$, which represents a possible solution of a combinatorial optimization problem. For the class of problems considered in this paper, at such a solution, $\mathbf{v}$ must be of the form $\mathbf{v}(\mathbf{p})$. This section analyses how it is possible to ensure that the network output, starting from a random state, will always converge to a hypercube corner of the form $\mathbf{v}(\mathbf{p})$. The key to this is to confine $\mathbf{v}$ to a subspace which contains only those hypercube corners which are of the form $\mathbf{v}(\mathbf{p})$. Let this subspace be termed the **valid subspace.** [3]

Analysis of Valid Subspace

A simple 3-dimensional example

Fig 2 shows a simplified illustration of the key principle of this analysis: that a subspace can be constructed which contains only those hypercube corners which correspond to a valid solution. Let the

[3] Strictly speaking this should be the valid affine subspace or valid linear manifold, since as later analysis reveals, it does not pass through the origin.

valid solutions for the 3-dimensional network output shown in Fig 2, be either the set,

$$\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$$

or the set

$$\{(1,1,0),\ (1,0,1),\ (0,1,1)\}$$

The lighter shaded plane is in effect a 2-dimensional valid subspace which only contains cube corners from the first set, while the darker shaded plane only contains cube corners from the second set.

Let,

$$S = \sum_{k=1}^{3} v_k$$

Another way of viewing the lighter shaded plane is that it is the plane where $S = 1$ and the darker shaded plane is where $S = 2$. Clearly if $S = 1$ and $\mathbf{v}$ is a cube corner then $\mathbf{v}$ must belong to the first set of valid solutions, likewise if $S = 2$, $\mathbf{v}$ must belong to the second set.

Recalling the definitions of $\mathbf{R}^n$ and $\mathbf{o}^n$ (see equations (17) and (15)), let $\mathbf{v}$ satisfy the equation,

$$\mathbf{v} = \tfrac{1}{3}\mathbf{o}^3 + \mathbf{R}^3\mathbf{v} \tag{26}$$

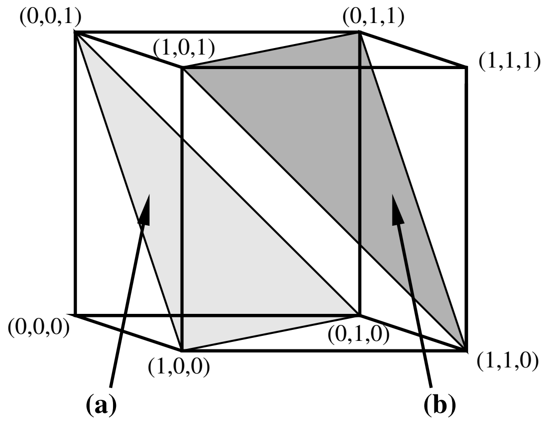
Since from (21),

$$\begin{aligned} \sum_{k=1}^{3}[\mathbf{R}^3\mathbf{v}]_k &= 0 \\ \text{and} \quad \sum_{k=1}^{3}[\tfrac{1}{3}\mathbf{o}^3]_k &= 1 \end{aligned}$$

it can be seen that if $\mathbf{v}$ satisfies (26) then it also satisfies $S = 1$. Therefore (26) is the equation of the valid subspace for the first set of valid solution cube corners. Essentially the $\mathbf{R}^3\mathbf{v}$ term of equation (26) is just a projection of $\mathbf{v}$ onto the subspace orthogonal to $\mathbf{o}^3$, i.e the plane where $S = 0$, hence it is the $\frac{1}{3}\mathbf{o}^3$ term of (26) that controls the value of S. Thus changing this term to $\frac{2}{3}\mathbf{o}^3$ ensures that $S = 2$ and hence

$$\mathbf{v} = \tfrac{2}{3}\mathbf{o}^3 + \mathbf{R}^3\mathbf{v} \tag{27}$$

is the equation of the valid subspace for the second set of valid solution cube corners.



(a) the plane or 2D valid subspace where, $\sum_{k=1}^{2} v_k = 1$ which contains the cube corners

$$\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$$

(b) the plane or 2D valid subspace where, $\sum_{k=1}^{2} v_k = 2$ which contains the cube corners

$$\{(1,1,0),\ (1,0,1),\ (0,1,1)\}$$

Figure 2: A Simplified 3D illustration of the valid subspace

Generalization to NM-dimensional valid solutions represented by $\mathbf{v}(\mathbf{p})$

The following analysis is based upon the properties of the $N \times M$ matrix $\mathbf{V}$ which is related to the NM element network output vector $\mathbf{v}$ by the expression:

$$\mathbf{v} = \mathrm{vec}(\mathbf{V}^T)$$

The properties of interest are the row and column sums of $\mathbf{V}$. Let these be defined as follows:

Let $c(i)$, where $i \in \{1, \cdots, M\}$, be the sum of the i^{th} column of $\mathbf{V}$. i.e,

$$c(i) = \sum_{x=1}^{N} V_{xi} \tag{28}$$

Let $r(x)$, where $x \in \{1, \cdots, N\}$, be the sum of the x^{th} row of $\mathbf{V}$. i.e,

$$r(x) = \sum_{i=1}^{M} V_{xi} \tag{29}$$

From (25) is can be seen that if $\mathbf{v}$ is a valid solution $\mathbf{v}(\mathbf{p})$, then

$$\mathbf{V} = \mathbf{V}(\mathbf{p})$$

Hence if $\mathbf{v}$ is a valid solution then the row and column sums of $\mathbf{V}$ are given by the row and column sums of $\mathbf{V}(\mathbf{p})$. These have the following properties for TSPs and GPPs:

For the case of N-city Travelling Salesman type optimization problems, $M = N$ and $\mathbf{p}$ is a permutation vector where the elements of $\mathbf{p}$ are drawn from the set $\{1, \cdots, N\}$. With these conditions on $\mathbf{p}$ it can be seen from the definition of $\mathbf{V}(\mathbf{p})$ given in (6), that $\mathbf{V}$ must be a permutation matrix. Hence for this class of problems, given that $\mathbf{v} = \mathrm{vec}(\mathbf{V}^T)$ is a hypercube corner, the condition

$$c(i) = r(x) = 1 \tag{30}$$

is sufficient to guarantee that $\mathbf{v}$ is of the form $\mathbf{v}(\mathbf{p})$.

For the case of N-node Graph Partitioning Problems with M partitions, p_x where $x \in \{1, \cdots, N\}$ represents which partition node x belongs to. Node x can only belong to one partition, thus assuming $\mathbf{v}$ is a hypercube corner, this means

$$r(x) = 1 \quad x \in \{1, \cdots, N\} \tag{31}$$

The column sums, $c(i)$, now represent the number of nodes in the i^{th} partition. For certain GPPs it is desirable to have partitions of equal size. In this case $c(i)$ will be given by

$$c(i) = N/M \quad i \in \{1, \cdots, M\} \tag{32}$$

It is however possible to set $c(i)$ to any value as long as the sum of the column sums, i.e $\sum_{i=1}^{M} c(i)$ is equal to N.

The Zerosum Subspace

In order to define the valid subspace it first necessary to define the **zerosum subspace** (zs). This has the property that if $\mathbf{v}$ lies in the zerosum subspace, then $\mathbf{V}$, where $\mathbf{v} = \mathrm{vec}(\mathbf{V}^T)$, has row and column sums equal to zero.

Let $\mathbf{v}^{zs}$ be the vector given by,

$$\mathbf{v}^{zs} = \mathbf{T}^{zs}\mathbf{v} \tag{33}$$

where $\mathbf{T}^{zs}$ is given by,

$$\mathbf{T}^{zs} = (\mathbf{R}^n \otimes \mathbf{R}^m) \tag{34}$$

It will now be shown that $\mathbf{T}^{zs}$ is a projection matrix which projects $\mathbf{v}$ onto the zerosum subspace, giving $\mathbf{v}^{zs}$ i.e. the component of $\mathbf{v}$ in the zerosum subspace.

This will be done by evaluating the row and column sums of $\mathbf{V}^{zs}$ where

$$\mathbf{v}^{zs} = \mathrm{vec}(\mathbf{V}^{zs\,T})$$

First note that from (24),

$$V_{xi} = (\boldsymbol{\delta}^x \otimes \boldsymbol{\delta}^i)^T\mathbf{v}$$

Using the fact that,

$$\sum_{x=1}^{N} \boldsymbol{\delta}^x = \mathbf{o}^n, \quad \sum_{i=1}^{M} \boldsymbol{\delta}^i = \mathbf{o}^m$$

allows (28) and (29) to be rewritten as follows,

$$\begin{aligned} c(i) &= \sum_{x=1}^{N} (\boldsymbol{\delta}^x \otimes \boldsymbol{\delta}^i)^T\mathbf{v} = (\mathbf{o}^n \otimes \boldsymbol{\delta}^i)^T\mathbf{v} \\ r(x) &= \sum_{i=1}^{M} (\boldsymbol{\delta}^x \otimes \boldsymbol{\delta}^i)^T\mathbf{v} = (\boldsymbol{\delta}^x \otimes \mathbf{o}^m)^T\mathbf{v} \end{aligned}$$

Thus if $r^{zs}(x)$ and $c^{zs}(i)$ are the row and column sums of $\mathbf{V}^{zs}$, then:

$$\begin{aligned} c^{zs}(i) &= (\mathbf{o}^n \otimes \boldsymbol{\delta}^i)^T\mathbf{v}^{zs} \\ r^{zs}(x) &= (\boldsymbol{\delta}^x \otimes \mathbf{o}^m)^T\mathbf{v}^{zs} \end{aligned}$$

But $\mathbf{v}^{zs} = (\mathbf{R}^n \otimes \mathbf{R}^m)\mathbf{v}$ and from (19), $\mathbf{R}^n\mathbf{o}^n = 0$, hence,

$$\begin{aligned} c^{zs}(i) &= (\mathbf{o}^n \otimes \boldsymbol{\delta}^i)^T(\mathbf{R}^n \otimes \mathbf{R}^m)\mathbf{v} \\ &= \left[(\mathbf{R}^n \otimes \mathbf{R}^m)^T(\mathbf{o}^n \otimes \boldsymbol{\delta}^i)\right]^T\mathbf{v} \\ &= (\mathbf{0} \otimes \mathbf{R}^m\boldsymbol{\delta}^i)^T\mathbf{v} \\ &= 0 \end{aligned}$$

and,

$$\begin{aligned} r^{zs}(x) &= (\boldsymbol{\delta}^x \otimes \mathbf{o}^m)^T(\mathbf{R}^n \otimes \mathbf{R}^m)\mathbf{v} \\ &= \left[(\mathbf{R}^n \otimes \mathbf{R}^m)^T(\boldsymbol{\delta}^x \otimes \mathbf{o}^m)\right]^T\mathbf{v} \\ &= (\mathbf{R}^n\boldsymbol{\delta}^x \otimes \mathbf{0})^T\mathbf{v} \\ &= 0 \end{aligned}$$

General expression for the Valid Subspace

Let $\mathbf{v}$ satisfy the equation

$$\mathbf{v} = \mathbf{s} + \mathbf{v}^{zs} = \mathbf{s} + \mathbf{T}^{zs}\mathbf{v} \tag{35}$$

where
$\mathbf{s} = (\mathbf{a} \otimes \mathbf{o}^m) + (\mathbf{o}^n \otimes \mathbf{b})$
$\mathbf{T}^{zs} = (\mathbf{R}^n \otimes \mathbf{R}^m)$
$\mathbf{a}$ is an arbitrary N-element vector
$\mathbf{b}$ is an arbitrary M-element vector

The space spanned by all vectors, $\mathbf{v}$, that satisfy (35) is the definition of the valid subspace.

The key property of this subspace is that the row and column sums of $\mathbf{V}$ are a purely a function of $\mathbf{s}$.

To see this, consider the row and column sums of $\mathbf{V}$ assuming $\mathbf{v}$ satisfies (35):

$$
\begin{aligned}
c(i) &= (\mathbf{o}^n \otimes \boldsymbol{\delta}^i)^T \mathbf{v} \\
&= (\mathbf{o}^n \otimes \boldsymbol{\delta}^i)^T \mathbf{s} + c^{zs}(i) \\
&= (\mathbf{o}^n \otimes \boldsymbol{\delta}^i)^T [(\mathbf{a} \otimes \mathbf{o}^m) + (\mathbf{o}^n \otimes \mathbf{b})] \\
&= ({\mathbf{o}^n}^T \otimes {\boldsymbol{\delta}^i}^T)[(\mathbf{a} \otimes \mathbf{o}^m) + (\mathbf{o}^n \otimes \mathbf{b})] \\
&= ({\mathbf{o}^n}^T \mathbf{a} \otimes {\boldsymbol{\delta}^i}^T \mathbf{o}^m) + ({\mathbf{o}^n}^T \mathbf{o}^n \otimes {\boldsymbol{\delta}^i}^T \mathbf{b})
\end{aligned}
$$

$$
\Rightarrow \boxed{c(i) = {\mathbf{o}^n}^T \mathbf{a} + N b_i} \tag{36}
$$

and,

$$
\begin{aligned}
r(x) &= (\boldsymbol{\delta}^x \otimes \mathbf{o}^m)^T \mathbf{v} \\
&= (\boldsymbol{\delta}^x \otimes \mathbf{o}^m)^T \mathbf{s} + r^{zs}(x) \\
&= (\boldsymbol{\delta}^x \otimes \mathbf{o}^m)^T [(\mathbf{a} \otimes \mathbf{o}^m) + (\mathbf{o}^n \otimes \mathbf{b})] \\
&= ({\boldsymbol{\delta}^x}^T \otimes {\mathbf{o}^m}^T)[(\mathbf{a} \otimes \mathbf{o}^m) + (\mathbf{o}^n \otimes \mathbf{b})] \\
&= ({\boldsymbol{\delta}^x}^T \mathbf{a} \otimes {\mathbf{o}^m}^T \mathbf{o}^m) + ({\boldsymbol{\delta}^x}^T \mathbf{o}^n \otimes {\mathbf{o}^m}^T \mathbf{b})
\end{aligned}
$$

$$
\Rightarrow \boxed{r(x) = M a_x + {\mathbf{o}^m}^T \mathbf{b}} \tag{37}
$$

Equations (36) and (37) show that the values $c(i)$ and $r(x)$ are completely independent of $\mathbf{v}$ and can be set to almost arbitrary values by carefully choosing the vectors $\mathbf{a}$ and $\mathbf{b}$.

Thus:

For a N-city TSP, if,

$$
\begin{aligned}
\mathbf{a} &= 0 \\
\mathbf{b} &= \tfrac{1}{N}\mathbf{o}^n
\end{aligned}
$$

then,

$$
\begin{aligned}
c(i) &= {\mathbf{o}^n}^T \mathbf{a} + N b_i &= 1 \\
r(x) &= M a_x + {\mathbf{o}^m}^T \mathbf{b} &= 1
\end{aligned}
$$

Hence if $\mathbf{v}$ is a hypercube corner and $\mathbf{v}$ satisfies,

$$
\mathbf{v} = \tfrac{1}{N}(\mathbf{o}^n \otimes \mathbf{o}^n) + (\mathbf{R}^n \otimes \mathbf{R}^n)\mathbf{v} \tag{38}
$$

then $\mathbf{v}$ must represent a valid-solution of a N-city TSP.

For a N-nodes GPP if,

$$
\begin{aligned}
\mathbf{a} & &= 0 \\
{\mathbf{o}^m}^T \mathbf{b} &= \textstyle\sum_{k=1}^{M} b_k &= 1
\end{aligned}
$$

then,

$$
\begin{aligned}
c(i) &= {\mathbf{o}^n}^T \mathbf{a} + N b_i &= N b_i \\
r(x) &= M a_x + {\mathbf{o}^m}^T \mathbf{b} &= 1
\end{aligned}
$$

Hence, assuming ${\mathbf{o}^m}^T \mathbf{b} = 1$, if $\mathbf{v}$ is a hypercube corner and $\mathbf{v}$ satisfies,

$$
\mathbf{v} = (\mathbf{o}^n \otimes \mathbf{b}) + (\mathbf{R}^n \otimes \mathbf{R}^m)\mathbf{v} \tag{39}
$$

then $\mathbf{v}$ must represent the valid-solution of a N-nodes GPP with M partitions, where the i^{th} partition has exactly $N b_i$ nodes.

A framework for deriving the free parameters of the network

In the light of the subspace analysis, the Hopfield network, in solving combinatorial optimization problems, should ideally operate as follows:

(i) The network is initialized to a random state such that $\mathbf{v}$ is both within the unit hypercube and $\mathbf{v}$ is on the valid subspace.

(ii) The minimisation of E should then cause $\mathbf{v}$ to be moved towards the bounds of the unit hypercube, whilst being confined to the valid subspace.

(iii) As $\mathbf{v}$ approaches the bounds of the unit hypercube, the interaction between the minimisation of E and the non-linear output threshold functions should force $\mathbf{v}$ into a hypercube corner that is both a valid solution and a solution of optimum 'cost'.

The continuous Hopfield network as proposed in [4] and shown in Fig 1, operates so as to minimise (by a sort of Gradient Descent) the Liapunov equation (2):

$$
E = -\tfrac{1}{2}\mathbf{v}^T \mathbf{T} \mathbf{v} - (\mathbf{i}^b)^T \mathbf{v} + \frac{1}{\tau} \sum \int_0^v g^{-1}(\alpha) d\alpha
$$

Assuming the $\frac{1}{\tau}$ term is negligible (this can be ensured by making τ very large), this equation can be reduced to:

$$
E = -\tfrac{1}{2}\mathbf{v}^T \mathbf{T} \mathbf{v} - (\mathbf{i}^b)^T \mathbf{v}
$$

The aim in this section is to derive $\mathbf{T}$ and $\mathbf{i}^b$ in such a way that in minimising the above Liapunov function the network follows steps (i),(ii) and (iii). An intermediate approach is employed, in which the form of the valid subspace is used as the basis for developing analytic expressions for $\mathbf{T}$ and $\mathbf{i}^b$. It is then shown that these expressions result in a Liapunov function which ensures that steps (i),(ii) and (iii) are followed.

Let E be split up into a confinement (cn) and optimization (op) term, such that in minimizing E^{cn} the network confines $\mathbf{v}$ to the valid subspace, and in minimising E^{op} the network moves $\mathbf{v}$ towards an optimum 'cost' solution. This can be achieved by splitting $\mathbf{T}$ and $\mathbf{i}^b$ so that,

$$\begin{aligned} \mathbf{T} &= \mathbf{T}^{cn} + \mathbf{T}^{op} \\ \mathbf{i}^b &= \mathbf{i}^{cn} + \mathbf{i}^{op} \end{aligned} \tag{40}$$

Thus the Liapunov function splits into,

$$\begin{aligned} E &= E^{cn} + E^{op} \\ \text{where } E^{cn} &= -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{cn}\mathbf{v} - (\mathbf{i}^{cn})^T\mathbf{v} \quad (41) \\ \text{and } E^{op} &= -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{op}\mathbf{v} - (\mathbf{i}^{op})^T\mathbf{v} \quad (42) \end{aligned}$$

Expressions for $\mathbf{T}^{cn}$ and $\mathbf{i}^{cn}$

If $\mathbf{T}^{cn}$ and $\mathbf{i}^{cn}$ are then set according to,

$$\begin{aligned} \mathbf{T}^{cn} &= \theta(\mathbf{T}^{zs} - \mathbf{I}^{nm}) \quad (43) \\ \mathbf{i}^{cn} &= \theta\mathbf{s} \quad (44) \\ \text{where } \theta &\text{ is an arbitrary positive constant} \end{aligned}$$

then it can be shown that if $\mathbf{v}$ minimises E^{cn} it must satisfy the valid subspace equation (35), i.e,

$$\mathbf{v} = \mathbf{s} + \mathbf{T}^{zs}\mathbf{v}$$

The proof of this is as follows:

Let the subspace orthogonal to the zerosum subspace be termed the **nonzero subspace** (nz). Any NM-dimensional vector can be decomposed into its components in each of these subspaces, i.e,

$$\mathbf{v} = \mathbf{v}^{nz} + \mathbf{v}^{zs} \quad \text{with} \quad \mathbf{v}^{nz\,T}\mathbf{v}^{zs} = 0$$

Clearly $\mathbf{s}$ is in the nonzero subspace, since,

$$\begin{aligned} \mathbf{v}^{zs\,T}\mathbf{s} &= \mathbf{v}^T\mathbf{T}^{zs}\mathbf{s} \\ &= \mathbf{v}^T(\mathbf{R}^n \otimes \mathbf{R}^m)[(\mathbf{a} \otimes \mathbf{o}^m) + (\mathbf{o}^n \otimes \mathbf{b})] \\ &= \mathbf{v}^T[(\mathbf{R}^n\mathbf{a} \otimes \mathbf{R}^m\mathbf{o}^m) + (\mathbf{R}^n\mathbf{o}^n \otimes \mathbf{R}^m\mathbf{b})] \\ &= 0 \end{aligned}$$

Substituting the decomposition of $\mathbf{v}$ in (41), rewriting $\mathbf{T}^{cn}$ with (43) and $\mathbf{i}^{cn}$ with (44), leads to,

$$\begin{aligned} E^{cn} &= -\tfrac{1}{2}\theta(\mathbf{v}^{nz} + \mathbf{v}^{zs})^T(\mathbf{T}^{zs} - \mathbf{I}^{nm})(\mathbf{v}^{nz} + \mathbf{v}^{zs}) \\ &\quad -\theta\mathbf{s}^T(\mathbf{v}^{nz} + \mathbf{v}^{zs}) \end{aligned}$$

But $\mathbf{T}^{zs}$ is a projection matrix (by definition symmetric),

$$\begin{aligned} \text{hence } \mathbf{T}^{zs}\mathbf{v}^{zs} &= \mathbf{T}^{zs}\mathbf{T}^{zs}\mathbf{v} = \mathbf{v}^{zs} \\ \text{and } \mathbf{v}^{zs\,T}\mathbf{T}^{zs} &= \mathbf{v}^T\mathbf{T}^{zs}\mathbf{T}^{zs} = \mathbf{v}^{zs\,T} \end{aligned}$$

Also,

$$\mathbf{v}^{nz\,T}\mathbf{T}^{zs} = \mathbf{0}^T \quad \text{and} \quad \mathbf{T}^{zs}\mathbf{v}^{nz} = \mathbf{0}$$

Thus,

$$\begin{aligned} E^{cn} &= -\tfrac{1}{2}\theta[\mathbf{v}^{zs\,T}\mathbf{v}^{zs} - \mathbf{v}^{zs\,T}\mathbf{v}^{zs} - \mathbf{v}^{nz\,T}\mathbf{v}^{nz}] \\ &\quad -\theta\mathbf{s}^T\mathbf{v}^{nz} \\ &= \tfrac{1}{2}\theta[\mathbf{v}^{nz\,T}\mathbf{v}^{nz} - 2\mathbf{s}^T\mathbf{v}^{nz}] \\ \Rightarrow E^{cn} &= \tfrac{1}{2}\theta|\mathbf{v}^{nz} - \mathbf{s}|^2 - \tfrac{1}{2}\theta|\mathbf{s}|^2 \end{aligned} \tag{45}$$

Therefore, since $\mathbf{s}$ is fixed, E^{cn} is minimised when

$$\mathbf{v}^{nz} = \mathbf{s}$$

which corresponds to $\mathbf{v}$ satisfying the valid subspace equation,

$$\mathbf{v} = \mathbf{s} + \mathbf{v}^{zs} = \mathbf{s} + \mathbf{T}^{zs}\mathbf{v}$$

Expressions for $\mathbf{T}^{op}$ and $\mathbf{i}^{op}$

Solving a combinatorial optimization problem also involves finding a valid solution which is in some sense of optimum 'cost'. Since the network operates by minimising a quadratic energy function, this reduces to finding some matrix $\mathbf{T}^{op}$ and vector $\mathbf{i}^{op}$ such that E^{op} monotonically corresponds to the cost of valid solution $\mathbf{v}(\mathbf{p})$.

Naturally the form of E^{op} is highly problem specific. Consider as examples the TSP and GPP.

Let,

$$\begin{aligned} \mathbf{T}^{op} &= -\mathbf{T}^{pq} + \beta\mathbf{I}^{nm} \\ \mathbf{i}^{op} &= \mathbf{0}\text{where } \mathbf{T}^{pq} = (\mathbf{P} \otimes \mathbf{Q}) \end{aligned} \tag{46}$$

For a TSP with N cities let:

$\mathbf{P}$ be the $N \times N$ matrix of intercity distances given by

$$P_{xy} = \text{distance between city } x \text{ and city } y \tag{47}$$

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

$$Q_{ij} = \delta_{j-1,i} + \delta_{j-1,i-2} \tag{48}$$

where $i, j \in \{1, \cdots, N\}$
and where all the subscripts of δ in (48) are given modulo N.

If $N = 5$ then $\mathbf{Q}$ would be given by,

$$\mathbf{Q} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

For a GPP with N nodes let:

P be the $N \times N$ matrix of edge weights given by

$$P_{xy} = \begin{array}{l}\text{weight of edge between node } x \\ \text{and node } y\end{array} \tag{49}$$

Q be minus the $M \times M$ identity matrix, i.e,

$$\mathbf{Q} = -\mathbf{I}^m \tag{50}$$

Now consider the value of $E^{pq} = \frac{1}{2}\mathbf{v}(\mathbf{p})^T(\mathbf{P}\otimes\mathbf{Q})\mathbf{v}(\mathbf{p})$ with the above settings for **P** and **Q**:

For the TSP case, if $\mathbf{v}(\mathbf{p})$ is a TSP valid-solution with a corresponding tour length $l(\mathbf{p})$ then,

$$l(\mathbf{p}) = \frac{1}{2}\mathbf{v}(\mathbf{p})^T(\mathbf{P}\otimes\mathbf{Q})\mathbf{v}(\mathbf{p}) = E^{pq} \tag{51}$$

Hence E^{pq} is a monotonic function of the 'cost' of **p** i.e. the corresponding tour length.

For the GPP case, let,

S^A be the sum of the weights of all edges.

S^S be the sum of the weights of all edges linking nodes in the same partition.

S^D be the sum of the weights of all edges linking nodes in different partitions, i.e. the edges which are cut by a partition.

If $\mathbf{v}(\mathbf{p})$ is a GPP valid-solution then,

$$-S^S = \frac{1}{2}\mathbf{v}(\mathbf{p})^T(\mathbf{P}\otimes\mathbf{Q})\mathbf{v}(\mathbf{p}) = E^{pq} \tag{52}$$

The object in a GPP is to minimise the number of cut edges which corresponds to minimising S^D. Note that,

$$S^A = S^S + S^D$$

Hence, since S^A is constant, E^{pq} is a monotonic function of the 'cost' of **p**, i.e. S^D.

Thus

$$\begin{aligned} E^{op}(\mathbf{v}(\mathbf{p})) &= -\tfrac{1}{2}\mathbf{v}(\mathbf{p})^T\mathbf{T}^{op}\mathbf{v}(\mathbf{p}) - (\mathbf{i}^{op})^T\mathbf{v}(\mathbf{p}) \\ &= E^{pq} - \beta\mathbf{v}(\mathbf{p})^T\mathbf{I}^{nm}\mathbf{v}(\mathbf{p}) \\ &= E^{pq} - \beta|\mathbf{v}(\mathbf{p})|^2 \end{aligned}$$

But $|\mathbf{v}(\mathbf{p})|^2$ is constant for both TSPs and GPPs, hence E^{op} must also be a monotonic function of the 'cost' of $\mathbf{v}(\mathbf{p})$.

Setting $\beta = 0$ makes the formulation of T^{op} for the TSP identical to the formulation used for $\mathbf{T}^d$ by Hopfield and Tank in [4]. The reason for including the β term is that it allows a constant to be added to all the eigenvalues of $\mathbf{T}^{op}$ without affecting the properties of E^{op}. The value of β, as will be shown later, plays a crucial role in enforcing convergence to a hypercube corner.

A Modified network with increased efficiency

If T^{cn} and i^{cn} are set according to (43) and (44), it can be seen that the network as proposed by Hopfield and Tank in [4] does indeed roughly follow steps (i)$\cdots$(iii). This is because the minimisation of E^{cn} should always ensure that **v** is on the valid subspace, while the minimisation of E^{op} should move **v** out towards a hypercube corner that is of minimum 'cost'.

However a more detailed analysis of how the network follows steps (i)$\cdots$(iii) reveals that it does this in an inefficient and unreliable way. This is due to the fact that the minimisation of E^{op} interferes with the minimisation of E^{cn}. Consequently θ in (43) has to be made large enough to ensure that E^{cn} is sufficiently dominant to enforce confinement to the valid subspace.

In computer simulations of the Hopfield network, a discrete approximation to the continuous differential equation (1) has to be used in which,

$$\begin{aligned} \Delta\mathbf{v} &= g(\mathbf{u}+\Delta\mathbf{u}) - g(\mathbf{u}) \\ \text{where } \Delta\mathbf{u} &= -\Delta t\nabla E = -\Delta t\nabla(E^{cn}+E^{op}) \end{aligned}$$

The need to ensure E^{cn} dominates over E^{op}, means that most of the change in **v** caused by $\Delta\mathbf{v}$ (via $\Delta\mathbf{u}$) will be concerned with the minimisation of E^{cn} and very little with the minimisation of E^{op}. This makes the simulation very inefficient, since most of the time will be spent enforcing the valid subspace confinement. In order to overcome this problem we propose the network implementation shown in Fig 3.

The key advantage of the modified network, is that it confines **v** to the valid subspace directly by the projection operation (1), rather than by the indirect minimisation of E^{cn}. The nonlinear operation (2) then ensures **v** is kept within the unit hypercube, by applying a 'symmetric ramp' threshold function to each of the elements of **v**. i.e,

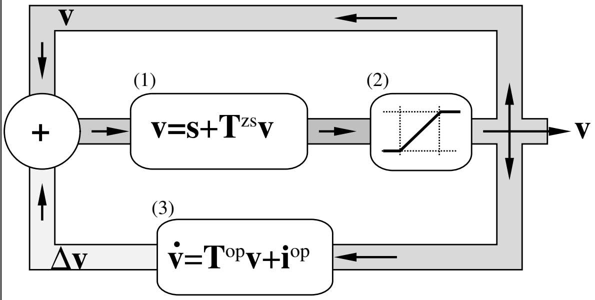
$$\begin{aligned} v_q &\rightarrow g(v_q) \\ \text{where } g(v_q) &= \begin{cases} 1 & \text{if } v_q > 1 \\ v_q & \text{if } 0 \le v_q \le 1 \\ 0 & \text{if } v_q < 0 \end{cases} \end{aligned}$$

The minimisation of E^{op} is achieved by operation (3). This computes the change in **v** as:

$$\Delta\mathbf{v} = \Delta t\frac{d}{dt}(\mathbf{v})$$

where

$$\frac{d}{dt}(\mathbf{v}) = \dot{\mathbf{v}} = \mathbf{T}^{op}\mathbf{v} + \mathbf{i}^{op} \tag{53}$$



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

$$\mathbf{v} = \mathbf{s} + \mathbf{T}^{zs}\mathbf{v}$$

$$\begin{array}{lrl} \text{where} & \mathbf{s} = & (\mathbf{a} \otimes \mathbf{o}^m) + (\mathbf{o}^n \otimes \mathbf{b}) \\ \text{and} & \mathbf{T}^{zs} = & (\mathbf{R}^n \otimes \mathbf{R}^m) \end{array}$$

(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

$$\Delta\mathbf{v} = \Delta t \frac{d}{dt}(\mathbf{v}) = \Delta t \dot{\mathbf{v}} = \Delta t(\mathbf{T}^{op}\mathbf{v} + \mathbf{i}^{op})$$

Figure 3: Schematic diagram of modified network implementation

Recalling from (42) that,

$$E^{op} = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{op}\mathbf{v} - (\mathbf{i}^{op})^T\mathbf{v}$$

it can be seen that,

$$\begin{array}{rcl} \Delta\mathbf{v} & = & -\Delta t \nabla E^{op} \\ \Rightarrow \Delta\mathbf{v}^T \nabla E^{op} & \leq & 0 \end{array}$$

Hence the network will always change $\mathbf{v}$ in a way that minimises E^{op}. In contrast to the Hopfield network, $\Delta\mathbf{v}$ is exclusively 'concerned' with the minimisation of E^{op}, and consequently computer simulations of the modified network converge in far fewer iterations than equivalent simulations of the Hopfield network. Typically, for 30-city TSPs, simulations of the modified network converge in ~ 2000 iterations as opposed to ~ 20000 iterations for the Hopfield network.

The modified network has the further important advantage of having just one parameter, Δt [4]. This compares with at worst five parameters (A, B, C, D and Δt) for the network as formulated by Hopfield and Tank in their 1985 paper [4] and at best two parameters (θ and Δt) for the Hopfield network as formulated in the previous subsection.

Finally the basic operations of the modified network - multiplication by a matrix followed by addition of a fixed vector - are identical to those of the original Hopfield network. Hence the modified network shares the same properties in terms of parallel analogue hardware implementation as the original Hopfield network.

Confinement to the valid subspace and the unit hypercube

The assumption that the repeated application of operations (1) and (2) will confine $\mathbf{v}$ both to the valid subspace and the unit hypercube is crucial to the viability of the modified network. This subsection is concerned with the proof of this assumption.

First assume that $\mathbf{v}$ at time t satisfies both these constraints. The value of $\mathbf{v}$ at time $t+\Delta t$ is given by $\mathbf{v}+\Delta\mathbf{v}$ where $\Delta\mathbf{v}$ is given by the dynamic equation (53),

$$\Delta\mathbf{v} = \Delta t \dot{\mathbf{v}} = \Delta t[\mathbf{T}^{op}\mathbf{v} + \mathbf{i}^{op}]$$

If $\mathbf{v}+\Delta\mathbf{v}$ is to remain both within the valid subspace and the unit hypercube, then $\Delta\mathbf{v}$ must satisfy the following conditions:

(c1) $\Delta\mathbf{v}$ is such that $\mathbf{v}+\Delta\mathbf{v}$ remains within the unit hypercube.

(c2) $\Delta\mathbf{v}$ lies wholly in the zerosum subspace.

Unfortunately it is very unlikely that $\Delta\mathbf{v}$ will satisfy both (c1) and (c2). However there will always be a component of $\Delta\mathbf{v}$, say $\Delta\hat{\mathbf{v}}$, which does satisfy both these conditions.

It is shown in Appendix A that there exists a projection matrix $\hat{\mathbf{T}}$ such that,

$$\Delta\hat{\mathbf{v}} = \hat{\mathbf{T}}\Delta\mathbf{v} \tag{54}$$

and that repeated application of operations (1) and (2) on $\mathbf{v}+\Delta\mathbf{v}$, is equivalent to the multiplication of $\Delta\mathbf{v}$ by $\hat{\mathbf{T}}$. i.e,

$$\mathbf{v} + \Delta\mathbf{v} \overset{(1)(2)\cdots(1)(2)}{\longrightarrow} \mathbf{v} + \hat{\mathbf{T}}\Delta\mathbf{v} = \mathbf{v} + \Delta\hat{\mathbf{v}}$$

Since $\mathbf{v}+\Delta\hat{\mathbf{v}}$ satisfies (c1) and (c2) it follows that repeated application of operations (1) and (2) does indeed confine $\mathbf{v}$ to both the valid subspace and the unit hypercube, regardless of the change in $\mathbf{v}$ effected by $\Delta\mathbf{v}$.

[4] In itself the value of Δt is a very significant factor in the overall efficiency of the network simulation, and even with the modified network, finding the best value is somewhat of a black art. A description of the algorithm used for 30 and 50-city TSPs is given in Appendix B.

Convergence to a hypercube corner

So far it has been assumed that by confining $\mathbf{v}$ to the valid subspace the network will eventually converge to a hypercube corner. There is, however, no guarantee that this will be the outcome. The aim of this subsection is to show how by choosing an appropriate value for β in (46) that convergence to a hypercube corner can indeed be guaranteed.

If $\mathbf{v}$ is confined to the valid subspace and $\mathbf{v}$ must remain within the unit hypercube, then it can be shown that $|\mathbf{v}|$ is maximised when $\mathbf{v}$ is a valid solution hypercube corner, i.e. $\mathbf{v} = \mathbf{v}(\mathbf{p})$. Hence a sufficient condition that will ensure the network converges to a valid solution hypercube corner is that the change in $\mathbf{v}$ at each time step Δt should always increase $|\mathbf{v}|$. Equation (54) gives this change, $\Delta\hat{\mathbf{v}}$, in a form that accounts for the confining effects of operations (1) and (2). Thus if the addition of $\Delta\hat{\mathbf{v}}$ is to increase the magnitude of $|\mathbf{v}|$ then,

$$\mathbf{v}^T \Delta\hat{\mathbf{v}} \geq 0 \tag{55}$$

$$\begin{aligned} \Rightarrow 0 &\leq \mathbf{v}^T \hat{\mathbf{T}} \Delta \mathbf{v} \\ \Rightarrow 0 &\leq \mathbf{v}^T \hat{\mathbf{T}} \Delta t (\mathbf{T}^{op}\mathbf{v} + \mathbf{i}^{op}) \\ \Rightarrow 0 &\leq \mathbf{v}^T \hat{\mathbf{T}} \mathbf{T}^{op}\mathbf{v} + \mathbf{v}^T \hat{\mathbf{T}} \mathbf{i}^{op} \end{aligned} \tag{56}$$

Substituting (46) in the condition for convergence to a valid solution hypercube corner given by (56), leads to,

$$\begin{aligned} 0 &\leq \mathbf{v}^T \hat{\mathbf{T}} (-\mathbf{T}^{pq} + \beta \mathbf{I}^{nm})\mathbf{v} + \mathbf{v}^T \hat{\mathbf{T}} \mathbf{i}^{op} \\ \Rightarrow 0 &\leq -\mathbf{v}^T \hat{\mathbf{T}} \mathbf{T}^{pq}\mathbf{v} + \mathbf{v}^T \hat{\mathbf{T}} \mathbf{i}^{op} + \beta \mathbf{v}^T \hat{\mathbf{T}} \mathbf{v} \\ \Rightarrow 0 &\leq -\mathbf{v}^T \hat{\mathbf{T}} \mathbf{T}^{pq}\mathbf{v} + \mathbf{v}^T \hat{\mathbf{T}} \mathbf{i}^{op} + \beta |\hat{\mathbf{v}}|^2 \end{aligned} \tag{57}$$

The $\beta|\hat{\mathbf{v}}|^2$ term in (57) is always positive, so there exists some value of β which will guarantee that the condition specified by (57) is satisfied.

Hence by choosing a large enough positive value for β it is possible to ensure that the modified network will always converge to a valid solution hypercube corner.

Isomorphism with Quadratic Programming Problems

The operation of the modified network can be summarised as follows:

$$\begin{array}{llcl} \text{minimise} & E^{op} & = & -\frac{1}{2}\mathbf{v}^T \mathbf{T}^{op}\mathbf{v} - \mathbf{i}^{op^T}\mathbf{v} \\ \text{subject to} & \mathbf{v}^{nz} & = & \mathbf{s} \\ \text{and} & 0 \leq & v_q & \leq 1 \quad q \in \{1 \cdots NM\} \end{array} \tag{58}$$

But,

$$\mathbf{v}^{nz} = \mathbf{v} - \mathbf{v}^{zs} = (\mathbf{I}^{nm} - \mathbf{T}^{zs})\mathbf{v}$$

Hence

$$\begin{aligned} \mathbf{v}^{nz} &= \mathbf{s} \\ \Rightarrow (\mathbf{I}^{nm} - \mathbf{T}^{zs})\mathbf{v} &= \mathbf{s} \end{aligned} \tag{59}$$

Equation (59) is equivalent to a set of equality conditions on $\mathbf{v}$, while the conditions specified by,

$$0 \leq v_q \leq 1 \quad q \in \{1 \cdots NM\}$$

are simply a set of inequality conditions on $\mathbf{v}$, hence (58) is exactly the form of a Quadratic Programming Problem [7, ch10].

N.B. If $\mathbf{T}^{op} = \mathbf{0}$ then the function being minimised in (58) is simply $-\mathbf{i}^{op}\mathbf{v}$. This is linear and hence (58) reduces to exactly the form of a linear programming problem.

Non-convexity: Global Minimum versus Local Minima

If the function to be minimised in a Quadratic Programming problem is convex, then the problem reduces to a Convex Programming Problem and has a single global minimum [7, ch9.4]. This is always the case if $|\mathbf{T}^{op}| = 0$ and the problem is of the Linear Programming class. For this reason, if the network is used to solve problems of the Linear Programming class, it should always achieve the global optimum solution.

On the other hand Quadratic Programming problems of the form (58) only reduce to Convex Programming problems if $\mathbf{T}^{op}$ is negative semi-definite. If this is not the case, and $\mathbf{T}^{op}$ is indefinite, then instead of one global minimum there will be many local minima: the number of which increases until $\mathbf{T}^{op}$ is positive definite, in which case it is possible that all the hypercube corners in the valid subspace are local minima.

Recall from (46) that,

$$\mathbf{T}^{op} = -\mathbf{T}^{pq} + \beta \mathbf{I}^{nm}$$

Clearly whatever the form of $\mathbf{T}^{pq}$, the value of β can be used to control whether $\mathbf{T}^{op}$ is negative definite or positive definite.

Unfortunately, as shown previously, to ensure the network converges to a valid solution hypercube corner, β has to made large enough to ensure (57) is satisfied, which is likely to make $\mathbf{T}^{op}$ positive definite.

Consequently there is a conflict between the need to ensure that the network converges to a valid solution hypercube corner and the desirability of keeping the number of local minima small. This will severely degrade the performance of the network in solving optimization problems, since it is very likely the network will converge to a sub-optimal local minimum.

Matrix Graduated Non-Convexity

Matrix Graduated Non-Convexity, or MGNC, is a way of overcoming the conflict between the need to ensure that the network converges to a valid solution hypercube corner and the desirability of keeping the number of local minima small. It is based on the concept of Graduated Non-Convexity developed in [8] and also on the ideas of [10].

The principle behind MGNC is that initially keeping E^{op} as convex as possible reduces the overall likelihood that the network will fall into a sub-optimal local minimum. As $\mathbf{v}$ grows towards the bounds of the unit hypercube it becomes necessary to gradually increase β to ensure that condition (57) is satisfied. Increasing β makes E^{op} 'more non-convex' and hence increases the chance of falling into a sub-optimal local minima. However, by this time $\mathbf{v}$ should lie in a region of space well away from these sub-optimal local minima, and so $\mathbf{v}$ should eventually converge to an optimal or near optimal valid solution.

The effect of changing β on the convexity of E^{op} can be seen by expressing E^{op} in terms of the eigenvectors and eigenvalues of $\mathbf{T}^{op}$.

Let ψ_p be an eigenvalue of $\mathbf{T}^{pq}$ with a corresponding normalised eigenvector $\mathbf{x}^p$. It can be seen that $\mathbf{x}^p$ is also an eigenvector of $\mathbf{T}^{op}$ with an eigenvalue of ϕ_p, since,

$$\begin{aligned}
\mathbf{T}^{op}\mathbf{x}^p &= (-\mathbf{T}^{pq} + \beta\mathbf{I}^{nm})\mathbf{x}^p \\
&= (-\psi_P + \beta)\mathbf{x}^p \\
\Rightarrow \mathbf{T}^{op}\mathbf{x}^p &= \phi_p \mathbf{x}^p \\
\text{with } \phi_p &= -\psi_p + \beta \quad p \in \{1, \cdots, NM\}
\end{aligned}$$

Now assume the eigenvalues of $\mathbf{T}^{pq}$ are labelled in order of decreasing magnitude, i.e,

$$\psi_1 > \psi_2 > \cdots > \psi_{NM}$$

Let $\mathcal{P}$ be the set of all positive eigenvalues of $\mathbf{T}^{op}$, in other words:

$$\mathcal{P} = \{q : \phi_q \geq 0\} = \{q : \psi_q < \beta\} \qquad (60)$$

Since $\mathbf{T}^{op}$ is symmetric, its normalised eigenvectors form an orthogonal set which fully spans NM dimensional space. Hence $\mathbf{v}$ and E^{op} can be expressed in terms of the $\mathbf{x}^p$, with,

$$\mathbf{v} = \sum_{p=1}^{NM} \alpha_p \mathbf{x}^p \quad \text{where} \quad \alpha_p = \mathbf{v}^T\mathbf{x}^p \qquad (61)$$

$$\begin{aligned}
E^{op} &= -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{op}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{op} \\
&= -\tfrac{1}{2}(\sum_p \alpha_p\mathbf{x}^p)^T\mathbf{T}^{op}(\sum_q \alpha_q\mathbf{x}^q) - \mathbf{v}^T\mathbf{i}^{op} \\
&= -\tfrac{1}{2}\sum_{p=1}^{NM} \phi_p(\alpha_p)^2 - \mathbf{v}^T\mathbf{i}^{op} \qquad (62)
\end{aligned}$$

(62) can be rewritten as follows,

$$\begin{aligned}
E^{op} &= E^+ + E^- - \mathbf{v}^T\mathbf{i}^{op} \\
\text{where } E^+ &= -\tfrac{1}{2}\sum_{p\notin\mathcal{P}} \phi_p(\alpha_p)^2 \qquad (63) \\
\text{and } E^- &= -\tfrac{1}{2}\sum_{q\in\mathcal{P}} \phi_q(\alpha_q)^2 \qquad (64)
\end{aligned}$$

From the definition of E^+ in (63) it follows that E^+ is a positive definite (i.e. convex) function over a $NM - |\mathcal{P}|$ dimensional subspace of $\mathbf{v}$. Similarly E^- is a negative definite (i.e. concave) function over a $|\mathcal{P}|$ dimensional subspace of $\mathbf{v}$. The size of $\mathcal{P}$, (i.e. $|\mathcal{P}|$) is determined by the value of β. For example if $\beta \geq \psi_1$ then $\mathcal{P} = \{1, \cdots, NM\}$ and if $\beta < \psi_{NM}$ then $\mathcal{P} = \emptyset$. Thus by modifying β the dimensionality of the subspaces over which E^{op} is convex and concave can be varied. It is in this sense that varying β changes the convexity of E^{op}, hence the term Matrix Graduated Non-Convexity, since the convexity of E^{op} is altered by changing the matrix $\mathbf{T}^{op}$.

Implementation of MGNC [5] on the Modified Network

(i) The modified network is initialised with a $\mathbf{v}$ which is both within the unit hypercube and the valid subspace. Ideally

$$\mathbf{v}(0) = \mathbf{s} + \mathbf{T}^{zs}\mathbf{v}^{rnd}$$

where $\mathbf{v}^{rnd}$ is a randomly generated vector of small magnitude ($|\mathbf{v}^{rnd}| \approx 10^{-6}$). Also β is set to a value such that $\psi_{NM-1} > \beta > \psi_{NM}$. In this case $\mathcal{P}$ has one element and E^{op} will be almost a convex function.

[5] It is possible to view the MGNC algorithm as a type of annealing, and in a sense varying β corresponds to varying the temperature (or neural gain) in Mean Field Annealing networks as proposed in [11] and [12]. However the way in which MGNC varies the convexity of E is very different to the variation produced by the annealing of temperature (or neural gain) in Mean Field Annealing networks.

(ii) The network is allowed to run until $|\mathbf{v}|$ starts to decrease (or increase very slowly). This corresponds to condition (57) no longer being satisfied. To correct this β is increased to a value just large enough to ensure (57) is satisfied and the network continues to increase $|\mathbf{v}|$.

(iii) (ii) is repeated until the network converges to a valid solution.

Results of simulations using the modified network to solve 30 and 50-city TSPs

The aim in presenting these results is to provide an overall confirmation of the theory developed in this paper. A detailed study, together with a theoretical analysis of how the MGNC algorithm ensures near optimal solutions, will be presented in a future paper. The results are based on computer simulations of the modified network being used to solve 30-city and 50-city TSPs, and are shown in Figs 4,5,6 and 7.

In Fig 4, the evolution of the output of a 30-city TSP network is presented as a set of 3D mesh plots of the matrix representation $\mathbf{V}$ of the network output $\mathbf{v}$, where,

$$\mathbf{v} = \mathrm{vec}(\mathbf{V}^T)$$

The co-ordinates of the cities are identical to the ones used by Hopfield and Tank for the 30-city TSP in [4], and hence act as a useful benchmark test. As an illustration of the effect of confinement to the valid subspace, Fig 5 shows the evolution of the values of a single row of $\mathbf{V}$. It can be seen that, as predicted theoretically, the sum across the row is always fixed at 1, regardless of the values of the individual elements of $\mathbf{V}$. Further, as the number of iterations increases, the number of non-zero elements in the row decreases, until a final solution is achieved. The tour corresponding to the final solution achieved by the network is shown in Fig 6. The results for the 30-city tour are summarised in Fig 7 along with some results on 50-city TSPs based on Durbin and Wilshaw's co-ordinates [5]. For details of the computer simulation implementation, see Appendix B.

Overall the results show the network achieving solutions within 2% of the (experimentally determined) optimum. This is of comparable quality to both Durbin and Wilshaw's elastic net [5] and to simulated annealing (based on the values given in [5]), whilst being significantly better than both the simple nearest neighbour heuristic and the results obtained by Hopfield and Tank in [4].

Conclusions

A general expression has been developed for the valid subspace, which is applicable to a large class of combinatorial optimization problems. This class includes the Travelling Salesman problem and the Graph Partitioning problem. It also includes many other combinatorial optimization problems, such as certain types of Dynamic Programming problems, which have so far not been mapped onto the Hopfield model. A new approach has then been described, which uses this expression as the basis for deriving the form of the network connection matrix and input bias vector. Unlike the traditional approaches based on first deriving the network energy function, this approach ensures confinement to the valid subspace and hence eventual convergence to a valid solution. Further this approach allows a much clearer picture to be developed of how the Hopfield network solves combinatorial optimization problems. A consequence of this is the proposal of a modified network and the Matrix Graduated Non-convexity algorithm. Finally the main theoretical findings of this paper are confirmed in computer simulations of a modified network using the Matrix Graduated Non-convexity algorithm. These show the modified network efficiently achieving solutions within 2% of the global optimum for 30 and 50-city Travelling Salesman problems.

References

[1] Aiyer,S.V.B, Niranjan,M, Fallside,F, *A Theoretical Investigation into the performance of the Hopfield Model* IEEE Trans. Neural Networks, Vol NN-1, Issue 2, p204-215, **1990**.

[2] Hopfield,J.J, *Neural networks and physical systems with emergent collective computational abilities* Proc.Natl.Acad.Sci. USA 79, 2554-2558, **1982**.

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

[4] Hopfield,J.J, Tank,D.W, *Neural Computation of Decisions in Optimization Problems* Biological Cybernetics. 52, 1-25, **1985**.

[5] Durbin,R, Wilshaw,D *An Analogue Approach to the Travelling Salesman Problem Using an Elastic Net Method* Nature 326(6114) p689-691, April **1987**.

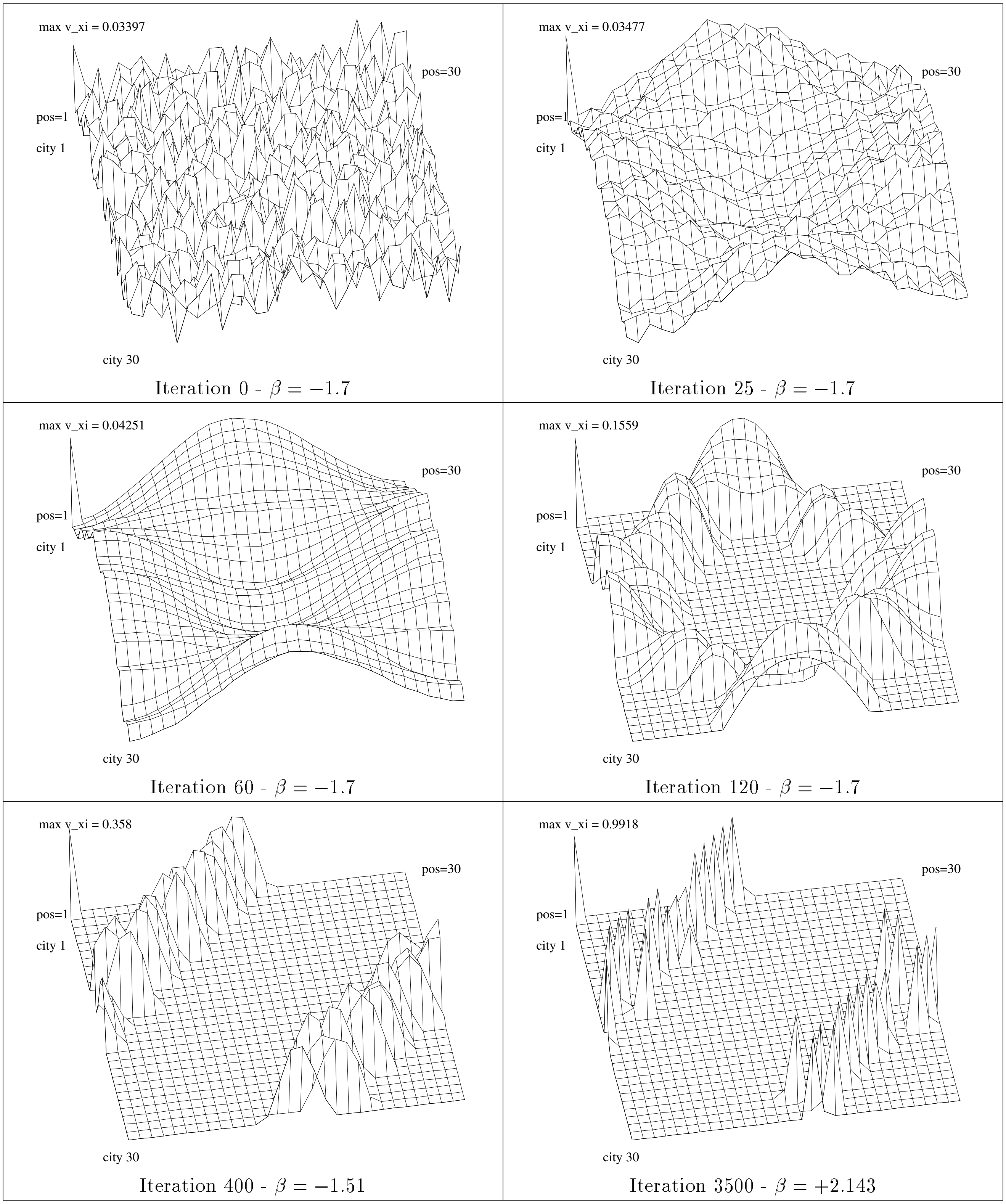
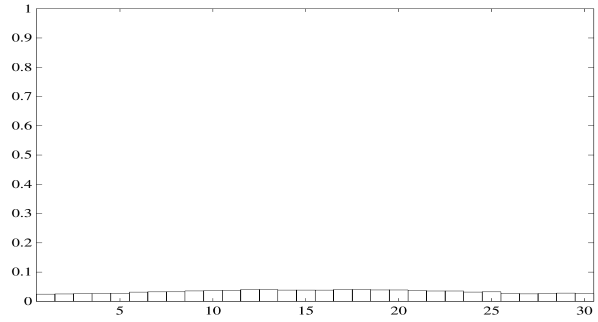
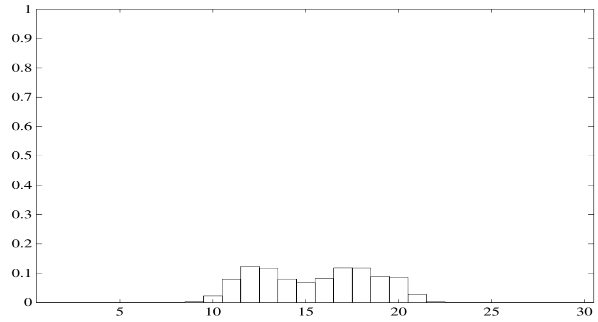


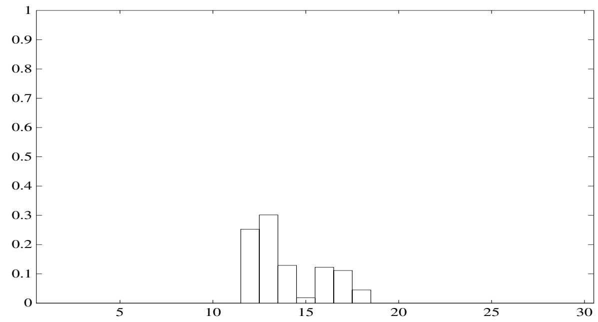
Figure 4: 3D Mesh Plots showing the evolution of $\mathbf{V}$ (the matrix representation of the network output vector $\mathbf{v}$, where $\mathrm{vec}(\mathbf{V}^T) = \mathbf{v}$) in a simulation of the modified network solving Hopfield and Tank's 30 city TSP. **N.B.** The value of β shown here does not correspond exactly to the description given in the paper; for the exact description see Appendix B.



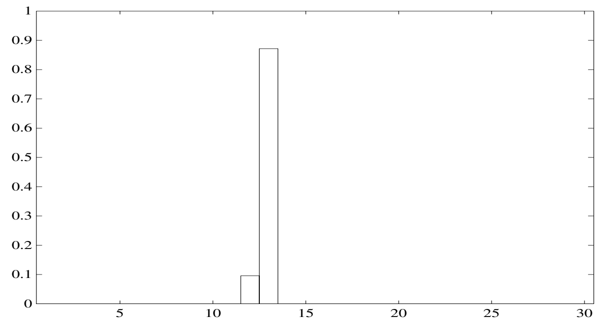
Iteration 60



Iteration 120



Iteration 400



Iteration 3500

Figure 5: Plot of the 2^{nd} row of $\mathbf{V}$, for the 30-city TSP shown in Fig 5, showing how the size of the group of non-zero elements decreases with the number of iterations, while the total sum across the row remains constant at 1.

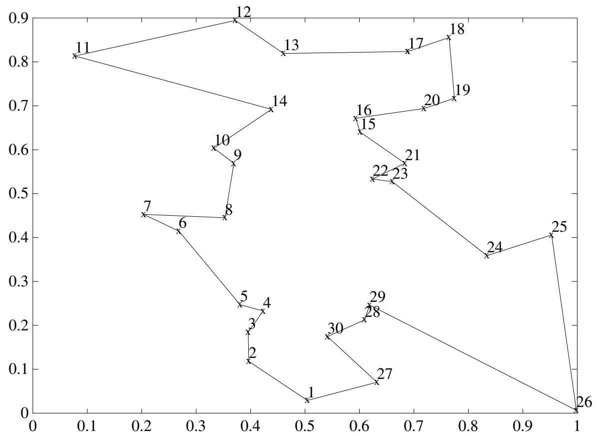


Figure 6: Plots of the final tour achieved by the modified network for Hopfield and Tank's 30-city TSP. The tour length is 4.34. This compares with 4.26 for the optimal tour, which follows the order of the city numbers.

	Hopfield & Tank 30 city TSP	Durbin & Wilshaw 50a city TSP	Durbin & Wilshaw 50c city TSP
SUN IV CPU time	2 mins	10 mins	10 mins
Modified Net. best	**4.34**	**5.92**	**5.67**
Hopfield & Tank's best	5.07		
Elastic Net. best	4.26	5.97	5.74
Simulated Annealing		5.88	5.65
Nearest Neighbour	4.75	6.58	6.28
Expt. Det. Optimum	4.26	5.84	5.57

Figure 7: Table summarising the results achieved by the modified network on 30 and 50-city TSPs.

[6] Wilson,V, Pawley,G.S *On the Stability of the TSP Problem Algorithm of Hopfield and Tank* Biological Cybernetics 58, p63-70 **1988**.

[7] Fletcher,R, *Practical Methods of Optimization* Wiley-Interscience Publications, John Wiley & Sons **1987**.

[8] Blake,A, Zisserman,A, *Visual Reconstruction* The MIT Press series in artificial intelligence, London **1987**.

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

[10] Styblinski,M.A, Tang,T.-S, *Experiments in Non-Convex Optimization: Stochastic Approximation with Function Smoothing and Simulated-Annealing* Neural Networks, Vol 3, No. 4, p467-483, Pergamon Press **1990**.

[11] Van den Bout, D.E, Miller III,T.K, *Graph Partitioning Using Annealed Networks* IEEE Trans. Neural Networks, Vol NN-1, Issue 2, p192-203, **1990**.

[12] Peterson,C, Soderberg,B, *A new method for mapping optimization problems onto neural networks* Int. Journal Neural Systems, Vol 1, No. 1, **1989**.

Appendix A

Proof of the equivalence of operations (1) and (2) of the modified network with the multiplcation of $\Delta\mathbf{v}$ by $\hat{\mathbf{T}}$

Let $\mathcal{A}^0$ denote the set of elements of $\mathbf{v}$ for which the inequality constraint $0 \leq v_q$ is active i.e. $v_q = 0$ and $\Delta v_q < 0$. The set $\mathcal{A}^0$ is defined formally by,

$$\mathcal{A}^0 = \{q : v_q = 0 \text{ and } \Delta v_q < 0\} \quad q \in \{1, \cdots, NM\} \tag{65}$$

Let $\mathcal{A}^1$ denote the set of elements of $\mathbf{v}$ for which the inequality constraint $1 \geq v_q$ is active i.e. $v_q = 1$ and $\Delta v_q > 0$. The set $\mathcal{A}^1$ is defined formally by,

$$\mathcal{A}^1 = \{q : v_q = 1 \text{ and } \Delta v_q > 0\} \quad q \in \{1, \cdots, NM\} \tag{66}$$

Let $\mathcal{A}$ denote the set given by the union of $\mathcal{A}^0$ and $\mathcal{A}^1$, i.e,

$$\mathcal{A} = \mathcal{A}^0 \cup \mathcal{A}^1 \tag{67}$$

It can be seen that if,

$$\Delta\hat{v}_q = 0 \ \ \forall\, q \in \mathcal{A} \tag{68}$$

and $|\Delta\hat{\mathbf{v}}|$ is small enough to ensure that,

$$0 \leq v_p + \Delta\hat{v}_p \leq 1 \ \text{ for all } p \notin \mathcal{A} \tag{69}$$

then $\Delta\hat{\mathbf{v}}$ satisfies (c1).

Let $\mathbf{T}^{ia}$ be a $NM \times NM$ matrix where,

$$[\mathbf{T}^{ia}]_{pq} = \begin{cases} 0 & \text{if } p \neq q \\ 0 & \text{if } p = q \quad \text{and } p \in \mathcal{A} \\ 1 & \text{if } p = q \quad \text{and } p \notin \mathcal{A} \end{cases} \tag{70}$$

It can be seen $\mathbf{T}^{ia}$ is a projection matrix since,

$$\mathbf{T}^{ia}\mathbf{T}^{ia} = \mathbf{T}^{ia}$$

Further if $\Delta\hat{\mathbf{v}}$ satisfies (68) then,

$$\Delta\hat{\mathbf{v}} = \mathbf{T}^{ia}\Delta\hat{\mathbf{v}} \tag{71}$$

Similarly if $\Delta\hat{\mathbf{v}}$ satisfies (c2) (i.e. $\Delta\hat{\mathbf{v}}$ lies in the zerosum subspace) then,

$$\Delta\hat{\mathbf{v}} = \mathbf{T}^{zs}\Delta\hat{\mathbf{v}} \tag{72}$$

Let the subspace spanned by all vectors $\mathbf{u}$ which satisfy $\mathbf{u} = \mathbf{T}^{ia}\mathbf{u}$ be termed the inactive subspace. It can be seen that $\hat{\mathbf{T}}$ is simply the matrix which projects $\mathbf{v}$ into the subspace which is the intersection of the zerosum subspace and the inactive subspace. Hence $\hat{\mathbf{T}}$ satisfies,

$$\begin{aligned} \hat{\mathbf{T}} &= \mathbf{T}^{zs}\hat{\mathbf{T}} \\ \hat{\mathbf{T}} &= \mathbf{T}^{ia}\hat{\mathbf{T}} \end{aligned} \tag{73}$$

Now assuming $\Delta\mathbf{v}^{(0)} = \Delta\mathbf{v}$, let,

$$\Delta\mathbf{v}^{(n)} = \Delta\hat{\mathbf{v}} + \mathbf{u}^{(n)}$$

where,

$$\begin{aligned} \Delta\mathbf{v}^{(n)} &= \begin{cases} \mathbf{T}^{zs}\Delta\mathbf{v}^{(n-1)} & \text{if } n \text{ is odd} \\ \mathbf{T}^{ia}\Delta\mathbf{v}^{(n-1)} & \text{if } n \text{ is even} \end{cases} \\ &= \begin{cases} \Delta\hat{\mathbf{v}} + \mathbf{T}^{zs}\mathbf{u}^{(n-1)} & \text{if } n \text{ is odd} \\ \Delta\hat{\mathbf{v}} + \mathbf{T}^{ia}\mathbf{u}^{(n-1)} & \text{if } n \text{ is even} \end{cases} \\ \Rightarrow \mathbf{u}^{(n)} &= \begin{cases} \mathbf{T}^{zs}\mathbf{u}^{(n-1)} & \text{if } n \text{ is odd} \\ \mathbf{T}^{ia}\mathbf{u}^{(n-1)} & \text{if } n \text{ is even} \end{cases} \end{aligned} \tag{74}$$

Since $\mathbf{T}^{zs}$ and $\mathbf{T}^{ia}$ are projection matrices,

$$\begin{aligned} |\mathbf{T}^{zs}\mathbf{u}^{(n-1)}| &\leq |\mathbf{u}^{(n-1)}| \\ \text{and } |\mathbf{T}^{ia}\mathbf{u}^{(n-1)}| &\leq |\mathbf{u}^{(n-1)}| \end{aligned}$$

Hence

$$|\mathbf{u}^{(n)}| \leq |\mathbf{u}^{(n-1)}| \tag{75}$$

Multiplying by $\mathbf{T}^{zs}$ corresponds to operation (1) while multiplying by $\mathbf{T}^{ia}$ corresponds to operation (2). From equation (75) it is clear that as $n \to \infty$, $\Delta\mathbf{v}^{(n)} \to \Delta\hat{\mathbf{v}}$. Hence the repeated application of operations (1) and (2) is equivalent to the projection performed on $\Delta\mathbf{v}$ by multiplication with $\hat{\mathbf{T}}$. (**N.B.** This proof assumes that (74) does not affect the set of active inequality constraints, $\mathcal{A}$, and hence $\mathbf{T}^{ia}$ remains the same for all n. Assuming (69) holds, this can be proved, although for conciseness the proof is not given here.)

Appendix B

Details of implementation for computer simulations using the Modified Network with MGNC to solve the Travelling Salesman Problem

Essentially the modified network was implemented exactly as shown in Fig 3 with $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{s}$ set according to (47), (48) and (38).

The main details of the implementation for 30 and 50-city TSPs are:

- Instead of setting $\mathbf{T}^{op}$ according to (46), the following formulation of $\mathbf{T}^{op}$ was used.

$$\begin{aligned} \mathbf{T}^{op} &= -\mathbf{T}^{pq} - \beta(\mathbf{P} \otimes \mathbf{I}^n) \\ \Rightarrow \mathbf{T}^{op} &= -(\mathbf{P} \otimes [\mathbf{Q} + \beta\mathbf{I}^n]) \qquad (76) \\ \text{with } \mathbf{T}^{pq} &= (\mathbf{P} \otimes \mathbf{Q}) \end{aligned}$$

  The reason for using (76) is that the properties of the eigenvalues of $\mathbf{P}$ for the TSP are such that varying β in (76) varies the convexity of E in a very similar way to varying β in (46). However because of the properties of $\mathbf{Q}$ for the TSP, using (76) with the MGNC algorithm results in a better final solution. A detailed analysis and justification of this will be presented in a future paper.

- For each traversal of the bottom loop in Fig 3 it is necessary to make several traversals of the top loop in order to ensure that operations (1) and (2) are applied enough times to enforce confinement within the valid subspace and unit hypercube. Rigid enforcement of these confinement conditions would in theory require an infinite number of loops, however in practise it was found sufficient to go round the top loop enough times to ensure that,

$$\left| \sum_{p=1}^{N^2} v_q - N \right| < 0.1N \qquad (77)$$

- The time step size Δt was dynamically computed so that

$$\frac{|\frac{\mathrm{d}}{\mathrm{dt}}\hat{\mathbf{v}}|}{|\mathbf{v}^{zs}|}\Delta t = 0.06$$

  An exact calculation of $\frac{\mathrm{d}}{\mathrm{dt}}\hat{\mathbf{v}}$ is very difficult, hence an estimate was obtained by applying operations (1) and (2) to $\mathbf{v} + \frac{\mathrm{d}}{\mathrm{dt}}\mathbf{v}$ and subtracting $\mathbf{v}$ from the result.

- All the matrix multiplications in Fig 3 are of the form

$$\mathbf{u} = (\mathbf{P} \otimes \mathbf{Q})\mathbf{v}$$

  Assuming $\mathbf{u} = \mathrm{vec}(\mathbf{U}^T)$, $\mathbf{v} = \mathrm{vec}(\mathbf{V}^T)$, the following Kronecker product identity can be applied (see [9]):

$$\begin{aligned} \mathbf{u} &= (\mathbf{P} \otimes \mathbf{Q})\mathbf{v} \\ \Rightarrow \mathrm{vec}(\mathbf{U}^T) &= \mathrm{vec}(\mathbf{Q}^T\mathbf{V}^T\mathbf{P}) \\ \Rightarrow \mathbf{U} &= \mathbf{P}^T\mathbf{V}\mathbf{Q} \end{aligned}$$

  Hence operation (1) becomes,

$$\mathbf{V} = \mathbf{S} + \mathbf{R}^n\mathbf{V}\mathbf{R}^n \quad \text{where } \mathbf{s} = \mathrm{vec}(\mathbf{S}^T)$$

  and operation (3) becomes,

$$\Delta\mathbf{V} = -\Delta t\mathbf{P}^T\mathbf{V}(\mathbf{Q} + \beta\mathbf{I}^n)$$

- The MGNC was implemented by starting β at -1.7 and incrementing it by 0.002 each time that,

$$\frac{\Delta\hat{\mathbf{v}}^T\mathbf{v}}{|\mathbf{v}^{zs}|} < 0.001$$