PROBLEM SOLVING WITH OPTIMIZATION NETWORKS

CUED/F-INFENG/TR 150

Andrew Howard Gee

Queens' College
Cambridge

July 1993

This dissertation is submitted for consideration towards the degree of Doctor of Philosophy at the University of Cambridge

Summary

Combinatorial optimization problems, which are characterized by a discrete set as opposed to a continuum of possible solutions, occur in many areas of engineering, science and management. Such problems have so far resisted efficient, exact solution, despite the attention of many capable researchers over the last few decades. It is not surprising, therefore, that most practical solution algorithms abandon the goal of finding the optimal solution, and instead attempt to find an approximate, useful solution in a reasonable amount of time. A recent approach makes use of highly interconnected networks of simple processing elements, which can be programmed to compute approximate solutions to a variety of difficult problems. When properly implemented in suitable parallel hardware, these *optimization networks* are capable of extremely rapid solutions rates, thereby lending themselves to real-time applications.

This thesis takes a detailed look at problem solving with optimization networks. Three important questions are identified concerning the applicability of optimization networks to general problems, the convergence properties of the networks, and the likely quality of the networks' solutions. These questions are subsequently answered using a combination of rigorous analysis and simple, illustrative examples. The investigation leads to a clearer understanding of the networks' capabilities and shortcomings, confirmed by extensive experiments. It is concluded that optimization networks are not as attractive as they might have previously seemed, since they can be successfully applied to only a limited number of problems exhibiting special, amenable properties.

Key words: Combinatorial optimization, neural networks, mean field annealing.

Acknowledgements

I am greatly indebted to my supervisor, Richard Prager, for initially allowing me the freedom to explore various research areas, and subsequently providing invaluable support as my work progressed. Members of the Speech, Vision and Robotics Group at the Cambridge University Department of Engineering have provided a stimulating and friendly environment to work in: special thanks must go to Patrick Gosling and Tony Robinson for maintaining a superb computing service, and to Sree Aiyer for both setting me on the right course and for numerous helpful discussions since then. I would like to thank the Science and Engineering Research Council of Great Britain, the Cambridge University Department of Engineering and Queens' College, Cambridge for their financial assistance. The last word must go to my parents, for without their constant encouragement and support I would not be where I am today.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration with any other party.

Contents

List of Figures

List of Tables

Notation

Matrix and vector notation

A_{ij}, $[\mathbf{A}]_{ij}$	Element in i^{th} row and j^{th} column of $\mathbf{A}$
$\mathbf{A}^T$	Transpose of matrix $\mathbf{A}$
x_i, $[\mathbf{x}]_i$	i^{th} element of vector $\mathbf{x}$
$\mathbf{x}^T$	Transpose of vector $\mathbf{x}$
$\mathbf{I}$	Identity matrix
$\mathbf{o}$	Column vector of ones
$\mathbf{O}$	Square matrix of ones
$\mathbf{R}$	Projection matrix which removes the $\mathbf{o}$ component of a vector
$\delta_{i,j}$	Kronecker impulse function

Network parameters

N	Size of optimization network's state vector
$\mathbf{u}$	Internal vector of optimization network
$\mathbf{v}$	State vector of optimization network
$\mathbf{v}_{\mathrm{o}}$	Initial value of $\mathbf{v}$
$\mathbf{V}$	Matrix equivalent of state vector $\mathbf{v}$
$\mathbf{T}$	Connection matrix of optimization network
$\mathbf{i}^{\mathrm{b}}$	Input bias vector of optimization network
E^{liap}	Extended Liapunov function of Hopfield network/MFA
E	Standard Liapunov function of optimization network
$g(u_i)$	Transfer function of optimization network
ϕ	Slope of transfer function at the initial position
η	Decay parameter of Hopfield network
γ	Hysteretic annealing parameter
T^p	Temperature annealing parameter
Δt	Time-step used in network simulations

Optimization parameters

E^{op}	Optimization objective function
$E^{\mathrm{op}\prime}$	Alternative optimization objective function
E^{cns}	Constraint penalty function
E^{alp}	Auxiliary linear problem objective function
E^{ann}	Annealing objective function
c_i	Weighting given to penalty function
$\mathbf{T}^{\mathrm{op}}$	Hessian of optimization objective function
$\mathbf{i}^{\mathrm{op}}$	Linear coefficient vector of optimization objective function
$\mathbf{T}^{\mathrm{val}}$	Valid subspace projection matrix
$\mathbf{s}$	Valid subspace offset vector
$\mathbf{A}^{\mathrm{eq}}$	Equality constraint matrix
$\mathbf{A}^{\mathrm{in}}$	Inequality constraint matrix
$\mathbf{b}^{\mathrm{eq}}$	Equality constraint vector
$\mathbf{b}^{\mathrm{in}}$	Inequality constraint vector

$\mathbf{w}$	Vector of slack variables
$\mathbf{v}^+$	Extended state vector for slack variable mappings
$\mathbf{T}^{\mathrm{op}+}$	Extended Hessian of objective function for slack variable mappings
$\mathbf{i}^{\mathrm{op}+}$	Extended linear coefficient vector of objective function for slack variable mappings
$\mathbf{A}^+$	Extended constraint matrix for slack variable mappings
$\mathbf{b}^+$	Extended constraint vector for slack variable mappings
P	Constraint polytope
P^+	Extended constraint polytope for slack variable mappings
Ω^n	Polytope with permutation matrices at its vertices
$\mathbf{v}^*$	Optimal solution to LP-relaxation problem

Valid subspace/eigenvector analysis notation

$\mathbf{v}^{\mathrm{val}}$	Component of $\mathbf{v}$ in valid subspace
$\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$	Initial value of $\mathbf{v}^{\mathrm{val}}$
$\mathbf{T}^{\mathrm{opr}}$	Hessian of optimization objective for $\mathbf{v}$ on the valid subspace
$\mathbf{i}^{\mathrm{opr}}$	Linear coefficient vector of optimization objective for $\mathbf{v}$ on the valid subspace
λ_i, λ_{kl}	Eigenvalue of $\mathbf{T}^{\mathrm{opr}}$
$\tilde{\lambda}_i$	Scaled eigenvalue of $\mathbf{T}^{\mathrm{opr}}$: $\tilde{\lambda}_i \equiv (\phi\lambda_i/T^p - \eta)$
$\mathcal{Z}$	Set of indices of zero eigenvalues of $\mathbf{T}^{\mathrm{opr}}$
$\mathbf{x}^i$, $\mathbf{x}^{kl}$	Eigenvector of $\mathbf{T}^{\mathrm{opr}}$
$\mathbf{x}^{\mathrm{max}}$	Eigenvector of $\mathbf{T}^{\mathrm{opr}}$ with largest positive eigenvalue
α_i, A_{kl}	Component of $\mathbf{v}^{\mathrm{val}}$ along $\mathbf{x}^i$, $\mathbf{x}^{kl}$
α_i^{o}, A_{kl}^{o}	Component of $\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$ along $\mathbf{x}^i$, $\mathbf{x}^{kl}$
β_i, B_{kl}	Component of $\mathbf{i}^{\mathrm{opr}}$ along $\mathbf{x}^i$, $\mathbf{x}^{kl}$
A'_{kl}	Squared magnitude of component of $\mathbf{v}^{\mathrm{val}}$ along $\mathbf{x}^{kl}$: $A'_{kl} \equiv A^2_{kl}$
Z_{kl}	Complex component of $\mathbf{v}^{\mathrm{val}}$ in degenerate eigenplane of $\mathbf{T}^{\mathrm{opr}}$
Z'_{kl}	Squared magnitude of component of $\mathbf{v}^{\mathrm{val}}$ in degenerate eigenplane: $Z'_{kl} \equiv \|Z_{kl}\|^2$
ζ_{kl}	Eigenvalue corresponding to degenerate eigenplane of $\mathbf{T}^{\mathrm{opr}}$
$\mathbf{Z}^{10}$	Prototypical decomposition of network's solution to 10-city TSP

Tabu search parameters

E_t^{op}	Time-varying objective function for tabu search dynamics
F_t	Time-varying penalty function which penalizes states already visited
$p(\mathbf{a},\mathbf{b})$	Proximity function for two vectors $\mathbf{a}$ and $\mathbf{b}$
$\mathbf{f}(\mathbf{a})$	Small displacement function such that $\mathbf{f}(\mathbf{a}) \approx \mathbf{a}$
α	Tabu search decay parameter
β	Tabu search weighting parameter
θ	Magnitude of tabu search diagonal term
$\mathbf{h}$	Tabu search linear term
ϵ	Tabu search offset parameter

Problem specific notation

n	Size of problem
$\mathbf{p}$	Permutation vector
$\mathbf{V}(\mathbf{p})$	Permutation matrix
$\mathbf{P}$	First Kronecker product submatrix of $\mathbf{T}^{\mathrm{op}}$

$\mathbf{Q}$	Second Kronecker product submatrix of $\mathbf{T}^{\mathrm{op}}$
$\mathbf{P}^{\mathrm{val}}$	Projected version of $\mathbf{P}$: $\mathbf{P}^{\mathrm{val}} \equiv \mathbf{RPR}$
$\mathbf{Q}^{\mathrm{val}}$	Projected version of $\mathbf{Q}$: $\mathbf{Q}^{\mathrm{val}} \equiv \mathbf{RQR}$
$\mathbf{C}$	Matrix of planar coordinates
$\mathcal{P}$	Unidentified pattern
$\mathcal{Q}$	Template pattern
$G_{\mathcal{P}}$	Graphical representation of pattern $\mathcal{P}$
$G_{\mathcal{Q}}$	Graphical representation of pattern $\mathcal{Q}$

Chapter 1

Introduction

1.1 Motivation

Combinatorial optimization problems rank among the most difficult known to the mathematical community, since many of them have proved practically impossible to solve exactly. A typical example is the travelling salesman problem, where we desire to find the shortest tour visiting a set of cities, starting and finishing at the same city. This is a *combinatorial* optimization problem since there are only a finite number of valid tours to consider, as opposed to a continuum of possible solutions. Hence the techniques developed over the years to solve continuous optimization problems are largely inapplicable. To this day, no way of finding the optimal tour has been discovered, short of calculating the length of every possible tour and choosing the shortest. Unfortunately, the number of tours to be considered rises exponentially with the number of cities. Even if the problem is limited to twenty cities, there are roughly 60 million billion possible tours to check. It would take a typical, modern computer workstation about 25 thousand years to evaluate this number of possibilities. It is not surprising, therefore, that much research has been directed towards discovering techniques which find good (though not necessarily optimal) solutions in reasonable amounts of time. There is considerable incentive to do so, since combinatorial optimization problems crop up in many areas of science, engineering and management: for example resource scheduling, communications routing, visual stereo correspondence and invariant pattern recognition can all be formulated as combinatorial optimization problems.

1.2 Optimization networks

While there already exist many effective algorithms for approximately solving combinatorial optimization problems (see Appendix A), this thesis is concerned with just one family of techniques, which we term *optimization networks*. The origins of optimization networks can be traced back to Hopfield and Tank's pioneering work in 1985 [71], though it would perhaps be more appropriate to start with a review of general Artificial Neural Networks.

It has long been noted that while conventional computers are very good at performing numerically intensive tasks such as complex arithmetic, they are less well adapted to the tasks which humans find straightforward, like speech and vision recognition. For this reason a new type of computer, modelled loosely on the architecture of the human brain, was proposed. These Artificial Neural Networks (ANNs) comprise a large number of simple processing elements (corresponding to single neurons in the human brain) connected together in massive, parallel arrays. An ANN can be trained to perform tasks such as speech and vision recognition and, like the human brain, has the ability to learn from experience. Moreover, with the highly parallel structure of the ANN properly exploited in electronic hardware, extremely high information processing speeds are possible, giving ANNs a huge advantage over conventional computers.

One type of ANN is the Hopfield network [69], originally proposed as a form of content addressable memory (a device which allows stored patterns to be recalled by presentation of noisy, corrupted versions of the same patterns). The Hopfield network is an example of a *feedback* neural network, where the outputs of the individual processing units are fed back to the inputs via a dense array of interconnections, producing a nonlinear, continuous dynamic system. In 1985, Hopfield and Tank demonstrated how the Hopfield network could be applied to the solution of the travelling salesman problem [71]. Their work was not conclusive, in that the network frequently failed to find valid solutions, let alone high quality ones, though a new methodology had clearly been defined. There then followed a considerable research effort, both to improve the performance of the Hopfield network on the travelling salesman problem, and to find ways of applying the network to more useful optimization problems. With the potential for very high speed implementations, researchers soon realized that it might be possible to find good solutions to difficult problems in a matter of milliseconds. This was a goal well worth aiming for.

At about the same time as the emergence of the Hopfield network, physicists were proposing a revolutionary new optimization technique called *simulated annealing*. Inspired by phenomena in statistical physics, simulated annealing provides a method for finding very good solutions to the most general combinatorial optimization problems, though the algorithm typically takes a long time to run. To overcome this time problem, *mean field annealing* (MFA)[1] was proposed as an approximation to simulated annealing, trading solution quality for improved speed of execution. It soon became clear that the MFA algorithm resembles a discrete-time simulation of the Hopfield network, thus establishing a firm link between the two optimization techniques. In this thesis we will study MFA, Hopfield networks and other related methods under the umbrella term of *optimization networks*, using a common, unified analysis for them all.

1.3 Justification

It is always important to justify any proposed research before embarking upon it. There already exist many good algorithms for solving a wide variety of combinatorial optimization problems: these are surveyed in Appendix A. Can optimization networks find a niche in this marketplace?

According to the literature, optimization networks have several features to recommend them[2]:

- Apparent applicability to a wide variety of problems.
- Easily reprogrammed for new problems.

[1]MFA is also sometimes referred to as *deterministic annealing*.

[2]It transpires that some of these points will be challenged in this thesis, though this of course does not affect any justification for the research itself.

- Fast, parallel implementation in analogue or digital hardware.

...and a serious disadvantage:

- When using optimization networks, it is often necessary to adopt inefficient problem representations: for example, a network of n^2 processing elements is required to solve an n-city travelling salesman problem. This leads to poor time complexity when simulating the network on a standard computer, or poor space complexity in parallel hardware implementations.

Other optimization techniques also exhibit general applicability and ease of adaptation to new problems: for example cutting plane techniques, branch-and-bound, simulated annealing and genetic algorithms can all be readily applied to a wide variety of problems. Where optimization networks have the apparent edge is in their very fast, parallel implementations: the potential ability to obtain good solutions to large problems in a matter of milliseconds is unique to optimization networks. Furthermore, we can envisage real-time applications which would greatly benefit from this, such as solving stereo correspondence problems in computer vision, graph labelling problems for invariant pattern recognition and routing problems in communications systems. This is surely enough to justify further investigation of optimization networks.

But how does the single disadvantage affect this claim? The poor time complexity in simulation is certainly a serious shortcoming: for example, using a conventional digital computer, it is laborious to simulate an optimization network solving a travelling salesman problem in more than about 100 cities. In contrast, an alternative, state of the art algorithm can solve a 2392-city problem to within 10% of optimality in 16 minutes on a modern computer workstation [46]. Thus, optimization networks are hardly competitive in simulation, relying on hardware implementations for their appeal. In hardware there is a corresponding problem with space complexity, though given the ever improving compactness of VLSI technology, this should not be debilitating for problems of moderate size. Nevertheless, it is clearly far-fetched to envisage an electrical circuit solving a 2000-city travelling salesman problem, since this would require a network of 4 million processing elements. Optimization networks are therefore restricted in their appeal to problems of moderate size requiring extremely rapid solution. Since at least several useful problems fall into this category (as listed above), a further investigation of optimization networks would appear to be justified.

1.4 Themes and contributions

Before attempting any original work, we have to identify the remaining questions concerning the use of optimization networks. Since most of the published work (which is surveyed in Appendix A) has focussed on the networks' application to a few specific problems, with varying degrees of success, it would seem sensible to take a more general look at the field. Specifically, there are three obvious questions:

- Is there a simple way to program an optimization network for an arbitrary problem, so that the network will never find a solution violating any of the problem's hard constraints?

- Given this method, is it always possible to force the network to converge to an interpretable, valid solution?

- If so, will the quality of this valid solution be high enough to compete with the other optimization techniques?

These are the key questions addressed in this thesis. We shall attempt to answer them using a rigorous, analytical approach, though without recourse to too much heavy mathematics: where possible, we will use simple, 2-dimensional examples to illustrate our arguments. Furthermore, we shall avoid experimenting with large scale problems where possible: shortcomings of optimization networks can be easily confirmed with small problems, while there is appropriate evidence in the literature for the networks' success on certain large problems, which it would be wasteful to reproduce. The broad outline of this thesis is as follows:

Chapter 2 presents a review of optimization networks, including their implementation in analogue hardware and their simulation on digital machines.

Chapter 3 answers the first of the three key questions. We present a *mapping*, by which an optimization network can be easily programmed to solve any 0-1 quadratic programming problem (it is standard practice to express combinatorial optimization problems in this form).

Chapter 4 addresses the second of these questions. We show that convergence to a valid solution can only be guaranteed for a small class of problems whose constraints define an *integral polytope*. Fortunately, many useful problems belong to this class, though many equally useful problems do not.

Chapters 5 and 6 tackle the final question for those problems over integral polytopes. We show that the quality of the networks' solutions is highly variable, depending largely on certain properties of the problem itself.

Chapter 7 presents some new ideas on how optimization networks can be sensibly applied to the solution of problems over non-integral polytopes. This requires modifying the networks so that they *search* for valid solutions, instead of attempting to find a valid solution in a single attempt, which we show to be impossible.

Chapter 8 closes the thesis by presenting our main conclusions.

Appendix A presents a literature survey covering the solution of combinatorial optimization problems, and the theory and application of optimization networks.

Appendix B contains derivations, proofs and experimental details referred to in the text.

Appendix C presents a brief review of Kronecker products, which are used at several points in the thesis.

Figures and tables can be found at the end of each chapter.

Chapter 2

Optimization Networks

The starting point for this chapter is the premise that many combinatorial optimization problems can be posed as the minimization of a single quadratic objective function over a set of 0-1 variables. This suggests a new family of heuristic, approximate solution techniques. Referred to collectively as *optimization networks*, these techniques all perform some sort of *continuous* descent on the objective function within the bounds of the unit hypercube. Ideally, the descent converges to a hypercube corner, which, being a 0-1 point, can be interpreted as a solution to the combinatorial problem. Optimization networks appear to offer a good compromise between solution quality and speed of operation: their inherent parallelism leads to hardware implementations which could solve large problems in a matter of milliseconds.

In this chapter we introduce three optimization networks. In Section 2.2 we review the well known Hopfield network, first proposed as a means of approximately solving the travelling salesman problem in 1985 [71]. The steepest descent network, described in Section 2.3, is a stripped down version of the Hopfield network with dynamics which are particularly easy to visualize. In Section 2.4 we review the mean field annealing algorithm, which has its roots in statistical physics but is shown to be very closely related to the other optimization networks. Finally, in Section 2.5 we demonstrate how all the networks can be implemented in high-speed analogue or digital hardware.

2.1 Quadratic objective functions for combinatorial optimization

Many combinatorial optimization problems can be posed as the minimization of a single quadratic objective function over a set of 0-1 variables. The manner in which problems are expressed in this form is not trivial and will be the subject of Chapter 3. In the meantime, however, let us assume that we have succeeded in expressing the problem in the following form:

$$\begin{array}{ll} \text{minimize} & E(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^T\mathbf{T}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{b}} \\ \text{where} & v_i \in \{0,1\} \end{array} \tag{2.1}$$

The problem of finding the 0-1 point which minimizes E is typically $\mathcal{NP}$-complete. This means that if we really want to find the best solution, we have little choice but to exhaustively search through every 0-1 point. Unfortunately, the number of 0-1 points grows exponentially with the size of the problem, and so exhaustive search is infeasible for all but the smallest of problems. A far more sensible approach to $\mathcal{NP}$-complete problems is to attempt to find a reasonably good solution in an acceptably short amount of time.

In the rest of this chapter we consider a family of solution techniques referred to collectively as *optimization networks*. All optimization networks can be viewed as feedback systems with $\mathbf{v}$ as a state vector. The networks' dynamics guide $\mathbf{v}$ through some sort of bounded descent on the objective function E (by *bounded* we mean that the descent is limited to the interior of the unit hypercube). Throughout the descent, the elements of $\mathbf{v}$ are allowed to take *any* value in the range 0-1, thus extending the search space from the vertices to the whole interior of the unit hypercube. Typically E will have many local minima in which $\mathbf{v}$ could easily become trapped, and so it is common to employ some sort of *annealing* mechanism to free $\mathbf{v}$ from such local minima and guide $\mathbf{v}$ towards a hypercube corner: we shall discuss annealing procedures in Chapters 4, 5 and 6. When $\mathbf{v}$ is at a hypercube corner all its elements are either 0 or 1, and so we have an interpretable solution to the combinatorial problem. Since the solution point was arrived at through some sort of descent, we might expect this point to represent a relatively good solution, though this will depend very much on the nature of the objective function E and its relationship to underlying features of the optimization problem: such issues will be comprehensibly addressed in Chapters 5 and 6. In the meantime, however, let us consider optimization networks without worrying about their likely efficacy, since they would appear to offer a reasonable compromise between solution quality and speed of execution, especially given the potential for fast hardware implementations.

2.2 The Hopfield network

A schematic diagram of the continuous Hopfield network[1] [70] is shown in Figure 2.1. The network comprises a set of simple processing elements, or *neurons*, connected together to form a dense, parallel array. Neuron i has input u_i, output v_i, and is connected to neuron j with a weight T_{ij}. Associated with each neuron is also an input bias term i_i^b. The dynamics of the network are governed by the following equations:

$$\dot{\mathbf{u}} = -\eta\mathbf{u} + \mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}} \tag{2.2}$$

$$v_i = g(u_i) \tag{2.3}$$

In equation (2.3), v_i is a continuous variable in the interval 0 to 1, and $g()$ is a monotonically increasing function which constrains v_i to this interval, usually a shifted hyperbolic tangent of the form

$$g(u_i) = \frac{1}{1 + \exp(-u_i/T^p)} \tag{2.4}$$

If $\mathbf{T}$ is symmetric, then the network has a Liapunov function [70]

$$E^{\mathrm{liap}}(\mathbf{v}) = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{b}} + \eta\sum_i \int_0^{v_i} g^{-1}(V)\, dV \tag{2.5}$$

If transfer functions of the form (2.4) are employed, then

$$g^{-1}(V) = T^p\left(\ln(V) - \ln(1 - V)\right) \tag{2.6}$$

[1] The Hopfield network is in fact a special case of the *additive model* developed by Grossberg in the 1960's (see [63] for a historical survey).

and the last term in the Liapunov function gives a measure of the system entropy, which can sometimes be exploited in specialist optimization applications [75, 76, 102]. However, for the vast majority of applications this term is a nuisance and is usually suppressed, either by setting $\eta = 0$[2], or by ensuring that the transfer functions (2.4) have sufficiently high gain (ie. low T^p), in which case the entropy term becomes negligible for $\mathbf{v}$ within the unit hypercube [70]. The Liapunov function then becomes

$$E(\mathbf{v}) = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{b}} \tag{2.7}$$

Throughout the rest of this thesis we shall refer predominantly to the shortened Liapunov function (2.7), since this is more readily applicable to the solution of combinatorial optimization problems than the full Liapunov function (2.5).

Since the network's dynamics guide $\mathbf{v}$ in a manner which never allows E to increase, $\mathbf{v}$ will converge to some minimal point of E within the unit hypercube. This process is illustrated in Figures 2.3 and 2.4, which show full $\mathbf{u}$ and $\mathbf{v}$ trajectories for a Hopfield network operating with a Liapunov function

$$\begin{aligned} E(\mathbf{v}) &= -\tfrac{1}{2}\mathbf{v}^T\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\mathbf{v} - \mathbf{v}^T\begin{bmatrix} -1 \\ 2 \end{bmatrix} \\ &= -\tfrac{1}{2}(v_1^2 + v_2^2) + v_1v_2 + v_1 - 2v_2 \end{aligned} \tag{2.8}$$

A contour plot of E is given in Figure 2.2. For the simulation in Figure 2.3 the Liapunov function was achieved with low gain transfer functions (2.4) by setting $\eta = 0$, though, as the traces indicate, this means that the $\mathbf{u}$ variables are unbounded and will continue to grow in magnitude indefinitely: this is clearly not practical for anything but computer simulation. For the simulation in Figure 2.4 the Liapunov function was achieved using a nonzero η and high gain transfer functions (2.4) with $T^p = 0.01$. In this case the $\mathbf{u}$ variables converge to finite values, while the $\mathbf{v}$ variables follow an almost identical trajectory to that achieved with $\eta = 0$, though in considerably less time. This is how any hardware implementation of the Hopfield network would operate.

The Hopfield network was originally proposed as a form of content addressable memory (CAM). The connection weights and input biases were programmed so that the minima of the network's Liapunov function corresponded to the items to be stored [69, 70]. If the network was initialized with $\mathbf{v}$ lying somewhere within the unit hypercube, then it was expected that $\mathbf{v}$ would converge to the nearest local minimum, thus recalling the stored item most resembling the starting point. Even though Hopfield networks offered the potential for very fast, real-time CAMs [66], it is now generally accepted that they have low storage capacity [145] which cannot be improved without creating spurious local minima [9, 136]. In particular, it has been demonstrated that a parallel architecture implementing a simple Hamming classification scheme easily outperforms the Hopfield network as a CAM [136].

In [71, 72], the Hopfield network was subsequently proposed as a means of approximately solving combinatorial optimization problems which can somehow be expressed as the minimization of a quadratic objective function over a set of 0-1 variables, as in equation (2.1). The idea is that the network's Liapunov function (2.7) is associated with the problem's objective function (2.1) by setting the connection weights and input biases appropriately. The network is then allowed to converge to a stable state, usually with some

[2] This can be achieved in simulation (although the u_i variables then become unbounded), but not in analogue circuit implementations of the network — see Section 2.5.

sort of annealing mechanism which frees $\mathbf{v}$ from local minima of E and drives $\mathbf{v}$ towards a hypercube corner. The corner thus reached, being a 0-1 point, is interpreted as a solution to the problem. The original work [71, 72] focussed on the travelling salesman problem, showing how the network's output can be used to represent a solution to that problem, and how the interconnection weights and input biases can be programmed appropriately. The particular attraction of the network lies in the existence of an analogue electrical circuit with dynamics corresponding to (2.2) and (2.3), which we shall describe in Section 2.5. Using such circuits, it should be possible to find good solutions to large problems in a matter of milliseconds.

The Liapunov function (2.7) is valid only if $\mathbf{T}$ is symmetric. However, it is also possible to use the network for problems where $\mathbf{T}$ is not symmetric, by noting that

$$E = \tfrac{1}{2}(E + E^T) = -\tfrac{1}{2}\mathbf{v}^T \left[\tfrac{1}{2}(\mathbf{T} + \mathbf{T}^T)\right] \mathbf{v} - \mathbf{v}^T \mathbf{i}^{\mathrm{b}} \tag{2.9}$$

So we simply replace $\mathbf{T}$ with $\frac{1}{2}(\mathbf{T} + \mathbf{T}^T)$, which is symmetric: in so doing we do not change E.

2.3 The steepest descent network

Also of interest are steepest descent dynamics, previously studied in [5, 112]:

$$\dot{v}_i = \begin{cases} 0 & \text{if } \left[\mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}}\right]_i < 0 \text{ and } v_i = 0 \\ 0 & \text{if } \left[\mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}}\right]_i > 0 \text{ and } v_i = 1 \\ \left[\mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}}\right]_i & \text{otherwise} \end{cases} \tag{2.10}$$

Since $\dot{\mathbf{v}}^T \nabla E \leq 0$, the dynamics (2.10) share the same Liapunov function (2.7) as the Hopfield network, and the $\mathbf{v}$ variables are also limited to the range $0 \leq v_i \leq 1$. Thus (2.10) provides an alternative method for performing a descent on E within the unit hypercube. The steepest descent dynamics are particularly easy to visualize: $\mathbf{v}$ follows the path of steepest descent on E until it reaches one of the hypercube faces. Here, it continues along the path of steepest *available* descent, without leaving the interior of the hypercube. Equilibrium is attained when $\mathbf{v}$ reaches a point where it can go no further without leaving the hypercube or climbing uphill on E. Figure 2.5 shows the steepest descent dynamics operating on the Liapunov function (2.8). As expected, the system converges to the same point as the Hopfield network did. The steepest descent dynamics can also be implemented in analogue hardware (see Section 2.5) and have the additional attraction of an efficient simulation on digital computers, as we shall see in Section 3.4.

2.4 Mean field annealing

Since its conception in 1983, simulated annealing [86] has enjoyed considerable popularity as a general purpose optimization tool. The technique, which is described in more detail in Appendix A, embodies a *stochastic* search process, and often takes an unacceptably long amount of time to find a good solution. In contrast, the mean field annealing (MFA) algorithm operates on *average* statistics of the annealing process to provide a *deterministic* approximation to simulated annealing: the result is improved execution speed at the expense of solution quality. Though not strictly a continuous descent technique, MFA is closely related to the Hopfield network and therefore warrants discussion here.

For a full derivation of the mean field annealing algorithm from its roots in statistical physics see [5, 118]; here we shall take as our starting point the *saddle point equations*:

$$v_i = \frac{1}{1+\exp(-u_i/T^p)} \tag{2.11}$$

$$u_i = \left(\sum_j T_{ij} v_j + i_i^b\right) \tag{2.12}$$

The MFA algorithm involves solving (2.11) and (2.12) at a series of progressively lower temperatures T^p: this process is known as *temperature annealing*. The solutions to (2.11) and (2.12) correspond to stable states of the Hopfield network with transfer functions of the form (2.4) and $\eta = 1$, so both techniques appear to be seeking the same solution points. Hence, in the limit of low temperature T^p, when the Hopfield network has a Liapunov function E (2.7), it is apparent that the MFA algorithm also seeks points which minimize E within the unit hypercube. The reason for tracing the saddle point solutions through a series of progressively lower temperatures, instead of simply solving the equations once at a low temperature, will be explained in Chapter 5 when we come to consider annealing procedures in more detail.

Equations (2.11) and (2.12) can be solved at each temperature using Hopfield dynamics (2.2)–(2.3), in which case the overall system is best described as a *temperature annealed* Hopfield network. However, more common is to solve the equations using an iterative replacement procedure, which we shall henceforth refer to as the MFA algorithm:

$$\mathbf{u}|_{t+1} = \mathbf{T}\mathbf{v}|_t + \mathbf{i}^{\mathrm{b}} \tag{2.13}$$

$$v_i|_{t+1} = \frac{1}{1+\exp(-\left. u_i\right|_{t+1}/T^p)} \tag{2.14}$$

The update rules (2.13) and (2.14) define a discrete algorithm which has no convergence guarantee and can in fact increase E at any iteration: this should be contrasted with the continuous descent networks which are guaranteed to converge to some minimal point of E within the unit hypercube. We shall investigate convergence of the MFA algorithm more fully in Chapter 5: at this point we simply note that while there is no convergence guarantee, in practice MFA can be made to work for many optimization problems and exhibits very rapid convergence to solutions of (2.11) and (2.12).

The relationship between MFA and the Hopfield network can be clarified by noting that if we choose to simulate the Hopfield network using an Euler approximation with a time-step Δt, we obtain from (2.2)

$$\mathbf{u}|_{t+\Delta t} = \mathbf{u}|_t + \Delta t(-\eta\mathbf{u}|_t + \mathbf{T}\mathbf{v}|_t + \mathbf{i}^{\mathrm{b}}) \tag{2.15}$$

Furthermore, if we choose $\Delta t = 1$ and $\eta = 1$, then the Euler approximation becomes the MFA algorithm update rule (2.13) [5, 117]. Throughout this thesis we shall strive to consider MFA and the continuous descent networks from a common standpoint, and not pay too much attention to the links between MFA and simulated annealing. Viewing MFA in this way also allows us to assess the detrimental effect of the approximations which lie between MFA and simulated annealing, since the latter is guaranteed to find the optimal solution to any combinatorial problem in the (impractical) limit of an infinitely slow cooling schedule.

The action of the MFA algorithm on the Liapunov function (2.8) is illustrated in Figure 2.6. Here we see the results of solving the saddle point equations (2.11) and (2.12) at a series of progressively lower temperatures until $v_i \in \{0, 1\}$ when $T^p \approx 0.1$. Rapid convergence of the update rules (2.13) and (2.14) was always observed with this particular objective E, and only a few iterations were required at each temperature to find a stable solution point. As expected, both $\mathbf{u}$ and $\mathbf{v}$ converge to the same points found by a Hopfield network running with $\eta = 1.0$ (see Figure 2.4).

2.5 Implementation of optimization networks

The main appeal of optimization networks lies in their potential implementation in analogue hardware, making possible the extremely rapid solution of large problems. In this section we consider both hardware and software implementations of the networks introduced in this chapter.

2.5.1 Analogue circuit implementations

An analogue circuit implementation of the Hopfield network would relax to a stable state within a few time constants of the circuit components, thus raising the possibility of obtaining good solutions to very large problems within a matter of milliseconds. Figure 2.7 shows a representative part of an analogue circuit with dynamics [70]

$$C_i \dot{u}_i = \sum_j T_{ij} v_j + i_i^b - \frac{u_i}{r_i} \tag{2.16}$$

where

$$T_{ij} = \frac{1}{R_{ij}^{\text{ex}}} - \frac{1}{R_{ij}^{\text{in}}} \quad \text{and} \quad \frac{1}{r_i} = \sum_j \left(\frac{1}{R_{ij}^{\text{ex}}} + \frac{1}{R_{ij}^{\text{in}}} \right) + \frac{1}{R_i} \tag{2.17}$$

The provision of inverted amplifier outputs allows the interconnections T_{ij} to take either positive (excitatory) or negative (inhibitory) sign. The amplifiers all have identical, monotonically increasing input/output transfer functions, which limit their outputs v_i to a certain range. If we assume that $C_i = C$ and $r_i = r$ for all i, then we can write the dynamics in vector form as follows:

$$\dot{\mathbf{u}} = \frac{1}{C} \left(-\frac{\mathbf{u}}{r} + \mathbf{T}\mathbf{v} + \mathbf{i}^{\text{b}} \right) \tag{2.18}$$

Thus, by comparison with equation (2.2), it is apparent that the circuit has dynamics equivalent to those of a Hopfield network with $\eta = 1/r$, and a speed of operation governed by the values of the circuit's capacitors and resistors. If the amplifiers have sufficiently high gain, then the circuit will have a Liapunov function of the form (2.7) and can therefore be used for combinatorial optimization. Since it is not possible to design the circuit to give $\eta = 0$, the only way to achieve the Liapunov function (2.7) is to use high gain amplifiers. Small circuits of this design have been used to solve optimization problems in analogue-to-digital conversion and linear programming [135].

Steepest descent networks can also be implemented in analogue hardware — see Figure 2.8. The circuit is very similar to that of the Hopfield network, except that the amplifiers now have a constant gain of one (and are effectively just current sourcing buffers) and a pair of diodes limits each of the v_i voltages to the range $V_L \leq v_i \leq V_H$. If we define the current input to each capacitor as

$$I_i = \sum_j T_{ij} v_j + i_i^b - \frac{v_i}{r_i} \tag{2.19}$$

then the dynamics of the circuit can be expressed as

$$C_i \dot{v}_i = \begin{cases} 0 & \text{if } I_i < 0 \text{ and } v_i = V_L \\ 0 & \text{if } I_i > 0 \text{ and } v_i = V_H \\ I_i & \text{otherwise} \end{cases} \tag{2.20}$$

where T_{ij} and r_i are defined in equation (2.17). Making the earlier assumption that $C_i = C$ and $r_i = r$ for all i, the dynamics reduce to

$$\dot{v}_i = \begin{cases} 0 & \text{if } I_i < 0 \text{ and } v_i = V_L \\ 0 & \text{if } I_i > 0 \text{ and } v_i = V_H \\ \frac{1}{C} I_i & \text{otherwise} \end{cases} \tag{2.21}$$

Setting $V_L = 0$ and $V_H = 1$, and adjusting T_{ii} to cancel the v_i/r_i term in (2.19), equations (2.21) and (2.19) are equivalent to the steepest descent dynamics (2.10).

While circuits like the ones in Figures 2.7 and 2.8 are interesting from a theoretical and illustrative standpoint, their applicability to practical implementations of large networks is doubtful. The analysis of the circuits assumed that all the components were ideal, and in particular that the amplifiers had no propagation delay. When real components are considered, finding effective circuits is nowhere near so simple, though still possible [128]. Real components aside, there remain several serious problems to overcome: for instance, programmable resistors are not easy to implement in VLSI. Thus the move has been away from operational amplifier based circuits and towards circuits built around MOSFET analogue multipliers, which require the weights to be stored as voltages. Weight storage can then be achieved using capacitors, which typically hold their charge long enough to give a network time to converge [41]. Alternatively, optimization networks can be implemented in pulse-stream VLSI [65, 110], using pulse-width modulation to carry out the required multiplications.

2.5.2 Network simulation using digital computers

While the MFA algorithm is naturally implemented on a digital computer, the Hopfield and steepest descent networks are described as continuous systems which map most conveniently onto analogue electrical circuits. However, until suitable hardware becomes generally available, optimization networks are bound to be studied in simulation on conventional, digital computers. The simulation of continuous systems on digital computers is a tricky process, the precise details of which can have significant bearings on any subsequent experimental results.

The range of simulation techniques available is well documented: see, for example, [44, 64, 90]. The most straightforward way to simulate a Hopfield or steepest descent network would be to use a standard Euler approximation with a finite time-step Δt. However, greater accuracy is possible using higher order techniques: for example, the fourth order Runge-Kutta approximation has been popular with researchers.

It remains only to point out that, whatever the simulation technique employed, most of the computational effort is expended on large scale matrix multiplications. These require a number of multiply and accumulate operations, which can be performed very rapidly using specialized digital signal processing hardware. In addition, the separate multiply and accumulate operations associated with a single matrix multiplication can be performed in parallel, offering further significant speed-ups. This applies also to the mean field annealing algorithm, for which there was no analogue circuit implementation. A parallel, digital CMOS circuit operating along these lines was used to simulate a 64-neuron Hopfield-type network in [79].

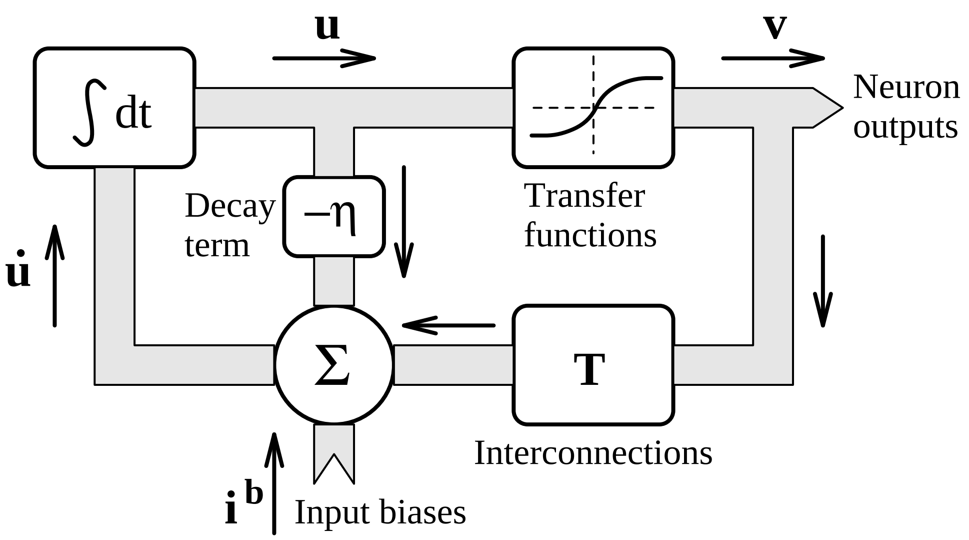


Figure 2.1: Schematic diagram of the continuous Hopfield network.

The Hopfield network comprises a set of 'neurons'. Neuron i has input u_i and output v_i, related by a monotonically increasing transfer function $v_i = g(u_i)$. The network's dynamics are governed by the first order differential equation $\dot{\mathbf{u}} = -\eta\mathbf{u} + \mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}}$. The $\mathbf{T}$ matrix defines a set of interconnections between the neurons, while the $\mathbf{i}^{\mathrm{b}}$ vector corresponds to a set of individual input biases to each neuron. The η term introduces an element of decay into the network's dynamics, which can be used to ensure that the $\mathbf{u}$ variables remain bounded in magnitude.

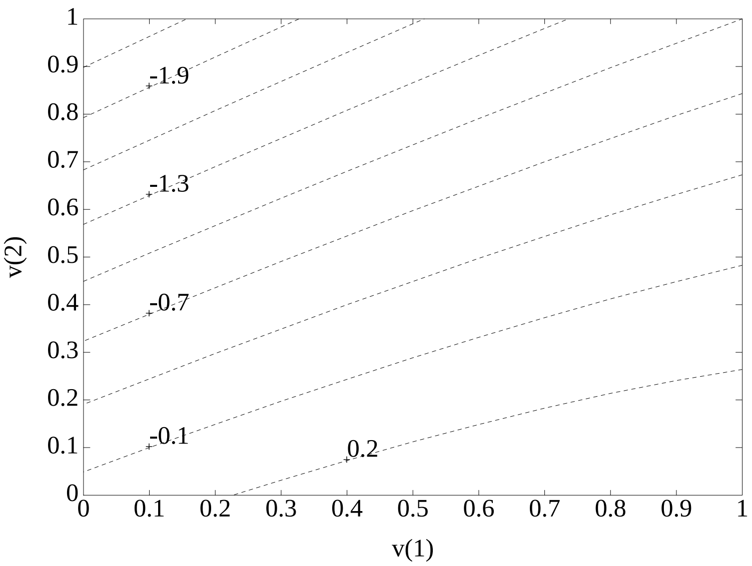


Figure 2.2: Contours of a simple Liapunov function.

The contours are for the function $E(\mathbf{v}) = -\frac{1}{2}(v_1^2 + v_2^2) + v_1 v_2 + v_1 - 2v_2$. *This function is used to compare and contrast the dynamics of the various optimization networks.*

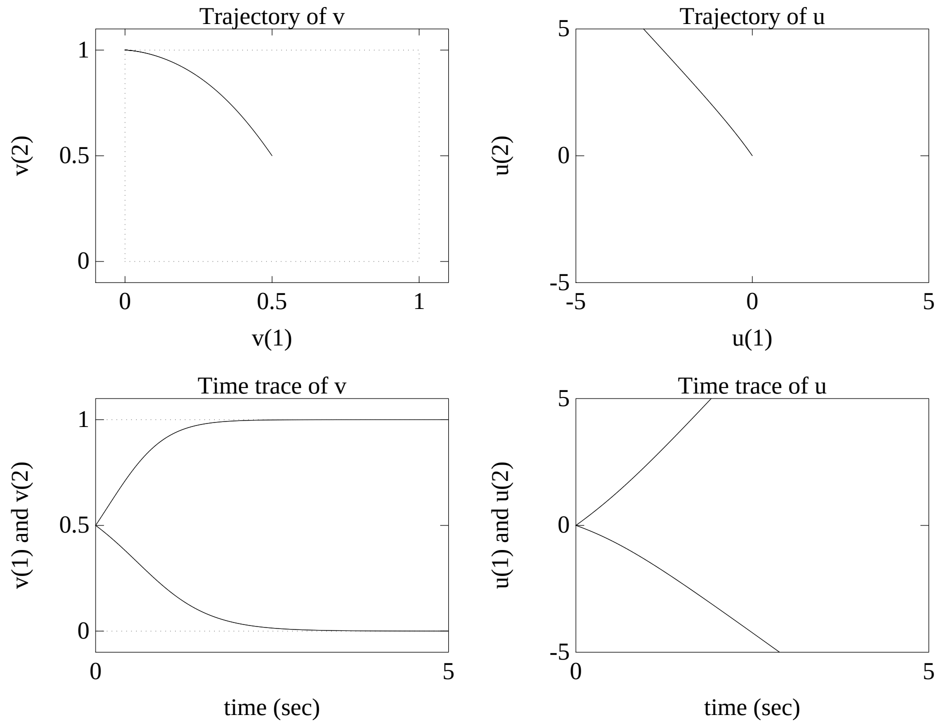


Figure 2.3: Trajectories from Hopfield network dynamics.

The traces are for a Hopfield network running with a Liapunov function E (2.8) and parameters $\eta = 0$ *and* $T^p = 1.0$. *Since there is no decay term, the* **u** *variables are unbounded. The* **v** *variables converge to the point (0,1), which is the minimum of the Liapunov function within the unit square.*

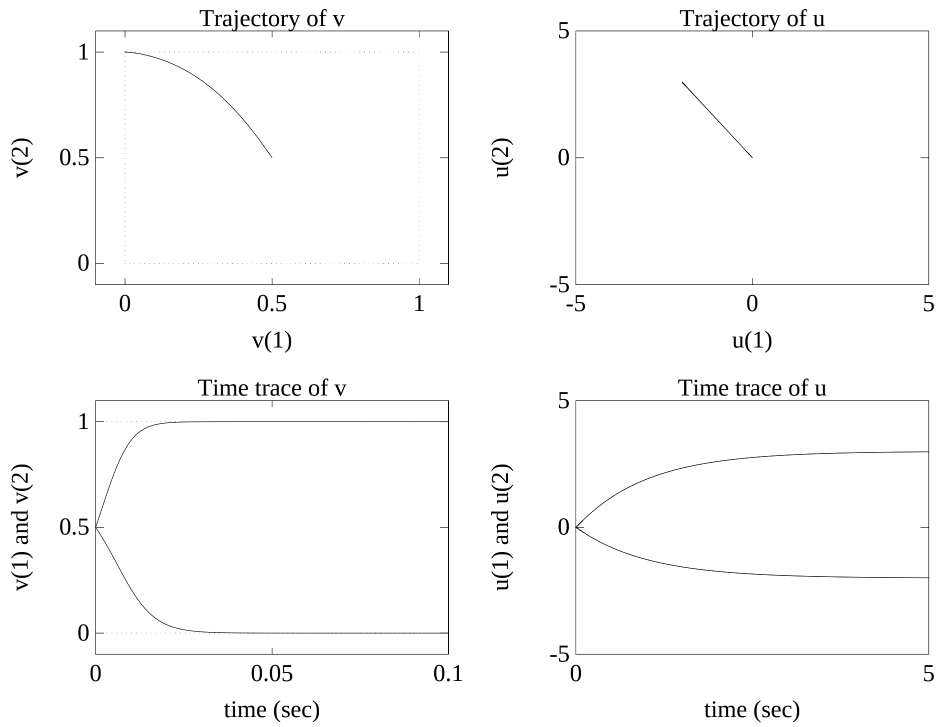


Figure 2.4: Trajectories from alternative Hopfield network dynamics.

The traces are for a Hopfield network running with parameters $\eta = 1$ *and* $T^p = 0.01$*. The high gain transfer functions ensure that the network still has a Liapunov function E (2.8), despite the presence of a nonzero decay term. In this case the* **u** *variables are bounded, while the* **v** *variables converge, as before, to the minimum of the Liapunov function within the unit square. Any hardware implementation of a Hopfield network would operate in this mode. Note the time scale on the* **v** *time trace.*

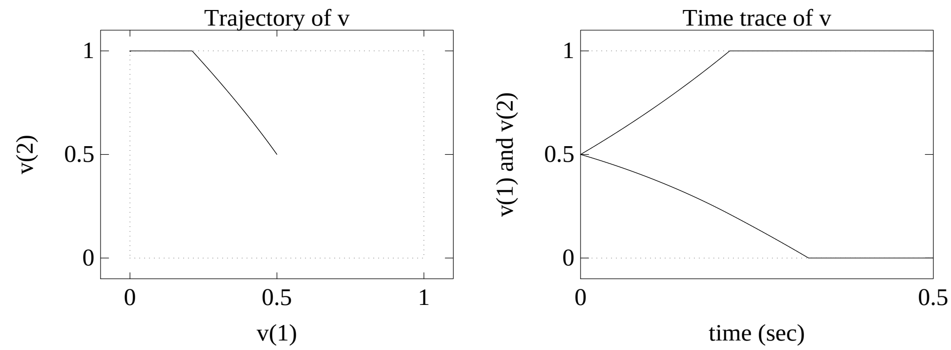


Figure 2.5: Trajectories from steepest descent dynamics.

For steepest descent dynamics there are no **u** *variables.* **v** *simply follows a path of steepest descent on E, until a hypercube face (in this simple example an edge of the unit square) is reached. At this point a path of steepest* available *descent is followed within the unit hypercube. As before,* **v** *converges to the minimum of the Liapunov function within the unit square.*

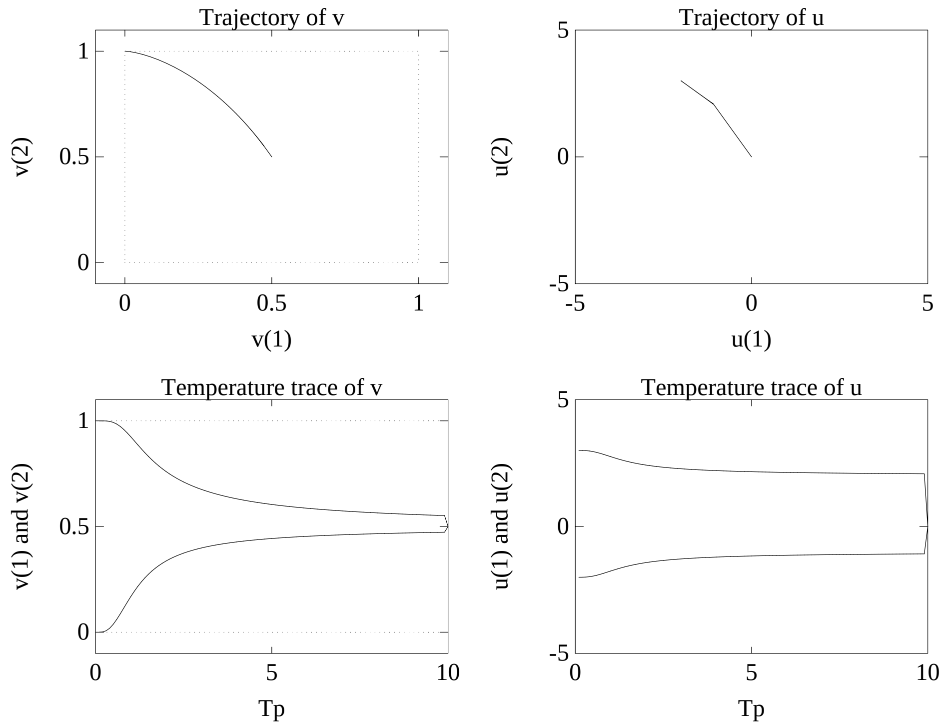


Figure 2.6: Trajectories from the MFA algorithm.

The MFA algorithm requires the solution of the saddle point equations at a series of progressively lower temperatures. The traces show how the **u** *and* **v** *solutions evolve as the temperature is lowered from 10 to 0.1 in 99 equal intervals of 0.1. The solutions at each temperature were obtained using the iterative update rule, which is equivalent to an Euler approximation of the Hopfield network dynamics with* $\eta = 1$ *and time-step* $\Delta t = 1$*. The solutions could equally have been obtained using continuous Hopfield dynamics, in which case the system is best described as a* temperature annealed *Hopfield network.*

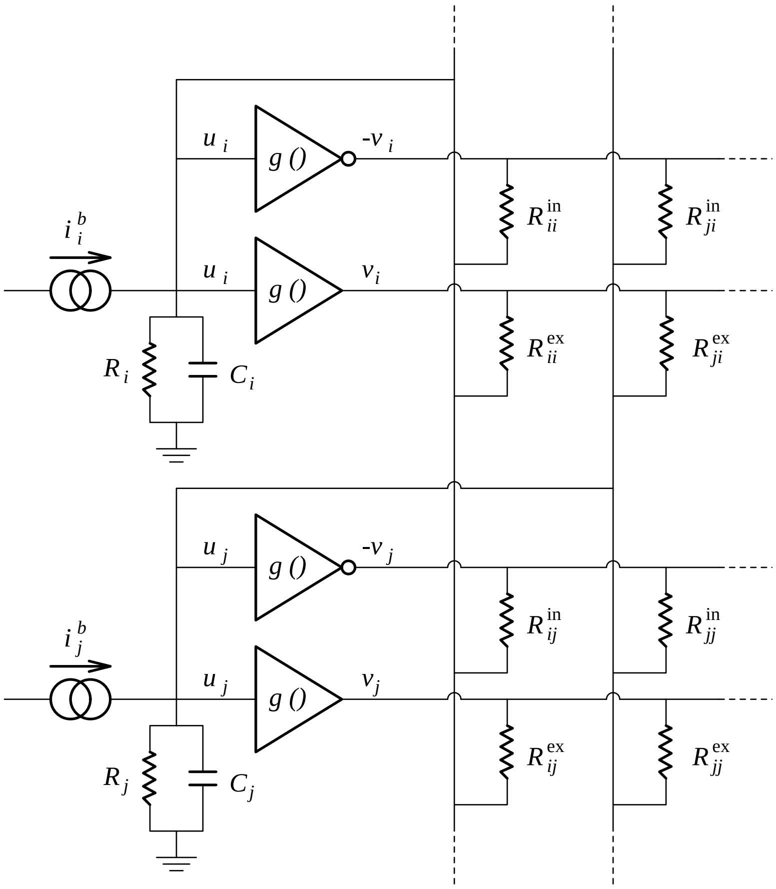


Figure 2.7: Analogue circuit implementation of the continuous Hopfield network.

With appropriate settings of the component values, this circuit exhibits the continuous Hopfield dynamics with nonzero decay parameter η. *However, existing practical hardware implementations of Hopfield networks tend to differ significantly from this simple circuit, since programmable resistors are difficult to incorporate directly in VLSI. More convenient is to build the circuits around MOSFET analogue multipliers, in which case the interconnection weights can be held as charges on capacitors.*

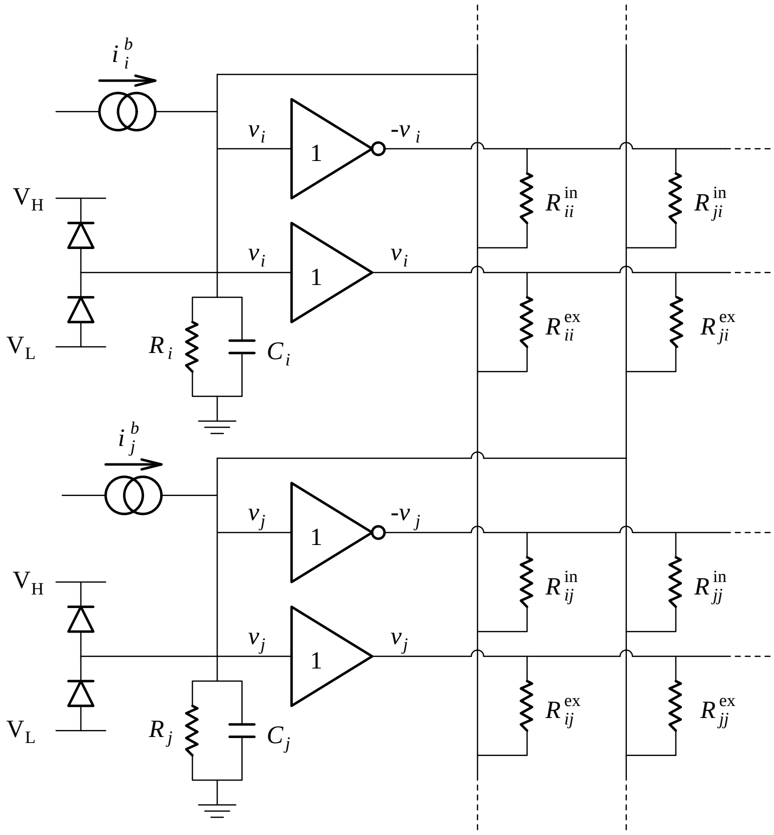


Figure 2.8: Analogue circuit implementation of the steepest descent network.

This modification of the Hopfield network circuit is designed to implement the steepest descent dynamics. The amplifiers, which now have unit gain, act as current sourcing buffers. Simple diode circuits are attached to each amplifier input, limiting the voltages there to the range V_L*–*V_H*.*

Chapter 3

The Mapping Process

In this chapter we justify the premise that many combinatorial problems can be *mapped* onto optimization networks for solution. To do this, we must express the problem as the minimization of a single quadratic function over a set of 0-1 variables. Unfortunately, most combinatorial problems are more naturally posed in terms of *quadratic 0-1 programming*, requiring the satisfaction of a set of linear constraints in addition to the minimization of an objective function. While it is clear how the objective can be associated with a network's Liapunov function, it is less obvious how to guarantee satisfaction of the constraints. The mapping becomes easier if the constraints are recast in terms of quadratic *penalty functions*: functions which attain their minimal values when the constraints are satisfied. In previous attempts at problem mapping, the penalty functions were often constructed in a rather *ad hoc* manner, resulting in poor network performance. In contrast, the mapping presented here is completely rigorous, and can cope with inequality as well as equality constraints. Networks running under the mapping will never find invalid solutions: this constitutes the main original contribution of the chapter.

We begin in Section 3.1 by making explicit what we mean by a quadratic 0-1 programming problem, and then proceed to show how the travelling salesman problem can be expressed in this form. While this process is very much intuitive, it can in fact be repeated for many useful combinatorial problems. A rigorous mapping for general quadratic 0-1 programming problems is presented in Section 3.2. An appealing feature of the mapping is that it can handle arbitrary inequality constraints using conventional optimization networks. Alternative strategies for inequality constraints, reviewed in Section 3.3, appeal to more complicated networks and algorithms, sacrificing the potential for simple hardware implementations. Section 3.4 describes how steepest descent networks, running under the new mapping, can be simulated on digital computers with increased efficiency. Finally, in Section 3.5, we show how the mapping works for the travelling salesman, Hamilton path, graph labelling and knapsack problems, which will be used as illustrative examples throughout this thesis.

3.1 Combinatorial optimization as quadratic 0-1 programming

Before deriving a general purpose mapping, it is necessary to obtain a standard form for the mathematical expression of combinatorial optimization problems. The most convenient format is the *quadratic 0-1 programming problem*:

$$\begin{aligned}
\text{minimize} \quad & E^{\text{op}}(\mathbf{v}) = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{\text{op}}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\text{op}} && (3.1)\\
\text{subject to} \quad & \mathbf{A}^{\text{eq}}\mathbf{v} = \mathbf{b}^{\text{eq}} && (3.2)\\
\text{and} \quad & \mathbf{A}^{\text{in}}\mathbf{v} \leq \mathbf{b}^{\text{in}} && (3.3)\\
\text{and} \quad & 0 \leq v_i \leq 1, \quad i \in \{1 \ldots n\} && (3.4)\\
\text{and} \quad & \mathbf{v} \text{ integral} && (3.5)\\
\text{where} \quad & \mathbf{v} \in \mathbb{R}^N, \mathbf{b}^{\text{eq}} \in \mathbb{R}^{m^{\text{eq}}}, \mathbf{b}^{\text{in}} \in \mathbb{R}^{m^{\text{in}}}
\end{aligned}$$

Most useful combinatorial problems are easily cast in this form. With reference to condition (3.5), we define an integral vector to be one whose elements are all integers. The system (3.1)–(3.5) describes an optimization problem where it is required to minimize the quadratic objective (3.1) over a set of N variables, subject to a set of m^{eq} linear equality constraints (3.2) and m^{in} linear inequality constraints (3.3). The final two conditions (3.4)–(3.5) ensure that $v_i \in \{0, 1\}$. The objective function E^{op} is *not* the same as the objective E minimized by optimization networks. The aim of the mapping process is to show how to construct E so that the network's solution not only minimizes E^{op}, but also satisfies the problem's constraints. While both E and E^{op} are correctly referred to as *objectives*, we will use their mathematical symbols to distinguish between them where necessary.

There are no hard rules for expressing combinatorial optimization problems in the form (3.1)–(3.5), but rather an individual assessment is required for each problem. For example, consider the planar Euclidean travelling salesman problem:

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

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 C_{j1} is the x-coordinate of city j, and C_{j2} is the y-coordinate of the same city. We could represent the solution as an n-element permutation vector $\mathbf{p}$, where $p_i = j$ if the city initially in list position j is to be placed in position i of the tour. We must also enforce

$$p_i \neq p_j \quad i, j \in \{1 \ldots n\}, i \neq j \tag{3.6}$$

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

$$V_{ij} = \begin{cases} 1 & \text{if } p_i = j \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

With the above definition of $\mathbf{V}$, the $n \times 2$ matrix $\mathbf{C}' = \mathbf{VC}$ contains the same city coordinates as $\mathbf{C}$, but reordered in the manner defined by $\mathbf{p}$: this process is illustrated in Figure 3.1.

Since $\mathbf{V}$ is a permutation matrix, its elements are all either 1's or 0's, and each row and column contains only a single 1. This can by expressed mathematically as follows:

$$V_{ij} \in \{1,0\} \tag{3.8}$$

$$\sum_{i=1}^{n} V_{ij} = \sum_{j=1}^{n} V_{ij} = 1 \quad i,j \in \{1\dots n\} \tag{3.9}$$

We can now use $\mathbf{V}$ to obtain an expression for the total path length of a tour specified by the permutation $\mathbf{p}$. Let $\mathbf{P}$ be the $n \times n$ matrix of negated inter-city distances given by

$$P_{ij} = -\ (\text{distance between cities } i \text{ and } j) \tag{3.10}$$

Let $\mathbf{Q}$ be the $n \times n$ matrix given by[1]

$$Q_{ij} = \delta_{j\ominus 1,i} + \delta_{j\oplus 1,i} \quad i,j \in \{1\dots n\} \tag{3.11}$$

where

$$\begin{aligned} \delta_{i,j} &= \begin{cases} 1 & \text{if } i=j \\ 0 & \text{otherwise} \end{cases} \\ a \oplus b &= a+b \quad \text{except} \quad n \oplus 1 = 1 \\ a \ominus b &= a-b \quad \text{except} \quad 1 \ominus 1 = n \end{aligned}$$

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

$$E^{\text{op}} = -\tfrac{1}{2}\,\text{trace}\left[\mathbf{V}\mathbf{P}\mathbf{V}^T\mathbf{Q}\right] \tag{3.12}$$

and the optimization problem can be concisely expressed as $\min_{\mathbf{p}} \; E^{\text{op}}$. $\mathbf{Q}$ effectively picks out the distances travelled in a tour from the permuted distance matrix $\mathbf{V}\mathbf{P}\mathbf{V}^T$. Note that we require $N = n^2$ variables to represent the solution to an n-city problem.

We have now succeeded in expressing the travelling salesman problem as a quadratic 0-1 programming problem. We have a quadratic objective (3.12), which we must minimize over a set of N 0-1 variables V_{ij}, subject to a set of linear constraints (3.9). For the travelling salesman problem all the constraints happened to be equality constraints, though this is not always the case: for example, inequality constraints are required for problems of the knapsack [53, 68, 113] and scheduling [58] variety.

In order to cast the problem in the standard form (3.1)–(3.5), we must rewrite it in terms of a vector of 0-1 variables, instead of the matrix we have at the moment. This is easily carried out using Kronecker product notation, which is reviewed in Appendix C. If we define

$$\mathbf{v} = \text{vec}(\mathbf{V}) \tag{3.13}$$

then the objective function becomes[2]

$$E^{\text{op}} = -\tfrac{1}{2}\mathbf{v}^T(\mathbf{P}\otimes\mathbf{Q})\mathbf{v} \tag{3.14}$$

[1] For example, if $n=5$ then $\mathbf{Q}$ is $\begin{bmatrix} 0&1&0&0&1 \\ 1&0&1&0&0 \\ 0&1&0&1&0 \\ 0&0&1&0&1 \\ 1&0&0&1&0 \end{bmatrix}$.

[2] This expression is derived using equations (C.6) and (C.7), giving the result $E^{\text{op}} = -\frac{1}{2}\mathbf{v}^T(\mathbf{P}^T\otimes\mathbf{Q})\mathbf{v}$. Since $\mathbf{P}$ is symmetric, this is equivalent to (3.14). Throughout this thesis, symmetric matrices are routinely replaced by their transposes, reducing the amount of mathematical exposition required for many derivations.

and the constraints become

$$\sum_{i=1}^{n} v_{n(j-1)+i} = 1 \quad j \in \{1 \ldots n\} \tag{3.15}$$

$$\sum_{j=1}^{n} v_{n(j-1)+i} = 1 \quad i \in \{1 \ldots n\} \tag{3.16}$$

In terms of the standard form (3.1)–(3.5), we have

$$\mathbf{T}^{\text{op}} = \mathbf{P} \otimes \mathbf{Q} \tag{3.17}$$

$$\mathbf{i}^{\text{op}} = \mathbf{0} \tag{3.18}$$

To complete the standard form description, we can easily construct $\mathbf{A}^{\text{eq}}$ and $\mathbf{b}^{\text{eq}}$ from (3.15) and (3.16).

The process of expressing the travelling salesman problem as a quadratic 0-1 programming problem proved to be relatively straightforward. It is possible to express many other problems in this form, including the Hamilton path [51, 74], graph partitioning [5, 117, 118, 143], graph labelling [51, 50, 52, 96], N-Queens [10], knapsack [53, 68, 113], assignment [41, 131, 132] and scheduling [58] problems (see also Appendix A.5). Once we have expressed the combinatorial problem as quadratic 0-1 programming, the mapping onto optimization networks, described in Section 3.2, is straightforward and mechanical.

3.2 Mapping quadratic 0-1 programming problems

It is clear that if we set up an optimization network with $E = E^{\text{op}}$, then the network will find solutions which locally minimize the problem's objective E^{op}. However, it is less obvious how to ensure that these solutions also satisfy the problem's constraints. One approach is to code the constraints as terms in E which are minimized when the constraints are satisfied: such terms are commonly known as *penalty functions*. By seeking out low energy states, the network will tend to find solutions which do not violate the constraints.

In the early work on optimization networks, the penalty functions were constructed in a rather *ad hoc* manner, usually employing one term for each constraint. A typical Liapunov function would take the form

$$E = E^{\text{op}} + c_1 E_1^{\text{cns}} + c_2 E_2^{\text{cns}} + \ldots \tag{3.19}$$

where E^{op} is the objective function to be minimized (eg. the length of a travelling salesman tour) and the E^{cns} terms are the penalty functions. The c_i parameters in equation (3.19) are constant weightings given to the various terms, usually set by trial and error. Unfortunately, the multiplicity of terms tend to frustrate one another, and the success of the network is highly sensitive to the relative values of the c_i parameters. Networks running under such naïve mappings rarely found valid solutions, let alone high quality ones [81, 149].

In this section we shall develop a mapping using a far more rigorous approach to penalty functions. Our goal is that the mapping should guarantee satisfaction of the problem's constraints. This can be made more explicit in geometric terms. We note that the conditions (3.2)–(3.4), if feasible, define a bounded polyhedron, or polytope, within which $\mathbf{v}$ must remain if it is to represent a valid solution: let us denote this polytope by the symbol P. Furthermore, all 0-1 points within P are vertices of P, though it does not necessarily follow that all vertices of P are 0-1 points. We can achieve our goal by ensuring that, under the problem mapping, $\mathbf{v}$ remains strictly within P at all times. We would also like to arrange that, while remaining within P, $\mathbf{v}$ performs some sort of descent on the problem's objective function E^{op} (3.1). We would probably employ an annealing procedure to free $\mathbf{v}$ from local minima of E^{op} and drive $\mathbf{v}$ towards the boundary of P, hopefully forcing convergence to a 0-1 vertex of P: convergence issues will be investigated in Chapter 4.

We begin the derivation of the mapping by considering the simple case where there are no inequality constraints (as for the travelling salesman problem), leaving only equality constraints of the form (3.2). The equality constraints constitute a system of linear equations which can be solved to obtain an affine subspace of solutions. If the rows of $\mathbf{A}^{\mathrm{eq}}$ are linearly independent (as is invariably the case for a set of feasible, irredundant constraints), then we can describe this subspace in the form (see any linear algebra text, for example [45] or [88, p. 49])

$$\begin{aligned} \mathbf{v} &= \mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s} && (3.20)\\ \text{where} \quad \mathbf{T}^{\mathrm{val}} &= \mathbf{I} - \mathbf{A}^{\mathrm{eq}T}(\mathbf{A}^{\mathrm{eq}}\mathbf{A}^{\mathrm{eq}T})^{-1}\mathbf{A}^{\mathrm{eq}} && (3.21)\\ \text{and} \quad \mathbf{s} &= \mathbf{A}^{\mathrm{eq}T}(\mathbf{A}^{\mathrm{eq}}\mathbf{A}^{\mathrm{eq}T})^{-1}\mathbf{b}^{\mathrm{eq}} && (3.22) \end{aligned}$$

Equation (3.20) defines an affine subspace, on which $\mathbf{v}$ must lie if it is to satisfy the constraints (3.2); so all valid solution points must necessarily be 0-1 points on this subspace. The subspace, which was originally termed the *valid subspace* [5], is completely specified by the projection matrix $\mathbf{T}^{\mathrm{val}}$ and the offset vector $\mathbf{s}$[3]. While we have described just one way of finding $\mathbf{T}^{\mathrm{val}}$ and $\mathbf{s}$, it should be realized that they are simply obtained by solving the set of linear equations (3.2): this can be carried out using a variety of standard techniques [119], including Gaussian elimination and singular value decomposition.

From a polyhedral perspective, the subspace (3.20), together with the hypercube faces (3.4), define the polytope P for this problem. We know that the network's transfer functions will ensure that $\mathbf{v}$ always remains within the unit hypercube, so if we can ensure that $\mathbf{v}$ is also pinned to the valid subspace, then confinement of $\mathbf{v}$ within P is guaranteed. We can achieve this by using a carefully constructed penalty function

$$E^{\mathrm{cns}} = \tfrac{1}{2}\left\|\mathbf{v} - (\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s})\right\|^2 \tag{3.25}$$

Note that when $\mathbf{v}$ lies on the valid subspace, E^{cns} is zero, while E^{cns} grows rapidly as $\mathbf{v}$ moves away from the valid subspace. The overall objective function is then

$$E = E^{\mathrm{op}} + cE^{\mathrm{cns}} \tag{3.26}$$

In the limit of large c, the penalty term E^{cns} will dominate over E^{op}, and any descent procedure on E will ensure that $\mathbf{v}$ remains pinned to the valid subspace (and therefore

[3]Note that $\mathbf{T}^{\mathrm{val}}$ is a projection matrix, so

$$\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{val}} = \mathbf{T}^{\mathrm{val}} \tag{3.23}$$

Also, s lies in the nullspace of $\mathbf{T}^{\mathrm{val}}$, so

$$\mathbf{T}^{\mathrm{val}}\mathbf{s} = \mathbf{0} \tag{3.24}$$

within P) throughout convergence. Furthermore, for $\mathbf{v}$ within P we see that $E = E^{\mathrm{op}}$, so the penalty term does not interfere with the objective term so long as the constraints are satisfied. The objective function E, as constructed in equation (3.26), effectively mimics E^{op} within P, while building high walls around P to ensure that $\mathbf{v}$ cannot escape into invalid regions — this principle is illustrated in Figure 3.2.

Substituting for E^{op} and E^{cns} in equation (3.26), we obtain

$$\begin{aligned} E &= -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{\mathrm{op}}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{op}} + \tfrac{1}{2}c\,\|\mathbf{v} - (\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s})\|^2 \\ &= -\tfrac{1}{2}\mathbf{v}^T\left[\mathbf{T}^{\mathrm{op}} + c(\mathbf{T}^{\mathrm{val}} - \mathbf{I})\right]\mathbf{v} - \mathbf{v}^T(\mathbf{i}^{\mathrm{op}} + c\mathbf{s}) + \tfrac{1}{2}c\mathbf{s}^T\mathbf{s} \qquad (3.27) \end{aligned}$$

Discarding the constant term, we see that (3.27) can be associated with the Liapunov function (2.7) of an optimization network, by setting the network's parameters as follows:

$$\mathbf{T} = \mathbf{T}^{\mathrm{op}} + c(\mathbf{T}^{\mathrm{val}} - \mathbf{I}) \qquad (3.28)$$

$$\mathbf{i}^{\mathrm{b}} = \mathbf{i}^{\mathrm{op}} + c\mathbf{s} \qquad (3.29)$$

The mapping can be extended to cope with inequality constraints of the form (3.3). This is possible using the *slack variable* technique, by which inequality constraints are converted into equality constraints. Slack variables were first proposed for the assignment problem in [132] and subsequently generalized in [131]. Here we improve the technique so that each inequality constraint requires the introduction of only *one* slack variable[4]. A typical constraint

$$A^{\mathrm{in}}_{i1}v_1 + A^{\mathrm{in}}_{i2}v_2 + \ldots + A^{\mathrm{in}}_{in}v_n \leq b^{\mathrm{in}}_i \qquad (3.30)$$

becomes

$$A^{\mathrm{in}}_{i1}v_1 + A^{\mathrm{in}}_{i2}v_2 + \ldots + A^{\mathrm{in}}_{in}v_n + k_i w_i = b^{\mathrm{in}}_i, \quad k_i w_i \geq 0 \qquad (3.31)$$

In (3.31) w_i is the slack variable, and k_i is a positive constant which is set to ensure that w_i is bounded between 0 and 1, and can therefore be treated in the same manner as the v variables. This is achieved by setting k_i as follows[5]:

$$k_i = b^{\mathrm{in}}_i - \sum_{\{j \mid A^{\mathrm{in}}_{ij} < 0\}} A^{\mathrm{in}}_{ij} \qquad (3.32)$$

Thus we can transform the system of inequality constraints into a system of equality constraints acting on an extended set of variables $\mathbf{v}^{+T} = [\mathbf{v}^T\ \mathbf{w}^T]$, where $\mathbf{w}$ is the vector of slack variables $[w_1\ w_2\ \ldots\ w_{m^{\mathrm{in}}}]^T$: the size of $\mathbf{v}^+$ is therefore $n^+ = n + m^{\mathrm{in}}$. The optimization problem can now be expressed as

$$\begin{aligned} \text{minimize} \quad & E^{\mathrm{op}}(\mathbf{v}) = -\tfrac{1}{2}\mathbf{v}^{+T}\mathbf{T}^{\mathrm{op}+}\mathbf{v}^{+} - \mathbf{v}^{+T}\mathbf{i}^{\mathrm{op}+} && (3.33) \\ \text{subject to} \quad & \mathbf{A}^{+}\mathbf{v}^{+} = \mathbf{b}^{+} && (3.34) \\ \text{and} \quad & 0 \leq v^{+}_i \leq 1, \quad i \in \{1 \ldots n^{+}\} && (3.35) \\ \text{and} \quad & \mathbf{v} \text{ integral} && (3.36) \end{aligned}$$

[4] A technique requiring only one slack variable per inequality constraint is also proposed in [3]. However, the slack variables are not bounded between 0 and 1, and therefore require modified transfer functions to confine them to the correct range. This constitutes an unnecessary complication, especially as regards any hardware implementation of the network.

[5] If equation (3.32) gives a negative value for k_i, then the inequality constraint (3.30) cannot be satisfied by any 0-1 vector $\mathbf{v}$.

where $\mathbf{T}^{\mathrm{op}+} = \begin{bmatrix} \mathbf{T}^{\mathrm{op}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, $\mathbf{i}^{\mathrm{op}+} = \begin{bmatrix} \mathbf{i}^{\mathrm{op}} \\ \mathbf{0} \end{bmatrix}$, $\mathbf{v}^+ \in \mathbb{R}^{n^+}$ and $\mathbf{b}^+ \in \mathbb{R}^{(m^{\mathrm{eq}}+m^{\mathrm{in}})}$. Note that E^{op} has no dependency on the slack variables $\mathbf{w}$, and the equality constraints (3.34) embody both the original equality and inequality constraints (3.2) and (3.3). The formulation (3.33)–(3.36) contains only equality constraints, and so the mapping technique described earlier is applicable. We can therefore find an $n^+ \times n^+$ matrix $\mathbf{T}^{\mathrm{val}}$, and an n^+-element vector $\mathbf{s}$, such that if the network's parameters are set as in (3.28)–(3.29) (substituting $\mathbf{T}^{\mathrm{op}+}$ for $\mathbf{T}^{\mathrm{op}}$ and $\mathbf{i}^{\mathrm{op}+}$ for $\mathbf{i}^{\mathrm{op}}$), the state vector $\mathbf{v}^+$ will perform a descent on E^{op} while remaining strictly within the polytope P^+ defined by (3.34) and (3.35). If we observe only the $\mathbf{v}$ variables within the extended set $\mathbf{v}^+$, we shall see that they perform a descent on E^{op} while remaining strictly within the polytope P, as required.

The operation of the slack variable mapping is illustrated in Figure 3.3. We take as an example a simple 2-dimensional problem with a single inequality constraint $v_1 + v_2 \leq 1.7$. There is a one-to-one correspondence between the vertices of P and P^+, with P being a projection of P^+ onto the v_1–v_2 plane. The slack variable mapping effectively transforms the inequality constraint into the constraint $w \geq 0$, which is directly enforced by the network's transfer functions.

Note that the integrality conditions $v_i \in \{0, 1\}$ apply only to the original variables and not to the slack variables, which may take any value between 0 and 1 at a valid solution point. For example, in Figure 3.3 w takes non-integral values at the two valid solution points B and D. Thus it is not necessary (and indeed incorrect) to force the slack variables towards 0 or 1 using an annealing technique.

The entire mapping process is summarized for quick reference in Table 3.1.

3.3 Alternative strategies for inequality constraints

In [113] an alternative technique for mapping inequality constraints is described. For a typical linear constraint

$$\mathbf{a}_i^T \mathbf{v} \leq \mathbf{b}_i \tag{3.37}$$

the authors propose a penalty function of the form

$$E_i^{\mathrm{cns}} = \begin{cases} 0 & \text{if } \mathbf{a}_i^T \mathbf{v} \leq \mathbf{b}_i \\ \mathbf{a}_i^T \mathbf{v} - \mathbf{b}_i & \text{otherwise} \end{cases} \tag{3.38}$$

This approach has the advantage of not requiring any additional slack variables, leading to more efficient problem representations. However, E^{cns} is not quadratic and therefore cannot be mapped onto standard optimization networks. Instead, a modified MFA-type algorithm is proposed for optimizing under E^{cns}, though no simple analogue circuit implementation is possible. The slack variable mapping, although introducing what could be a large number of extra variables (though for many useful problems this is *not* the case), preserves the quadratic Liapunov function associated with simple analogue electrical circuits. The cost of the extra amplifiers for the slack variables would not be particularly high, and the advantage of an analogue implementation considerable.

Modified networks allowing non-quadratic penalty functions of the form (3.38) have also been proposed in [16, 99, 135, 144], though they are usually far more complicated than the standard optimization networks. In the remainder of this thesis, we shall deal exclusively with the quadratic objective function offered by the slack variable mapping, since this appears to offer a good compromise between implementation size and complexity.

3.4 Efficient simulation of steepest descent networks

The details of a problem mapping might affect the way we choose to simulate the continuous descent networks on digital computers. A standard Euler approximation of the dynamic equations, though feasible, would require a very small time-step at each iteration, and would therefore be slow to converge. This is because c in equations (3.28) and (3.29) must be large to guarantee confinement to the polytope P^+, resulting in correspondingly large values of $\dot{\mathbf{v}}^+$ when $\mathbf{v}^+$ strays marginally outside P^+. Hence a large time-step is bound to lead to unstable oscillations of $\mathbf{v}^+$ around the valid subspace.

Figure 3.4 shows a schematic illustration of a far more efficient simulation algorithm (previously presented in [5]) for the steepest descent dynamics (2.10). Each iteration has two distinct stages. First, as depicted in box (1), $\mathbf{v}^+$ is updated over a finite time-step Δt using the gradient of the objective term E^{op} alone. The time-step used here can be fairly large, since the problematic E^{cns} term is not involved. However, updating $\mathbf{v}^+$ in this manner will typically take $\mathbf{v}^+$ outside the polytope P^+. Hence, after every update, $\mathbf{v}^+$ is directly projected back onto P^+. This is an iterative process, requiring several passes around the loop (2), in which $\mathbf{v}^+$ is first orthogonally projected onto the valid subspace (3.20), and then thresholded so that its elements lie in the range 0 to 1. For more details of projection onto polytopes the reader is referred to [4, 37, 77, 109]. The resulting algorithm is highly efficient and has general applicability to any quadratic 0-1 programming problem. Throughout this thesis we shall make extensive use of the algorithm in simulating the steepest descent network applied to a variety of problems. The availability of such an algorithm is the key advantage of the steepest descent network over the Hopfield network, as far as simulation on digital computers is concerned.

3.5 Mapping some common problems

In this section we show how the mapping process works for some common problems used as examples throughout this thesis. The problems we shall discuss are the travelling salesman, Hamilton path, graph labelling and knapsack problems. Where possible, we shall derive expressions for the matrices $\mathbf{T}^{\mathrm{op}}$ and $\mathbf{T}^{\mathrm{val}}$, and the vectors $\mathbf{i}^{\mathrm{op}}$ and $\mathbf{s}$. We can then use equations (3.28) and (3.29) to set the network parameters $\mathbf{T}$ and $\mathbf{i}^{\mathrm{b}}$.

The travelling salesman problem

We have seen how the travelling salesman problem can be expressed as quadratic 0-1 programming. We derived expressions for $\mathbf{T}^{\mathrm{op}}$ (3.17) and $\mathbf{i}^{\mathrm{op}}$ (3.18), and formulated a set of linear equality constraints (3.15)–(3.16). The constraint equations can be solved to give a valid subspace with parameters [5]

$$\mathbf{T}^{\mathrm{val}} = \mathbf{R} \otimes \mathbf{R} \tag{3.39}$$

$$\mathbf{s} = \frac{1}{n}(\mathbf{o} \otimes \mathbf{o}) \tag{3.40}$$

where

$$\begin{aligned} \mathbf{R} &= \mathbf{I} - \tfrac{1}{n}\mathbf{O} && (3.41) \\ o_i &= 1, \quad i \in \{1 \dots n\} && (3.42) \\ O_{ij} &= 1, \quad i,j \in \{1 \dots n\} && (3.43) \end{aligned}$$

The constraint polytope associated with this mapping has the permutation matrices at its vertices, and is often referred to as Ω^n. We shall examine Ω^n in greater detail in Section 4.6.

The Hamilton path problem

The planar, Euclidean Hamilton path problem can be expressed as follows:

> Given n cities in a plane, 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.[6]

Thus the Hamilton path problem is identical to the travelling salesman problem in all respects, except that we require an open tour and not a closed one. We can therefore use a permutation matrix to represent the solution, leading to an optimization over Ω^n, with $\mathbf{T}^{\text{val}}$ and $\mathbf{s}$ as given in equations (3.39) and (3.40) respectively. We need only modify the objective function E^{op} to give the correct length for an open tour. We define a matrix $\mathbf{P}$ to be the $n \times n$ matrix of negated inter-city distances, as with the travelling salesman problem, but a slightly different matrix $\mathbf{Q}$[7]

$$Q_{ij} = \delta_{j-1,i} + \delta_{j+1,i} \qquad i,j \in \{1 \dots n\} \tag{3.44}$$

It is easily verified that the Hamilton path length corresponding to a permutation of cities $\mathbf{V}$ is given by [51]

$$E^{\text{op}} = -\tfrac{1}{2}\,\text{trace}\left[\mathbf{V}\mathbf{P}\mathbf{V}^T\mathbf{Q}\right] \tag{3.45}$$

This is an identical expression to that obtained for the objective of the travelling salesman problem, so we can set $\mathbf{T}^{\text{op}}$ and $\mathbf{i}^{\text{op}}$ as in equations (3.17) and (3.18) respectively.

The graph labelling problem

The graph labelling problem has application in certain invariant pattern recognition systems [21, 51, 52, 96]. Patterns are often described by means of a graph, where the graph nodes represent features of the pattern, and the edge connections encode relational descriptors between the features. We can arrange for the descriptors to be invariant to a wide variety of transformations of the patterns. The simplest example concerns planar dot patterns, where each of the graph nodes represents a dot, and the edge weight between each of the nodes is the Euclidean distance between the two dots in the plane — see

[6] Strictly speaking, we should refer to this as the *shortest* Hamilton path problem, though we will adopt the more compact nomenclature throughout this thesis.

[7] For example, if $n = 5$ then $\mathbf{Q}$ is $\begin{bmatrix} 0&1&0&0&0 \\ 1&0&1&0&0 \\ 0&1&0&1&0 \\ 0&0&1&0&1 \\ 0&0&0&1&0 \end{bmatrix}$.

Figure 3.5. Such a representation is invariant to translation and rotation of the patterns, and can also be made invariant to enlargement by a simple normalization operation.

Invariant pattern recognition can then be performed by template comparison. We compare two graphs $G_{\mathcal{P}}$ and $G_{\mathcal{Q}}$, representing patterns $\mathcal{P}$ and $\mathcal{Q}$, and compute some measure of similarity between them. However, before the similarity can be computed, each of the nodes in one graph has to be matched with a particular node in the other. Suppose the graphs contain the same number of nodes n, and let the edge weight matrix of $G_{\mathcal{P}}$ be $\mathbf{P}$, and that of $G_{\mathcal{Q}}$ be $\mathbf{Q}$. Then we can match nodes in $G_{\mathcal{P}}$ with nodes in $G_{\mathcal{Q}}$ using a permutation $\mathbf{p}$ of the nodes in $G_{\mathcal{P}}$. A quantity which measures the dissimilarity between the two graphs is then

$$E^{\mathrm{op}} = \left\| \mathbf{V}\mathbf{P}\mathbf{V}^T - \mathbf{Q} \right\|^2 \tag{3.46}$$

where $\mathbf{V}(\mathbf{p})$ is the permutation matrix corresponding to the permutation $\mathbf{p}$. The graph labelling problem is to find the permutation which minimizes E^{op}: this is illustrated in Figure 3.5.

Since we are yet again optimizing over Ω^n, we can use the same $\mathbf{T}^{\mathrm{val}}$ and $\mathbf{s}$ as we did for the travelling salesman problem (3.39)–(3.40). Expanding E^{op}, we obtain

$$E^{\mathrm{op}} = \|\mathbf{P}\|^2 + \|\mathbf{Q}\|^2 - 2\ \mathrm{trace}\left[\mathbf{V}\mathbf{P}\mathbf{V}^T\mathbf{Q}\right] \tag{3.47}$$

Discarding the two constant terms, we see that E^{op} is once again of the same form as the objective for the travelling salesman problem, so we set $\mathbf{T}^{\mathrm{op}}$ and $\mathbf{i}^{\mathrm{op}}$ as in (3.17) and (3.18) respectively.

The knapsack problem

The knapsack problem can be expressed as follows:

> We are presented with a knapsack of capacity C, and a set of n items. Item i has size x_i and usefulness y_i. We have to decide which items to pack in the knapsack to obtain the maximum usefulness from the collection, without overfilling the knapsack.

This problem can be posed as quadratic 0-1 programming as follows. Let $v_i = 1$ if item i is to be packed, and $v_i = 0$ otherwise. Then the problem is

$$\begin{array}{rl} \text{minimize} & E^{\mathrm{op}}(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^T\mathbf{y} \\ \text{subject to} & \mathbf{v}^T\mathbf{x} \le C \\ \text{and} & v_i \in \{0,1\} \quad i \in \{1 \ldots n\} \end{array}$$

where C and all the elements of $\mathbf{x}$ and $\mathbf{y}$ are nonnegative. We require one slack variable w to transform the single inequality constraint into an equality constraint, after which we can use equations (3.21) and (3.22) to find $\mathbf{T}^{\mathrm{val}}$ and $\mathbf{s}$. Note that E^{op} is linear for the knapsack problem, so $\mathbf{T}^{\mathrm{op}} = \mathbf{0}$. Unlike all the other problems we have met so far, the knapsack problem is not over Ω^n.

Problem	$$\begin{array}{rl} \text{minimize} & E^{\mathrm{op}}(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^T\mathbf{T}^{\mathrm{op}}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{op}} \\ \text{subject to} & \mathbf{A}^{\mathrm{eq}}\mathbf{v} = \mathbf{b}^{\mathrm{eq}} \\ \text{and} & \mathbf{A}^{\mathrm{in}}\mathbf{v} \leq \mathbf{b}^{\mathrm{in}} \\ \text{and} & v_i \in \{1,0\}, \quad i \in \{1\dots n\} \end{array}$$
Convert inequality constraints	$$\begin{array}{rcl} A^{\mathrm{in}}_{i1}v_1 + A^{\mathrm{in}}_{i2}v_2 + \ldots + A^{\mathrm{in}}_{in}v_n & \leq & b^{\mathrm{in}}_i \\ & \Downarrow & \\ A^{\mathrm{in}}_{i1}v_1 + A^{\mathrm{in}}_{i2}v_2 + \ldots + A^{\mathrm{in}}_{in}v_n + k_i w_i & = & b^{\mathrm{in}}_i, \quad k_i w_i \geq 0 \\ k_i = b^{\mathrm{in}}_i - \displaystyle\sum_{\{j \mid A^{\mathrm{in}}_{ij} < 0\}} A^{\mathrm{in}}_{ij} & \Rightarrow & 0 \leq w_i \leq 1 \end{array}$$
Equality-constrained problem	$$\begin{array}{rl} \text{minimize} & E^{\mathrm{op}}(\mathbf{v}) = -\frac{1}{2}\mathbf{v}^{+T}\mathbf{T}^{\mathrm{op}+}\mathbf{v}^{+} - \mathbf{v}^{+T}\mathbf{i}^{\mathrm{op}+} \\ \text{subject to} & \mathbf{A}^{+}\mathbf{v}^{+} = \mathbf{b}^{+} \\ \text{and} & 0 \leq v^{+}_i \leq 1, \quad i \in \{1\dots n^{+}\} \\ \text{and} & \mathbf{v} \text{ integral} \\ \text{where} & \mathbf{v}^{+T} = [\mathbf{v}^T \; \mathbf{w}^T] \end{array}$$
Solve equality constraints	$$\begin{array}{rrcl} & \mathbf{A}^{+}\mathbf{v}^{+} & = & \mathbf{b}^{+} \\ & & \Downarrow & \\ & \mathbf{v}^{+} & = & \mathbf{T}^{\mathrm{val}}\mathbf{v}^{+} + \mathbf{s} \\ \text{where} & \mathbf{T}^{\mathrm{val}} & = & \mathbf{I} - \mathbf{A}^{+T}(\mathbf{A}^{+}\mathbf{A}^{+T})^{-1}\mathbf{A}^{+} \\ \text{and} & \mathbf{s} & = & \mathbf{A}^{+T}(\mathbf{A}^{+}\mathbf{A}^{+T})^{-1}\mathbf{b}^{+} \end{array}$$
Problem as a single quadratic objective	$$\text{minimize } E = E^{\mathrm{op}} + \tfrac{1}{2}c\,\lVert\mathbf{v}^{+} - (\mathbf{T}^{\mathrm{val}}\mathbf{v}^{+} + \mathbf{s})\rVert^2$$ where c is a large weighting given to the penalty function
Set network interconnections and biases	$$\begin{array}{rcl} \mathbf{T} & = & \mathbf{T}^{\mathrm{op}+} + c(\mathbf{T}^{\mathrm{val}} - \mathbf{I}) \\ \mathbf{i}^{\mathrm{b}} & = & \mathbf{i}^{\mathrm{op}+} + c\mathbf{s} \end{array}$$

Table 3.1: Mapping quadratic 0-1 programming problems.

The table shows how any quadratic 0-1 programming problem can be mapped onto a standard optimization network for solution. Most useful combinatorial problems can be expressed as quadratic 0-1 programming.

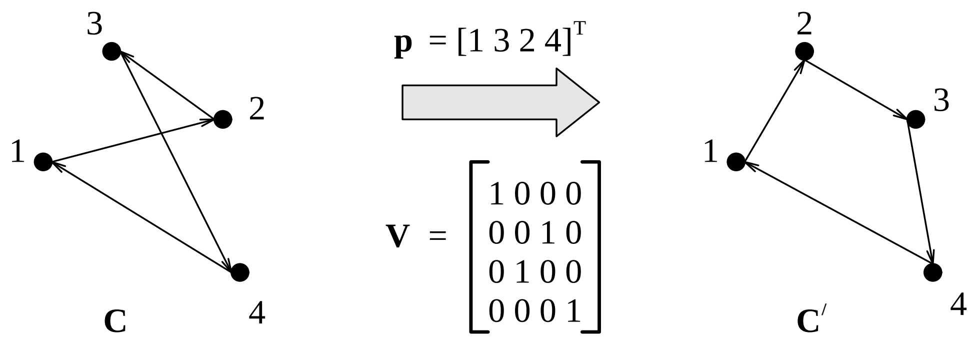


Figure 3.1: The action of permutation vectors and matrices.

A permutation is compactly expressed by means of a permutation vector $\mathbf{p}$. *The corresponding permutation matrix* $\mathbf{V}$ *can operate on a list of coordinates* $\mathbf{C}$ *to produce a reordered list* $\mathbf{C}'$.

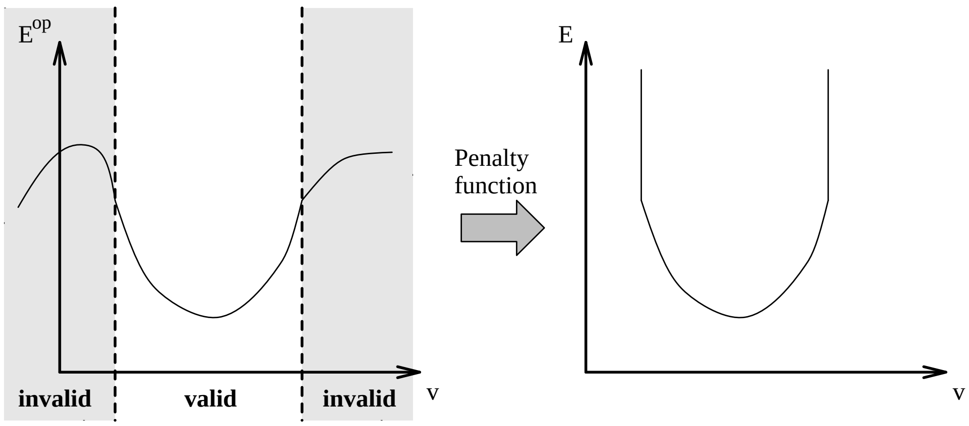


Figure 3.2: A rigorous penalty function in one dimension.

The left hand figure shows a simple optimization problem where it is required to minimize E^{op} *within a limited range of validity. To express the problem as the minimization of a single function* E*, it is necessary to absorb the validity constraints into a* penalty function. *A proper penalty function, when added to* E^{op}*, builds high energy walls around the region of validity, leaving* E^{op} *unchanged within that region.*

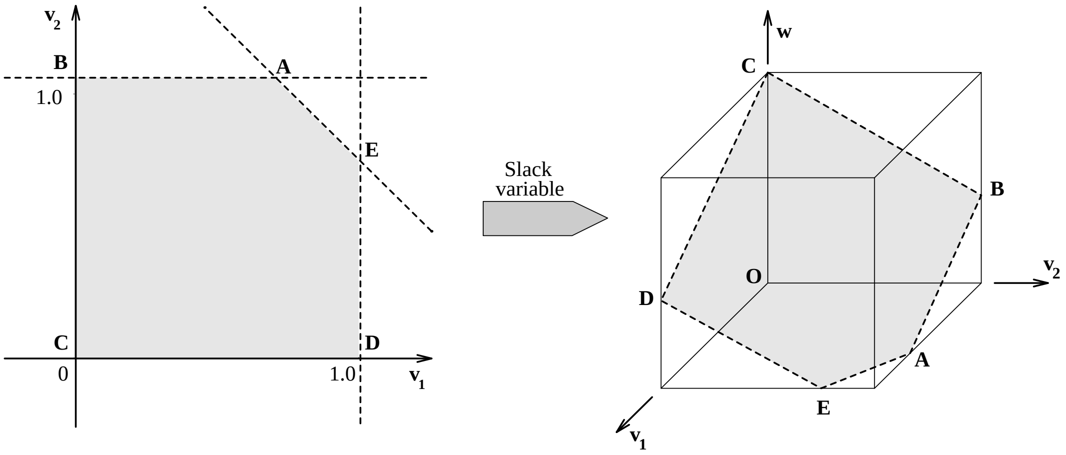


Figure 3.3: The slack variable mapping process.

On the left we see the polytope P for a simple 2-variable problem with one inequality constraint $v_1 + v_2 \leq 1.7$. On the right we see the polytope P^+ obtained using the slack variable mapping. The mapping effectively transforms the inequality constraint into the condition $w > 0$, which is directly enforced by an optimization network's transfer functions.

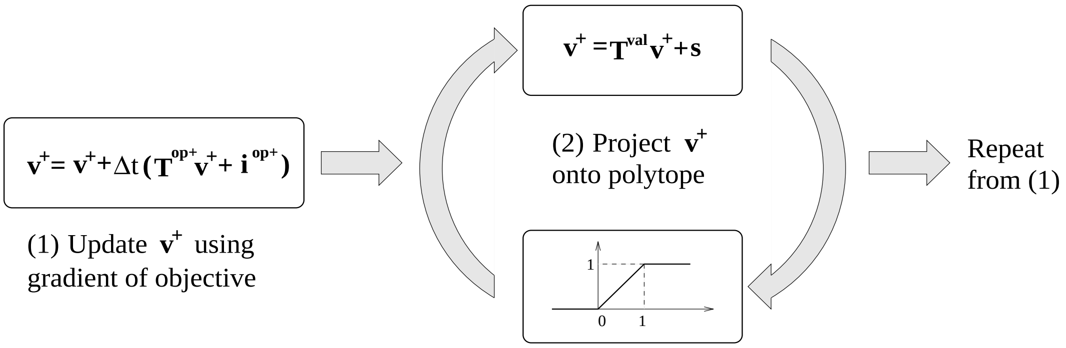


Figure 3.4: Schematic diagram of the efficient descent algorithm.

Each iteration of the algorithm has two distinct stages. First, as depicted in box (1), $\mathbf{v}^+$ is updated over a finite time-step Δt using the gradient of the objective term E^{op} alone. Updating $\mathbf{v}^+$ in this manner will typically take $\mathbf{v}^+$ outside the polytope P^+. Hence, after every update, $\mathbf{v}^+$ is directly projected back onto P^+. This is an iterative process, requiring several passes around the loop (2), in which $\mathbf{v}^+$ is first orthogonally projected onto the subspace $\mathbf{v}^+ = \mathbf{T}^{\mathrm{val}}\mathbf{v}^+ + \mathbf{s}$, and then thresholded so that its elements lie in the range 0 to 1.

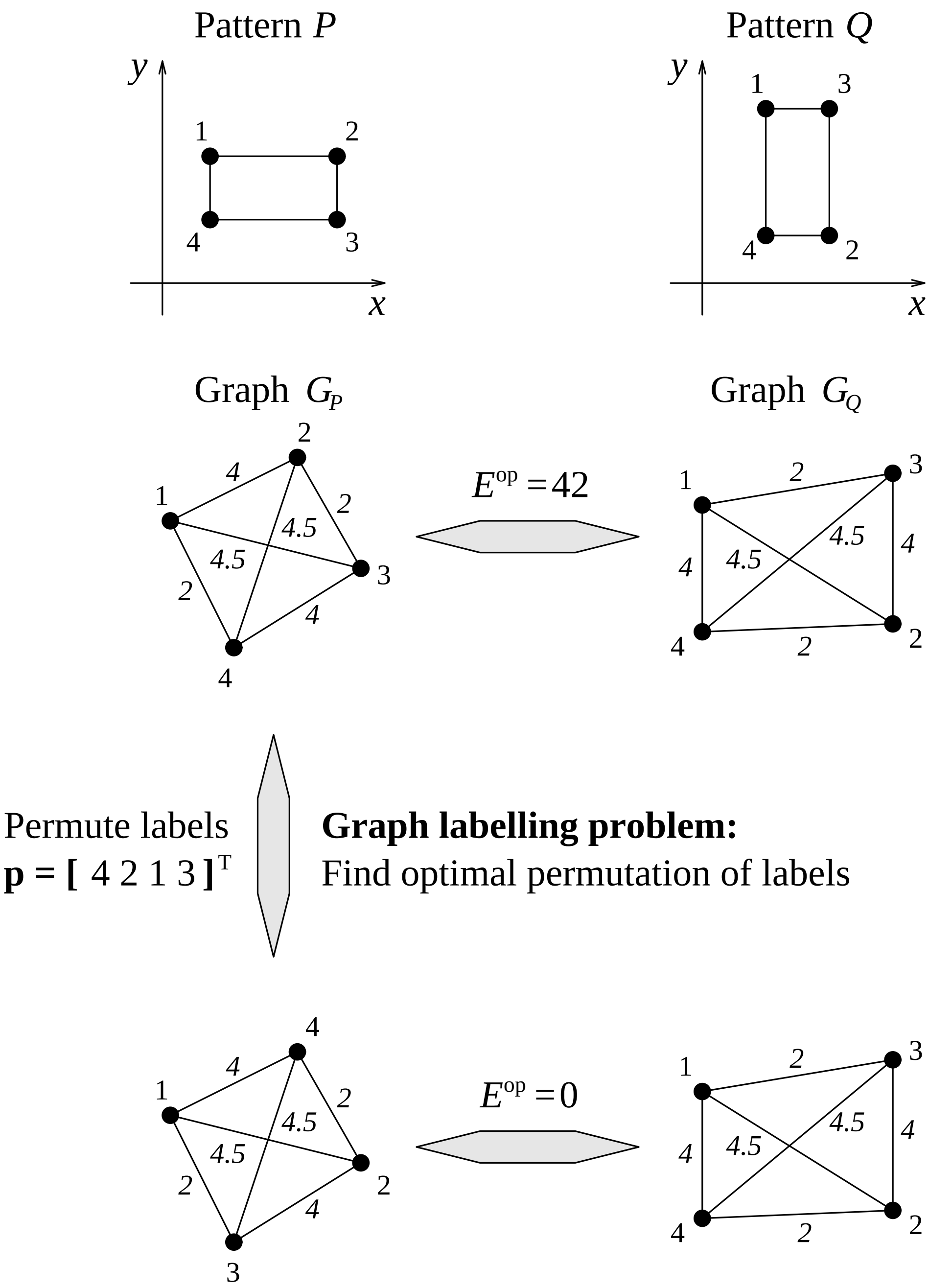


Figure 3.5: Invariant pattern recognition by graph matching.

The two patterns $\mathcal{P}$ and $\mathcal{Q}$ are identical up to a rotation of $90°$. A pattern recognition system invariant to rotation must recognize the congruence of the two patterns. The graphical representations $G_\mathcal{P}$ and $G_\mathcal{Q}$ are insensitive to rotation and translation of the patterns. For each graph, the edge weight between nodes i and j represents the Euclidean distance between points i and j in the corresponding pattern. However, before the two graphs can be compared, each of the nodes in $G_\mathcal{P}$ must be associated with the correct node in $G_\mathcal{Q}$. This is achieved by solving the Graph Labelling Problem*: find the permutation* $\mathbf{p}$ *which minimizes $E^{\text{op}} = \|\mathbf{VPV}^T - \mathbf{Q}\|^2$, where $\mathbf{P}$ and $\mathbf{Q}$ are the edge weight matrices of $G_\mathcal{P}$ and $G_\mathcal{Q}$ respectively, and $\mathbf{V}$ is the permutation matrix corresponding to $\mathbf{p}$. The function E^{op} computes a measure of dissimilarity between $G_\mathcal{Q}$ and the relabelled graph $G_\mathcal{P}$. Only when the correct labelling is found is the similarity between $\mathcal{P}$ and $\mathcal{Q}$ revealed.*

Chapter 4

Polytopes and Convergence

So far we have deliberately skirted round the issue of how we might force an optimization network to converge to a 0-1 point, merely stating that this is one of the goals of an annealing procedure. In this chapter we present a thorough examination of the convergence properties of optimization networks running under rigorous problem mappings. In particular, we ask under what conditions we can expect an annealing process to force convergence to a 0-1 point. It becomes apparent that a 0-1 point cannot be found except in special cases where the optimization is over an *integral polytope*, which has *all* its vertices at integral points. Moreover, this failure is not a consequence of any minor, correctable detail, but a direct result of a very basic design misconception: that optimization networks should be able to converge to a valid solution point in one attempt. Drawing on powerful results from mathematical programming theory, we show that finding valid solution points is generally an $\mathcal{NP}$-complete problem. It would therefore be unreasonable to expect an optimization network to find such points with a one-shot descent procedure. This forces us to reasses the application of optimization networks, and consider how they might be modified to work with a broader class of problems.

The argument will be illustrated using small example problems, so that we can easily visualize the solution space. In Section 4.2 we present a simple knapsack problem in two variables, for which an optimization network will fail to converge to a 0-1 point. Furthermore, in Section 4.3 we show that none of the common annealing techniques can help. These findings are related to issues in $\mathcal{NP}$ theory in Sections 4.4 and 4.5, where we introduce the concept of integral polytopes and discuss their significance. In Section 4.6 we direct our attention to the polytope Ω^n, and present a proof of its integrality: hence convergence to a 0-1 point can be guaranteed for the travelling salesman and other problems over Ω^n. Finally, in Section 4.7 we reasses the application of optimization networks in the light of the polyhedral issues, and propose alternative strategies for optimization over non-integral polytopes. Since polyhedral issues have been largely ignored in the relevant literature so far, most of this chapter represents an original contribution to optimization network theory.

4.1 $\mathcal{NP}$ theory and complexity

We briefly introduced the concept of an $\mathcal{NP}$-complete problem in Chapter 2, mentioning that to find the optimal solution to such a problem we have little choice but to search every possible solution and pick the best. The number of candidate solutions scales exponentially with the size of the problem, and so exhaustive search is infeasible for all but the smallest of problems. In this section we shall briefly review the ideas behind $\mathcal{NP}$ theory, and discuss the bearing it has on the field of optimization networks.

$\mathcal{NP}$-complete problems have the property that they can all be transformed into one another in polynomial time. It follows that if one of them can be solved in polynomial time then so can the rest. It has become a popular field of mathematics to demonstrate that particular problems are $\mathcal{NP}$-complete, with more problems being added to the list all the time. Typical examples of $\mathcal{NP}$-complete problems are the travelling salesman, graph partitioning, knapsack and scheduling problems. In [47], the authors compiled a useful list of all the problems proved $\mathcal{NP}$-complete up to 1979. While some of these problems may not seem particularly important in everyday life, many have real practical significance. The search for an exact, polynomial-time solution algorithm has therefore been very intensive. However, despite the best efforts of many researchers over many years, no such algorithm has been found for any of the $\mathcal{NP}$-complete problems. It has therefore been widely accepted that the classes of polynomially solvable problems and $\mathcal{NP}$-complete problems are mutually exclusive, that is $\mathcal{P} \neq \mathcal{NP}$, though no proof has yet been found [47].

It has become common practice to make use of the assertion that $\mathcal{P} \neq \mathcal{NP}$. We shall do this in two distinct ways. Primarily, as proposed in Chapter 2, we shall not waste our time trying to derive polynomial-time algorithms to exactly solve $\mathcal{NP}$-complete problems, but shall instead hope that optimization networks find *good*, though not necessarily *optimal*, solutions in reasonable amounts of time. At this point we should note that the operating speed of an optimization network is at best independent of problem size (when implemented in analogue hardware), and at worst scales with the square of the number of neurons (when simulated on a digital computer)[1]. We shall also make use of $\mathcal{NP}$ theory to place bounds on the capabilities of such networks. For instance, if we can demonstrate that by performing a particular task an optimization network is effectively solving an $\mathcal{NP}$-complete problem, we could conclude that it is unreasonable to expect the network to perform such a task.

4.2 A simple knapsack problem

We begin our study of convergence issues with a simple example. Consider the following knapsack problem in two variables

$$\begin{aligned} \text{minimize} \quad & E^{\text{op}}(\mathbf{v}) = -(v_1 + 2v_2) && (4.1)\\ \text{subject to} \quad & v_1 + v_2 \leq 1.7 && (4.2)\\ \text{and} \quad & 0 \leq v_i \leq 1, \quad i \in \{1, 2\} && (4.3)\\ \text{and} \quad & \mathbf{v} \text{ integral} && (4.4) \end{aligned}$$

In knapsack terminology we have two items, v_1 and v_2, of equal size 1, which we wish to pack into a knapsack of capacity 1.7. Associated with each item is a profit, 1 for v_1 and 2 for v_2, indicating how useful that item is. We have to decide which items to pack to maximize the profit, while at the same time not overfilling the knapsack. For knapsack

[1]The complexity of finding stable states of optimization networks is discussed in [105]. It is shown that for a network of arbitrary connectivity (ie. not necessarily symmetrical connectivity, so there is no Liapunov function and convergence guarantee) stable states can be computed using a simplex-like algorithm. This algorithm is strongly polynomial in practice, though not necessarily so in theory.

problems the objective function E^{op} is linear and therefore has no local minima. It is clear that the optimal solution to our simple problem is $\mathbf{v} = [0\ 1]^T$, though the general knapsack problem is $\mathcal{NP}$-complete [47, 123].

The problem is easily mapped onto an optimization network for solution, by introducing one slack variable w to cope with the single inequality constraint (4.2). The slack variable expression of the problem is

$$\begin{aligned} \text{minimize} \quad & E^{\mathrm{op}}(\mathbf{v}) = -(v_1 + 2v_2) && (4.5) \\ \text{subject to} \quad & v_1 + v_2 + 1.7w = 1.7 && (4.6) \\ \text{and} \quad & 0 \le v_i^+ \le 1, \quad i \in \{1,2,3\} && (4.7) \\ \text{and} \quad & \mathbf{v} \text{ integral} && (4.8) \end{aligned}$$

Figure 4.1 shows the polytopes P and P^+ for this problem. Also marked within P are contours of E^{op} and a typical descent trajectory within the polytope. We see that the descent converges to the point A $= [0.7\ 1.0]^T$, which is a vertex of P. The corresponding descent path within P^+ is also illustrated. The vertex A is in fact the optimal solution to the problem (4.1)–(4.3), without the integrality condition (4.4). This less restricted problem is known as the *LP-relaxation* (linear programming relaxation) of the knapsack problem. So the correct mapping of the knapsack problem leads the network to solve the LP-relaxation accurately[2], though a valid 0-1 solution is not found.

4.3 Annealing techniques

It is often the case that some sort of annealing technique is used to free $\mathbf{v}$ from local minima of E^{op} and force $\mathbf{v}$ towards a hypercube corner. For our simple knapsack example, if $\mathbf{v}$ is to represent a valid solution, we require convergence to one of B, C or D, the 0-1 points in P. We shall start by considering the effect of hysteretic annealing [41] (variants have been dubbed convex relaxation [112] and matrix graduated non-convexity [5]), which can be used in conjunction with any of the optimization networks. In its most generally applicable form[3], hysteretic annealing involves adding a term

$$E^{\mathrm{ann}} = -\tfrac{1}{2}\gamma \sum_{i=1}^{N} (v_i - 0.5)^2 \tag{4.9}$$

to the objective function E^{op}. The function E^{ann} is either convex or concave, depending on the sign of γ, and has full spherical symmetry around the point $\mathbf{v}^T = [0.5\ 0.5\ \dots\ 0.5]$. A consequence is that E^{ann} has the same value at all 0-1 points, and therefore does not invalidate the objective for 0-1 programming problems[4]. The annealing process is usually

[2] Indeed, an optimization network correctly set up using the mapping in Table 3.1 will reliably solve linear programming problems, without needing any of the modifications proposed in [33, 99, 135, 144].

[3] In Section 5.3 we shall examine the original form of hysteretic annealing, as presented in [41], and point out its shortcomings. The expression for E^{ann} considered here is for a modified hysteretic annealing scheme with more general applicability.

[4] Note that E^{ann} depends only on the v variables and not on any slack variables. Since slack variables may take any value at a valid solution point, summing over the slack variables as well would invalidate the objective. For two-dimensional problems like the one in Figure 4.1, contours of E^{ann} would appear in P as concentric circles around the point (0.5, 0.5). For a slack variable mapping these would become concentric cylinders in the unit cube, aligned along the w-axis: their intersection with P^+ would be a set of ellipses.

initialized with a large negative value of γ, in which case E^{op} is convex and $\mathbf{v}$ converges to a point within the interior of P. Subsequently the value of γ is gradually increased, eventually E^{op} becomes concave, and $\mathbf{v}$ is driven towards the boundary of P. For quadratic objectives this process allows $\mathbf{v}$ to escape from local minima of E^{op} and can help guide $\mathbf{v}$ towards good solutions.

Applied to the knapsack problem in Figure 4.1, however, hysteretic annealing has no useful effect. Vertex A is also a local minimum of E^{ann} within P, so no amount of increasing γ will drive $\mathbf{v}$ away from A, at least until the magnitude of γ becomes comparable with that of c in (3.28) and (3.29), at which point $\mathbf{v}$ will escape from P and converge to the point $[1.0\ 1.0]^T$, though this is clearly undesirable. Negative values of γ will succeed in freeing $\mathbf{v}$ from the vertex A, though only to guide $\mathbf{v}$ back towards the interior of P, where it will stabilize: this, too, is clearly not very useful.

We might well ask whether any other form of annealing would be more successful. For example, it seems feasible that using temperature annealing with a Hopfield network (increasing the gain of the neuron transfer functions (2.4) by reducing T^p) would eventually force the outputs v_i to either 0 or 1. However, even with very high gain transfer functions, we are still operating on the Liapunov function E (2.7). Taking the knapsack problem in Figure 4.1 as an example, any direction away from the vertex A is 'uphill' in the sense of E. Since $\mathbf{v}$ evolves in such a way that E is non-increasing, it is clear that $\mathbf{v}$ will not move from this vertex, however steep the transfer functions become. As a means of forcing convergence to a 0-1 point, varying the neuron gains is therefore ineffective. The same argument can also be applied to MFA, which, in the low temperature limit, seeks the same solution points as a Hopfield network running with high gain transfer functions and $\eta = 1$. Thus we would expect that if the MFA algorithm converges at all, it will also converge to the vertex A.

The only other form of annealing to consider is MFA with neuron normalization, as proposed for the travelling salesman and graph partitioning problems in [118, 143]. In this technique constraints of the form

$$\sum_{i \in \mathcal{S}_j} v_i = 1, \qquad j \in \{1 \ldots N_{\mathcal{S}}\} \tag{4.10}$$

are implicitly enforced using update rules

$$v_i|_{t+1} = \frac{\exp(-\left.\frac{\partial E}{\partial v_i}\right|_t / T^p)}{\sum_{k \in \mathcal{S}_j} \exp(-\left.\frac{\partial E}{\partial v_k}\right|_t / T^p)} \qquad \text{for all } i \in \mathcal{S}_j \tag{4.11}$$

where $-\frac{\partial E}{\partial v_k} = [\mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}}]_k$. The sets $\mathcal{S}_j$ must be non-intersecting and their union must cover all the elements of $\mathbf{v}$. As the annealing progresses, the temperature T^p is gradually lowered until, when fully converged, $\mathbf{v}$ represents a set of 0-1 solutions to the constraint equations (4.10). The annealing prevents $\mathbf{v}$ from getting trapped in local minima of E^{op} and forces convergence to a 0-1 point which satisfies (4.10). However, the general problem of finding a 0-1 solution to the equation

$$\mathbf{a}^T \mathbf{v} = \beta \qquad (\mathbf{a} \text{ being a rational vector and } \beta \text{ a rational scalar}) \tag{4.12}$$

is $\mathcal{NP}$-complete [123], and so we would not expect to be able to find update rules like (4.11) to enforce more general linear constraints of the form (4.12)[5]. We therefore conclude that

[5]Specifically, update rules which guarantee convergence to a 0-1 point satisfying the problem's constraints can only be found if the constraints define an *integral polytope* — see Section 4.4.

this version of MFA is not relevant to our discussion of problems with general linear constraints.

4.4 Polytopes and $\mathcal{NP}$ theory

There is a strong connection between polytopes, the convergence of optimization networks and $\mathcal{NP}$ theory. In this section we shall see that the ability of an optimization network to perform certain tasks is well reflected in the complexity of the tasks.

Returning to the knapsack problem, it is not surprising that the network failed to find the optimal 0-1 solution, since doing so is an $\mathcal{NP}$-complete problem. The network did however succeed in finding the optimal solution to the LP-relaxation of the knapsack problem. This is a standard linear programming problem, involving the minimization of a linear objective subject to linear constraints but with no integrality condition. Linear programming can be performed in polynomial time using Khachiyan's method [123], although in practice the (non-polynomial) simplex method is usually more efficient [32, 123]. It is the integrality condition (4.4) which makes the knapsack problem $\mathcal{NP}$-complete. Indeed, while all linear programming problems can be solved in polynomial time, the general *integer* linear programming problem is $\mathcal{NP}$-complete [123]. So the failure of the network to solve the knapsack problem is fully in line with what we would expect from $\mathcal{NP}$ theory.

More surprising than the network's failure to find the *optimal* solution, was its failure to find *any* 0-1 point within P, converging as it did to the non-integral vertex A. We also concluded that no annealing procedure was likely to alleviate this problem. However, this too is in full agreement with $\mathcal{NP}$-theory. It is proved in [123] that the following problem is $\mathcal{NP}$-complete:

> Given rational matrix $\mathbf{A}$ and rational vector $\mathbf{b}$, does $\mathbf{Av} \leq \mathbf{b}$ have an integral solution $\mathbf{v}$?

Furthermore, in [47, 120] it is stated that the variant in which all the components of $\mathbf{v}$ are required to belong to $\{0, 1\}$ is also $\mathcal{NP}$-complete, even if all the components of $\mathbf{A}$ and $\mathbf{b}$ are in {0,1} as well. Now, the system $\mathbf{Av} \leq \mathbf{b}$ describes a set of linear constraints like the ones found in quadratic 0-1 programming problems[6]. So if an optimization network could reliably find a 0-1 point satisfying the problem's constraints, then it would effectively be solving the above $\mathcal{NP}$-complete problem in polynomial time. The observed failure of the network to do this is therefore not surprising.

Since finding any 0-1 point within P is typically $\mathcal{NP}$-complete, we should not look too hard for an annealing procedure which does this. The best we can expect from annealing is to force convergence to some vertex of P. This is in fact what hysteretic annealing does, since increasing γ in (4.9) will eventually make E^{op} concave. It is easy to prove (and is intuitively obvious) that if E^{op} is linear or concave, then convergence to some vertex of P is guaranteed, except in highly pathological cases where the path of descent is exactly orthogonal to a face of P. However, for 0-1 programming problems we hope that $\mathbf{v}$ will converge to a 0-1 point, which will be an *integral* vertex of P. This can only be guaranteed if *all* the vertices of P are 0-1 points, that is if *all* the vertices of P are integral: a polytope exhibiting this property is called an *integral polytope* [123].

[6] In our earlier formulation, the problem's constraints (3.2)–(3.4) also featured a set of equality constraints $\mathbf{A}^{\mathrm{eq}}\mathbf{v} = \mathbf{b}^{\mathrm{eq}}$. However, these can be rewritten as two sets of inequality constraints $\mathbf{A}^{\mathrm{eq}}\mathbf{v} \leq \mathbf{b}^{\mathrm{eq}}$ and $-\mathbf{A}^{\mathrm{eq}}\mathbf{v} \leq -\mathbf{b}^{\mathrm{eq}}$, leading to an overall constraint set of the form $\mathbf{A}\,\mathbf{v} \leq \mathbf{b}$.

4.5 The significance of integral polytopes

Figure 4.2 illustrates the difference between integral and non-integral polytopes. Integral polytopes are very important in the field of mathematical programming. For example, it follows that integer linear programming over integral polytopes is not $\mathcal{NP}$-complete, since the optimal solution to the LP-relaxation, which can be found in polynomial time using Khachiyan's method [123], will necessarily be a vertex of P and therefore an integral point. Likewise, we now see that integral polytopes are highly relevant to optimization networks, which will reliably converge to a valid solution point only if the optimization is over an integral polytope. Optimization networks, in their conventional form, are quite unsuited to the solution of 0-1 programming problems over non-integral polytopes, since there is no guarantee that a valid solution point can be found, let alone the optimal one.

Given that integral polytopes are the exception and not the rule, it would be dangerous to assume the integrality of a polytope associated with any particular problem mapping. Unfortunately, the identification of integral polytopes is itself an $\mathcal{NP}$-complete problem [115]. However, it *is* possible to recognize some special classes of integrality, covering some of the polytopes implicitly studied by the optimization network community. In particular, the polytope Ω^n, associated with the mapping of the travelling salesman problem [71, etc], is indeed integral. This result can be proved using standard linear algebra techniques [23, 101, 147], though the proof is rather complicated. More recently a large set of results has been developed for recognizing special cases of integrality, and it is now envisaged that any investigation of integrality would work from these results and not start from first principles. As an example, we present here some of these results and use them to prove the integrality of Ω^n.

4.6 The polytope Ω^n

The polytope Ω^n, which arises from the constraints of the travelling salesman problem, is defined by the following equations:

$$V_{ij} \geq 0 \tag{4.13}$$

$$\sum_{i=1}^{n} V_{ij} = 1 \tag{4.14}$$

$$\sum_{j=1}^{n} V_{ij} = 1 \tag{4.15}$$

where $i, j \in \{1 \ldots n\}$. Note that the combination of (4.13)–(4.15) makes the usual limiting constraints $V_{ij} \leq 1$ redundant. If we add the condition that $\mathbf{V}$ is integral, we obtain the same constraints that we considered in Chapter 3 for the travelling salesman problem. It is clear that Ω^n contains all the doubly stochastic matrices[7] and has the permutation matrices at (at least) some of its vertices. Various features of Ω^n have been well studied in [26, 27, 28]; here we shall restrict ourselves to proving its integrality, that is proving that *all* its vertices are 0-1 points.

We start with the following results from mathematical programming theory, all of which can be found in [123]:

[7] A doubly stochastic matrix is a matrix of positive real numbers with row and column sums equal to one.

Result 1 Let $\mathbf{A}$ be a totally unimodular matrix and let $\mathbf{b}$ be an integral vector. Then the polyhedron $P = \{\mathbf{v} \mid \mathbf{A}\mathbf{v} \le \mathbf{b}\}$ is integral.

Result 2 Total unimodularity is preserved under the following operations: (i) repeating a row or column; (ii) multiplying a row or column by -1; (iii) adding a row or column with one nonzero element, that element being ± 1.

Result 3 (Hoffman and Kruskal's theorem) Let $G = (V, E)$ be an undirected graph, and let $\mathbf{M}$ be the $V \times E$ incidence matrix of G (ie. M is the $\{0,1\}$-matrix with rows and columns indexed by the vertices and edges of G, respectively, where $M_{v,e} = 1$ if and only if $v \in e$). Then M is totally unimodular if and only if G is bipartite. So $\mathbf{M}$ is totally unimodular if and only if the rows of $\mathbf{M}$ can be split into two classes so that each column contains a 1 in each of these classes.

We are going to prove the integrality of Ω^n by first proving that Ω^n can be described in the form $\Omega^n = \{\mathbf{v} \mid \mathbf{A}\mathbf{v} \le \mathbf{b}\}$ where $\mathbf{A}$ is totally unimodular. In [123] it is proved that any given matrix can be tested for total unimodularity in polynomial time, so total unimodular matrices give rise to one of the special classes of integral polytopes which can be easily recognized.

To apply the above results we must first rewrite the constraint system (4.13)–(4.15) in terms of the vector equivalent of $\mathbf{V}$. Following the approach we took in Chapter 3, we obtain

$$\mathbf{v} = \mathrm{vec}(\mathbf{V}) \tag{4.16}$$

$$v_i \ge 0 \quad i \in \{1 \dots n^2\} \tag{4.17}$$

$$\sum_{i=1}^{n} v_{n(j-1)+i} = 1 \quad j \in \{1 \dots n\} \tag{4.18}$$

$$\sum_{j=1}^{n} v_{n(j-1)+i} = 1 \quad i \in \{1 \dots n\} \tag{4.19}$$

In matrix form we can write (4.18)–(4.19) as $\mathbf{A}^{\mathrm{eq}}\mathbf{v} = \mathbf{b}^{\mathrm{eq}}$, where $b_i^{\mathrm{eq}} = 1$ for all i. Taking the case $n = 3$ as an example, $\mathbf{A}^{\mathrm{eq}}$ would be given as follows:

$$\mathbf{A}^{\mathrm{eq}} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

We see that the rows of $\mathbf{A}^{\mathrm{eq}}$ can be split into two classes, rows 1–3 and rows 4–6, corresponding to the constraints (4.18) and (4.19) respectively. Furthermore, the columns of $\mathbf{A}^{\mathrm{eq}}$ contain a 1 in each of these classes. Thus, by Result 3 above, $\mathbf{A}^{\mathrm{eq}}$ is totally unimodular. It is clear that this holds for any problem size n.

The constraints (4.17) can be written in matrix form as $\mathbf{A}^{\mathrm{in}}\mathbf{v} \le \mathbf{b}^{\mathrm{in}}$, where $\mathbf{A}^{\mathrm{in}} = -\mathbf{I}$ and $b_i^{\mathrm{in}} = 0$ for all i. Rewriting the equality constraints $\mathbf{A}^{\mathrm{eq}}\mathbf{v} = \mathbf{b}^{\mathrm{eq}}$ as two sets of inequality constraints $\mathbf{A}^{\mathrm{eq}}\mathbf{v} \le \mathbf{b}^{\mathrm{eq}}$ and $-\mathbf{A}^{\mathrm{eq}}\mathbf{v} \le -\mathbf{b}^{\mathrm{eq}}$, the entire constraint system (4.17)–(4.19) becomes $\mathbf{A}\mathbf{v} \le \mathbf{b}$, where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^{\mathrm{in}} \\ \mathbf{A}^{\mathrm{eq}} \\ -\mathbf{A}^{\mathrm{eq}} \end{bmatrix} \tag{4.21}$$

and

$$\mathbf{b} = \begin{bmatrix} \mathbf{b}^{\mathrm{in}} \\ \mathbf{b}^{\mathrm{eq}} \\ -\mathbf{b}^{\mathrm{eq}} \end{bmatrix} \tag{4.22}$$

We can now use Result 2 to prove the total unimodularity of $\mathbf{A}$. Starting with $\mathbf{A}^{\mathrm{eq}}$, which we have already proved to be totally unimodular, we use Results 2(i) and 2(ii) to first repeat the rows of $\mathbf{A}^{\mathrm{eq}}$ and then negate the new set of rows, creating the lower two-thirds of the matrix $\mathbf{A}$ (4.21) and preserving total unimodularity. Finally, we complete $\mathbf{A}$ by adding the matrix $\mathbf{A}^{\mathrm{in}}$, which has rows with only a single nonzero in them, that being -1; by Result 2(iii) $\mathbf{A}$ is totally unimodular. Since $\mathbf{b}$ is integral, we can now use Result 1 to conclude that the polytope $\Omega^n = \{\mathbf{v} \mid \mathbf{A}\mathbf{v} \leq \mathbf{b}\}$, equivalent to (4.17)–(4.19), is integral.

So, for the travelling salesman problem and other problems over Ω^n, we can be sure that an optimization network, running with a suitable annealing procedure, will converge to a valid solution point. If the objective function of the travelling salesman problem had been linear, then this point would also represent the optimal solution. However, the objective function is quadratic, so convergence to the optimal solution point cannot be guaranteed: this is consistent with the $\mathcal{NP}$-completeness of the travelling salesman problem.

This approach to the travelling salesman problem, traditionally pursued by the optimization network community, should be contrasted with the mappings studied by the mathematical programming community, where a linear objective over a non-integral polytope is considered [91, 123, etc]. Unfortunately, to map the travelling salesman problem in this manner requires either a number of constraints which scales exponentially with the size of the problem, or a large number of supplementary variables with a complicated constraint set. So while such mappings provide a very convenient basis for theoretical studies, they are not well suited for use with optimization networks. Throughout the rest of this thesis we shall consider only the mapping of the travelling salesman problem where we optimize a quadratic objective over Ω^n.

4.7 Polytopes and optimization strategy

It is now necessary to reasses our use of optimization networks in the light of the polyhedral issues. To do this, we must ask ourselves exactly what we aim to achieve with such networks, and at what cost. Looking at the summary in Table 4.1, it is clear that the complexity of the problem and the suggested network solution technique are determined more by the nature of the polytope than by the order of the objective function.

Starting with integral polytopes, we have seen that an optimization network operating under a good mapping is guaranteed to find a valid solution point every time. Further aspects of network performance will depend on the nature of the problem's objective function. If E^{op} is linear, then the network will reliably find the optimal solution to the problem, which is not $\mathcal{NP}$-complete: an example of such a problem is one-to-one linear assignment, as studied in [41]. If E^{op} is quadratic, then the problem is $\mathcal{NP}$-complete and there is no guarantee that the network will find the optimal solution. Problems falling into this category include the travelling salesman, graph partitioning and graph labelling problems. However, a valid solution will be found, and given the nature of the descent process leading to this solution, there is a reasonable chance that the solution will be quite good. An annealing procedure might further enhance the network's performance, resulting in the sensible compromise we are seeking between solution quality and search time. We shall closely examine the network's performance on such problems in Chapters 5 and 6.

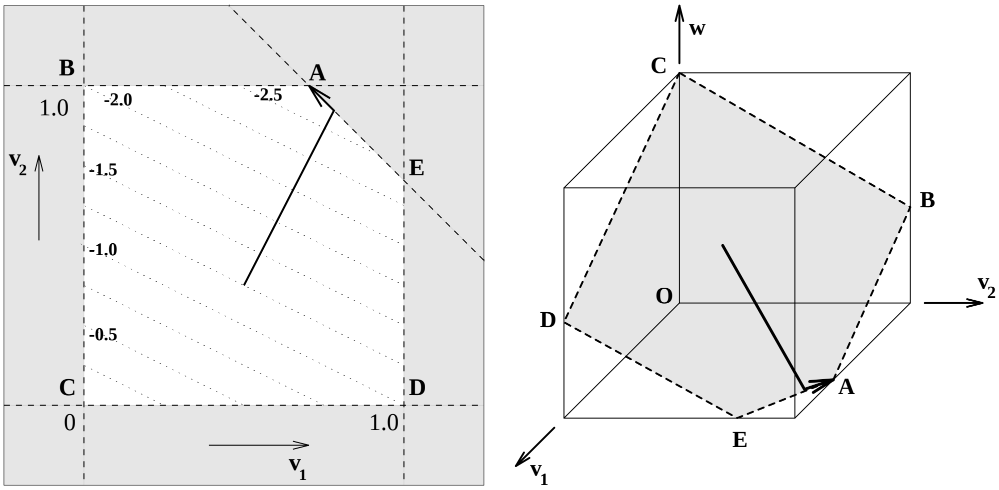
Turning now to optimization over non-integral polytopes, we see a very different picture. Convergence to a 0-1 point cannot be guaranteed, and we must ask how useful optimization networks are in these situations. We shall restrict the discussion to cases where the objective is linear or altogether absent (the latter being the case for pure constraint satisfaction problems, where we desire to find *any* 0-1 point within P), since, thankfully, no useful problems appear to require the minimization of a quadratic objective over a non-integral polytope. For linear objectives, an optimization network will converge to the optimal LP-relaxation solution. In other words optimization networks, running under a good mapping, can perform linear programming: this is of considerable value on its own. At the very least, the LP-relaxation solution can be used to place a lower bound on the optimal 0-1 solution. Moreover, it might be the case that a simple heuristic can be applied to the LP-relaxation solution to obtain a good 0-1 solution: in Chapter 7 we examine one such heuristic for the knapsack problem. Failing this, if we require a valid 0-1 solution at all costs, we must resign ourselves to exponential-time techniques, since finding such a point generally constitutes an $\mathcal{NP}$-complete problem. In Chapter 7 we present modified network dynamics which perform a *search* of the constraint polytope's vertices, as opposed to a one-shot descent within the polytope. While this system has its drawbacks, it at least embodies a coherent approach to optimization over non-integral polytopes, paying all due attention to the polyhedral nature of the problem.

Polytope	Integral		
Objective	Quadratic	Linear	None
Complexity	$\mathcal{NP}$-complete	$\mathcal{P}$-solvable	$\mathcal{P}$-solvable
Network	Annealing	Simple descent	Simple descent
Examples	TSP [71, etc] GPP [117, 118, 143]	One-to-one assignment [41]	Crossbar switching [134]

Polytope	Non-integral		
Objective	Quadratic	Linear	None
Complexity	At least $\mathcal{NP}$-complete	$\mathcal{NP}$-complete	$\mathcal{NP}$-complete
Network	Vertex search	Vertex search	Vertex search
Examples		TSP [91, 123, etc] Knapsack [68]	'Teachers and classes' [58]

Table 4.1: Classification of quadratic 0-1 programming problems.

The complexity of the problem and the suggested network solution technique are determined more by the nature of the polytope than by the order of the objective function. For pure constraint satisfaction problems over integral polytopes, where there is no objective function, simple descent on any *linear or concave function will find a valid solution point. Abbrevations used in this table: "TSP" stands for the travelling salesman problem, "GPP" for the graph partitioning problem.*



Trajectory of $\mathbf{v}$ in P Trajectory of $\mathbf{v}^+$ in P^+

Figure 4.1: Network convergence for a simple knapsack problem.

The left hand figure shows the polytope P for a simple 2-dimensional knapsack problem, along with contours of E^{op} (shown in dotted lines). For this linear objective function, any continuous descent procedure converges to the vertex A*. The corresponding process for a slack variable mapping is illustrated in the right hand figure.*

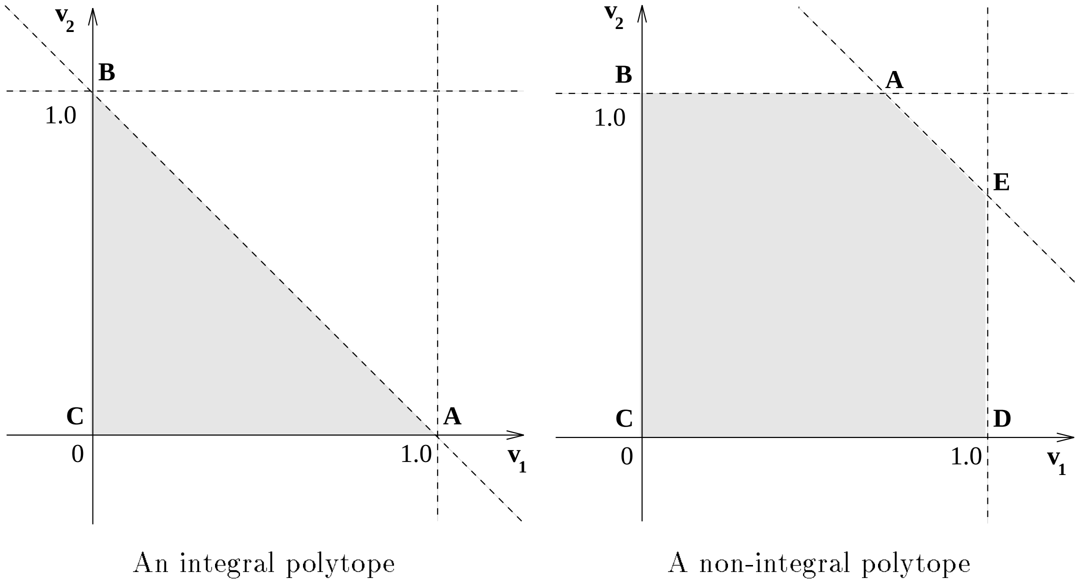


Figure 4.2: Integral and non-integral polytopes within the unit square.

An integral *polytope has all its vertices at integral points (points whose coordinates are all integers). It follows that an integral polytope within the unit hypercube has all its vertices at 0-1 points.*

Chapter 5

Optimization over Integral Polytopes

In this chapter we restrict attention to optimization over integral polytopes. This covers all problems over Ω^n, such as the travelling salesman, Hamilton path, graph labelling and assignment problems, as well as the graph partitioning problem which is not over Ω^n. For such problems we can be sure that an optimization network, running under a rigorous mapping, will succeed in converging to a valid 0-1 solution point. We can subsequently begin to assess the *quality* of network solutions, in terms of the value of the objective at the solution point. Our methodology is as follows: we attempt to glean as much information as possible about the network's dynamics, and then ask whether such dynamics are likely to lead to good solutions which minimize the objective function. The inherent nonlinearity of the dynamics makes their analysis very difficult, but we shall see that it is possible to predict the state vector's initial trajectory away from its starting position. Knowledge of this initial direction alone allows us to identify certain difficulties with the operation of optimization networks: in some cases the initial direction systematically leads the state vector away from the optimal solution point, in others the initial direction is overly sensitive to the random starting position, resulting in highly variable network performance. For problems of the latter variety, we describe how annealing procedures are of some help, since they make the network's behaviour less dependent on the precise location of the starting position.

The state vector's initial direction is identified by means of a linearized analysis of the network's dynamics, presented in Sections 5.1 and 5.2. Great care is taken to make explicit the assumptions behind the analysis, and the consequences thereof. A new result is that, for many problems, the initial direction is heavily influenced by a linear bias term, which we denote $\mathbf{i}^{\mathrm{opr}}$. The linearized dynamics can also be used to cast light on the various annealing procedures. In Section 5.3 we show how annealing makes the network's behaviour less dependent on the random starting position. The similarities between temperature and hysteretic annealing are also investigated. The main original contribution of the chapter begins in Section 5.5, where movement along $\mathbf{i}^{\mathrm{opr}}$ is associated with the solution of a certain *auxiliary linear problem*. The relationship between the auxiliary linear problem and the parent problem is shown to have considerable bearing on the network's performance. In particular, by presenting cases where the auxiliary linear problem and the parent problem are bound to have very different solutions, we show that $\mathbf{i}^{\mathrm{opr}}$ can systematically lead the state vector away from the parent problem's optimal solution.

5.1 Linearized analysis of network dynamics

In this section we present a linearized analysis of the network's dynamics, valid for small excursions of $\mathbf{v}$ from its starting point $\mathbf{v}_o$. We shall consider only problems which have no inequality constraints, so we can drop the $\mathbf{v}^+$ notation, which indicated that the network was operating on an extended variable set including slack variables. This is not a severe restriction; as we stated earlier, we are dealing exclusively with problems over integral polytopes; the most common of these, such as the problems over Ω^n and the graph partitioning problem, feature only equality constraints. Before we can start the analysis, it is necessary to make two assumptions about the starting point $\mathbf{v}_o$, since it is not possible to proceed very far otherwise. The assumptions are:

Assumption 1 $\mathbf{v}_o$ lies within a small, random displacement of a uniform vector, so $[\mathbf{v}_o]_i \approx [\mathbf{v}_o]_j$ for all i, j. We assume that such a point exists within the polytope P.

Assumption 2 The uniform vector near $\mathbf{v}_o$ is given by the vector $\mathbf{s}$ in equation (3.20).

Let us briefly consider the implications of these assumptions. Assumption 1 states that we initialize the network near a point which does not bias any subsequent network behaviour, since all the elements of $\mathbf{v}$ have approximately the same value. This would be a clear first choice for a starting position, though we have to assume that such a point exists within P, which is by no means guaranteed. Assumption 2 says that the vector $\mathbf{s}$ in equation (3.20), used to define the valid subspace for the rigorous problem mapping, provides us with a suitable uniform vector within P. Looking at equation (3.40), we see that this is indeed the case when operating over Ω^n; furthermore, this also holds for the graph partitioning problem [5]. In fact Assumptions 1 and 2 are hardly restrictive, since they are valid for all the common problems over integral polytopes listed in the introduction to this chapter.

Using Assumption 1 we can linearize the networks' dynamics by writing (see Figure 5.1)

$$v_i = (\phi/T^p)u_i + \theta \quad \text{and} \quad \dot{v}_i = (\phi/T^p)\dot{u}_i$$

Without Assumption 1 the ϕ's and θ's would have been different for each v_i, leading to a complicated and unrevealing analysis. Linearization of the Hopfield network's dynamics (2.2)–(2.3) gives

$$\frac{T^p\dot{\mathbf{v}}}{\phi} = \mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}} - \eta\left[\frac{T^p(\mathbf{v} - \theta\mathbf{o})}{\phi}\right] \tag{5.1}$$

where $\mathbf{o}$ is a vector of ones, that is $o_i = 1$ for all i. Equation (5.1) is valid for small excursions of $\mathbf{v}$ from the starting position. We can also use equation (5.1) with $\eta = 0$ and $\phi/T^p = 1$ to model the steepest descent dynamics (2.10), valid until $\mathbf{v}$ approaches a hypercube face. In addition, an Euler simulation of (5.1) with $\eta = 1$ and a time-step $\Delta t = 1$ approximates the MFA algorithm, again valid for small excursions of $\mathbf{v}$ from the starting position.

Under the influence of a penalty function of the form (3.25), $\mathbf{v}$ will converge rapidly to the valid subspace, and then stay there indefinitely. The transient behaviour as $\mathbf{v}$ moves towards the valid subspace is of little interest; furthermore, $\mathbf{v}$ is usually initialized *on* the valid subspace. Far more significant are the dynamics of $\mathbf{v}$ *in* the valid subspace. If we write $\mathbf{v} = \mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s} \equiv \mathbf{v}^{\mathrm{val}} + \mathbf{s}$, then we can extract $\dot{\mathbf{v}}^{\mathrm{val}}$, the component of $\dot{\mathbf{v}}$ in the valid subspace, by multiplying both sides of equation (5.1) by $\mathbf{T}^{\mathrm{val}}$. Doing this, and substituting the expressions for $\mathbf{T}$ (3.28) and $\mathbf{i}^{\mathrm{b}}$ (3.29) for a rigorous problem mapping, we obtain

$$\dot{\mathbf{v}}^{\mathrm{val}} = \left(\frac{\phi}{T^p}\mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}} - \eta\mathbf{I}\right)\mathbf{v}^{\mathrm{val}} + \frac{\phi}{T^p}\mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}}\mathbf{s} + \mathbf{i}^{\mathrm{op}}) \tag{5.2}$$

where we have made use of Assumption 2 to state that $\mathbf{T}^{\rm val}\mathbf{o} \propto \mathbf{T}^{\rm val}\mathbf{s} = \mathbf{0}$. We can simplify (5.2) by defining the following quantities

$$\mathbf{T}^{\rm opr} \equiv \mathbf{T}^{\rm val}\mathbf{T}^{\rm op}\mathbf{T}^{\rm val} \tag{5.3}$$

$$\mathbf{i}^{\rm opr} \equiv \mathbf{T}^{\rm val}(\mathbf{T}^{\rm op}\mathbf{s} + \mathbf{i}^{\rm op}) \tag{5.4}$$

which gives

$$\dot{\mathbf{v}}^{\rm val} = \left(\frac{\phi}{T^p}\mathbf{T}^{\rm opr} - \eta\mathbf{I}\right)\mathbf{v}^{\rm val} + \frac{\phi}{T^p}\mathbf{i}^{\rm opr} \tag{5.5}$$

Suppose $\mathbf{T}^{\rm opr}$ has eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_N$ with unit eigenvectors $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^N$.[1] We also assume that the eigenvalues are ordered so that $\lambda_1 \le \lambda_2 \le \ldots \le \lambda_N$. Let $\mathbf{v}^{\rm val}$, $\mathbf{v}^{\rm val}_{\rm o}$ and $\mathbf{i}^{\rm opr}$ be decomposed along $\mathbf{T}^{\rm opr}$'s eigenvectors as follows:

$$\mathbf{v}^{\rm val} = \sum_{i=1}^{N} \alpha_i \mathbf{x}^i \tag{5.8}$$

$$\mathbf{v}^{\rm val}_{\rm o} = \sum_{i=1}^{N} \alpha_i^{\rm o} \mathbf{x}^i \tag{5.9}$$

$$\mathbf{i}^{\rm opr} = \sum_{i=1}^{N} \beta_i \mathbf{x}^i \tag{5.10}$$

It is possible to solve the dynamic equation (5.5), expressing $\mathbf{v}^{\rm val}(t)$ in the eigenvector basis of $\mathbf{T}^{\rm opr}$. The detailed derivation can be found in Appendix B.1, in which we conclude that

$$\alpha_i(t) = \begin{cases} \alpha_i^{\rm o} + \dfrac{\phi}{T^p}\beta_i t & \text{for } \tilde{\lambda}_i = 0 \\ \left(\alpha_i^{\rm o} + \dfrac{\phi\beta_i}{T^p\tilde{\lambda}_i}\right)\exp(\tilde{\lambda}_i t) - \dfrac{\phi\beta_i}{T^p\tilde{\lambda}_i} & \text{for } \tilde{\lambda}_i \neq 0 \end{cases} \tag{5.11}$$

where $\tilde{\lambda}_i \equiv (\phi\lambda_i/T^p - \eta)$.

5.2 The initial direction of v

We now proceed to use the linearized analysis to investigate the direction in which $\mathbf{v}$ initially moves. It will become apparent in Section 5.4 that the initial behaviour of the network is a critical factor in its overall performance. To simplify the analysis, we consider the cases $\mathbf{i}^{\rm opr} = \mathbf{0}$ and $\mathbf{i}^{\rm opr} \neq \mathbf{0}$ separately.

---

[1] For later use, we shall distinguish between two types of eigenvector of $\mathbf{T}^{\rm opr}$, namely those lying in the range of $\mathbf{T}^{\rm val}$ and those in its nullspace. Defining the set $\mathcal{Z}$ such that $i \in \mathcal{Z}$ if $\mathbf{x}^i$ is in the nullspace of $\mathbf{T}^{\rm val}$, we see that

$$\mathbf{T}^{\rm opr}\mathbf{x}^i = \mathbf{T}^{\rm val}\mathbf{T}^{\rm op}\mathbf{T}^{\rm val}\mathbf{x}^i = \mathbf{0} \quad \text{for } i \in \mathcal{Z} \tag{5.6}$$

So, for $i \in \mathcal{Z}$, $\mathbf{x}^i$ is an eigenvector of $\mathbf{T}^{\rm opr}$ with a corresponding eigenvalue $\lambda_i = 0$. Since the $\mathbf{x}^i$'s form a mutually orthogonal set, it follows that all the other eigenvectors lie in the range of $\mathbf{T}^{\rm val}$. Furthermore, since $\mathbf{T}^{\rm val}$ is a projection matrix, we can state that

$$\mathbf{T}^{\rm val}\mathbf{x}^i = \mathbf{x}^i \quad \text{for } i \notin \mathcal{Z} \tag{5.7}$$

It is the eigenvectors and eigenvalues with $i \notin \mathcal{Z}$ which are of relevance to the network dynamics, since $\dot{\mathbf{v}}^{\rm val}$ lies wholly in the range of $\mathbf{T}^{\rm val}$. The components of $\mathbf{v}$ in the nullspace of $\mathbf{T}^{\rm val}$ are fixed by confinement to the polytope P.

$\mathbf{i}^{\text{opr}} = \mathbf{0}$

It transpires that $\mathbf{i}^{\text{opr}} = \mathbf{0}$ for many common problems, including the travelling salesman problem (see Appendix B.2). This means that E^{op} has a stationary point very near to where $\mathbf{v}$ is initialized[2]. In this case the linearized dynamics (5.11) become

$$\alpha_i(t) = \alpha_i^{\text{o}} \exp(\tilde{\lambda}_i t) \quad \text{for all } \tilde{\lambda}_i \tag{5.12}$$

Looking at equation (5.12), it is apparent that only those α_i with corresponding positive $\tilde{\lambda}_i$ are going to increase in magnitude. This is equivalent to saying that $\mathbf{v}$ moves only along those eigenvectors of $\mathbf{T}^{\text{opr}}$ for which $(\phi\lambda_i)/T^p > \eta$, with the α_i's corresponding to the most positive λ_i's dominating the dynamics. Indeed, if it were not for the network's nonlinear transfer functions, which make the result (5.12) invalid once $\mathbf{v}$ has moved significantly far from its starting position, it is clear that $\mathbf{v}^{\text{val}}$ will eventually align itself with $\mathbf{x}^{\max}$, the eigenvector of $\mathbf{T}^{\text{opr}}$ with the largest positive eigenvalue.

Assumptions 1 and 2 tell us that $\mathbf{v}_{\text{o}}$ lies a small, random displacement away from $\mathbf{s}$. In other words $\mathbf{v}_{\text{o}}^{\text{val}} \approx \mathbf{0}$ or, in terms of the eigenvector decomposition (5.9), $\alpha_i^{\text{o}} \approx 0$ for all i. Equation (5.12) indicates that the direction in which $\mathbf{v}$ moves along any eigenvector is governed solely by the sign of the corresponding α_i^{o}, which is random. Viewed geometrically, this reflects the fact that $\mathbf{v}$ can fall off the stationary point in one of many directions, depending on the random starting vector $\mathbf{v}_{\text{o}}^{\text{val}}$. It seems likely that $\mathbf{v}$ would subsequently converge to one of many 0-1 points, spread widely around the boundary of P. The success of a network in solving such problems therefore depends on somehow being able to exert more control on the initial direction of $\mathbf{v}$. In Section 5.3 we show that the various annealing techniques offer us exactly this sort of control.

$\mathbf{i}^{\text{opr}} \neq \mathbf{0}$

We turn now to the case $\mathbf{i}^{\text{opr}} \neq \mathbf{0}$, so there is no stationary point near the starting position of $\mathbf{v}$. Since Assumption 2 implies that $\mathbf{v}_{\text{o}} \approx \mathbf{s}$, it follows that $\mathbf{v}_{\text{o}}^{\text{val}} \approx \mathbf{0}$ and so all the α_i^{o}'s are small. If we therefore neglect the α_i^{o}'s in equation (5.11), and substitute $(1 + \tilde{\lambda}_i t)$ for $\exp(\tilde{\lambda}_i t)$ (valid for small t), we obtain

$$\alpha_i(t) = \frac{\phi}{T^p}\beta_i t \quad \Leftrightarrow \quad \mathbf{v}^{\text{val}}(t) = \frac{\phi}{T^p}\mathbf{i}^{\text{opr}} t \tag{5.13}$$

It follows that $\mathbf{i}^{\text{opr}}$ is largely responsible for the initial direction of $\mathbf{v}$. After a short while, however, the dynamics become heavily influenced by the dominant eigenvector $\mathbf{x}^{\max}$.

Since $\mathbf{v}^{\text{val}}$ initially moves in the direction of $\mathbf{i}^{\text{opr}}$, it seems more likely that the network might converge to a unique point, irrespective of the starting position. However, this is not the case if $\mathbf{i}^{\text{opr}}$ is orthogonal to some of the more important[3] eigenvectors of $\mathbf{T}^{\text{opr}}$. For example, consider what happens if $\mathbf{i}^{\text{opr}}$ is orthogonal to the dominant eigenvector $\mathbf{x}^{\max}$.

---

[2]To see this, we substitute $(\mathbf{T}^{\text{val}}\mathbf{v} + \mathbf{s})$ for $\mathbf{v}$ in the expression for E^{op} (3.1), to obtain

$$\begin{aligned} E^{\text{op}} &= -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{\text{opr}}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\text{opr}} + \text{terms independent of } \mathbf{v} \\ &= -\tfrac{1}{2}\mathbf{v}^{\text{val}^T}\mathbf{T}^{\text{opr}}\mathbf{v}^{\text{val}} - \mathbf{v}^{\text{val}^T}\mathbf{i}^{\text{opr}} + \text{terms independent of } \mathbf{v}^{\text{val}} \end{aligned}$$

Hence $\nabla E^{\text{op}} = -\mathbf{T}^{\text{opr}}\mathbf{v}^{\text{val}}$ for $\mathbf{i}^{\text{opr}} = \mathbf{0}$, and so there is a stationary point at $\mathbf{v}^{\text{val}} = \mathbf{0}$ (ie. at $\mathbf{v} = \mathbf{s}$), which, by Assumption 2, is near the starting position.

[3]In the sense that they have corresponding large, positive eigenvalues.

Under these conditions, the direction in which $\mathbf{v}^{\text{val}}$ moves along $\mathbf{x}^{\text{max}}$ is still dependent on the random starting position, and we might expect $\mathbf{v}$ to converge to one of two solution points, depending on which way $\mathbf{v}^{\text{val}}$ moves along $\mathbf{x}^{\text{max}}$. Viewed geometrically, the starting position lies on a sloping ridge which runs along $\mathbf{i}^{\text{opr}}$ and drops off on either side in the directions $\pm\mathbf{x}^{\text{max}}$. $\mathbf{i}^{\text{opr}}$ leads $\mathbf{v}$ downhill along the ridge, but $\mathbf{v}$ is free to fall off either side of the ridge, depending on which side the random starting position lies.

Such a situation arises with the Hamilton path problem, where we desire to find the shortest path passing through n cities. The Hamilton path problem is very similar to the travelling salesman problem, except that no closed tour is specified. There are therefore two optimal solutions, corresponding to traversing the shortest path in opposite directions. In Appendix B.3 we prove that $\mathbf{i}^{\text{opr}}$ is orthogonal to $\mathbf{x}^{\text{max}}$ for the Hamilton path problem. Figure 5.2 shows the decomposition of $\mathbf{v}^{\text{val}}$ in the early stages of convergence, as a steepest descent network attempts to solve a randomly generated 10-city Hamilton path problem, using the mapping described in Section 3.5. Two runs are illustrated, starting once at the point $\mathbf{v}_{\text{o}}^{\text{val}}$ and then again at $-\mathbf{v}_{\text{o}}^{\text{val}}$. As expected, in both cases $\mathbf{v}^{\text{val}}$ shows an initial tendency towards $\mathbf{i}^{\text{opr}}$, before realigning along the dominant eigenvector $\mathbf{x}^{\text{max}}$. In one run $\mathbf{v}^{\text{val}}$ moves along $+\mathbf{x}^{\text{max}}$, while in the other it moves along $-\mathbf{x}^{\text{max}}$. It transpires that the two runs converge to different solution points, corresponding to traversing the shortest path in opposite directions[4].

5.3 Annealing revisited

By investigating the effects of the various annealing techniques on the linearized dynamics, we can gain a better understanding of the benefits offered by annealing. We shall see that hysteretic and temperature annealing (as used in the MFA algorithm) have very similar effects on the initial direction of $\mathbf{v}$, though hysteretic annealing is more powerful in forcing eventual convergence to a vertex of P.

5.3.1 Hysteretic annealing

As originally proposed in [41], hysteretic annealing involves adding a variable amount of self-feedback to each element of the network. This is equivalent to adding a term of the form

$$E_{\text{o}}^{\text{ann}} = -\tfrac{1}{2}\gamma \sum_{i=1}^{N} v_i^2 = -\tfrac{1}{2}\gamma \mathbf{v}^T \mathbf{I} \mathbf{v}$$

to the objective function E^{op}. However, this will invalidate the objective function unless all feasible solution points lie the same Euclidean distance from the origin. While this is the case for all problems over Ω^n and the graph partitioning problem, we propose a modified version of hysteretic annealing with wider applicability. If instead we use a term

[4]Any of the problems studied in this chapter can be expressed in many different ways: for example, an n-city Hamilton path problem can be expressed in $n!$ ways, each one corresponding to a different ordering of the cities in the matrix $\mathbf{P}$. For the results to have any significance, it is necessary to prove that the decomposition of a particular solution's $\mathbf{v}^{\text{val}}$ along $\mathbf{i}^{\text{opr}}$ and the eigenvectors of $\mathbf{T}^{\text{opr}}$ is independent of this ordering. While this is indeed the case, the proof is rather long and tedious: interested readers should refer to [49].

of the form

$$E^{\text{ann}} = -\tfrac{1}{2}\gamma \sum_{i=1}^{N}(v_i - 0.5)^2 = -\tfrac{1}{2}\gamma \mathbf{v}^T \mathbf{I}\mathbf{v} + \tfrac{1}{2}\gamma \mathbf{v}^T \mathbf{o} + \text{a constant}$$

then E^{ann} has the same value at all 0-1 points, and does not invalidate the objective for any 0-1 programming problem. The annealing process is usually initialized with a large negative value of γ, in which case E^{op} is convex and $\mathbf{v}$ converges to a point within the interior of P. Subsequently the value of γ is gradually increased, eventually E^{op} becomes concave, and $\mathbf{v}$ is driven towards the boundary of P. As we mentioned in Chapter 4, convergence to a vertex of P is guaranteed if E^{op} is concave.

We can view hysteretic annealing as replacing the matrix $\mathbf{T}^{\text{op}}$ by $(\mathbf{T}^{\text{op}} + \gamma\mathbf{I})$, and the vector $\mathbf{i}^{\text{op}}$ by $(\mathbf{i}^{\text{op}} - \frac{1}{2}\gamma\mathbf{o})$. This has the following effects on $\mathbf{T}^{\text{opr}}$ and $\mathbf{i}^{\text{opr}}$:

$$\begin{aligned} \mathbf{T}^{\text{opr}} &\rightarrow \mathbf{T}^{\text{val}}(\mathbf{T}^{\text{op}} + \gamma\mathbf{I})\mathbf{T}^{\text{val}} \\ &= \mathbf{T}^{\text{val}}\mathbf{T}^{\text{op}}\mathbf{T}^{\text{val}} + \gamma\mathbf{T}^{\text{val}} \\ \mathbf{i}^{\text{opr}} &\rightarrow \mathbf{T}^{\text{val}}((\mathbf{T}^{\text{op}} + \gamma\mathbf{I})\mathbf{s} + \mathbf{i}^{\text{op}} - \tfrac{1}{2}\gamma\mathbf{o}) \\ &= \mathbf{T}^{\text{val}}(\mathbf{T}^{\text{op}}\mathbf{s} + \mathbf{i}^{\text{op}}) \quad \text{using Assumption 2} \end{aligned}$$

Hence $\mathbf{i}^{\text{opr}}$ is unchanged, while $\mathbf{T}^{\text{opr}}$ has the extra $\gamma\mathbf{T}^{\text{val}}$ term added to it[5]. We must investigate what effect this has on the eigenvectors and eigenvalues of $\mathbf{T}^{\text{opr}}$. It transpires that the eigenvectors of $\mathbf{T}^{\text{opr}}$ are unchanged, since, using (5.6) and (5.7)

$$\begin{aligned} (\mathbf{T}^{\text{val}}\mathbf{T}^{\text{op}}\mathbf{T}^{\text{val}} + \gamma\mathbf{T}^{\text{val}})\mathbf{x}^i &= \lambda_i\mathbf{x}^i + \gamma\mathbf{T}^{\text{val}}\mathbf{x}^i \\ &= \begin{cases} \mathbf{0} & \text{for } i \in \mathcal{Z} \\ (\lambda_i + \gamma)\mathbf{x}^i & \text{for } i \notin \mathcal{Z} \end{cases} \end{aligned} \tag{5.14}$$

The effect of the hysteretic annealing term is to shift all the eigenvalues for which $i \notin \mathcal{Z}$ (ie. those eigenvalues with eigenvectors in the range of $\mathbf{T}^{\text{val}}$) by a uniform amount γ. As we noted earlier, it is these eigenvalues, along with their corresponding eigenvectors, which affect $\mathbf{v}^{\text{val}}$.

It follows, therefore, that the action of hysteretic annealing is to modify the values of $\tilde{\lambda}_i$ as follows:

$$\tilde{\lambda}_i \rightarrow \frac{\phi}{T^p}(\lambda_i + \gamma) - \eta \quad \text{for } i \notin \mathcal{Z} \tag{5.15}$$

Let us consider what this means for the case $\mathbf{i}^{\text{opr}} = \mathbf{0}$. The annealing starts with a sufficiently negative value of γ such that all the $\tilde{\lambda}_i$ are negative; $\mathbf{v}^{\text{val}}$ therefore remains very small, since equation (5.12) tells us that $\alpha_i \rightarrow 0$ for all $\tilde{\lambda}_i < 0$. The parameter γ is then gradually increased, so that one of the $\tilde{\lambda}_i$ becomes positive, at which point the corresponding α_i is free to increase in magnitude. Thus, an effect of annealing is to control the initial direction of $\mathbf{v}$, ensuring that $\mathbf{v}^{\text{val}}$ initially moves only along the eigenvector of $\mathbf{T}^{\text{opr}}$ with the largest positive eigenvalue.

This effect is illustrated in Figure 5.3, where we see a steepest descent network (for which $\tilde{\lambda}_i = \lambda_i + \gamma$) operating on a simple, hypothetical problem. Figure 5.3(a) shows a two-dimensional integral polytope P within a three-dimensional unit cube; there are three valid solution points at the cube corners labelled A, B and C. In Figure 5.3(b) the contours

[5] We would obtain the same results for the original hysteretic annealing term $E_{\text{o}}^{\text{ann}}$, only without having to call upon Assumption 2.

of $E^{\rm op}$ have been marked within P. For this problem $\mathbf{i}^{\rm opr} = \mathbf{0}$, so $E^{\rm op}$ has a stationary point near the centre of P, where $\mathbf{v}^{\rm val} = \mathbf{0}$. The eigenvectors of $\mathbf{T}^{\rm opr}$, $\mathbf{x}^1$ and $\mathbf{x}^2$, are the axes of the conic sections which describe the contours of $E^{\rm op}$. We have assumed that both of the eigenvalues of $\mathbf{T}^{\rm opr}$, λ_1 and λ_2, are positive, so that the contours are ellipses. We have also assumed that $\lambda_1 > \lambda_2$, so the descent is steepest in the $\mathbf{x}^1$ direction. Examining the contours, it is clear that the corner C represents the unique global optimum to this problem. Looking at Figure 5.3(b), we see that, in the absence of an annealing process, any of the three corners A, B and C can be reached by descent dynamics starting from positions near the centre of P.

The effect of annealing is to make the network's behaviour less dependent on the random starting position and increase the chances of the network finding the optimal solution. Figure 5.3(c) shows the contours of $E^{\rm op}$ at the start of an annealing process, when the eigenvalues have been shifted so that only $\tilde{\lambda}_1$ is positive. The stationary point near the centre of P has been turned into a saddle point, so that moving along $\mathbf{x}^1$ reduces $E^{\rm op}$ while moving along $\mathbf{x}^2$ increases $E^{\rm op}$. The subsequent trajectory of $\mathbf{v}$ will therefore reduce the $\mathbf{x}^2$ component towards zero and increase the magnitude of the $\mathbf{x}^1$ component: such a trajectory, starting from a position $\mathbf{v}_{\rm o}^{\rm val}$, is shown in Figure 5.3(c). In Figure 5.3(d) we see that $\mathbf{v}$ continues its descent to converge, in this case, to the optimal solution point. Figure 5.3(e) shows the effect of starting from an initial position $-\mathbf{v}_{\rm o}^{\rm val}$. This time the $\mathbf{x}^1$ component is introduced with the opposite sign, leading to a suboptimal solution in Figure 5.3(f). Note that, for this second starting position, it was necessary to advance the annealing so that $\tilde{\lambda}_2 > 0$ (at which point $E^{\rm op}$ becomes concave) to force convergence to a hypercube corner.

This example illustrates the extra control offered by hysteretic annealing. With no annealing it would have been necessary to try many random starting vectors to be sure of finding the best possible solution. With annealing it was necessary to run the network only twice, starting once with $\mathbf{v}^{\rm val} = \mathbf{v}_{\rm o}^{\rm val}$, and then again with $\mathbf{v}^{\rm val} = -\mathbf{v}_{\rm o}^{\rm val}$. This may still seem a little expensive to solve a single problem, but at least it is feasible. Furthermore, for many common problems it is not necessary to run the network twice. Such problems include those with $\mathbf{i}^{\rm opr} \neq \mathbf{0}$, and those, like the travelling salesman problem, with repeated optimal solutions located at regular intervals around P.[6]

5.3.2 Temperature annealing

We have briefly mentioned temperature annealing in the context of the MFA algorithm in Chapter 2. Recall that the MFA algorithm involves solving the saddle point equations (2.11) and (2.12) at a series of progressively lower temperatures T^p. In the low temperature limit MFA seeks points which minimize E (2.7) within the unit hypercube. In this section we shall explain why we trace the saddle point solutions through a series of progressively lower temperatures, instead of simply solving them once at a low temperature.

Since at each temperature the solutions to the saddle point equations are exactly the stable states of a Hopfield network running with $\eta = 1$, we could envisage an analogous annealing process on a Hopfield network. This would involve gradually increasing the gain of the transfer functions (2.4) as the network converges. We mentioned in Chapter 4 that

[6] For an n-city travelling salesman problem, there are $2n$ optimal solutions spaced regularly around P [5]. The $2n$ solutions correspond to starting the shortest tour at each of the n cities, and traversing the tour in different senses.

this is not going to help force convergence to a 0-1 point, but the process does allow us to control the initial direction of $\mathbf{v}$, in much the same manner as hysteretic annealing does.

Once again we make use of the linearized dynamics, viewing the MFA algorithm as an Euler approximation of the Hopfield network with $\eta = 1$ and a time-step $\Delta t = 1$. Recall that, for $\mathbf{i}^{\mathrm{opr}} = \mathbf{0}$, $\mathbf{v}$ moves only along those eigenvectors of $\mathbf{T}^{\mathrm{opr}}$ for which $\tilde{\lambda}_i > 0$. Substituting for $\tilde{\lambda}_i$ we obtain the condition

$$T^p < \frac{\phi}{\eta}\lambda_i$$

if $\mathbf{v}$ is to move along $\mathbf{x}^i$. So temperature annealing has much the same effect as hysteretic annealing: the annealing starts at a sufficiently high temperature so that $T^p > (\phi/\eta)\lambda_i$ for all i[7], and then T^p is gradually lowered so that $\mathbf{v}$ moves first along that eigenvector with the largest positive eigenvalue.

Recalling the full Hopfield network Liapunov function (2.5), we see that temperature annealing involves adding a term of the form $\eta \sum_i \int_0^{v_i} g^{-1}(V)\, dV$ to the objective function E^{op}. Unless this term has the same value at all 0-1 points within P (as is the case for optimization over Ω^n), this means that the network will be operating on the wrong objective function until the low temperature limit is reached. This should be contrasted with the use of hysteretic annealing, which can operate at a fixed, low value of T^p, giving a valid objective function at all times. In the low temperature limit the η term added to E^{op} simply disappears. Therefore temperature annealing cannot be used to guarantee that E^{op} eventually becomes concave, as is possible with hysteretic annealing. It follows that $\mathbf{v}$ does not necessarily converge to a vertex of P, but could stabilize within the interior of P. This is equivalent to saying that in the limit of low T^p, not all the $\tilde{\lambda}_i$'s become positive, only those for which $\lambda_i > 0$.

We can also use the linearized analysis to identify a cause of instability in the MFA algorithm. If a particular $\tilde{\lambda}_i$ is very negative, then an Euler approximation of the linearized dynamics (B.1)

$$\dot{\alpha}_i = \tilde{\lambda}_i \alpha_i + \frac{\phi}{T^p}\beta_i$$

with a large time-step $\Delta t = 1$ could lead to unstable oscillations of $\mathbf{v}$. Instability of the MFA algorithm can be avoided by ensuring that none of the $\tilde{\lambda}_i$ become excessively negative. This can be achieved by mixing hysteretic annealing with temperature annealing, using a fixed value of γ to make the $\tilde{\lambda}_i$'s more positive. The stabilizing effect of adding a hysteretic annealing term has been noted in [118]. We could also use a positive γ term to ensure that $\mathbf{v}$ eventually converges to a vertex of P, since the term makes E^{op} more concave.

While the above analysis was developed for standard MFA, it is shown in [5] that it also applies to MFA with neuron normalization [118, 143]. Although this modified form of MFA does not have general applicability, we use it in the course of this thesis to solve travelling salesman problems. The update rule (4.11) implicitly enforces half of the problem's constraints, leading to simpler penalty functions and generally superior solutions [118].

[7] In many MFA texts, the temperature above which no motion of $\mathbf{v}$ occurs is referred to as the *critical temperature*. Drawing on analogies in statistical physics, it is often said that the system undergoes a *phase transition* at the critical temperature, when $\mathbf{v}$ moves along the dominant eigenvector of $\mathbf{T}^{\mathrm{opr}}$.

5.4 The initial direction of v and solution quality

In this chapter we have so far identified two major influences on the initial direction of $\mathbf{v}$, namely

- $\mathbf{i}^{\mathrm{opr}}$, responsible for the very early movement of $\mathbf{v}$.
- $\mathbf{x}^{\max}$, the dominant eigenvector of $\mathbf{T}^{\mathrm{opr}}$, which has a very significant influence on the subsequent movement of $\mathbf{v}$. $\mathbf{x}^{\max}$ is also responsible for the initial movement of $\mathbf{v}$ when $\mathbf{i}^{\mathrm{opr}} = \mathbf{0}$.

In this section we shall examine the significance of this initial direction on the overall network performance. It is difficult to predict the motion of $\mathbf{v}$ beyond its initial direction, since the linearized analysis would no longer be valid and the nonlinear dynamics are complicated to analyse. It is also easy to argue that $\mathbf{i}^{\mathrm{opr}}$ and $\mathbf{x}^{\max}$ are largely irrelevant, since their influence is totally dependent on $\mathbf{v}$ being initialized near $\mathbf{s}$. However, assuming $\mathbf{s}$ to be a uniform vector, this is a sensible point at which to initialize $\mathbf{v}$, since it implies no bias towards any particular solution point. Besides, an unbiased starting position is most widely used in the literature, and so for the sake of compatibility of results we shall continue with this policy.

Since different initial directions are going to lead $\mathbf{v}$ towards different solution points, it seems likely that the initial direction will have a very significant influence on the eventual solution quality. For example, we shall see in Section 5.5 that $\mathbf{i}^{\mathrm{opr}}$ can systematically guide $\mathbf{v}$ towards a certain type of solution which is not necessarily compatible with the combinatorial objective: in such cases we might say that E^{op} is not well suited to the underlying combinatorial problem. This is illustrated in Figure 5.4. We see two different objectives defined within a polytope P: the objectives are the same as the one used in Figure 5.3, except that nonzero $\mathbf{i}^{\mathrm{opr}}$ terms have been introduced. The effect of this on E^{op} is to shift the centre of the elliptical contours from the point $\mathbf{v}^{\mathrm{val}} = \mathbf{0}$, giving $\mathbf{i}^{\mathrm{opr}}$ full control over the initial direction of $\mathbf{v}$. In Figure 5.4(a) $\mathbf{i}^{\mathrm{opr}}$ is well suited to the underlying combinatorial problem and the network finds the optimal solution by descent. In Figure 5.4(b), however, $\mathbf{i}^{\mathrm{opr}}$ takes $\mathbf{v}$ into a poor region of P from the outset, and $\mathbf{v}$ does not converge to the optimal solution point. For such a small and trivial problem we would view this failure as fairly serious. Moreover, we cannot get around this problem by initializing $\mathbf{v}$ at the stationary point of E^{op} (and then annealing the network twice, starting at $\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$ and then again at $-\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$), since this point may not lie within P, as in Figure 5.4(a). As argued above, we can do no better than to persevere with the standard, unbiased starting position. In Chapter 6 we shall describe some alternative techniques for suppressing the $\mathbf{i}^{\mathrm{opr}}$ term.

In the rest of this chapter we study the $\mathbf{i}^{\mathrm{opr}}$ term and its relationship to the underlying combinatorial problem. We develop a framework for investigating this relationship from a largely analytical standpoint. When $\mathbf{i}^{\mathrm{opr}} = \mathbf{0}$, the initial direction is governed by $\mathbf{x}^{\max}$. Such cases are more difficult to analyse and will be addressed, for a restricted class of problem, through an experimental process in Chapter 6.

5.5 Auxiliary linear problems

For problems where $\mathbf{i}^{\mathrm{opr}} \neq \mathbf{0}$, we have shown theoretically that $\mathbf{i}^{\mathrm{opr}}$ is largely responsible for the initial direction of $\mathbf{v}^{\mathrm{val}}$, and have confirmed this experimentally in Figure 5.2. We must now investigate how $\mathbf{i}^{\text{opr}}$ relates to the underlying combinatorial problem, and decide whether $\mathbf{i}^{\text{opr}}$ is likely to guide $\mathbf{v}$ towards a good solution.

In order to gain a better understanding of the significance of $\mathbf{i}^{\text{opr}}$, we associate $\mathbf{i}^{\text{opr}}$ with an *auxiliary linear problem*. First, we rewrite the problem's objective function (3.1) for $\mathbf{v}$ lying within P, substituting $(\mathbf{T}^{\text{val}}\mathbf{v} + \mathbf{s})$ for $\mathbf{v}$:

$$E^{\text{op}} = -\tfrac{1}{2}(\mathbf{T}^{\text{val}}\mathbf{v} + \mathbf{s})^T \mathbf{T}^{\text{op}} (\mathbf{T}^{\text{val}}\mathbf{v} + \mathbf{s}) - (\mathbf{T}^{\text{val}}\mathbf{v} + \mathbf{s})^T \mathbf{i}^{\text{op}} \tag{5.16}$$

Simplifying equation (5.16), we obtain

$$E^{\text{op}} = -\tfrac{1}{2}\mathbf{v}^T \mathbf{T}^{\text{opr}} \mathbf{v} - \mathbf{i}^{\text{opr}T}\mathbf{v} + \text{terms independent of } \mathbf{v} \tag{5.17}$$

So we can associate the action of $\mathbf{i}^{\text{opr}}$ with the minimization of an auxiliary linear problem, with objective function

$$E^{\text{alp}} = -\mathbf{i}^{\text{opr}T}\mathbf{v} \tag{5.18}$$

We have already established that the initial dynamics of the network are heavily influenced by $\mathbf{i}^{\text{opr}}$, and so we conclude that the initial behaviour of the network is concerned largely with minimizing E^{alp}. We must now decide whether the solution of the auxiliary linear problem is in any way compatible with the solution of the parent optimization problem. This requires an individual assessment for each type of problem.

The Hamilton path problem

For the Hamilton path problem, we show in Appendix B.4 that E^{alp} can be expressed as follows:

$$E^{\text{alp}} = \tfrac{1}{n}\left[(\mathbf{Q}\mathbf{o})^T \mathbf{V}(-\mathbf{P}\mathbf{o})\right] + \text{terms independent of } \mathbf{V} \tag{5.19}$$

where $\mathbf{V}$ is the matrix equivalent of $\mathbf{v}$, a permutation matrix when $\mathbf{v}$ has converged to a 0-1 point, and $\mathbf{o}$ is the vector of ones defined in equation (3.42). Recall that the matrix $\mathbf{P}$ is the negated intercity distance matrix, and the matrix $\mathbf{Q}$ is a 0-1 matrix defined in equation (3.44). Now, element i of the vector $-\mathbf{P}\mathbf{o}$ gives the summed distances of all the other cities from city i, since $[-\mathbf{P}\mathbf{o}]_i = \sum_{j=1}^{n} -P_{ij}$. Similarly

$$[\mathbf{Q}\mathbf{o}]_i = \begin{cases} 1 & \text{if } i \in \{1, n\} \\ 2 & \text{if } 2 \le i \le (n-1) \end{cases} \tag{5.20}$$

For example, if $n = 5$ then $\mathbf{Q}\mathbf{o} = [\,1\ \ 2\ \ 2\ \ 2\ \ 1\,]^T$. In order to minimize E^{alp} (5.19), $\mathbf{V}$ must reorder the elements of $-\mathbf{P}\mathbf{o}$ so that the largest elements appear in the first and last rows of $\mathbf{V}(-\mathbf{P}\mathbf{o})$. 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 otherwise have been the case. 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 effectively assumes this strategy in the early stages of convergence.

Armed with this new insight, we can now address the question of how well suited $\mathbf{i}^{\text{opr}}$ is to the solution of the underlying Hamilton path problem. The auxiliary linear problem suggests that the network may fail to find an optimal tour which does not start and finish at remote cities. Such a problem is illustrated in Figure 5.5. Here we see sixteen cities arranged in a spiral pattern, along with the shortest Hamilton path through the cities: the optimal path length is 54.0 units. Figure 5.6 shows the evolution of $\mathbf{v}$ for a steepest descent network with hysteretic annealing attempting to solve the problem, using the mapping described in Section 3.5. The network behaves as predicted by the linearized analysis, putting in a large $\mathbf{i}^{\text{opr}}$ component from the start, followed by a significant $\mathbf{x}^{\text{max}}$ component. However, a suboptimal solution is reached (path length 56.2 units), with the journey starting and ending at outlying cities — see Figure 5.7. Clearly this is a result of the auxiliary linear problem being poorly related to the parent Hamilton path problem. Indeed, Figure 5.5 shows that the optimal solution's $\mathbf{v}^{\text{val}}$ contains a small *negative* component in the direction of $\mathbf{i}^{\text{opr}}$. An optimization network is most unlikely to converge to such a point, given that its initial dynamics move $\mathbf{v}$ along $\mathbf{i}^{\text{opr}}$ in a *positive* sense.

The graph labelling problem

We turn now to the graph labelling problem, and its application to simple invariant pattern recognition systems. We consider the comparison of planar dot patterns associated with examples of handwritten digits. Each of the digits is represented by 20 points placed around its outline. Subsequently, a 20-node graph is constructed for each digit, with the graph nodes representing the points and the edge weights equal to the Euclidean distance between the points. This representation is invariant to translation and rotation of the digits, and can also be made invariant to enlargement by a simple normalization operation. An unidentified pattern $\mathcal{P}$ (with graph $G_{\mathcal{P}}$) can be compared with a template pattern $\mathcal{Q}$ (with graph $G_{\mathcal{Q}}$), and a similarity measure computed for the potential match. However, first it is necessary to solve the *graph labelling problem*: find which nodes in $G_{\mathcal{P}}$ to match with which nodes in $G_{\mathcal{Q}}$, so that the similarity between the graphs is maximized.

A mapping for this problem was presented in Section 3.5, and is identical to the travelling salesman and Hamilton path mappings, except that the matrix $\mathbf{P}$ is the edge weight matrix of $G_{\mathcal{P}}$, and the matrix $\mathbf{Q}$ is the edge weight matrix of $G_{\mathcal{Q}}$. It follows that equation (5.19) gives us E^{alp} for the graph labelling problem as well:

$$E^{\text{alp}} = -\tfrac{1}{n}\left[(\mathbf{Qo})^T\mathbf{V}(\mathbf{Po})\right]$$

Now, the vectors $\mathbf{Qo}$ and $\mathbf{Po}$ compute the summed distances of all the other points from each point in $\mathcal{P}$ or $\mathcal{Q}$, since $[\mathbf{Qo}]_i = \sum_{j=1}^{n} Q_{ij}$ and $[\mathbf{Po}]_i = \sum_{j=1}^{n} P_{ij}$. In order to minimize E^{alp}, $\mathbf{V}$ must reorder the elements of $\mathbf{Po}$ so that large elements appear in the same rows as the large elements of $\mathbf{Qo}$, while small elements appear in the same rows as the small elements of $\mathbf{Qo}$. In other words, the auxiliary linear problem matches outlying points in pattern $\mathcal{P}$ with outlying points in pattern $\mathcal{Q}$, and central points in pattern $\mathcal{P}$ with central points in pattern $\mathcal{Q}$.

The auxiliary linear problem seems far better suited to the parent problem than was the case for Hamilton paths: it is hard to imagine two point sets for which the optimal match does not associate outlying points in one pattern with outlying points in the other, and central points in one pattern with central points in the other. Experimental evidence to support this hypothesis can be found in Figure 5.8, where we study several typical 20-point matching problems for which good solutions have been found. For each problem, Figure 5.9 shows the decomposition of the solutions' $\mathbf{v}^{\text{val}}$ along $\mathbf{i}^{\text{opr}}$ and the eigenvectors of $\mathbf{T}^{\text{opr}}$. It is apparent that the solutions contain significant components along $\mathbf{i}^{\text{opr}}$, indicating a good degree of sympathy between the auxiliary linear problem and the parent problem. Note that it is the *geometry* of the situation which is helping the optimization networks here.

If the graphs did not represent planar Euclidean patterns, then we would not be able to use the above argument to support the suitability of the auxiliary linear problem. In Section 6.7 we shall see that for *random* graph labelling problems an alternative objective function, which effectively removes the $\mathbf{i}^{\mathrm{opr}}$ term, often leads to improved performance.

Unfortunately, optimization networks experience a different kind of operating difficulty with graph labelling problems. Figure 5.10 shows the decomposition of $\mathbf{i}^{\mathrm{opr}}$ along the eigenvectors of $\mathbf{T}^{\mathrm{opr}}$, revealing that for some problems the vectors $\mathbf{i}^{\mathrm{opr}}$ and $\mathbf{x}^{\mathrm{max}}$ are nearly mutually orthogonal. As we saw for the Hamilton path problem in Section 5.2, this means that the motion of $\mathbf{v}^{\mathrm{val}}$ in the direction of $\mathbf{x}^{\mathrm{max}}$ is dependent on the random starting vector. However, unlike the Hamilton path problem, where each of the two possible routes could lead to an optimal solution, the point matching problem has no solution degeneracy, and so one of the routes is bound to lead to a suboptimal solution. This hypothesis is confirmed in Figure 5.11, which shows the result of comparing the patterns $\mathcal{P}_4$ and $\mathcal{Q}_4$ (see Figure 5.8) twice using a steepest descent network with hysteretic annealing, starting once at $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ and then again at $-\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$. As expected, the component of $\mathbf{v}^{\mathrm{val}}$ along $\mathbf{x}^{\mathrm{max}}$ develops different signs in the two runs, leading to two different solutions, one with $E^{\mathrm{op}} = 0.157$ (the good solution seen in Figure 5.8), the other with $E^{\mathrm{op}} = 0.346$. The second solution is sufficiently poor to hide the similarity between $\mathcal{P}_4$ and $\mathcal{Q}_4$, most likely resulting in a misclassification in the context of the pattern recognition system.

5.6 Summary

In this chapter we have identified two reasons why a network may fail to find a good solution to a particular optimization problem. These are

1. The auxiliary linear problem is poorly related to the parent problem. $\mathbf{i}^{\mathrm{opr}}$ takes $\mathbf{v}$ into a poor region of space from the outset, leading to a poor solution. This was the case for the spiral Hamilton path problem.

2. $\mathbf{i}^{\mathrm{opr}}$ is orthogonal to the dominant eigenvector of $\mathbf{T}^{\mathrm{opr}}$. $\mathbf{v}$ can therefore move along this eigenvector in either direction, depending on the random starting vector. $\mathbf{v}$ will converge to one of two possible solutions and, unless there is a suitable degeneracy of solutions, one of these is likely to be poor. It is possible to recover from this failure by running the network twice, starting once at $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ and then again at $-\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$.

We have so far studied, through largely analytical means, only the influence of $\mathbf{i}^{\mathrm{opr}}$ on the network's performance. In Chapter 6 we appeal to experimental means to investigate the relationship between the combinatorial problem and further aspects of E^{op}.

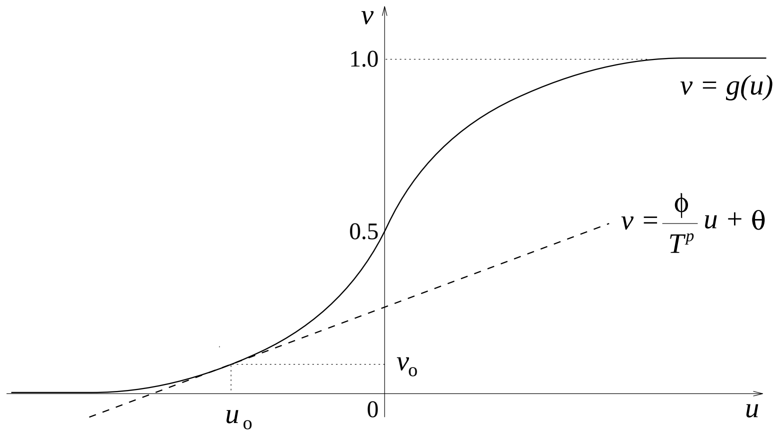


Figure 5.1: Linearizing the network's dynamics.

The network's dynamics are linearized by considering only small excursions of $\mathbf{u}$ *and* $\mathbf{v}$ *around their initial positions* $\mathbf{u}_\mathrm{o}$ *and* $\mathbf{v}_\mathrm{o}$*. It is therefore possible to replace the nonlinear transfer functions* $g()$ *by their tangents at the starting point. Differentiating equation (2.4) reveals that* $\phi = v_\mathrm{o}(1 - v_\mathrm{o})$*.*

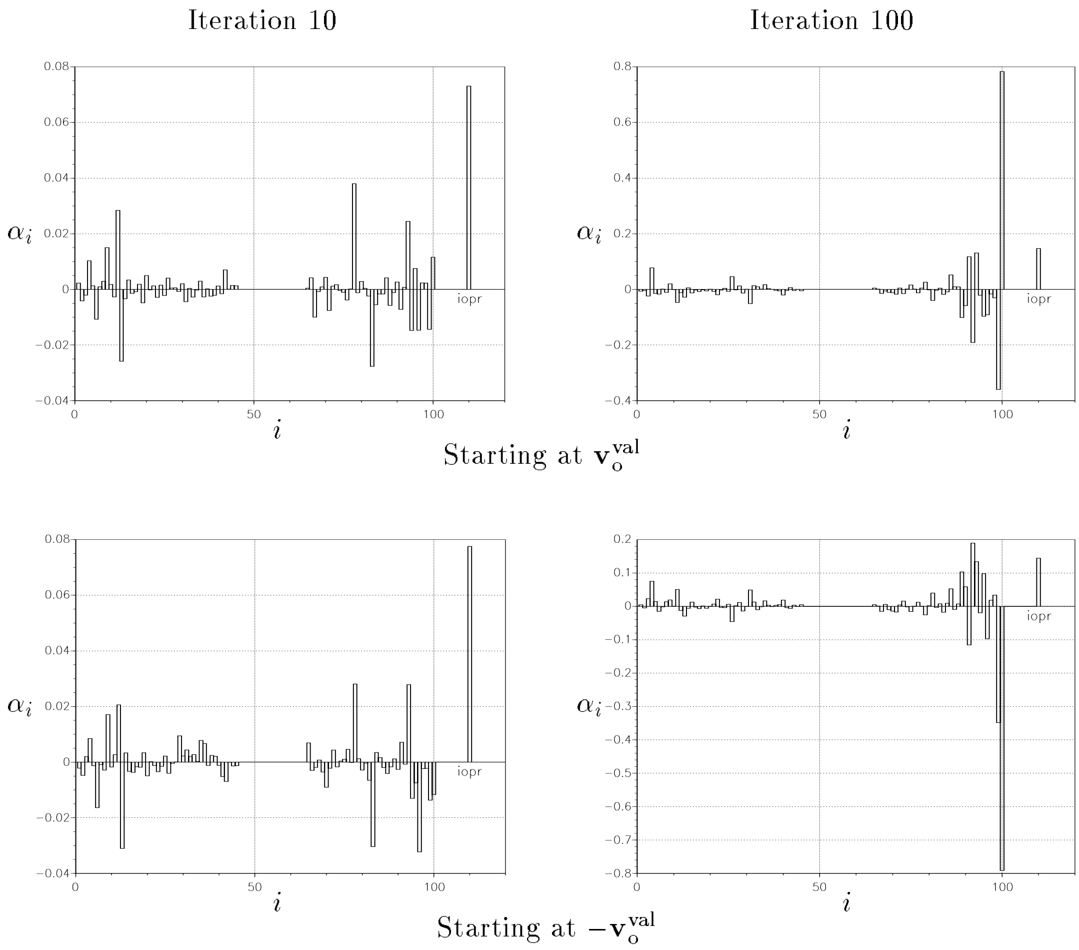
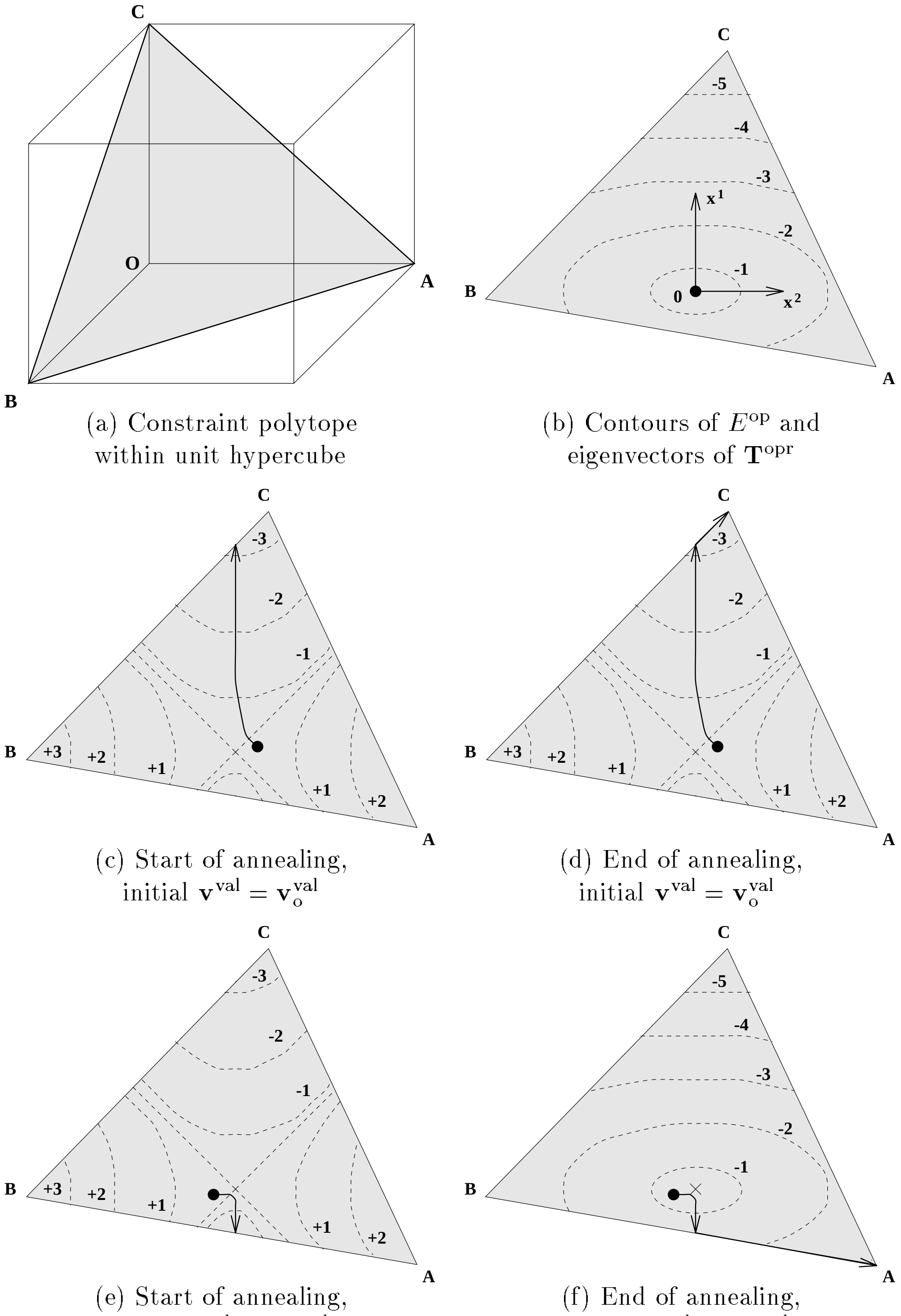


Figure 5.2: Evolution of $\mathbf{v}$ for a 10-city Hamilton path problem.

The figure shows the decomposition of $\mathbf{v}^\mathrm{val}$ *along the eigenvectors of* $\mathbf{T}^\mathrm{opr}$ *for a steepest descent network solving a 10-city Hamilton path problem, starting once at the point* $\mathbf{v}_\mathrm{o}^\mathrm{val}$ *and then again at the point* $-\mathbf{v}_\mathrm{o}^\mathrm{val}$*. The* α_i*'s, which represent the components of* $\mathbf{v}^\mathrm{val}$ *along the 100 eigenvectors* $\mathbf{x}^i$*, are plotted against* i*. The components along the eigenvectors with the more positive eigenvalues appear on the right. The bar marked* iopr *measures the component of* $\mathbf{v}^\mathrm{val}$ *along* $\mathbf{i}^\mathrm{opr}$*:* $\mathrm{iopr} = {\mathbf{v}^\mathrm{val}}^T \mathbf{i}^\mathrm{opr} / \|\mathbf{i}^\mathrm{opr}\|$*. The figure confirms that* $\mathbf{v}^\mathrm{val}$ *initially moves along* $\mathbf{i}^\mathrm{opr}$*, before turning towards* $\mathbf{x}^\mathrm{max}$*, the dominant eigenvector of* $\mathbf{T}^\mathrm{opr}$*. The direction in which* $\mathbf{v}^\mathrm{val}$ *moves along* $\mathbf{x}^\mathrm{max}$ *depends on the random starting position.*



(a) Constraint polytope within unit hypercube

(b) Contours of E^{op} and eigenvectors of $\mathbf{T}^{\mathrm{opr}}$

(c) Start of annealing, initial $\mathbf{v}^{\mathrm{val}} = \mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$

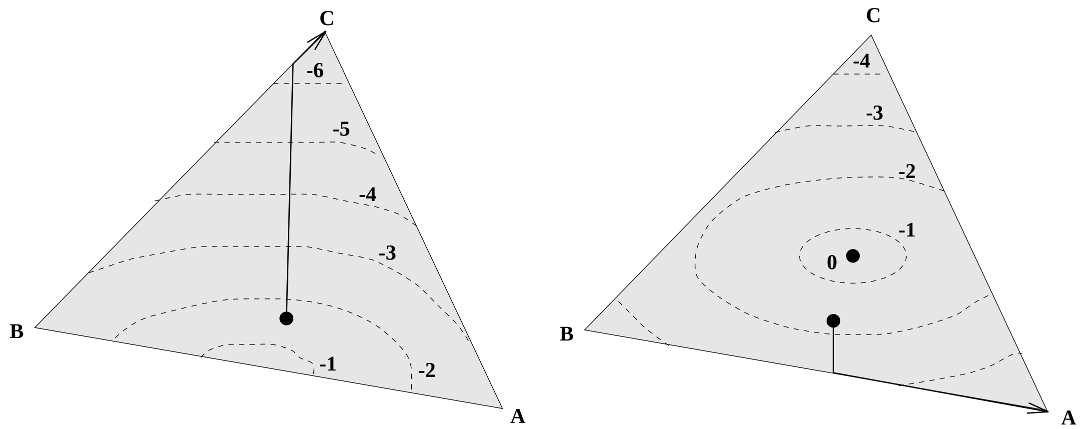
(d) End of annealing, initial $\mathbf{v}^{\mathrm{val}} = \mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$

(e) Start of annealing, initial $\mathbf{v}^{\mathrm{val}} = -\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$

(f) End of annealing, initial $\mathbf{v}^{\mathrm{val}} = -\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$

Figure 5.3: The effects of annealing and different random starting positions.

Figure (b) illustrates that, without annealing, any of the solution points A, B *and* C *can be reached by descent from starting positions near the centre of the polytope. Annealing makes the network's dynamics less dependent on the random starting position, by turning the local maximum at the centre of the polytope into a saddle point. Reversing the sign of the initial* $\mathbf{v}^{\mathrm{val}}$ *takes* $\mathbf{v}$ *down different sides of the saddle, leading to one of two different solution points.*



(a) E^{op} well suited (b) E^{op} ill-suited

Figure 5.4: The suitability of E^{op}.

The action of $\mathbf{i}^{\mathrm{opr}}$ *leads to just one way in which* E^{op} *can be ill-suited to the solution of the underlying combinatorial problem. Introducing an* $\mathbf{i}^{\mathrm{opr}}$ *term shifts the stationary point of* E^{op} *away from* $\mathbf{v}^{\mathrm{val}} = \mathbf{0}$*, giving* $\mathbf{i}^{\mathrm{opr}}$ *full control over the initial direction of* $\mathbf{v}$*. In (a)* $\mathbf{i}^{\mathrm{opr}}$ *is well suited to the underlying combinatorial problem, and the network finds the optimal solution by descent. In (b), however,* $\mathbf{i}^{\mathrm{opr}}$ *takes* $\mathbf{v}$ *into a poor region of* P *from the outset, and* $\mathbf{v}$ *does not converge to the optimal solution point. Initializing* $\mathbf{v}$ *at the stationary point of* E^{op} *is not an option, since this point may not lie within* P*, as in (a).*

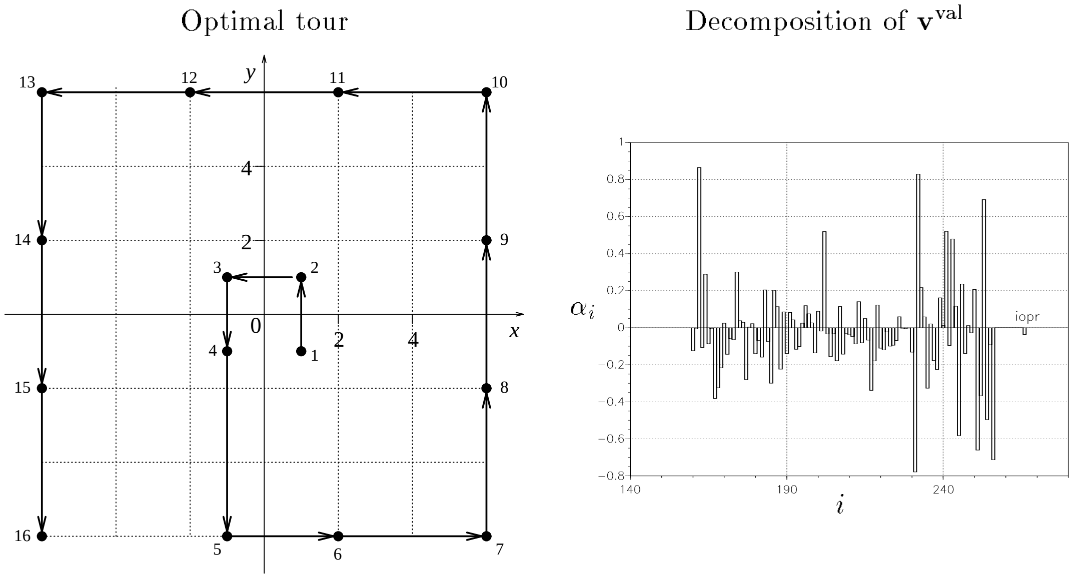


Figure 5.5: Optimal solution to the spiral Hamilton path problem.

The figure shows the optimal solution to the 16-city spiral Hamilton path problem, together with the decomposition of the corresponding $\mathbf{v}^{\mathrm{val}}$ *along* $\mathbf{i}^{\mathrm{opr}}$ *and the 256 eigenvectors of* $\mathbf{T}^{\mathrm{opr}}$*. See the caption to Figure 5.2 for an explanation of the decomposition plot: for clarity only the components along those eigenvectors with the most positive eigenvalues are displayed. Since the decomposition shows a small* negative *component along* $\mathbf{i}^{\mathrm{opr}}$*, while* $\mathbf{v}$ *initially moves along* $\mathbf{i}^{\mathrm{opr}}$ *in a* positive *sense, it is most unlikely that an optimization network will find this solution point. In the context of the auxiliary linear problem, the negative component along* $\mathbf{i}^{\mathrm{opr}}$ *indicates that the optimal tour does not start and finish at outlying cities.*

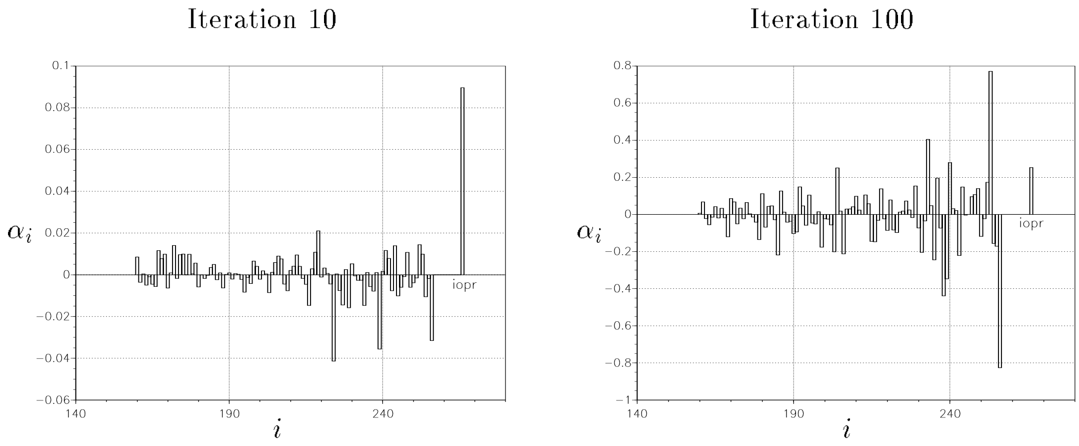


Figure 5.6: Evolution of $\mathbf{v}$ for the spiral Hamilton path problem.

The figure shows the decomposition of $\mathbf{v}^{\text{val}}$ *along* $\mathbf{i}^{\text{opr}}$ *and the 256 eigenvectors of* $\mathbf{T}^{\text{opr}}$ *for a steepest descent network with hysteretic annealing solving the 16-city spiral Hamilton path problem. See the caption to Figure 5.2 for an explanation of the decomposition plots: for clarity only the components along those eigenvectors with the most positive eigenvalues are displayed. As predicted by the theory,* $\mathbf{v}^{\text{val}}$ *initially moves along* $\mathbf{i}^{\text{opr}}$*, before turning towards the dominant eigenvector of* $\mathbf{T}^{\text{opr}}$*. Since Figure 5.5 indicates that the optimal solution has a negative component along* $\mathbf{i}^{\text{opr}}$*, it is clear that the steepest descent network initially moves* $\mathbf{v}$ *away from this solution.*

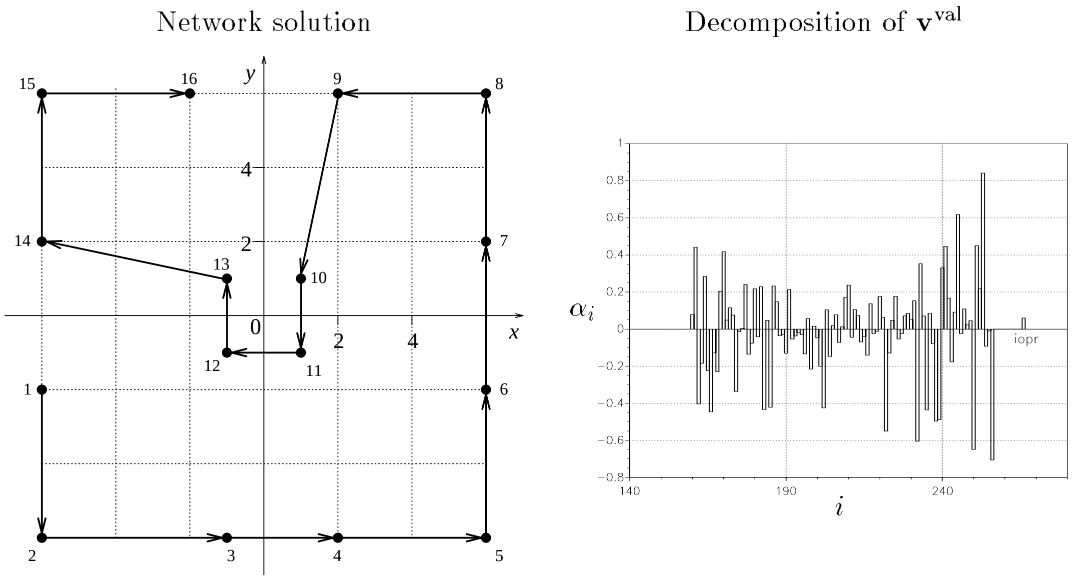


Figure 5.7: Network solution to the spiral Hamilton path problem.

The figure shows the solution found by the steepest descent network to the 16-city spiral Hamilton path problem, together with the decomposition of the corresponding $\mathbf{v}^{\text{val}}$ *along* $\mathbf{i}^{\text{opr}}$ *and the 256 eigenvectors of* $\mathbf{T}^{\text{opr}}$*. See the caption to Figure 5.2 for an explanation of the decomposition plot: for clarity only the components along those eigenvectors with the most positive eigenvalues are displayed. The decomposition shows a small* positive *component along* $\mathbf{i}^{\text{opr}}$*, which is consistent with the analysis of the network's dynamics. In the context of the auxiliary linear problem, the positive component along* $\mathbf{i}^{\text{opr}}$ *indicates that the tour starts and finishes at outlying cities, unlike the optimal tour.*

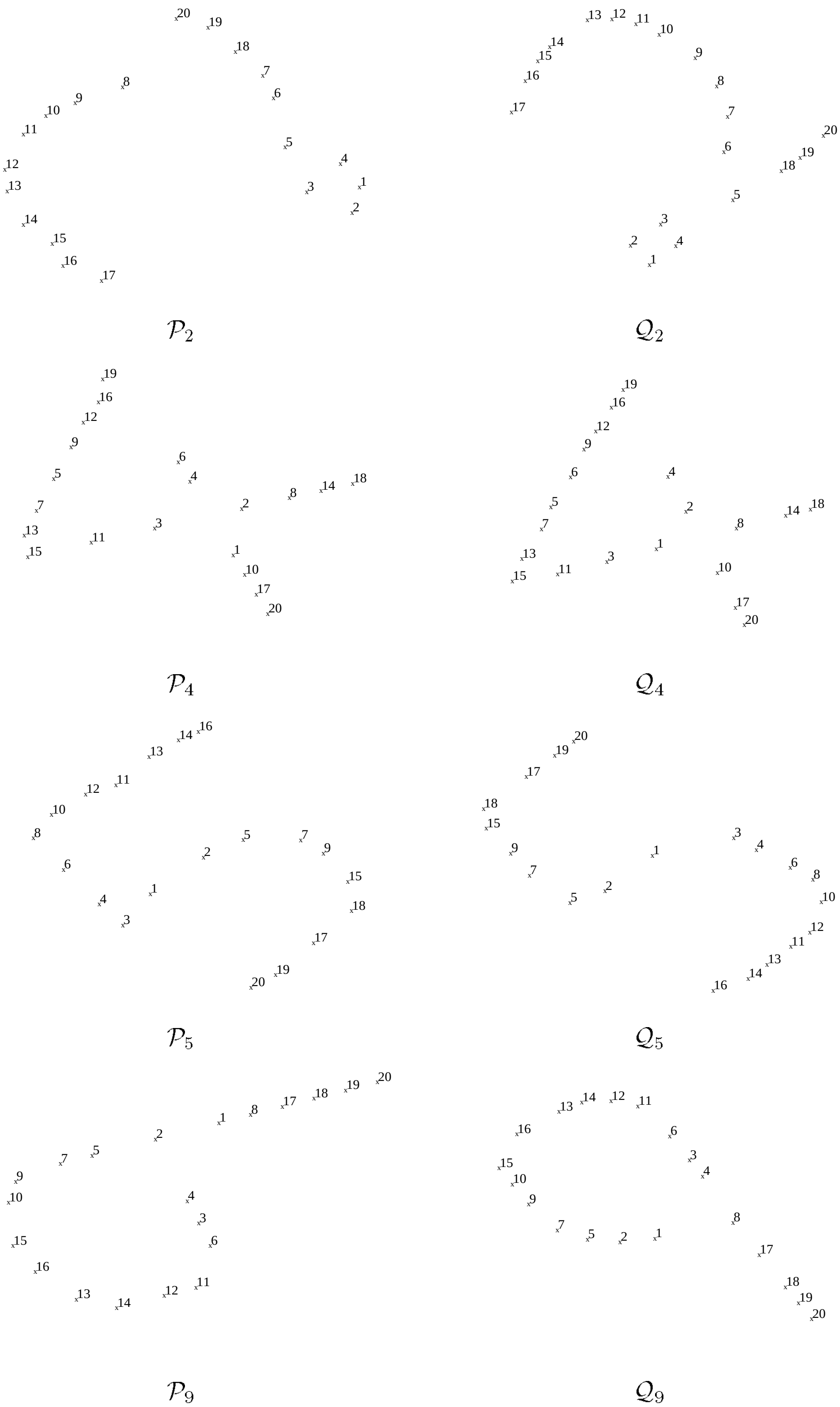


Figure 5.8: Pairs of patterns with labelling for a good match.

A cursory examination of the labellings confirms that outlying points in $\mathcal{P}$ are matched with outlying points in $\mathcal{Q}$, while central points in $\mathcal{P}$ are matched with central points in $\mathcal{Q}$. Thus it appears that the auxiliary linear problem is in sympathy with the parent problem.

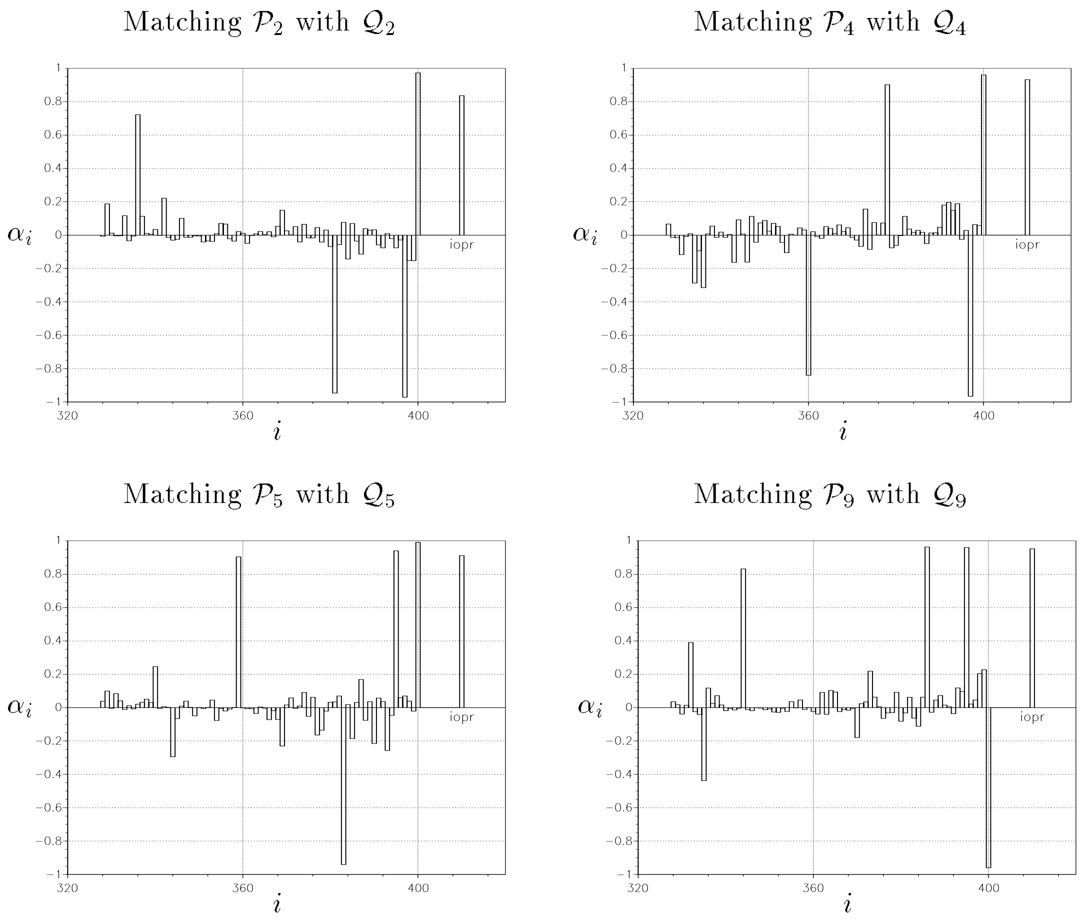


Figure 5.9: Solution decompositions for the graph labelling problems.

The figure shows the decompositions of $\mathbf{v}^{\mathrm{val}}$ *along* $\mathbf{i}^{\mathrm{opr}}$ *and the eigenvectors of* $\mathbf{T}^{\mathrm{opr}}$ *for the four solutions to the graph labelling problems shown in Figure 5.8. See the caption to Figure 5.2 for an explanation of the decomposition plots: for clarity only the components along those eigenvectors with the most positive eigenvalues are displayed. The large components along* $\mathbf{i}^{\mathrm{opr}}$ *confirm the apparent sympathy between the auxiliary linear problem and the parent problem.*

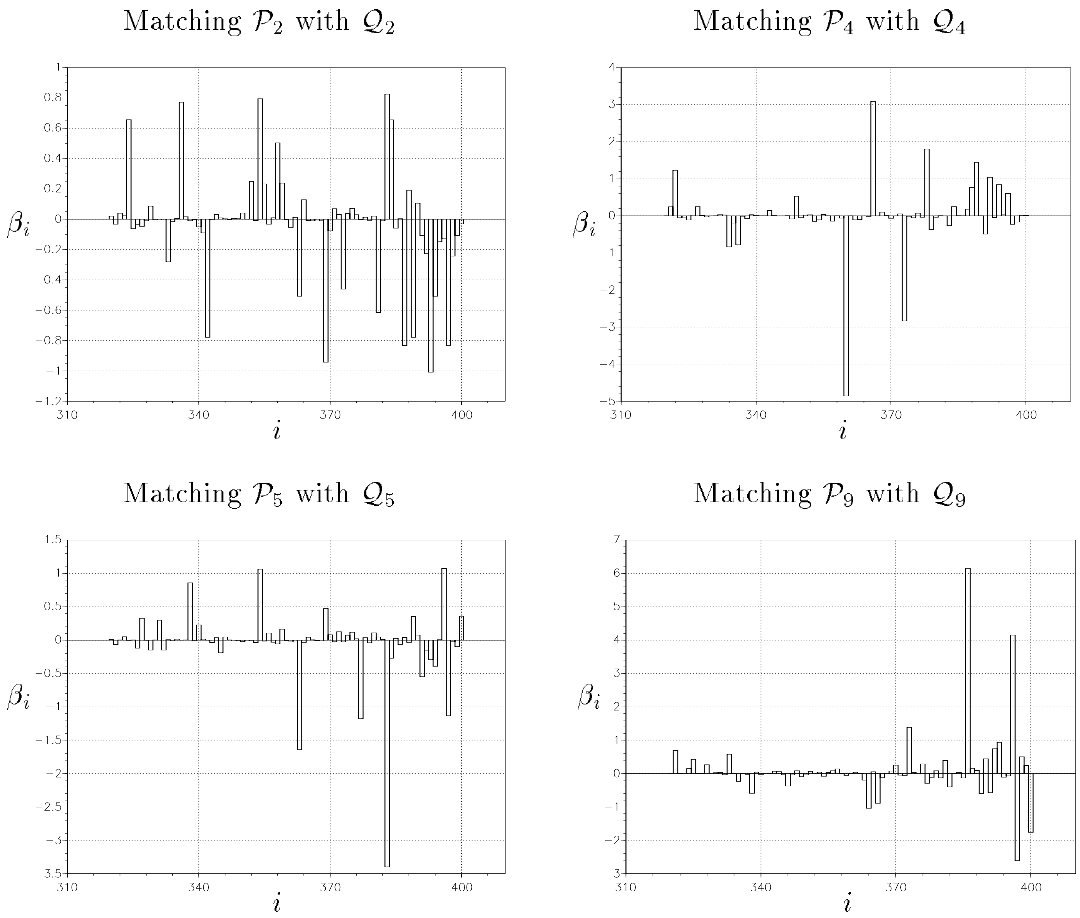


Figure 5.10: Decompositions of $\mathbf{i}^{\mathrm{opr}}$ for the graph labelling problems.

The figure shows the decompositions of $\mathbf{i}^{\mathrm{opr}}$ *along the eigenvectors of* $\mathbf{T}^{\mathrm{opr}}$ *for the four graph labelling problems shown in Figure 5.8. See the caption to Figure 5.2 for an explanation of the decomposition plots: for clarity only the components along those eigenvectors with the most positive eigenvalues are displayed. For the top two problems it is apparent that* $\mathbf{i}^{\mathrm{opr}}$ *is virtually orthogonal to the dominant eigenvector* $\mathbf{x}^{\max}$*, and so* $\mathbf{v}$ *will evolve along this eigenvector in a direction totally dependent on the random starting vector. One of these two directions will most probably lead to a poor quality solution.*

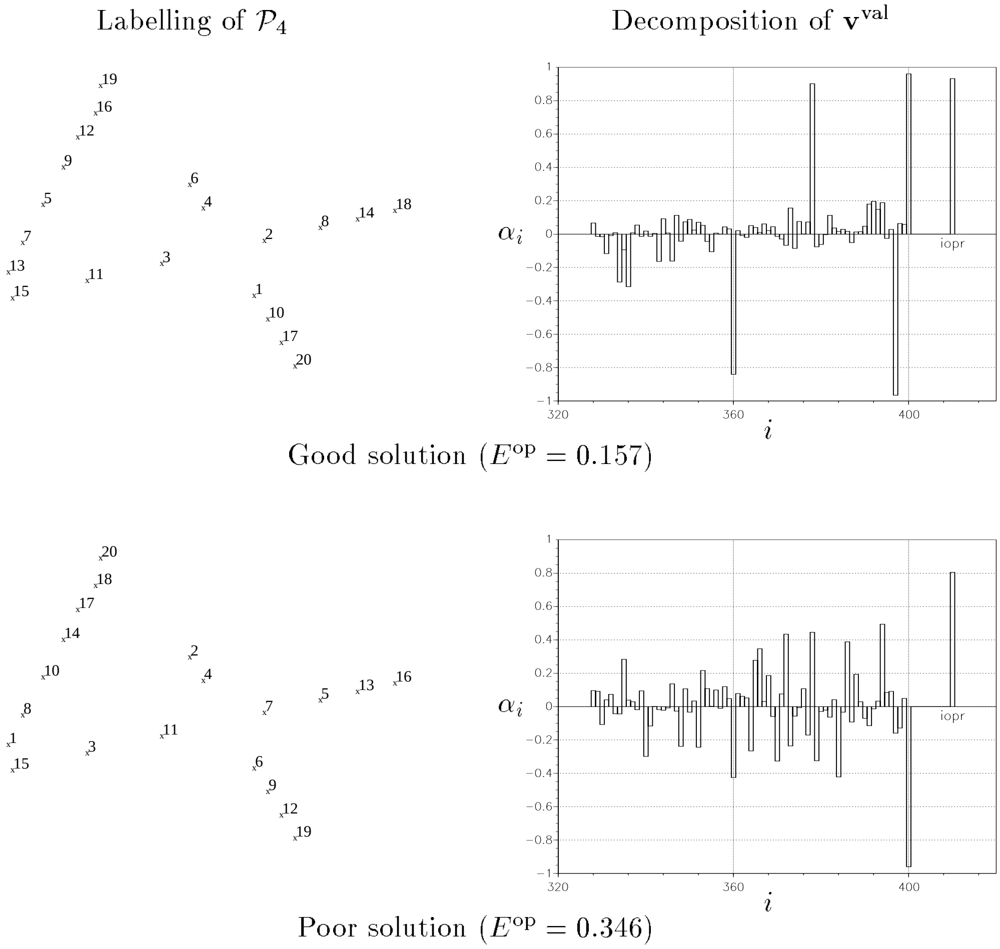


Figure 5.11: Network solutions for the matching of $\mathcal{P}_4$ with $\mathcal{Q}_4$.

The two solutions were found by a steepest descent network with hysteretic annealing attempting to match $\mathcal{P}_4$ *with* $\mathcal{Q}_4$*, starting once at* $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ *and then again at* $-\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$*. See the caption to Figure 5.2 for an explanation of the decomposition plots: for clarity only the components along those eigenvectors with the most positive eigenvalues are displayed. Since* $\mathbf{i}^{\mathrm{opr}}$ *is virtually orthogonal to the dominant eigenvector* $\mathbf{x}^{\mathrm{max}}$*,* $\mathbf{v}$ *moves along* $\mathbf{x}^{\mathrm{max}}$ *in different directions in the two runs, only one of which leads to a good solution. The other solution, if used by an invariant pattern recognition system, would most probably result in a misclassification. The labellings of* $\mathcal{P}_4$ *should be compared with the labelling of* $\mathcal{Q}_4$ *in Figure 5.8.*

Chapter 6

Kronecker Product Mappings over Ω^n

In Chapter 5 we made some progress towards predicting the performance of optimization networks on certain problems over integral polytopes. Our findings centred around the suitability of the $\mathbf{i}^{\mathrm{opr}}$ term in guiding the state vector away from its initial position. However, no attempt was made to investigate other features of the networks' dynamics and their influence on solution quality, since further analysis of the dynamics beyond the initial direction is difficult. This chapter concerns a broad class of problems over Ω^n with a common Kronecker product formulation, including the travelling salesman, Hamilton path and graph labelling problems. By restricting the study to such problems, a more detailed investigation of the networks' dynamics is possible. We describe how an optimization network will tend to move its state vector towards a 0-1 point exhibiting certain prototypical properties. The likely success of the network can then be assessed by examining good solutions (obtained by exhaustive search) for evidence of the same properties.

We begin the investigation by reviewing in Sections 6.1 and 6.2 a framework for investigating Kronecker product mappings, first presented in [5]. For these mappings, a valid solution has to exhibit certain constraining properties when decomposed along the eigenvectors of $\mathbf{T}^{\mathrm{opr}}$, as described in Section 6.3. This information can be used to make further deductions about the networks' dynamics, leading in Section 6.4 to a prototypical decomposition of $\mathbf{v}$ at full convergence. The original contribution of this chapter begins in Section 6.5, where we examine good solutions to large numbers of travelling salesman problems for evidence of the same decomposition. Our findings indicate that optimization networks are well suited to the solution of Euclidean, but not random, travelling salesman problems. In an attempt to improve performance, we propose in Section 6.7 a means of modifying the networks' dynamics by making use of alternative objective functions. In particular, it is possible to construct a valid objective for any problem such that E^{op} has a stationary point at the starting position: in this way, dependence on an $\mathbf{i}^{\mathrm{opr}}$ term can be removed. However, we conclude that the alternative objectives perform, on average, no better than the original objectives over a large set of problems.

6.1 A framework for Kronecker product mappings

In this chapter we consider only problems with the following mapping:

$$\begin{aligned}\mathbf{T}^{\mathrm{val}} &= \mathbf{R}\otimes\mathbf{R}\\ \mathbf{s} &= \tfrac{1}{n}(\mathbf{o}\otimes\mathbf{o})\\ \mathbf{T}^{\mathrm{op}} &= \mathbf{P}\otimes\mathbf{Q}\\ \mathbf{i}^{\mathrm{op}} &= \mathbf{0}\end{aligned}$$

where $\mathbf{R}$ and $\mathbf{o}$ are defined in equations (3.41) and (3.42) respectively. The expressions for $\mathbf{T}^{\mathrm{val}}$ and $\mathbf{s}$ restrict $\mathbf{v}$ to Ω^n, so we know that $\mathbf{v}$ will converge to a point representing a permutation matrix. For general $\mathbf{P}$ and $\mathbf{Q}$, the network solves the graph labelling problem for graphs with edge weight matrices $\mathbf{P}$ and $\mathbf{Q}$. For special $\mathbf{P}$ and $\mathbf{Q}$, the network can solve the travelling salesman or Hamilton path problems.

For such mappings it is possible to make further predictions concerning the network's dynamics, beyond the initial direction of $\mathbf{v}$. We start by looking again at the eigenvectors of $\mathbf{T}^{\mathrm{opr}}$. From Appendix B.3 we obtain the following expression for $\mathbf{T}^{\mathrm{opr}}$ (B.6):

$$\mathbf{T}^{\mathrm{opr}} = \mathbf{RPR}\otimes\mathbf{RQR}\equiv\mathbf{P}^{\mathrm{val}}\otimes\mathbf{Q}^{\mathrm{val}}$$

where $\mathbf{P}^{\mathrm{val}}\equiv\mathbf{RPR}$ and $\mathbf{Q}^{\mathrm{val}}\equiv\mathbf{RQR}$. We can now use the properties of Kronecker products to relate the eigenvalues and eigenvectors of $\mathbf{T}^{\mathrm{opr}}$ to those of $\mathbf{P}^{\mathrm{val}}$ and $\mathbf{Q}^{\mathrm{val}}$. We start by identifying the nullspaces of $\mathbf{Q}^{\mathrm{val}}$ and $\mathbf{P}^{\mathrm{val}}$, which are clearly both spanned by $\mathbf{o}$, since $\mathbf{Ro}=\mathbf{0}$. So let us define $\mathbf{w}^1$ and $\mathbf{h}^1$ to be eigenvectors of $\mathbf{P}^{\mathrm{val}}$ and $\mathbf{Q}^{\mathrm{val}}$ respectively, with associated eigenvalues μ_1 and θ_1, where

$$\mathbf{w}^1 = \tfrac{1}{\sqrt{n}}\mathbf{o} \quad\text{and}\quad \mu_1 = 0 \tag{6.1}$$
$$\mathbf{h}^1 = \tfrac{1}{\sqrt{n}}\mathbf{o} \quad\text{and}\quad \theta_1 = 0 \tag{6.2}$$

Let the remaining eigenvectors of $\mathbf{P}^{\mathrm{val}}$ be $\mathbf{w}^2\ldots\mathbf{w}^n$ with eigenvalues $\mu_2\ldots\mu_n$, and those of $\mathbf{Q}^{\mathrm{val}}$ be $\mathbf{h}^2\ldots\mathbf{h}^n$ with eigenvalues $\theta_2\ldots\theta_n$. Let us also assume that the eigenvalues are ordered as follows:

$$\mu_2\geq\mu_3\geq\ldots\geq\mu_n \tag{6.3}$$
$$\theta_2\geq\theta_3\geq\ldots\geq\theta_n \tag{6.4}$$

If $\mathbf{T}^{\mathrm{opr}}$ has eigenvectors and eigenvalues $\mathbf{x}^{kl}$ and λ_{kl} respectively, then

$$\mathbf{x}^{kl} = \mathbf{w}^k\otimes\mathbf{h}^l \tag{6.5}$$
$$\text{and}\quad \lambda_{kl} = \mu_k\theta_l \tag{6.6}$$

for $1\leq k\leq n$ and $1\leq l\leq n$. Note that

$$\begin{aligned}\mathbf{T}^{\mathrm{val}}\mathbf{x}^{kl} &= (\mathbf{R}\otimes\mathbf{R})(\mathbf{w}^k\otimes\mathbf{h}^l) = (\mathbf{Rw}^k\otimes\mathbf{Rh}^l)\\ \Rightarrow \mathbf{T}^{\mathrm{val}}\mathbf{x}^{kl} &= \begin{cases}\mathbf{0} & \text{for } k=1 \text{ or } l=1\\ \mathbf{x}^{kl} & \text{for } k\geq 2 \text{ and } l\geq 2\end{cases}\end{aligned} \tag{6.7}$$

Hence those $\mathbf{x}^{kl}$ with $k=1$ or $l=1$ are eigenvectors of $\mathbf{T}^{\mathrm{opr}}$ orthogonal to the valid subspace, while those $\mathbf{x}^{kl}$ with $k\geq 2$ and $l\geq 2$ lie within the valid subspace.

We turn now to the decompositions of $\mathbf{v}^{\mathrm{val}}$, $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ and $\mathbf{i}^{\mathrm{opr}}$ along the eigenvectors of $\mathbf{T}^{\mathrm{opr}}$. We replace the component vectors α, α^{o} and β with matrices $\mathbf{A}$, $\mathbf{A}^{\mathrm{o}}$ and $\mathbf{B}$, such that

$$\mathbf{v}^{\mathrm{val}} = \sum_{k=1}^{n}\sum_{l=1}^{n} A_{kl}\mathbf{x}^{kl} \tag{6.8}$$

$$\mathbf{v}_{\mathrm{o}}^{\mathrm{val}} = \sum_{k=1}^{n}\sum_{l=1}^{n} A_{kl}^{\mathrm{o}}\mathbf{x}^{kl} \tag{6.9}$$

$$\mathbf{i}^{\mathrm{opr}} = \sum_{k=1}^{n}\sum_{l=1}^{n} B_{kl}\mathbf{x}^{kl} \tag{6.10}$$

Note that the indices on the summations could equally run from 2 to n, reflecting the fact that $\mathbf{v}^{\mathrm{val}}$, $\mathbf{v}_{\mathrm{o}}^{\mathrm{val}}$ and $\mathbf{i}^{\mathrm{opr}}$ all lie in the valid subspace, and therefore have no components along those $\mathbf{x}^{kl}$ orthogonal to the valid subspace.

6.2 Eigenvalue degeneracy

For some problems it transpires that the eigenvalues of $\mathbf{P}^{\mathrm{val}}$ or $\mathbf{Q}^{\mathrm{val}}$ are degenerate; this degeneracy propagates through to the eigenvalues of $\mathbf{T}^{\mathrm{opr}}$. For example, consider the $\mathbf{Q}$ matrix for the travelling salesman problem (3.11). The (unordered) eigenvalues of the corresponding matrix $\mathbf{Q}^{\mathrm{val}}$ are given by [5]

$$\theta_l = \begin{cases} 0 & \text{for } l = 1 \\ 2\cos\left(\frac{2\pi}{n}(l-1)\right) & \text{for } l \geq 2 \end{cases} \tag{6.11}$$

It follows that many of the eigenvalues of $\mathbf{Q}^{\mathrm{val}}$ are degenerate, since $\theta_l = \theta_{\tilde{l}}$ for $\tilde{l} = n-l+2$. For example, if $n = 10$ then the (ordered) eigenvalues of $\mathbf{Q}^{\mathrm{val}}$ are approximately

$$\theta = [\ 0 \quad 1.6 \quad 1.6 \quad 0.6 \quad 0.6 \quad -0.6 \quad -0.6 \quad -1.6 \quad -1.6 \quad -2.0\ ]^T$$

and it is clear that there are four pairs of degenerate eigenvalues. We should really be examining the components of $\mathbf{v}^{\mathrm{val}}$ in the eigenplanes of $\mathbf{T}^{\mathrm{opr}}$, instead of along each degenerate eigenvector. Since we shall make extensive use of the 10-city travelling salesman problem as an illustrative example, it would be beneficial to introduce some notation to deal with this particular problem's eigenvalue degeneracy. Let us define a 10×6 matrix $\mathbf{Z}$ of complex elements such that

$$\mathrm{Re}(Z_{kl}) = \begin{cases} A_{kl} & \text{for } l = 1 \\ A_{k,2l-2} & \text{for } 2 \leq l \leq 6 \end{cases} \tag{6.12}$$

$$\mathrm{Im}(Z_{kl}) = \begin{cases} 0 & \text{for } l = 1 \text{ or } l = 6 \\ A_{k,2l-1} & \text{for } 2 \leq l \leq 5 \end{cases} \tag{6.13}$$

Hence, for the degenerate eigenvalues, the magnitude of Z_{kl} gives the magnitude of $\mathbf{v}^{\mathrm{val}}$ in the corresponding eigenplane, while the phase of Z_{kl} gives the direction in that eigenplane. For the simple eigenvalues, Z_{kl} is real and gives the component of $\mathbf{v}^{\mathrm{val}}$ along the corresponding eigenvector. Let us also introduce a reduced 10×6 matrix ζ of eigenvalues of $\mathbf{T}^{\mathrm{opr}}$, such that

$$\zeta_{kl} = \begin{cases} \lambda_{kl} & \text{for } l = 1 \\ \lambda_{k,2l-2} & \text{for } 2 \leq l \leq 6 \end{cases} \tag{6.14}$$

The eigenvalues ζ_{kl} relate to the eigenvectors and eigenplanes in the decomposition matrix $\mathbf{Z}$.

6.3 Permutation matrices and bounds on A

For Kronecker product problems over Ω^n, it is possible to place bounds on the final decomposition of $\mathbf{v}^{\mathrm{val}}$ (ie. on $\mathbf{A}$). These bounds are not a consequence of the problem's constraints, which simply fix the component of $\mathbf{v}$ *outside* the valid subspace, and have no bearing on $\mathbf{A}$[1]. However, we can say something about $\mathbf{A}$ if we consider that when converged, $\mathbf{v}$ must represent a permutation matrix, so

$$\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$$

where $\mathbf{v}=$ vec($\mathbf{V}$). It is shown in [5] that this condition implies the following bounds on $\mathbf{A}$:

$$\sum_{k=2}^{n} A'_{kl} = 1 \quad \text{for } 2 \leq l \leq n \tag{6.15}$$

$$\sum_{l=2}^{n} A'_{kl} = 1 \quad \text{for } 2 \leq k \leq n \tag{6.16}$$

where $\mathbf{A}'$ is a matrix of positive elements defined by $A'_{kl} \equiv A^2_{kl}$. Moreover, these conditions, which are valid when $\mathbf{v}$ has converged to a 0-1 point, can be relaxed for the unconverged state vector to [5]

$$\sum_{k=2}^{n} A'_{kl} \leq 1 \quad \text{for } 2 \leq l \leq n \tag{6.17}$$

$$\sum_{l=2}^{n} A'_{kl} \leq 1 \quad \text{for } 2 \leq k \leq n \tag{6.18}$$

Hence the sum of the elements in those rows and columns of $\mathbf{A}'$ associated with eigenvectors in the valid subspace cannot exceed one at any time, and must reach this value when $\mathbf{v}$ has converged to a 0-1 point. The same conditions can be expressed in terms of the matrix $\mathbf{Z}$ for the 10-city travelling salesman problem. Defining the matrix $\mathbf{Z}'$ such that $Z'_{kl} \equiv \|Z_{kl}\|^2$, the conditions are

$$\sum_{l=2}^{6} Z'_{kl} \leq 1 \quad \text{for } 2 \leq k \leq 10 \tag{6.19}$$

$$\sum_{k=2}^{10} Z'_{kl} \leq \begin{cases} 1 & \text{for } l = 6 \\ 2 & \text{for } 2 \leq l \leq 5 \end{cases} \tag{6.20}$$

with the strict equalities holding when $\mathbf{v}$ has converged to a 0-1 point.

6.4 Network dynamics for Kronecker product mappings

We can use the bounds on $\mathbf{A}$ to make further predictions about the dynamics of an optimization network running under Kronecker product mappings. To do this, we make use of the linearized analysis of the network's dynamics which we derived in Chapter 5. We shall deal exclusively with the case $\mathbf{i}^{\mathrm{opr}} = \mathbf{0}$, which covers the mapping of the travelling

[1] The constraints dictate that $\mathbf{v}$ must lie on the valid subspace with equation $\mathbf{v} = \mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s} \equiv \mathbf{v}^{\mathrm{val}} + \mathbf{s}$: they say nothing about the decomposition of $\mathbf{v}^{\mathrm{val}}$.

salesman problem (see Appendix B.2). In Section 6.7 we shall go on to see how any other problem can be remapped so that $\mathbf{i}^{\mathrm{opr}} = \mathbf{0}$, extending the applicability of this section to all Kronecker product problems over Ω^n.

In what follows we assume that an annealing procedure is being used, so $\mathbf{v}^{\mathrm{val}}$ moves first along those eigenvectors of $\mathbf{T}^{\mathrm{opr}}$ with more positive eigenvalues, then along those with more negative eigenvalues. Of course, once $\mathbf{v}$ has moved some distance from its starting position the linearized analysis breaks down. The order in which $\mathbf{v}^{\mathrm{val}}$ moves along the eigenvectors of $\mathbf{T}^{\mathrm{opr}}$ will not be governed solely by the corresponding eigenvalues, but will be influenced by the network's nonlinear transfer functions as well. However, we can use the linearized analysis as a rough guide to the network's dynamics, especially valid at the start of convergence.

To begin with, let us assume that the μ_k's and θ_l's are all positive for $k \geq 2$ and $l \geq 2$. Recalling that the eigenvalues of $\mathbf{T}^{\mathrm{opr}}$ are given by $\lambda_{kl} = \mu_k \theta_l$, and that the μ_k's and θ_l's are ordered as in (6.3) and (6.4), it follows that the most positive eigenvalue of $\mathbf{T}^{\mathrm{opr}}$ is λ_{22}. So we would expect the A_{22} component to be introduced at an early stage. Assuming the linearized analysis holds, this will continue until $A'_{22} = 1$, at which point the magnitude of A_{22} can increase no further without violating conditions (6.17) and (6.18). Since $A'_{22} = 1$, no further elements in the second row or column of $\mathbf{A}$ can be introduced without violating conditions (6.17) and (6.18). Hence, when the annealing progresses, it is the A_{33} component which is introduced next, since this is associated with the next most positive eigenvalue. The linearized analysis suggests that this process continues until $\mathbf{v}$ converges to a point where

$$A'_{kl} = \begin{cases} \delta_{k,l} & \text{for } k, l \geq 2 \\ 0 & \text{for } k = 1 \text{ or } l = 1 \end{cases} \tag{6.21}$$

In the above argument, we appealed solely to the linearized dynamics, which break down after $\mathbf{v}$ has moved some distance from its initial position. However, it is reasonable to propose that $\mathbf{v}$ will converge to a state similar to (6.21), especially around those elements of $\mathbf{A}'$ introduced early on, when the linear approximation is most applicable. We shall verify this proposition experimentally in Figures 6.1 and 6.2.

The argument can be extended to cover cases where $\mathbf{P}^{\mathrm{val}}$ and $\mathbf{Q}^{\mathrm{val}}$ have both positive and negative eigenvalues [5], with the same result that $\mathbf{A}'$ converges to a form not unlike (6.21), especially around those elements introduced early on. The corresponding result for the 10-city travelling salesman problem is that the network's solution will have a decomposition $\mathbf{Z}'$ resembling the matrix $\mathbf{Z}^{10}$, where

$$\mathbf{Z}^{10} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

To illustrate the annealing process, let us consider a particular 10-city Euclidean travelling salesman problem, where the cities have been randomly placed within a unit square.

The **P** and **Q** matrices are set as in (3.10) and (3.11), to give the following reduced eigenvalue matrix:

$$\zeta = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.58 & 1.37 & -1.37 & -3.58 & -4.42 \\ 0 & 1.69 & 0.64 & -0.64 & -1.69 & -2.08 \\ 0 & 0.87 & 0.33 & -0.33 & -0.87 & -1.08 \\ 0 & 0.62 & 0.24 & -0.24 & -0.62 & -0.76 \\ 0 & 0.39 & 0.15 & -0.15 & -0.39 & -0.48 \\ 0 & 0.24 & 0.09 & -0.09 & -0.24 & -0.29 \\ 0 & 0.19 & 0.07 & -0.07 & -0.19 & -0.24 \\ 0 & 0.18 & 0.07 & -0.07 & -0.18 & -0.22 \\ 0 & 0.16 & 0.06 & -0.06 & -0.16 & -0.20 \end{bmatrix} \qquad \mathrm{order}(\zeta) = \begin{bmatrix} - & - & - & - & - & - \\ - & 1 & 3 & 41 & 44 & 45 \\ - & 2 & 5 & 37 & 42 & 43 \\ - & 4 & 8 & 33 & 39 & 40 \\ - & 6 & 10 & 29 & 36 & 38 \\ - & 7 & 14 & 23 & 34 & 35 \\ - & 9 & 15 & 22 & 30 & 32 \\ - & 11 & 16 & 21 & 26 & 31 \\ - & 12 & 17 & 20 & 25 & 28 \\ - & 13 & 18 & 19 & 24 & 27 \end{bmatrix}$$

The right hand matrix shows the ordering of the elements of ζ; the dashes correspond to eigenvalues associated with the nullspace of $\mathbf{T}^{\mathrm{val}}$, which play no part in the dynamics of $\mathbf{v}^{\mathrm{val}}$.

Figure 6.1 shows the matrix $\mathbf{Z}'$ at various values of γ for a steepest descent network running with hysteretic annealing. The $\gamma = -4.0$ plot shows the decomposition of the random starting position, which is very near the point $\mathbf{v} = \mathbf{s}$, so all the components of $\mathbf{v}^{\mathrm{val}}$ are small. For $\gamma = -1.9$ only $\tilde{\zeta}_{22}$ is positive[2], and the dynamics increase the component Z_{22} as predicted. By the time $\gamma = -0.1$, the $\mathbf{Z}'$ matrix has evolved to a form not unlike the expected $\mathbf{Z}^{10}$, though those Z_{kl} with very negative ζ_{kl}'s have yet to be introduced. Finally, when $\gamma = 1.5$, $\mathbf{v}$ has successfully converged to a valid 0-1 point, and the final form of $\mathbf{Z}'$ is similar to $\mathbf{Z}^{10}$, especially around the Z_{22} corner of $\mathbf{Z}$. So this illustrative experiment is in good agreement with the theory presented above.

In order to demonstrate the similarity between hysteretic and temperature annealing, Figure 6.2 shows how the $\mathbf{Z}'$ matrix evolves for the same problem being solved using MFA with neuron normalization [5, 118, 143]. It is apparent that the state vector follows a similar trajectory to that of the steepest descent network, leading to an identical final solution.

6.5 Properties of good solutions

Now that we know more about the structure of solutions found by optimization networks, we are in a position to predict the likely success of the networks on specific problems. To do this, we examine the decompositions of optimal and near-optimal solutions to such problems, and compare the structure of the decompositions with the prototypical structure of a network's solution. If there is a good match between the two, especially around those components introduced early in the convergence process, then we would expect an optimization network to perform fairly well.

An analytical approach [5] suggests that the optimal solution will indeed have a decomposition not unlike that of the network's solution (6.21). If we constrain $\mathbf{V}$ so that its row and column sums both equal 1, and so that $\mathbf{V}\mathbf{V}^T = \mathbf{V}^T\mathbf{V} = \mathbf{I}$, then the $\mathbf{V}$ which minimizes E^{op} has a decomposition identical to (6.21). Unfortunately, these constraints are not enough to specify a permutation matrix: we need to impose the extra

[2] Recall that $\tilde{\zeta}_{kl} = \zeta_{kl} + \gamma$ for steepest descent dynamics. The linearized analysis predicts that the Z_{kl} component of $\mathbf{v}$ will not increase until $\tilde{\zeta}_{kl} > 0$.

constraint $V_{ij} > 0$ [5]. To understand the effect of this extra constraint on the decomposition of the optimal solution, we must revert to experimental means. In this section we study two databases of 10-city travelling salesman problems. The ten best solutions to these small problems were found by exhaustive search. One database contains 1000 Euclidean problems (so $\mathbf{P}$ is set to be the negated Euclidean distance matrix of the cities in the plane), the other contains 1000 random problems (in this case $\mathbf{P}$ is a random, symmetric 10×10 matrix, with zeros along its leading diagonal).

Figure 6.3 shows the decomposition of the optimal solutions to the 1000 Euclidean problems. The data is presented as the mean of the decomposition matrix $\mathbf{Z}'$, along with the standard deviation of the same data across the ensemble of 1000 solutions. We see that the decomposition does, roughly, resemble the $\mathbf{Z}^{10}$ form which an optimization network is likely to find. The resemblance is closest around the Z'_{22} corner, where the variance of the data is also fairly low. Figure 6.5(a) shows the mean of the eigenvalues ζ across the ensemble of 1000 problems. We see that the most positive eigenvalues correspond to eigenvectors in the Z'_{22} corner of $\mathbf{Z}'$, and so an optimization network will introduce these components first. It therefore seems likely that an optimization network will be successful with these problems, since the initial direction of $\mathbf{v}$ is compatible with the location of the optimal solution.

Turning now to the solutions to the random problems, we see a much more diffuse decomposition in Figure 6.4, with higher standard deviations. The peaks in the decomposition are of a lower magnitude than those for the Euclidean problems. Figure 6.5(b) shows the eigenvalues of $\mathbf{T}^{\mathrm{opr}}$ for these problems. Once again, the eigenvalues are most positive for components in the Z'_{22} corner of $\mathbf{Z}'$, so optimization networks will move $\mathbf{v}$ first in this direction. But the peak of the optimal solutions' mean decomposition at Z'_{22} is not so high, and the standard deviation here is also rather high, indicating that the optimal solutions do not necessarily have a large Z'_{22} component. So we might expect optimization networks to be less successful with these problems.

Figure 6.6 shows how the decompositions vary across the ten best solutions to each problem. The data is presented as the standard deviation of the $\mathbf{Z}'$ matrix for the ten best solutions, averaged over all 1000 problems in each database. For the Euclidean problems, the deviations are very low, especially around the Z'_{22} corner, indicating that good solutions tend to cluster in the solution space, and that they always have a large component along the dominant eigenvector. This is further evidence for the likely success of optimization networks on these problems, since their dynamics will guide $\mathbf{v}$ towards the cluster of good solutions. For the random problems, however, we see that the deviations are much larger, especially around the Z'_{22} corner. This suggests that there is no predictable decomposition for good solutions and, in particular, good solutions do not necessarily have large components along the dominant eigenvector. We would therefore expect network performance to be compromised on these problems. A different configuration space analysis [13] suggests that solutions to Euclidean and random graph partitioning problems exhibit similar clustering properties: good solutions are found in clusters for the Euclidean problems, but not for the random ones.

6.6 Empirical evaluation of network performance

The predictions of network performance can be checked against the results of suitable experiments. We used a steepest descent network and an MFA algorithm with neuron normalization to solve all 1000 Euclidean and random problems. The full results are displayed in Figures 6.7 and 6.8, and summarized in Tables 6.1 and 6.2. For comparison, the tables also show how a stochastic simulated annealing algorithm [119] and a 3-opt search algorithm [97] perform on the same problems. Experimental detail for all the optimization techniques can be found in Appendix B.5.

Both optimization networks solve the Euclidean problems fairly well, finding tours about 1% longer than the shortest tour on average. Moreover, it appears that this good performance is replicated with larger Euclidean problems: in [116, 118] MFA is used to solve problems in up to 200 cities, with performance comparable to that of the far slower simulated annealing algorithm.

As predicted, the optimization networks were not nearly so successful with the random problems, finding tours about 11% longer than optimal on average. We can attempt to explain this failure by looking at the decompositions of the networks' solutions: these are shown in Figures 6.9 and 6.10. As expected, both networks found solutions which resemble the $\mathbf{Z}^{10}$ form much more than the optimal solutions do (Figure 6.4(a)). In particular, the networks introduced a large Z'_{22} component with little variance across the 1000 problems: the standard deviation of the Z'_{22} component was 0.11 for the steepest descent solutions, 0.14 for the MFA solutions and 0.23 for the optimal solutions. In other words, optimization networks rely heavily on good solutions having large components along the dominant eigenvector, which is not always the case. Stochastic simulated annealing is not affected by such factors, finding near-optimal solutions for the random problems as it did for the Euclidean problems. Moreover, the networks' performance is even inferior to that of the simple 3-opt search technique, which, for random problems of less than fifty cities, finds solutions close to the presumed optima [87]. The poor performance of optimization networks running on other random problems (of the graph partitioning variety) has also been reported in [13, 14].

Of course, solution quality is not the only factor to consider when comparing different optimization techniques: there is also execution time. On this count, although optimization networks are uncompetitive in simulation, with execution time scaling as n^3 for an n-city travelling salesman problem[3], they would be the clear winners if implemented in suitable hardware, in which case the execution time would be independent of the problem size. Stochastic simulated annealing scales as n^2 [116, 118] and basic 3-opt search as n^3 [97] (although local search can be refined to scale as n^2 [98]). At first sight, it might appear that optimization networks, if implemented in parallel hardware, would be the first choice for large problems. However, the solution quality for random problems is likely to be unacceptable, and there is an n^2 space complexity which would limit their application to problems of moderate size. So optimization networks are not particularly appealing, especially since alternative, state-of-the-art algorithms for the Euclidean travelling salesman problem now have *linear* time and space complexity [46].

[3] In Chapter 4, we pointed out that the time taken to simulate an optimization network of size N scales as N^2. Since a network of size $N = n^2$ is required for an n-city travelling salesman problem, this gives a simulation complexity of n^4. However, more efficient simulations are possible for Kronecker product mappings over Ω^n. Calculating $\mathbf{T}^{\mathrm{val}}\mathbf{v}$ for the particular $\mathbf{T}^{\mathrm{val}}$ associated with Ω^n is only an order N operation; most of the computational effort is expended calculating $\mathbf{T}^{\mathrm{op}}\mathbf{v}$. For Kronecker product mappings, it is straightforward to show using (C.7) that $\mathbf{T}^{\mathrm{op}}\mathbf{v}$ becomes $\mathbf{QVP}^T$ in matrix form, where $\mathbf{v} = \mathrm{vec}(\mathbf{V})$ and $\mathbf{T}^{\mathrm{op}} = (\mathbf{P} \otimes \mathbf{Q})$: $\mathbf{T}^{\mathrm{op}}\mathbf{v}$ can therefore be calculated with n^3 complexity.

6.7 Alternative objective functions

We have now seen, and explained, several causes of network failure. Before we conclude that optimization networks are simply not suited to the solution of some problems, we can attempt to exploit one final degree of flexibility. For any particular problem, there are an infinite number of continuous functions E^{op} which provide a suitable objective at 0-1 points. We have already come across one family of such functions when we looked at hysteretic annealing: we saw that adding any multiple of E^{ann} (4.9) to E^{op} did not invalidate the objective at 0-1 points. This effectively gives us an infinite number of valid objectives for the problem. In this section we shall investigate other families of alternative objective functions, and attempt to use them to overcome some of the identified shortcomings of optimization networks.

Alternative objectives for general mappings

Recall the form of an objective function

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

where $\mathbf{v} \in \mathbb{R}^N$. Now let $\mathbf{d}$ be an arbitrary N-element vector, and let $\mathbf{D}$ be an $N \times N$ matrix such that $\mathbf{D} = \mathrm{diag}\,(\mathbf{d})$. Consider a new objective $E^{\mathrm{op}\prime}$ constructed as follows [2]:

$$E^{\mathrm{op}\prime} = -\tfrac{1}{2}\mathbf{v}^T(\mathbf{T}^{\mathrm{op}} + \mathbf{D})\mathbf{v} - \mathbf{v}^T(\mathbf{i}^{\mathrm{op}} - \tfrac{1}{2}\mathbf{d}) \tag{6.22}$$

If $v_i \in \{0,1\}$, then $\mathbf{v}^T\mathbf{D}\mathbf{v} = \mathbf{v}^T\mathbf{d}$, and so

$$E^{\mathrm{op}\prime} = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{\mathrm{op}}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{op}} \quad \text{for } v_i \in \{0,1\}$$

It follows that $E^{\mathrm{op}\prime} = E^{\mathrm{op}}$ at all 0-1 points, and so $E^{\mathrm{op}\prime}$ is a valid objective for the combinatorial optimization problem. This defines a whole family of alternative objective functions we could use, since $\mathbf{d}$ can be any N-element vector. The alternative objectives give rise to $\mathbf{T}^{\mathrm{opr}}$ having different eigenvectors and eigenvalues, and also alter $\mathbf{i}^{\mathrm{opr}}$, so we would expect the performance of descent procedures on these functions to be quite variable.

We can exploit this flexibility to eliminate $\mathbf{i}^{\mathrm{opr}}$ for any problem. Assuming no slack variables, and substituting $(\mathbf{T}^{\mathrm{op}} + \mathbf{D})$ for $\mathbf{T}^{\mathrm{op}}$ and $(\mathbf{i}^{\mathrm{op}} - \frac{1}{2}\mathbf{d})$ for $\mathbf{i}^{\mathrm{op}}$, $\mathbf{i}^{\mathrm{opr}}$ (5.4) becomes

$$\mathbf{i}^{\mathrm{opr}} = \mathbf{T}^{\mathrm{val}}(\mathbf{T}^{\mathrm{op}}\mathbf{s} + \mathbf{i}^{\mathrm{op}} + \mathbf{D}\mathbf{s} - \tfrac{1}{2}\mathbf{d}) \tag{6.23}$$

If we make the bracketed vector expression lie in the nullspace of $\mathbf{T}^{\mathrm{val}}$, then $\mathbf{i}^{\mathrm{opr}} = \mathbf{0}$. We can achieve this if we set the bracketed expression to be some multiple of $\mathbf{s}$, which we know lies in the nullspace of $\mathbf{T}^{\mathrm{val}}$. The appropriate values for the elements of $\mathbf{d}$ are

$$d_i = \frac{ks_i - [\mathbf{T}^{\mathrm{op}}\mathbf{s} + \mathbf{i}^{\mathrm{op}}]_i}{s_i - \frac{1}{2}} \tag{6.24}$$

for any scalar k.[4] This alternative objective function creates a stationary point near the initial position of $\mathbf{v}$ (ie. at $\mathbf{v} = \mathbf{s}$), so we no longer have an $\mathbf{i}^{\mathrm{opr}}$ term and all the problems it can entail. However, we have to be aware that $\mathbf{v}$ can now follow two different paths, depending on which direction it moves along the dominant eigenvector, so we may have to run the network twice for problems with no solution degeneracy.

[4] It is straightforward to show that this technique can also be used for mappings including slack variables, but only with $k = 0$.

Alternative objectives for Kronecker product mappings

For Kronecker product mappings, E^{op} can be expressed as $-\frac{1}{2}\mathrm{trace}(\mathbf{VPV}^T\mathbf{Q})$ (3.12). We propose a family of alternative objective functions

$$E^{\mathrm{op}\prime} = -\tfrac{1}{2}\mathrm{trace}\left(\mathbf{V}(\mathbf{P}+\phi\mathbf{I})\mathbf{V}^T(\mathbf{Q}+\mathbf{D}^{\mathrm{q}})\right) \tag{6.25}$$

where $\mathbf{D}^{\mathrm{q}}$ is an $N \times N$ diagonal matrix, and ϕ is a scalar constant. Using the fact that $\mathbf{V}$ is a permutation matrix when $\mathbf{v}$ has converged to a 0-1 point, we obtain

$$\begin{aligned} E^{\mathrm{op}\prime} &= E^{\mathrm{op}} - \tfrac{1}{2}\left[\mathrm{trace}\left(\mathbf{VPV}^T\mathbf{D}^{\mathrm{q}}\right) + \phi\ \mathrm{trace}\left(\mathbf{VIV}^T\mathbf{Q}\right) + \phi\ \mathrm{trace}\left(\mathbf{VIV}^T\mathbf{D}^{\mathrm{q}}\right)\right] \\ &= E^{\mathrm{op}} - \tfrac{1}{2}\left[\sum_{i=1}^{n}\left[\mathbf{V}^T\mathbf{D}^{\mathrm{q}}\mathbf{V}\right]_{ii} P_{ii} + \phi\sum_{i=1}^{n} Q_{ii} + \phi\sum_{i=1}^{n} D^{\mathrm{q}}_{ii}\right] \quad \text{for } \mathbf{VV}^T = \mathbf{I} \end{aligned}$$

If we assume that the elements on the leading diagonals of $\mathbf{P}$ and $\mathbf{Q}$ are all zero, as is the case for all the combinatorial problems we have mapped using Kronecker products so far, we obtain

$$E^{\mathrm{op}\prime} = E^{\mathrm{op}} - \tfrac{1}{2}\phi\ \mathrm{trace}(\mathbf{D}^{\mathrm{q}}) \tag{6.26}$$

Hence $E^{\mathrm{op}\prime}$ is a valid objective function for any $\mathbf{D}^{\mathrm{q}}$ or ϕ, since, to within a constant, the objective at 0-1 points is the same as that obtained using the original objective E^{op}. The symmetry of the formulation indicates that it is equally valid to add an arbitrary diagonal matrix $\mathbf{D}^{\mathrm{p}}$ to $\mathbf{P}$, and a multiple of the identity matrix to $\mathbf{Q}$.

These alternative objective functions offer us another way to eliminate $\mathbf{i}^{\mathrm{opr}}$, which, for Kronecker product mappings, is given by (B.5)

$$\mathbf{i}^{\mathrm{opr}} = \tfrac{1}{n}(\mathbf{RPo} \otimes \mathbf{RQo})$$

Consider adding a diagonal matrix $\mathbf{D}^{\mathrm{q}}$ to $\mathbf{Q}$, where

$$D^{\mathrm{q}}_{ii} = \theta - [\mathbf{Qo}]_i = \theta - \sum_{j=1}^{n} Q_{ij} \tag{6.27}$$

and θ is an arbitrary constant. This results in all the row sums of $(\mathbf{Q}+\mathbf{D}^{\mathrm{q}})$ equaling θ, in which case $(\mathbf{Q}+\mathbf{D}^{\mathrm{q}})\mathbf{o} = \theta\mathbf{o}$. $\mathbf{i}^{\mathrm{opr}}$ subsequently becomes

$$\mathbf{i}^{\mathrm{opr}} = \tfrac{1}{n}\left(\mathbf{RPo} \otimes \mathbf{R}(\mathbf{Q}+\mathbf{D}^{\mathrm{q}})\mathbf{o}\right) = \tfrac{1}{n}(\mathbf{RPo} \otimes \theta\mathbf{Ro}) = \mathbf{0} \tag{6.28}$$

In this manner, any Kronecker product problem can be formulated with no $\mathbf{i}^{\mathrm{opr}}$ term.

We can test the alternative objective functions on the 16-city Hamilton path problem of Figure 5.5, for which we decided that the $\mathbf{i}^{\mathrm{opr}}$ term was frustrating the solution of the underlying combinatorial problem. We used an alternative objective of the form (6.25) to remap the problem so that there was a stationary point near the initial position of $\mathbf{v}$. A steepest descent network was then run twice, starting once at $\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$ and then again at $-\mathbf{v}^{\mathrm{val}}_{\mathrm{o}}$. As usual, the two runs produced equivalent solutions, corresponding to traversing the same path in different directions; this path, which is again suboptimal, is shown in Figure 6.11(a). The network's poor performance is not surprising when we look at the decomposition of the optimal solution along the new eigenvectors of $\mathbf{T}^{\mathrm{opr}}$ — see Figure 6.11(b). There is no prominent peak at the A'_{22} position, which corresponds to the dominant eigenvector. It is therefore unlikely that an optimization network will find such a solution[5].

So, it is by no means certain that removing $\mathbf{i}^{\mathrm{opr}}$ will improve the network's performance, since there is no guarantee that $E^{\mathrm{op}\prime}$ will be any more suited to the underlying combinatorial problem than E^{op}. However, sometimes an improvement can be obtained. We used alternative objectives of the form (6.25) to suppress $\mathbf{i}^{\mathrm{opr}}$ in 1000 graph labelling problems, using randomly generated edge weight matrices for the graphs. For both objectives, a steepest descent network was run twice, negating the starting position between each run to be sure of obtaining the best possible solution. The results are displayed in Figure 6.12: in about 33% of the problems the removal of $\mathbf{i}^{\mathrm{opr}}$ was beneficial. On average, the original and alternative objectives performed more or less equally well. We conclude, therefore, that there is no certain advantage to be gained through the use of alternative objectives to suppress $\mathbf{i}^{\mathrm{opr}}$, though to be sure of obtaining the best possible performance such alternatives should be fully explored.

[5]In fact, the optimal solution has a large component A'_{42}, which an optimization network will find very difficult to emulate, since after the dynamics have introduced the A'_{22} component, other components in the second column of $\mathbf{A}'$ are blocked by the constraint (6.17).

Solution technique	**Mean error**	**Std. dev. of error**	**Number valid**
Steepest descent network	0.71%	1.64%	1000
Mean field annealing	1.36%	2.39%	987
Stochastic simulated annealing	0.01%	0.06%	1000
3-opt search (one pass)	0.02%	0.25%	1000

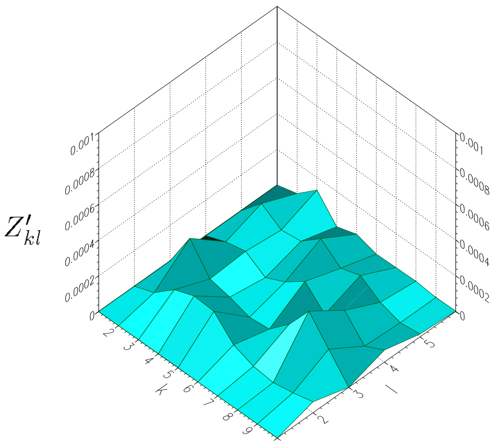
Table 6.1: Results of experiments on 10-city Euclidean travelling salesman problems.

The results confirm that optimization networks solve Euclidean travelling salesman problems fairly well. Although other optimization techniques perform even better, they could not run as fast as an optimization network implemented in parallel hardware. Since MFA is most easily stabilized using a moderate *weighting for the penalty function, valid solutions are not always found. The error measure is specified in the caption to Figure 6.7.*

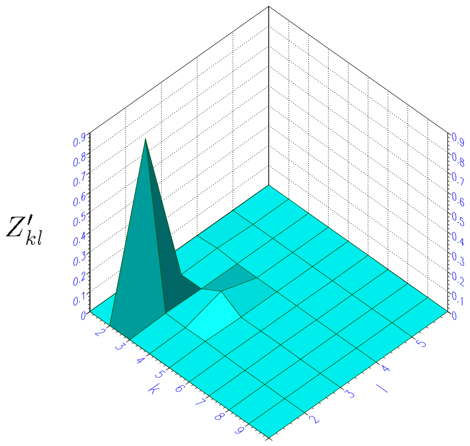
Solution technique	**Mean error**	**Std. dev. of error**	**Number valid**
Steepest descent network	13.5%	13.7%	1000
Mean field annealing	9.73%	10.7%	988
Stochastic simulated annealing	0.29%	1.35%	1000
3-opt search (one pass)	0.84%	2.99%	1000

Table 6.2: Results of experiments on 10-city random travelling salesman problems.

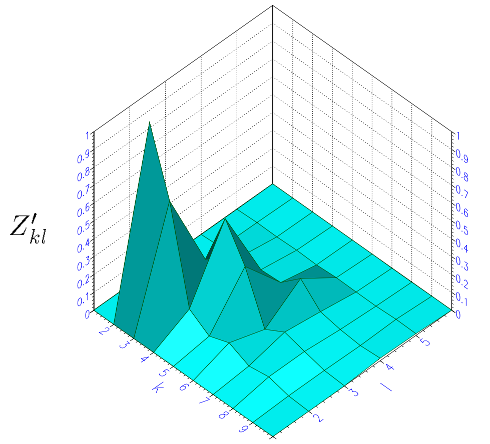
Optimization networks are not nearly so effective with random travelling salesman problems. In contrast, other optimization techniques have little difficulty with these problems, identifying a real weakness of optimization networks. Since MFA is most easily stabilized using a moderate *weighting for the penalty function, valid solutions are not always found. The error measure is specified in the caption to Figure 6.7.*



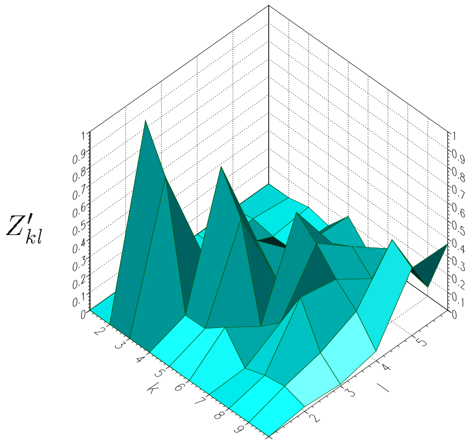
(a) $\gamma = -4.0$



(b) $\gamma = -1.9$



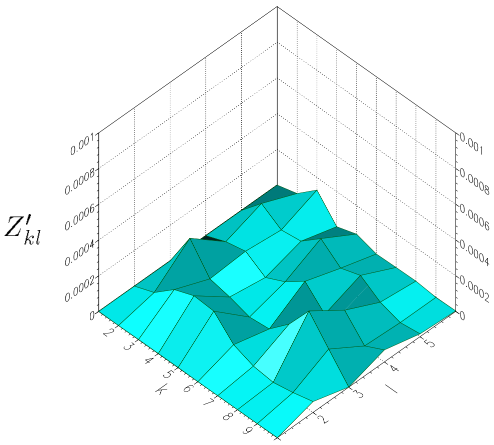
(c) $\gamma = -0.1$



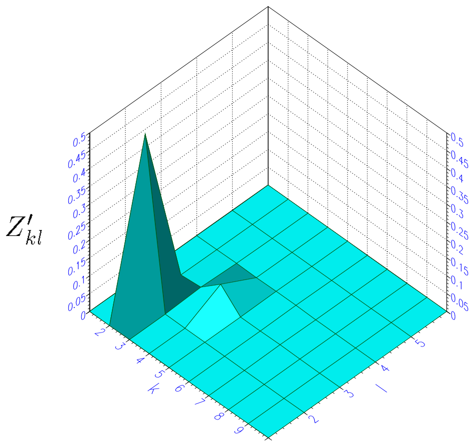
(d) $\gamma = 1.5$ (Fully converged)

Figure 6.1: Hysteretic annealing applied to a 10-city travelling salesman problem.

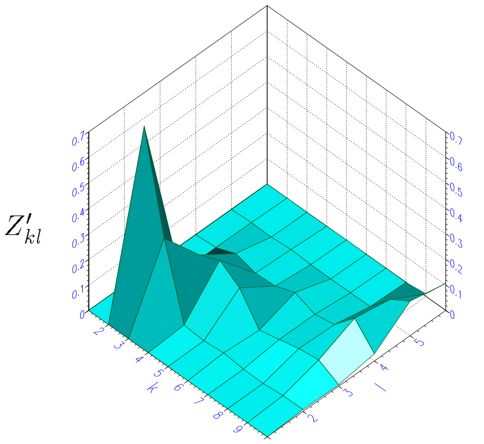
The figure shows the evolution of $\mathbf{v}$ *for a steepest descent network with hysteretic annealing solving a 10-city travelling salesman problem. Since many of the eigenvalues of* $\mathbf{T}^{\mathrm{opr}}$ *are degenerate, the figure displays the* $\mathbf{Z}'$ *matrix, which gives the components of* $\mathbf{v}^{\mathrm{val}}$ *in the degenerate eigenplanes instead of along each degenerate eigenvector. Starting from a random position near the point* $\mathbf{v} = \mathbf{s}$ *(a), the component along the dominant eigenvector is the first to be introduced when* $\gamma = -1.9$ *(b). By the time* $\gamma = -0.1$ *(c), the* $\mathbf{Z}'$ *matrix has evolved to a form not unlike the expected* $\mathbf{Z}^{10}$*, though those elements associated with the more negative eigenvalues have yet to be introduced. Finally, when* $\gamma = 1.5$ *(d),* $\mathbf{v}$ *has successfully converged to a 0-1 point, which is in fact optimal.*



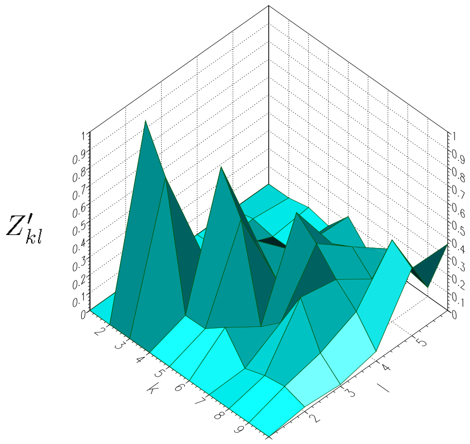
(a) $T^p = 0.5$



(b) $T^p = 0.35$



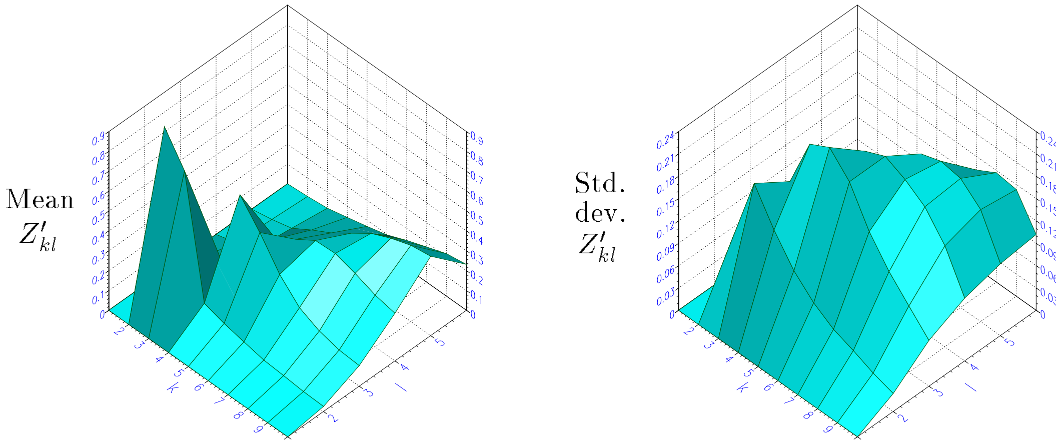
(c) $T^p = 0.30$



(d) $T^p = 0.20$ (Fully converged)

Figure 6.2: Mean field annealing applied to a 10-city travelling salesman problem.

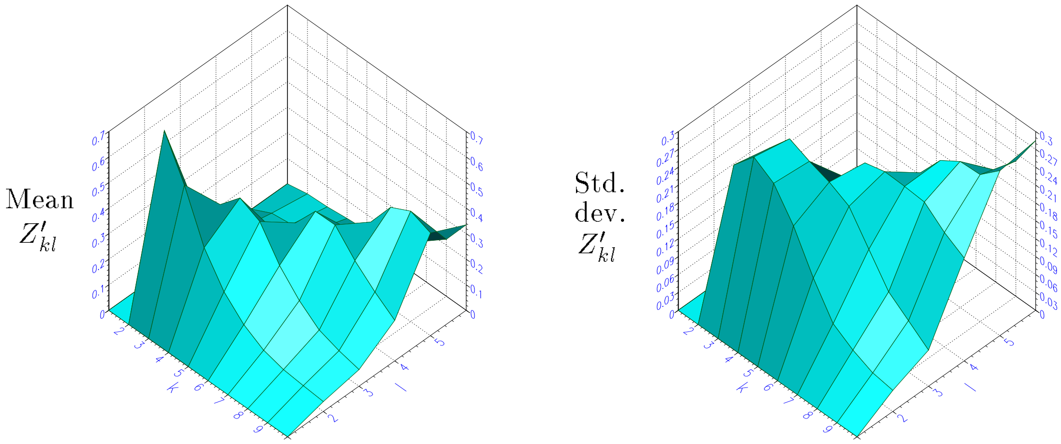
The figure shows the decomposition of $\mathbf{v}^{\mathrm{val}}$ *at several temperatures for the MFA algorithm solving the travelling salesman problem of Figure 6.1. The similarity between the decompositions in Figures 6.1 and 6.2 confirms the theory behind temperature and hysteretic annealing. The MFA algorithm finds the same optimal solution, seen decomposed in (d), as did the steepest descent network with hysteretic annealing.*



(a) Mean of $\mathbf{Z}'$ (b) Standard deviation of $\mathbf{Z}'$

Figure 6.3: Properties of optimal solutions to Euclidean travelling salesman problems.

The optimal solutions to 1000 10-city Euclidean travelling salesman problems were found by exhaustive search. The figure shows the mean and standard deviation of the $\mathbf{Z}'$ *decomposition along the eigenvectors of* $\mathbf{T}^{\mathrm{opr}}$*. The mean decomposition is similar to the* $\mathbf{Z}^{10}$ *form which an optimization network is likely to find, especially around the* Z'_{22} *corner, where the standard deviation of the data is fairly low. Since the components in this corner are associated with the more positive eigenvalues of* $\mathbf{T}^{\mathrm{opr}}$ *(see Figure 6.5(a)), an optimization network will introduce these components first. It therefore seems likely that an optimization network will be successful with these problems, since the initial direction of* $\mathbf{v}$ *is compatible with the location of the optimal solution.*

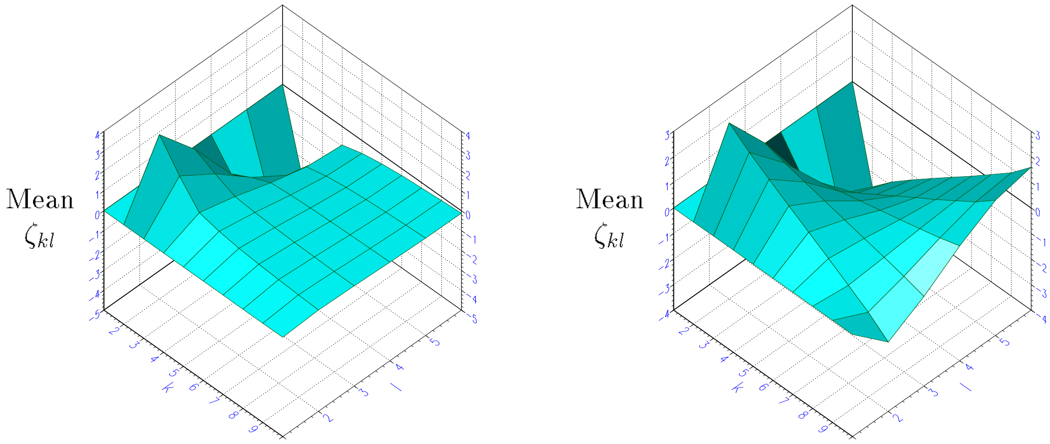


(a) Mean of $\mathbf{Z}'$

(b) Standard deviation of $\mathbf{Z}'$

Figure 6.4: Properties of optimal solutions to random travelling salesman problems.

The optimal solutions to 1000 10-city random travelling salesman problems were found by exhaustive search. The figure shows the mean and standard deviation of the $\mathbf{Z}'$ *decomposition along the eigenvectors of* $\mathbf{T}^{\mathrm{opr}}$. *Compared with the corresponding plots for Euclidean problems in Figure 6.3, the mean* $\mathbf{Z}'$ *matrix is much more diffuse, with higher standard deviations. In particular, the peak at* Z'_{22} *is not so high, and the standard deviation here is also rather high, indicating that the optimal solutions do not necessarily have a large* Z'_{22} *component. Since this component corresponds to the dominant eigenvector (see Figure 6.5(b)) and therefore to the initial direction of* $\mathbf{v}$, *it is expected that optimization networks might be less successful with these problems.*

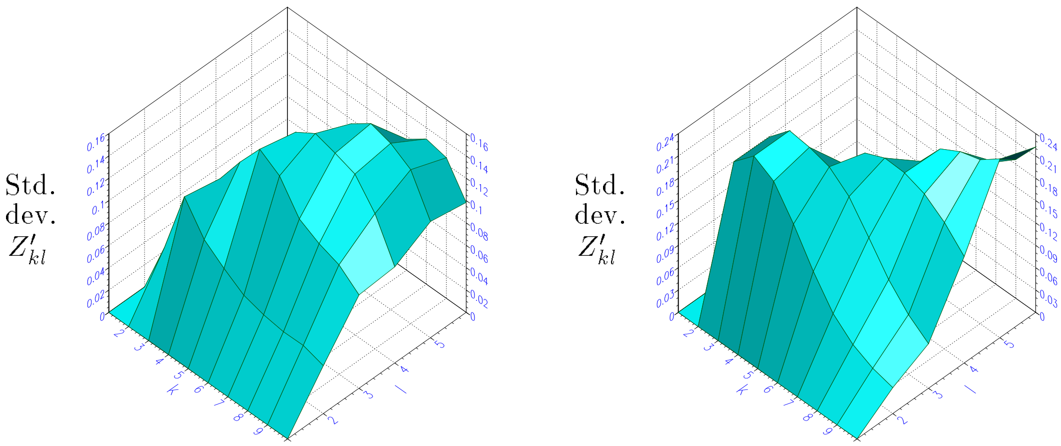


(a) Mean ζ for Euclidean problems

(b) Mean ζ for random problems

Figure 6.5: Eigenvalues of $\mathbf{T}^{\text{opr}}$ for travelling salesman problems.

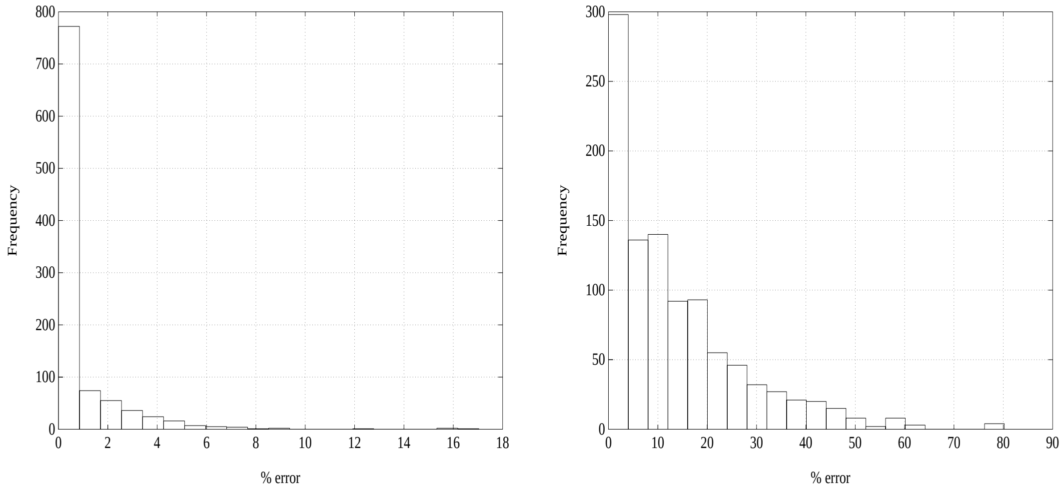
The figure shows the eigenvalue matrix ζ averaged across 1000 10-city Euclidean and random travelling salesman problems. For both classes of problem, the highest eigenvalues are in the (2,2) corner, so an optimization network is expected to introduce the Z_{22} component first.



(a) Euclidean problems (b) Random problems

Figure 6.6: Clustering of good travelling salesman solutions.

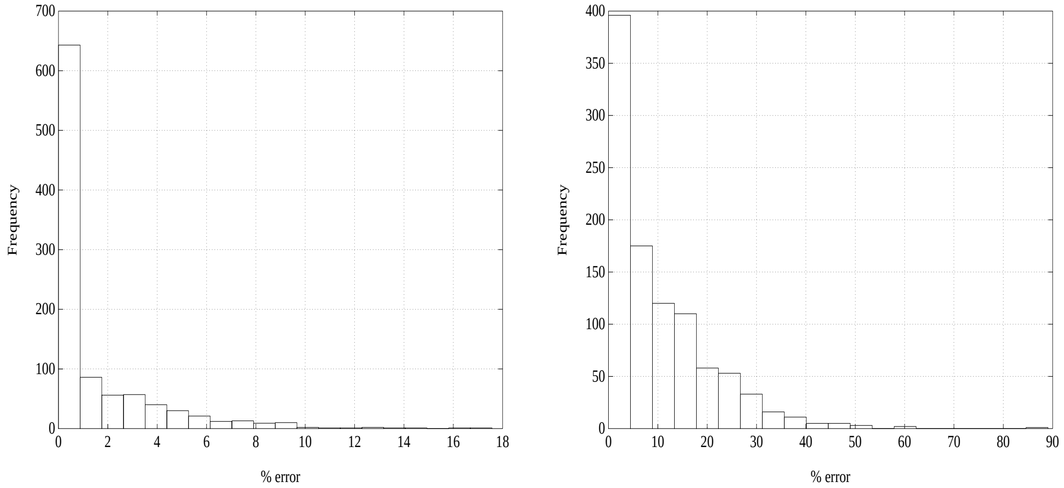
The plots show how the decomposition of $\mathbf{v}^{\mathrm{val}}$ *varies across the ten best solutions to each problem. The data is presented as the standard deviation of the* $\mathbf{Z}'$ *matrix for the ten best solutions, averaged over all 1000 problems in each database. For the Euclidean problems the deviations are low, indicating that good solutions tend to cluster in the solution space. For the random problems, however, the deviations are much larger. This suggests that good solutions to random problems have a far less predictable decomposition, which does not bode well for the likely performance of optimization networks.*



(a) 1000 10-city Euclidean problems (b) 1000 10-city random problems

Figure 6.7: Performance of the steepest descent network on travelling salesman problems.

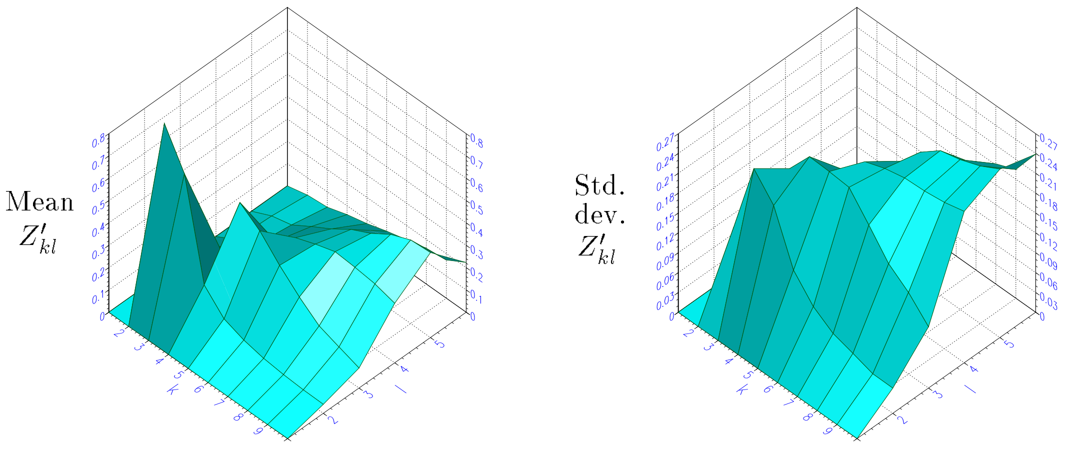
The results are for a steepest descent network running with a slow hysteretic annealing schedule. The error measure is $(E^{\mathrm{op}}_{\mathrm{found}} - E^{\mathrm{op}}_{\mathrm{optimal}})/E^{\mathrm{op}}_{\mathrm{optimal}} \times 100\%$. *The mean percentage error for the Euclidean problems is 0.71%, while the mean error for the random problems is considerably larger at 13.5%. Similar performances were predicted by examining properties of good and optimal solutions.*



(a) 987 10-city Euclidean problems (b) 988 10-city random problems

Figure 6.8: Performance of the MFA algorithm on travelling salesman problems.

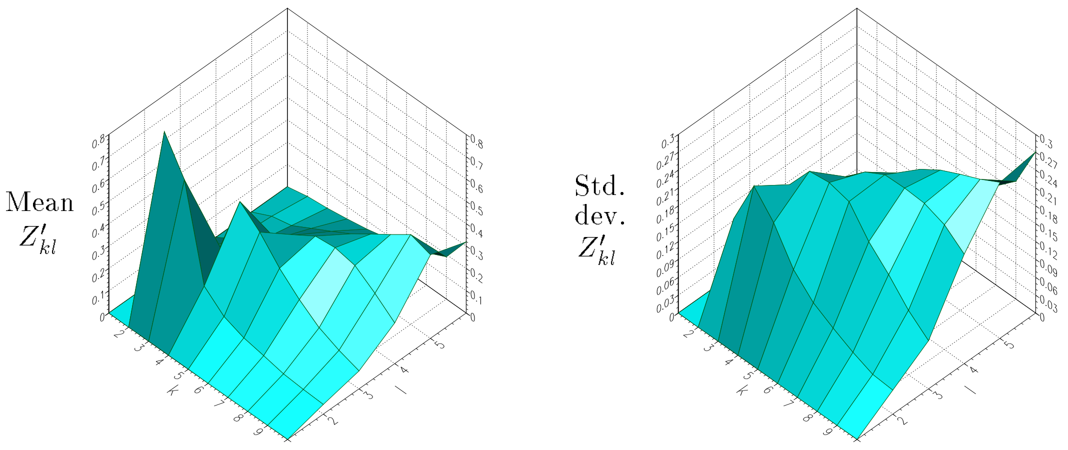
The results are for an MFA algorithm, with neuron normalization, running with a slow temperature annealing schedule: since MFA is most easily stabilized using a moderate *weighting for the penalty function, valid solutions were not found for all 1000 problems. The error measure is specified in the caption to Figure 6.7. The mean percentage error for the Euclidean problems is 1.36%, while the mean error for the random problems is considerably larger at 9.73%. The MFA algorithm performs comparably with the steepest descent network (Figure 6.7) on both Euclidean and random problems, which is not surprising given the similarity between temperature and hysteretic annealing.*



(a) Mean of $\mathbf{Z}'$ (b) Standard deviation of $\mathbf{Z}'$

Figure 6.9: Steepest descent network solutions to random travelling salesman problems.

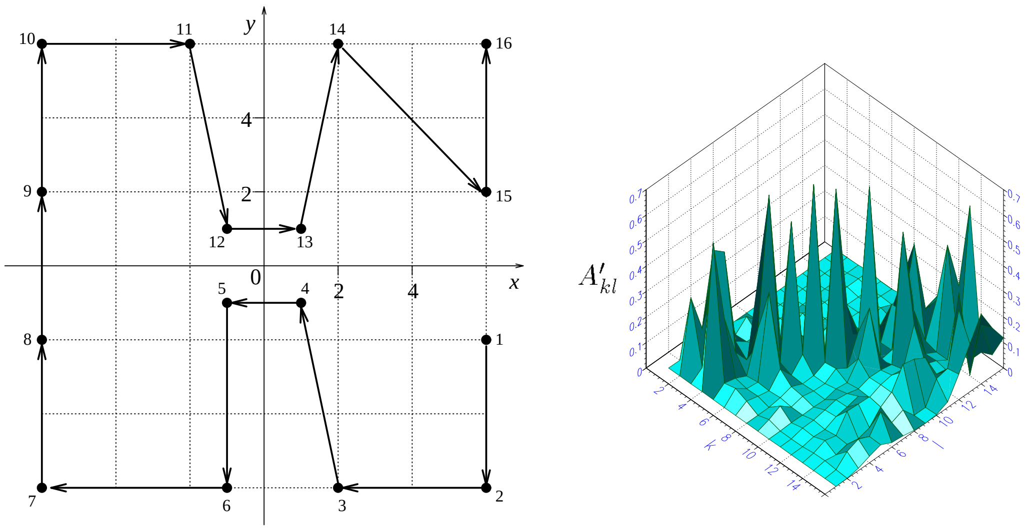
The plots show the decomposition of the solutions found by the steepest descent network to the 1000 random travelling salesman problems. The network found solutions which resemble the $\mathbf{Z}^{10}$ *form much more than the optimal solutions do (Figure 6.4(a)). In particular, optimization networks rely on good solutions having large components along the dominant eigenvector, which is not always the case.*



(a) Mean of $\mathbf{Z}'$ (b) Standard deviation of $\mathbf{Z}'$

Figure 6.10: MFA algorithm solutions to random travelling salesman problems.

The plots show the decomposition of the solutions found by the MFA algorithm to 988 of the random travelling salesman problems. As predicted by the theory, MFA appears to fail for the same reasons as the steepest descent network does (Figure 6.9).



(a) Network solution

(b) Decomposition of optimal solution (matrix $\mathbf{A}'$)

Figure 6.11: Performance of an alternative objective on the spiral Hamilton path problem.

The solution shown in (a) was found by a steepest descent network with hysteretic annealing. The spiral Hamilton path problem of Figure 5.5 was remapped using an alternative objective function so that $\mathbf{i}^{\mathrm{opr}} = \mathbf{0}$*: hence dependence on the auxiliary linear problem is eliminated. However, the solution found is still suboptimal. The decomposition of the optimal solution along the new eigenvectors of* $\mathbf{T}^{\mathrm{opr}}$ *is shown in (b). It is most unlikely that an optimization network will find this solution, since the component* A'_{22} *along the dominant eigenvector is not particularly prominent.*

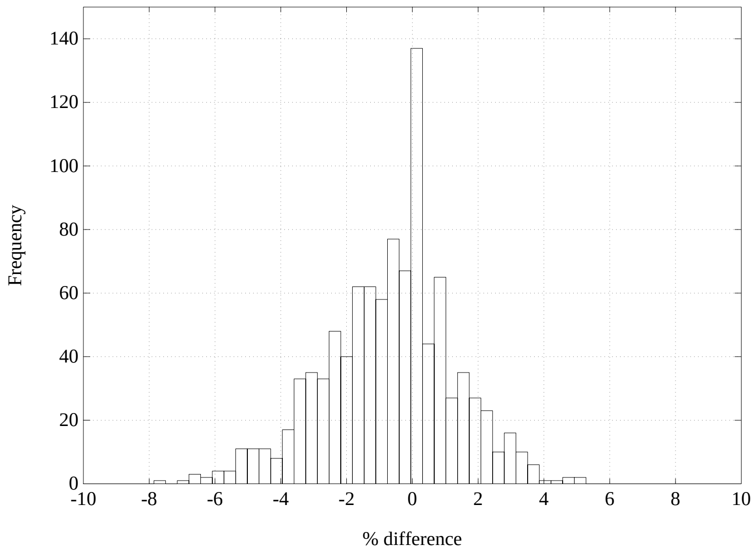


Figure 6.12: Performance of alternative objective functions on graph labelling problems.

The results are for 1000 graph labelling problems, using graphs with randomly generated edge weights. A steepest descent network with hysteretic annealing solved each problem twice, negating the starting position between runs to be sure of obtaining the best possible solution. For each problem two objectives were tried, one standard objective and another formulated so that $\mathbf{i}^{\rm opr} = \mathbf{0}$. $E^{\rm op}$ *was calculated at the best solution found using each objective. The histogram shows the distribution of* $E^{\rm op}$ *for the standard objective minus* $E^{\rm op}$ *for the alternative objective, expressed as a percentage of* $E^{\rm op}$ *for the standard objective. The results indicate that neither of the objectives gives consistently better performance, though improvements were obtained using alternative objectives in about 33% of all cases.*

Chapter 7

Optimization over Non-Integral Polytopes

In Chapter 4 we concluded that conventional optimization networks were largely unsuited to optimization over non-integral polytopes, since they had little chance of converging to a 0-1 point. Moreover, this was not a consequence of any minor, correctable detail, but a direct result of a very basic design misconception: that optimization networks should be able to converge to a solution point in one attempt. Since finding a valid solution point is generally an $\mathcal{NP}$-complete problem, the fundamental conception of optimization networks is flawed. Should we require a valid solution at all costs, we may have to resort to an exponential-time technique. Alternatively, we could attempt to use the LP-relaxation solution, as found by an optimization network, to prime a suitable heuristic which finds a nearby 0-1 point.

In this chapter we investigate two sound, connectionist approaches to solving simple knapsack problems. In Section 7.2 we describe a straightforward heuristic which obtains a good 0-1 solution from the LP-relaxation solution. However, we cannot rely on similar heuristics being available for arbitrary problems. Therefore, in Section 7.3 we present a fundamental redesign of optimization networks which enables them to *search* the vertices of polytopes for valid solution points. For knapsack problems, the performance of the search networks is fairly good, though inferior to that of the LP-relaxation heuristic. However, search networks have general applicability, and at least offer a coherent, connectionist approach to optimization over non-integral polytopes.

7.1 Strategies for optimization over non-integral polytopes

We have now established that a one-shot descent is unlikely to find a valid solution point within a non-integral polytope. We must therefore develop new strategies for optimization over non-integral polytopes. We shall find it useful to consider two cases separately: that the objective E^{op} is linear, as with the knapsack problem, and that the objective is altogether absent, as with pure constraint satisfaction problems like timetable scheduling.

Linear objective

For linear objectives, we argued in Chapter 4 that an optimization network, effectively performing linear programming, will reliably converge to the problem's LP-relaxation solution. This can be used to place a lower bound on the optimal 0-1 solution. Alternatively, we could look for a simple heuristic which uses the LP-relaxation solution to obtain a good 0-1 solution: a suitable heuristic for the knapsack problem is presented in Section 7.2. Failing this, if we require a valid 0-1 solution at all costs, we have no choice but to use an exponential-time technique. We could either find *all* the vertices of the constraint polytope P, using a suitable vertex searching algorithm [85, 103][1], or we could attempt to prune the search using branch-and-bound [93] or dynamic programming [38]. We could also ask ourselves whether a vertex search could be implemented on a modified optimization network. One possible scheme, based on the principle of tabu search, is presented in Section 7.3.

No objective

We must also consider pure constraint satisfaction problems, which have no objective to minimize: such problems include the $\mathcal{NP}$-complete 'teachers and classes' timetabling problem studied in [58], which is essentially an example of resource-constrained multiprocessor scheduling [47, p. 239]. The problem is defined as follows:

> A certain school has N_p teachers, N_q classes and N_x classrooms. It is required to schedule N_l lessons (formally defined as teacher-class pairs) within a time-limit of N_t lesson periods. The problem is to find a suitable schedule, such that no teacher or class is expected to be in more than one place at a time, and no classroom is expected to accommodate more than one lesson at a time.

In [58] the problem is mapped using a 0-1 $N_pN_qN_xN_t$-element solution vector $\mathbf{v}$, which describes how teacher-class pairs (ie. lessons) are to be associated with space-time slots. Note that the problem has no cost function, only hard constraints: all we have to do is find *any* integral vertex within the constraint polytope P. If we wish to apply an optimization network to this problem, we will have to impose an arbitrary objective on the network's dynamics to move $\mathbf{v}$ towards the boundary of P. Any linear or concave function will suffice, though $\mathbf{v}$ will of course converge to a random vertex of P, which will not necessarily be integral.

Timetabling problems can be characterized by their *sparseness*. After all the required lessons have been scheduled, there will be $N_s = (N_xN_t - N_l)$ spare space-time slots: the sparseness of the problem is defined as the ratio $N_s/(N_xN_t)$. Figure 7.1 shows the proportion of integral vertices on typical timetable scheduling polytopes. There is a clear correlation between the sparseness of the problem and the proportion of integral vertices on the corresponding polytope. For sparse problems, the probability of an arbitrary descent procedure converging to an integral vertex is near one, though this falls off rapidly for less sparse problems. For dense problems, there is an additional correlation involving the problem size: as the problem gets larger, the proportion of integral vertices gets smaller. So a one-shot descent procedure, operating on an arbitrary objective function, is not likely to be very successful with difficult constraint satisfaction problems, most probably converging to a non-integral vertex of P: we could, however, invoke the vertex search dynamics, described in Section 7.3, to allow $\mathbf{v}$ to explore further vertices of P.

For problems with a linear objective, the LP-relaxation solution was potentially of some use. However, for pure-constraint satisfaction problems, since the objective is entirely arbitrary, the LP-relaxation is likely to be fairly useless. For example, Table 7.1 shows

[1] For polytopes within the unit hypercube, the number of vertices scales exponentially with the dimension of the hypercube [121, 123].

the unresolved timetable corresponding to a non-integral vertex of a timetable scheduling polytope (an optimization network will probably converge to such a point). At the vertex, many of the elements of $\mathbf{v}$ are in fact zero, ruling out most of the possible schedules. The timetable shows the remaining possible schedules consistent with the nonzero elements of $\mathbf{v}$. It is *not* straightforward to extract a valid timetable from this, as suggested in [58]. But can we do better using other optimization techniques? The answer to this question is yes. The imperfect solution to a scheduling problem found by, say, simulated annealing is likely to be of some use. Since simulated annealing does not search over the interior of the unit hypercube, but only over its vertices, the solution it finds will be a unique timetable, possibly violating one or two constraints. We could at least remove some lessons to obtain a valid timetable which fulfills most of the teaching requirements. Even better, we might be able to swap a few lessons around and arrive at a timetable meeting all the original requirements. No such possibilities are offered by the network's partial solution in Table 7.1.

7.2 Using the LP relaxation solution

Let us remind ourselves of what a knapsack problem is. We have a knapsack of capacity C, and a set of n items. Item i has size x_i and profit y_i. We have to decide which items to pack in the knapsack to obtain the maximum profit from the collection, without overfilling the knapsack. This problem can be posed as quadratic 0-1 programming as follows. Let $v_i = 1$ if item i is to be packed, and $v_i = 0$ otherwise. Then the problem is:

$$\begin{aligned} \text{minimize} \quad & E^{\text{op}}(\mathbf{v}) = -\tfrac{1}{2}\mathbf{v}^T\mathbf{y} && (7.1)\\ \text{subject to} \quad & \mathbf{v}^T\mathbf{x} \leq C && (7.2)\\ \text{and} \quad & v_i \in \{0,1\} \quad i \in \{1\ldots n\} && (7.3) \end{aligned}$$

where C and all the elements of $\mathbf{x}$ and $\mathbf{y}$ are nonnegative. We saw in Chapter 4 that an optimization network, when applied to the solution of a knapsack problem, typically converges to the LP-relaxation solution and fails to find a valid 0-1 point. The LP-relaxation solution $\mathbf{v}^*$ represents the optimal solution to the problem (7.1)–(7.2), without the integrality condition (7.3). It is easy to show that $\mathbf{v}^*$ satisfies $\mathbf{x}^T\mathbf{v}^* = C$. Here we describe a very simple heuristic for obtaining a good 0-1 solution from $\mathbf{v}^*$.

Since E^{op} attains its minimal value within P at $\mathbf{v}^*$, it is expected that any 0-1 points near $\mathbf{v}^*$ will also have low values of E^{op}. In Appendix B.6 we prove that all vertices of a knapsack problem's constraint polytope have at most one non-integral coordinate: it follows that $\mathbf{v}^*$, being a vertex of P, will also have at most one non-integral coordinate. In fact, except in highly pathological cases, $\mathbf{v}^*$ has exactly one non-integral coordinate. If we reduce the value of this coordinate to zero, then we are left with a valid 0-1 solution which is bound to have a fairly low value of E^{op}. This forms the basis of the linear programming technique for solving knapsack problems.

We applied the technique to a set of randomly generated knapsack problems. Each item was allocated a random size and profit, both between 0 and 1, and the knapsack capacity was chosen to be some random fraction of the total size of the items (a different fraction for each problem). Initially small problems were considered, featuring only 10 items, so that exhaustive search could be used to find the optimal solution. For 1000 of these problems, a steepest descent network, running with one slack variable to map the inequality constraint (7.2), was used to find the LP-relaxation solution $\mathbf{v}^*$: see Appendix B.8 for experimental detail. The simple heuristic described above was then used to obtain a 0-1 solution.

The results in Figure 7.2 show that the technique is very effective. Mean solution errors around the 4% mark are perfectly satisfactory when reasonable quality solutions are required in reasonable search times, which is the best we can aim for when dealing with $\mathcal{NP}$-complete problems. Larger knapsack problems are considered in Section 7.3.

7.3 Tabu search networks

In this section we describe how tabu search dynamics, previously presented in [19, 20] as a viable neural search technique, can be modified to provide a means of searching the vertices of non-integral polytopes with an optimization network.

Tabu search [59] was originally developed as a *meta-heuristic*, to work in conjunction with other search techniques. The idea is that certain parts of the search space are designated tabu for some time, directing the search towards more fruitful areas. The choice of appropriate tabu strategies can lead to highly efficient searches, with some extremely impressive results in the field of combinatorial optimization [59].

Applied to descent networks, tabu search suggests a dynamic objective function, where we continuously update E to penalize points that $\mathbf{v}^+$ has already visited. In this way, if $\mathbf{v}^+$ gets trapped at any point, eventually E will build up locally to such an extent that $\mathbf{v}^+$ is driven away. In tabu terminology, we mark recently visited points as tabu, and direct the search into fresh areas of space. What emerges is a neural *search* technique, which no longer converges to a single point, and therefore needs constant monitoring to spot any valid solution points the search may pass through. This should be contrasted with the one-shot descent offered by conventional optimization networks, which typically converge to non-integral points. To be sure of finding an integral point the use of some sort of search procedure is imperative. The tabu methodology makes this search efficient, while a connectionist implementation in hardware could provide very rapid processing speeds.

The implementation of the search on feedback networks is particularly elegant. In what follows we build on the approach in [19, 20], adapting it where necessary to achieve our specific goal of escape from vertices of P^+. We consider a time-varying objective function

$$E_t^{\mathrm{op}}(\mathbf{v}^+, t) = E^{\mathrm{op}}(\mathbf{v}) + F_t(\mathbf{v}^+, t) \tag{7.4}$$

where

$$F_t(\mathbf{v}^+, t) = \beta \int_0^t e^{\alpha(s-t)} p\left(\mathbf{v}^+, \mathbf{f}(\mathbf{v}^+(s))\right) \, ds \tag{7.5}$$

In (7.5), α and β are positive constants, $p(\mathbf{a}, \mathbf{b})$ measures the proximity of the vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{v}^+(s)$ denotes the location of the network's state vector at time s, and $\mathbf{f}(\mathbf{a})$ is a small displacement function such that $\mathbf{f}(\mathbf{a}) \approx \mathbf{a}$. Thus F_t evolves in time to increase the objective in the vicinity of points recently visited. Let us now consider how F_t affects the descent dynamics for $\mathbf{v}^+$ trapped at a vertex $\mathbf{v}_\mathrm{o}^+$ of P^+ since time $t = 0$. In this case

$$F_t(\mathbf{v}^+, t) = \beta \int_0^t e^{\alpha(s-t)} p\left(\mathbf{v}^+, \mathbf{f}(\mathbf{v}_\mathrm{o}^+)\right) \, ds$$

If we set $\mathbf{f}(\mathbf{a}) = \mathbf{a}$, then even though the objective is increased at the vertex, escape is not guaranteed since the maximum of F_t will be at $\mathbf{v}_\mathrm{o}^+$, and therefore the gradient of F_t at $\mathbf{v}_\mathrm{o}^+$

will be zero. What is required is to place the maximum of F_t *near* $\mathbf{v}^+_{\mathrm{o}}$ but just *outside* the polytope P^+. This can be achieved if we set

$$\mathbf{f}(\mathbf{a}) = \mathbf{a} + \epsilon(\mathbf{a} - \mathbf{c}) \tag{7.6}$$

where $\mathbf{c}$ is some vector within P^+, and ϵ is a small, positive constant. If we choose the quadratic proximity function $p(\mathbf{a}, \mathbf{b}) = -\|\mathbf{a} - \mathbf{b}\|^2$, then F_t becomes

$$F_t(\mathbf{v}^+, t) = -\tfrac{1}{2}\theta \mathbf{v}^{+T}\mathbf{v}^+ - \mathbf{h}^T\mathbf{v}^+ + g(t) \tag{7.7}$$

$$\text{where} \quad \theta(t) = 2\beta \int_0^t e^{\alpha(s-t)}\, ds \tag{7.8}$$

$$\text{and} \quad \mathbf{h}(t) = -2\beta \int_0^t e^{\alpha(s-t)} \mathbf{f}(\mathbf{v}^+(s))\, ds \tag{7.9}$$

Equations (7.8) and (7.9) can be differentiated to obtain the following dynamic update equations for θ and $\mathbf{h}$

$$\dot{\theta} = 2\beta - \alpha\theta \tag{7.10}$$

$$\dot{\mathbf{h}} = -2\beta \mathbf{f}(\mathbf{v}^+(t)) - \alpha \mathbf{h} \tag{7.11}$$

with initial conditions $\theta(0) = 0$ and $\mathbf{h}(0) = \mathbf{0}$. The tabu search system is obtained by arranging for the overall objective function to be

$$E(\mathbf{v}^+) = -\tfrac{1}{2}\mathbf{v}^{+T}\mathbf{T}\mathbf{v}^+ - \mathbf{v}^{+T}\mathbf{i}^{\mathrm{b}} = E_t^{\mathrm{op}}(\mathbf{v}^+, t) - g(t) + \tfrac{1}{2}c\|\mathbf{v}^+ - (\mathbf{T}^{\mathrm{val}}\mathbf{v}^+ + \mathbf{s})\|^2 \tag{7.12}$$

where we have discarded $g(t)$, the part of F_t which was not dependent on $\mathbf{v}^+$. Remember that the c term confines $\mathbf{v}^+$ to the polytope P^+, as discussed in Section 3.2. The objective function (7.12) is achieved by setting

$$\mathbf{T} = \mathbf{T}^{\mathrm{op}+} + c(\mathbf{T}^{\mathrm{val}} - \mathbf{I}) + \theta(t)\mathbf{I} \tag{7.13}$$

$$\mathbf{i}^{\mathrm{b}} = \mathbf{i}^{\mathrm{op}+} + c\mathbf{s} + \mathbf{h}(t) \tag{7.14}$$

The state vector $\mathbf{v}^+$ can be updated using either Hopfield or steepest descent dynamics on E, though the dynamics of $\mathbf{v}^+$ will now depend on θ and $\mathbf{h}$. In turn, the dynamics of $\mathbf{h}$, given in (7.11), depend on $\mathbf{v}^+$. The system of coupled, first order differential equations is surprisingly simple and quite possibly retains the attraction of being implementable in analogue hardware. Note that only the diagonal interconnections and input bias currents are time-varying, with the former approaching an asymptotic value of $2\beta/\alpha$.

The tabu search dynamics are intended for use over non-integral polytopes with linear or concave objectives. Indeed, if E^{op} is linear or concave, then it is straightforward to show that E_t^{op} is also linear or concave for all time t, and so there are no local minima of E_t^{op} in which $\mathbf{v}^+$ can get trapped. Ability to escape from any vertex of P^+ can be guaranteed for appropriate settings of the parameters α, β and ϵ — see Appendix B.7. For moderate β, the trajectory of $\mathbf{v}^+$ is initially dominated by the gradient of E^{op}, until $\mathbf{v}^+$ gets trapped at a vertex of P^+. At this point the tabu objective F_t proceeds to dominate, and eventually $\mathbf{v}^+$ escapes from the vertex and moves towards another. We can therefore expect $\mathbf{v}^+$ to search through a number of vertices, with an initial bias towards those minimizing E^{op}. The search might exhibit limit cycles: it is a matter of future research to develop continuous dynamics which search *every* vertex of a polytope, given enough time.

The tabu search system was implemented with steepest descent dynamics (2.10). With tabu search, the steepest descent dynamics have the additional advantage over the Hopfield dynamics that reaction to changes in F_t is more rapid. This is because there are no $\mathbf{u}$ variables, which can stray far from the midpoint of the transfer functions (2.4), and subsequently take a long time to return and effect significant changes in $\mathbf{v}$. Figure 7.3 shows the trajectory of $\mathbf{v}^+$ under the tabu dynamics, with $\mathbf{T}$ and $\mathbf{i}^{\mathrm{b}}$ set to map the knapsack problem of Figure 4.1. We see that the search passes through all the vertices of P^+, including the valid solution points B, C and D. In fact, the first vertex to be visited after the LP-relaxation solution at A is vertex B, which is the optimal solution to the knapsack problem. Notice that the coupled dynamic system is rather slow, taking over one minute to search through all the vertices. Most of the time is expended when $\mathbf{v}^+$ is trapped at one of the vertices A or C, where it takes some time for θ and $\mathbf{h}$ to change to such an extent that $\mathbf{v}^+$ is freed from the vertex. However, it is possible to analytically integrate the tabu equations for stationary $\mathbf{v}^+$, leading to more efficient simulations if this possibility is exploited. Carrying out the integration reveals that the speed of the continuous system is approximately proportional to α, so significant speed-ups are possible with larger values of α (see Appendix B.7)[2].

The technique was applied to the database of solved knapsack problems studied in Section 7.2. For all 1000 problems, steepest descent dynamics were used on the time-varying objective function (7.12): see Appendix B.8 for experimental detail. The search was run for a fixed number of iterations on all problems, and the network output was continuously observed, so that valid solutions could be logged as the network found them.

The results in Figure 7.4 reflect the best solution found within the iteration limit. With valid solutions found over 99% of the time, and mean solution errors around the 5% mark, the results are comparable with those of the linear programming technique. The performance of the system was found to be largely insensitive to changes in the parameters α, β and ϵ, so long as the ability to escape from a vertex was maintained (see Appendix B.7). Limit cycles were observed in the state vector trajectory, though these typically included visits to many vertices.

For larger knapsack problems, even though we cannot search for the optimal solution, we can compare the performances of various approximate solution techniques. Table 7.2 shows how the tabu search and linear programming approaches compare with a simple greedy heuristic for problems in up to 250 items. The greedy approach is simply to pack items in order of decreasing profit, omitting any item if it causes the knapsack to overflow. The tabu search system works as well as the very effective linear programming approach, and consistently outperforms the greedy heuristic.

The work on tabu dynamics presented here is intended merely as a pointer to more coherent approaches to optimization over non-integral polytopes. The dynamics, as they stand, have the serious drawback that the resulting search does not encompass all the polytope's vertices. The reason why the technique is relatively successful with knapsack problems lies in the distribution of integral vertices on the corresponding polytopes. Figure 7.5 indicates that approximately half the vertices of knapsack polytopes are integral, independent of the problem size. So any search taking in just a few vertices stands a good chance of finding an integral one. Furthermore, good solution points (in terms of solution quality) are often located close to the LP-relaxation solution, which is where the

[2]Small values of α are used in the experiments here, since this simplifies the simulation of the continuous dynamics — see Appendix B.8.

descent initially leads $\mathbf{v}^+$. These special properties make the knapsack problem particularly amenable to solution by the tabu search technique. However, as we have already seen for scheduling problems in Figure 7.1, we cannot expect other problems to exhibit similar properties. It remains to stress that even if the dynamics can be modified to produce an exhaustive vertex search, they will have to be implemented in analogue hardware to appeal, in terms of speed, over alternative optimization techniques.

	Room 1	**Room 2**	**Room 3**	**Room 4**
Time 1	Mr. A./Class 1 Ms. D./Class 1	Mr. B./Class 3 Ms. C./Class 4	Mr. B./Class 2 Ms. C./Class 2	Mr. A./Class 4 Ms. D./Class 3
Time 2	Mr. A./Class 2 Mr. B./Class 1	Ms. C./Class 3 Ms. D./Class 4	Mr. B./Class 1 Ms. D./Class 3	Mr. A./Class 2 Ms. C./Class 4
Time 3	Mr. B./Class 1 Ms. D./Class 3	Mr. A./Class 4 Ms. C./Class 2	Mr. B./Class 3 Ms. C./Class 2	Mr. A./Class 1 Ms. D./Class 4
Time 4	Ms. C./Class 3 Ms. D./Class 3	Mr. B./Class 1 Ms. C./Class 2	Mr. A./Class 4	Mr. B./Class 2 Ms. D./Class 1

Table 7.1: Unresolved timetable for a non-integral vertex of a dense scheduling polytope.

An optimization network has converged to a non-integral vertex, as is typical for dense scheduling problems. However, many of the elements in the solution vector are in fact zero, ruling out the vast majority of schedules. The unresolved timetable shows all the remaining possible schedules; there are generally two lessons which could be placed in each space-time slot. It is not *straightforward, as suggested in [58], to extract a valid timetable from this, and so the network's partial solution is fairly useless.*

Solution technique	**Number of items**				
	50	100	150	200	250
Linear programming	0.0%	0.0%	0.0%	0.0%	0.0%
Tabu search network	+0.37%	+0.06%	−0.01%	−0.31%	−0.34%
Greedy heuristic	−11.7%	−7.1%	−6.0%	−6.9%	−14.1%

Table 7.2: Average solution qualities for large knapsack problems.

The table compares the performances of several solution techniques on large knapsack problems. The average solution qualities ($-E^{\rm op}$) over ten problems are given relative to the solutions found by the linear programming technique. The results show that the tabu search dynamics work as well as the linear programming approach, and consistently outperform the greedy heuristic. The tabu search parameters were $\alpha = 0.01, \beta = 1.0, \epsilon = 0.1$.

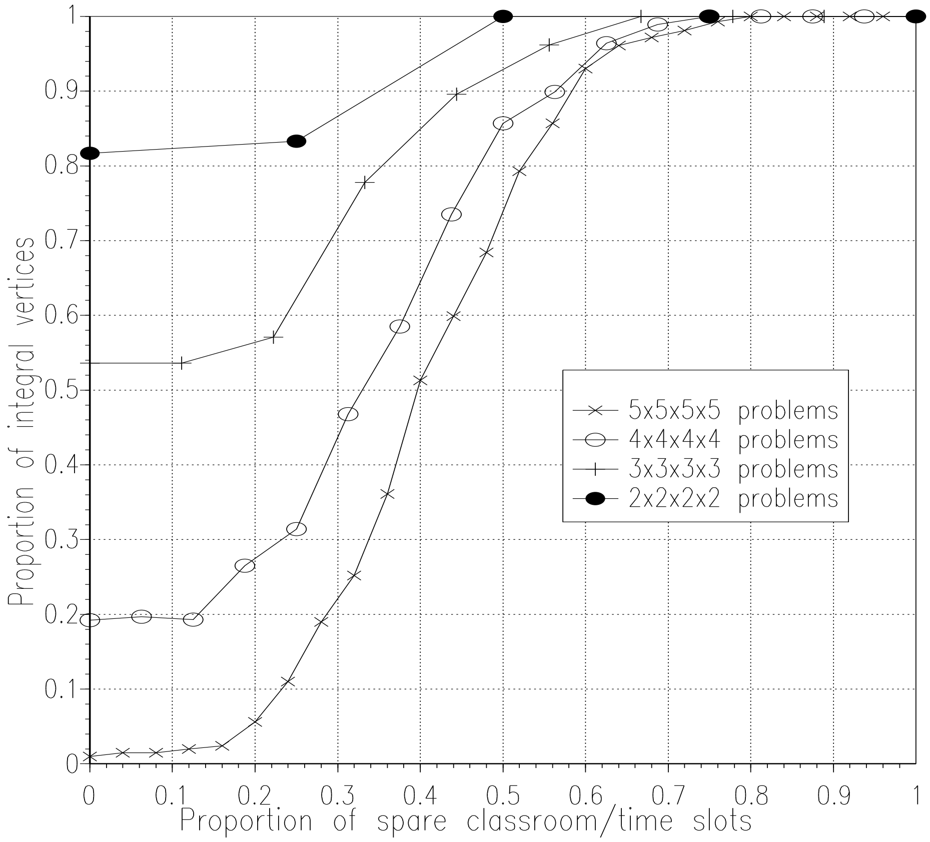


Figure 7.1: Proportion of integral vertices for timetable scheduling polytopes.

The data relates to timetable scheduling problems [58] of various sizes. Each problem considers equal numbers of teachers, classes, classrooms and time periods, ranging from two to five of each. For each size, problems of different sparseness *were generated, ranging from problems where all the space-time slots were to be filled with lessons, to problems where no lessons were to be scheduled at all. The resulting polytopes were examined with the aim of estimating the proportion of integral vertices. A simplex algorithm [119] was run with 1000 random, linear objectives to find 1000 vertices of each polytope (an optimization network could equally have been used to perform the linear programming, but the simplex algorithm runs much faster on a digital machine). The results can be used to infer the probability of an optimization network converging to a 0-1 vertex. It is clear that this probability is rapidly diminished for larger and less sparse problems.*

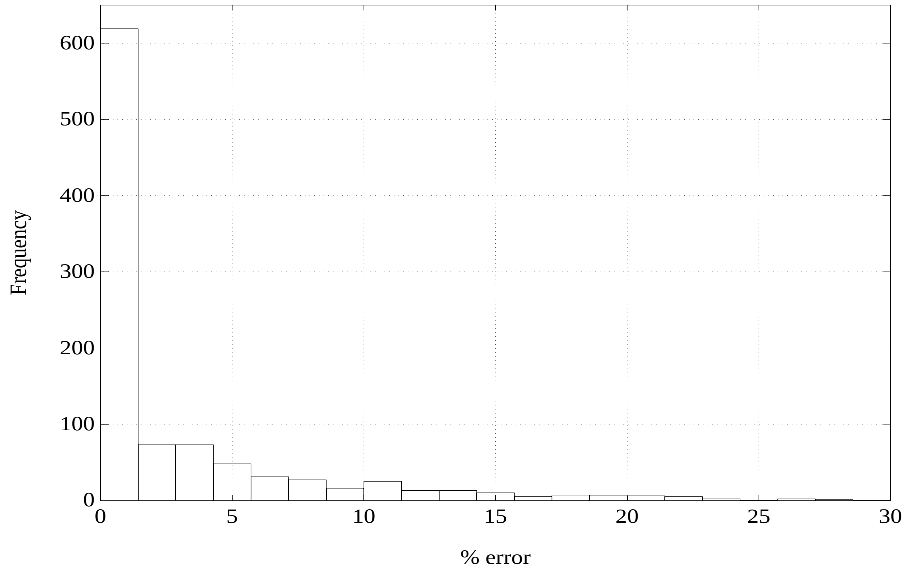
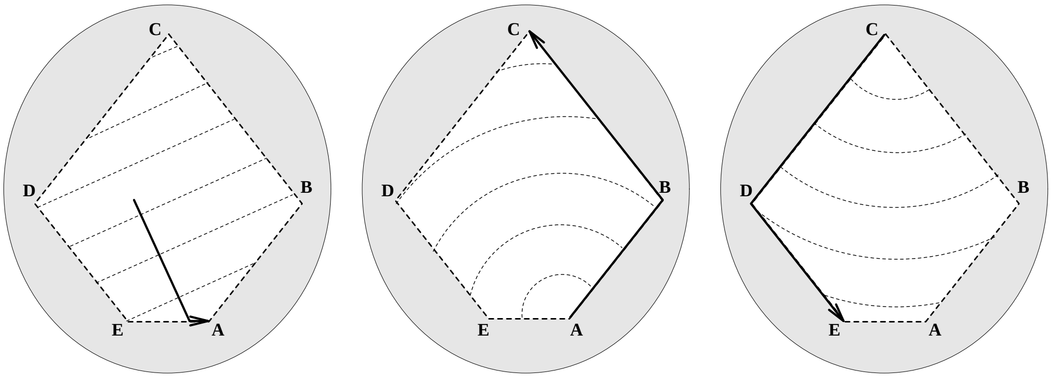


Figure 7.2: Performance of the linear programming technique on knapsack problems.

Solutions to 1000 10-item knapsack problems were found using linear programming combined with a simple heuristic. A steepest descent network was used to find the LP-relaxation solution $\mathbf{v}^*$*, the one nonzero element being subsequently reduced to zero to obtain a 0-1 solution. The error measure used to display the results is* $(E^{\rm op}_{\rm optimal} - E^{\rm op}_{\rm found})/E^{\rm op}_{\rm optimal} \times 100\%$*. The errors have a mean of 3.72% and a standard deviation of 9.24%.*



(a) $t = 0$ s (b) $t = 6.3$ s (c) $t = 76.4$ s

Figure 7.3: An illustration of tabu search.

The figure illustrates how tabu search can be applied to the simple knapsack problem first presented in Figure 4.1. Conventional descent dynamics converge to the non-integral vertex A*, which does not represent a valid solution. With tabu search dynamics, the objective function is continuously updated to penalize points that* $\mathbf{v}^+$ *has already visited: contours of* E *are shown in dotted lines. Hence* $\mathbf{v}^+$ *is ejected from* A *and passes through the optimal solution vertex* B*, before converging temporarily to* C*. After a further period of time the evolving objective ejects* $\mathbf{v}^+$ *from* C*, whereupon the remaining two vertices* D *and* E *are visited. The simulation was performed using steepest descent dynamics with tabu search parameters* $\alpha = 0.01, \beta = 2.0, \epsilon = 0.1$*. Faster search speeds are possible with larger values of* α*.*

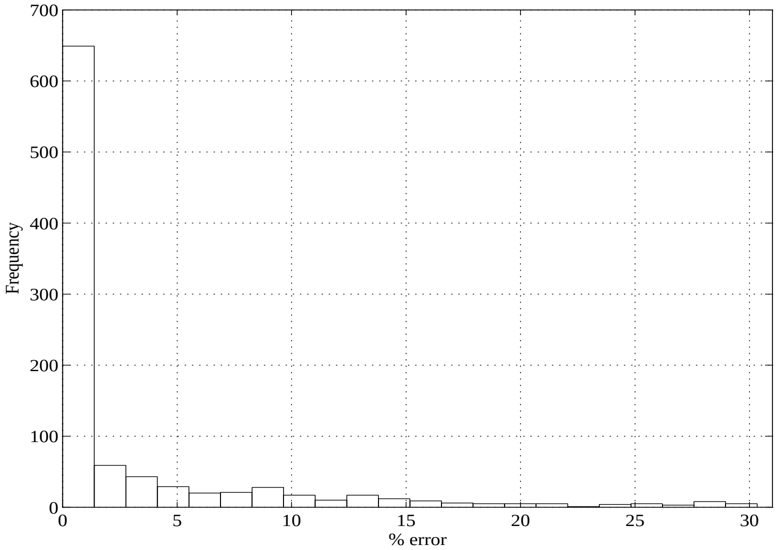


Figure 7.4: Performance of tabu search dynamics on knapsack problems.

Solutions to 10-item knapsack problems were found using steepest descent dynamics with tabu search. The error measure used to display the results is specified in the caption to Figure 7.2. Valid solutions were found for 995 out of the 1000 problems. The errors have a mean of 4.56% and a standard deviation of 10.3%. The tabu search parameters were $\alpha = 0.01, \beta = 1.0, \epsilon = 0.1$.

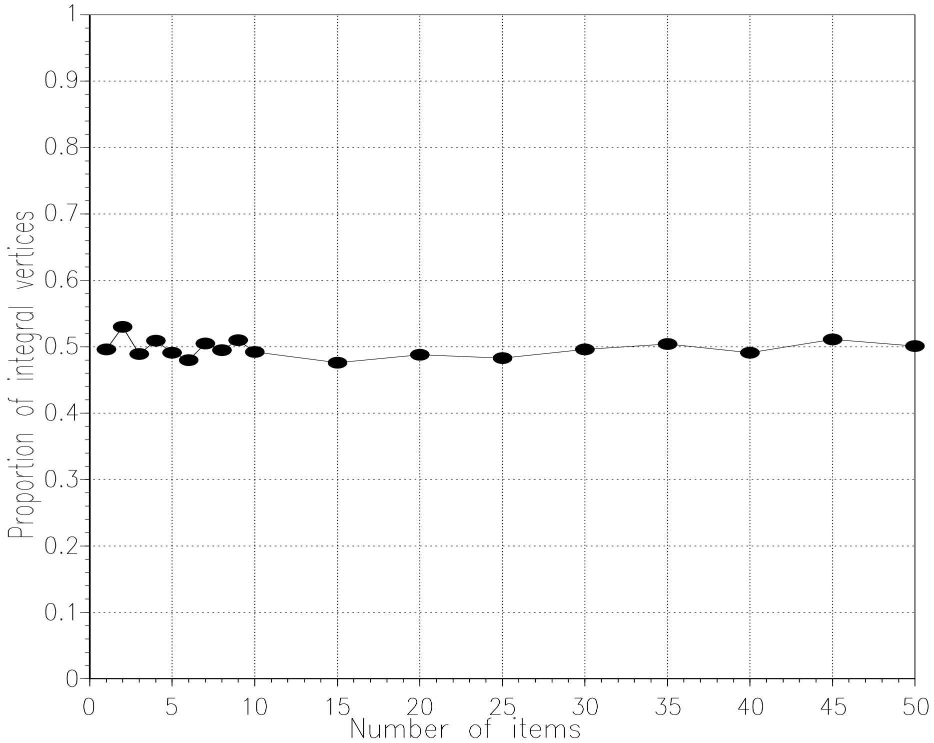


Figure 7.5: Proportion of integral vertices for knapsack polytopes.

The data relates to randomly generated knapsack problems: each item was allocated a random size and profit, both between 0 and 1, and the knapsack capacity was chosen to be some random fraction of the total size of the items. The proportion of integral vertices on the resulting polytopes was estimated using the simplex technique described in the caption to Figure 7.1. It is apparent that about 50% of the vertices are integral, independent of the problem size. This partly explains why the tabu search dynamics are relatively successful with these problems, even though the search takes in only a limited number of vertices. In addition, since good 0-1 solution points are located close to the LP-relaxation solution, which is where the descent dynamics initially lead $\mathbf{v}^+$, *the solutions found by the search are often correspondingly good.*

Chapter 8

Conclusions

In Chapter 1 we introduced this thesis with a clear justification for investigating optimization networks: they appeared to offer the unique facility of being able to solve small to medium sized problems of a general nature in a matter of milliseconds, with many useful applications. We then identified three key questions requiring authoritative answers:

- Is there a simple way to program an optimization network for an arbitrary problem, so that the network will never find a solution violating any of the problem's hard constraints?
- Given this method, is it always possible to force the network to converge to an interpretable, valid solution?
- If so, will the quality of this valid solution be high enough to compete with the other optimization techniques?

This thesis has been largely concerned with answering these questions.

The investigation into problem mappings came to a very positive conclusion: given a combinatorial optimization problem expressed in terms of quadratic 0-1 programming (which is fairly standard practice), there is a very straightforward way to program an optimization network for its solution. The mapping guarantees that any solution found by the network will not violate any of the problem's constraints.

Unfortunately, the study of network convergence came to a less positive conclusion. Drawing on results from complexity theory and mathematical programming, hitherto ignored by the neural network community, we showed that it is generally not possible to force convergence to a 0-1 point, except for special classes of problems over integral polytopes. One of the justifications for researching optimization networks is therefore refuted: they cannot be applied to such a wide variety of problems. The tabu search dynamics, which we presented for optimization over non-integral polytopes, were intended merely as a pointer towards a more coherent approach: they remain far from being competitive with other optimization techniques. Fortunately, many useful problems exhibit the amenable property of an integral constraint polytope. It therefore seemed worthwhile investigating the final question concerning the quality of network solutions.

This was perhaps the hardest question to answer, requiring a mixture of theoretical and experimental analysis to obtain some partial, though illuminating, answers. With the single exception of the Euclidean travelling salesman problem, we found no class of problem for which the network solution was completely reliable. When solving Hamilton path problems, the network was sometimes thwarted by an inappropriate auxiliary linear problem; to solve a graph labelling problem, it was often necessary to run the network twice to find just one good solution; random travelling salesman problems exhibited none of the properties that optimization networks require if they are to perform well. A novel study of alternative energy functions showed that their use could potentially, but not reliably, improve solution quality.

Optimization networks are therefore restricted in their application to problems with amenable properties over integral polytopes. Furthermore, they rely on suitable hardware implementation to compete with other optimization techniques, which, when tailored to specific problems, can perform extremely well. The Euclidean travelling salesman problem, the only problem considered in this thesis which optimization networks solved reliably well, serves to illustrate the point. To compete with even a 1965 algorithm [97], a network would have to find a good solution to a 100-city problem within a few seconds: this would certainly require implementation in special, parallel hardware. Even then, the space requirements of most mappings would limit the application of hardware networks to problems of moderate size. It is clearly far-fetched to envisage a circuit with four million processing elements solving a 2000-city travelling salesman problem, as can be done using alternative techniques [46].

Optimization networks emerge from this thesis with a narrow, *potential* field of application, conditional on the development of suitable hardware. They are therefore not as attractive as they might have originally seemed.

Appendix A

Bibliographical Notes

This appendix contains a review of the literature covering the solution of combinatorial optimization problems, and the theory and application of optimization networks.

A.1 Traditional approaches to combinatorial optimization

A.1.1 Cutting plane techniques

Exact approaches to combinatorial optimization divide into two main classes. First, there are the *cutting plane* techniques, originally due to Gomory [61]. From an integer linear programming perspective, the idea behind this method is to successively add extra constraints which reduce the size of the constraint polytope, without excluding any feasible integer points. After each new constraint (or *cut*) has been added, the LP-relaxation solution is found using the simplex algorithm. At each step, either the LP-relaxation solution is an integer, at which point the problem is solved, or else the LP-relaxation solution is used to generate a new cut. Cutting plane algorithms describe how to generate cuts so that no feasible points are excluded at each cut, and so that the algorithm converges in a finite (exponentially bounded) number of steps.

A.1.2 Branch-and-bound

In addition to cutting place techniques, there are the *enumerative* methods, based on intelligent enumeration of all feasible solutions. The most common enumerative procedure is *branch-and-bound*, well surveyed up to 1966 in [93]. At each stage of the branch-and-bound procedure, the set of possible solutions is partitioned into ever smaller mutually exclusive sets: this is the *branch* operation. An efficient algorithm is then used to compute a lower bound on the cost of any solution in each set: this is the *bound* operation. As the sets become smaller, and clearly in the limit of the sets containing only a single solution, it becomes possible to identify the best feasible solution in a set: at this point exploration of this set can cease. Exploration of a set can also be halted if the lower bound is inferior to any feasible solution found so far. Eventually it is possible to stop exploring all the remaining solution sets, for one of the above two reasons. At this point the problem is solved, since the best feasible solution found *must* be the optimal solution to the problem. The branch-and-bound algorithm provides a way of enumerating all feasible solutions without having to consider each and every one. However, this can still take some time, and it is often necessary to terminate the branch-and-bound algorithm before optimality is reached. In such a situation a lower bound on the optimal solution is available, as well as the best solution found so far, so the maximum error from the optimum can be calculated.

A.1.3 Dynamic programming

Another enumerative technique is *dynamic programming*, which is comprehensively surveyed in [38]. Before the technique can be applied, the problem must be posed as a *multistage decision process*: a process in which a sequence of decisions is made, the choices available being dependent on the previous decisions. Dynamic programming exploits the *principle of optimality*:

> An optimal sequence of decisions has the property that whatever the initial state and initial decision are, the remaining decisions must be an optimal sequence of decisions with regard to the state resulting from the first decision [120].

Essentially, this means that solutions to subproblems can be used to prune the search for solutions to larger subproblems, and finally to the problem itself. Dynamic programming works well with many common problems, providing, for example, a pseudo-polynomial time algorithm for the knapsack problem [47].

In practice, enumerative methods have found many more applications than cutting plane methods. Full descriptions of all these approaches can be found in integer programming texts [18, 48, 73, 83, 111, 122, 123, 153], or more general combinatorial optimization texts [106, 114, 120].

A.1.4 Local search

When exact techniques still take too long, inexact heuristics, tailored to particular problems, often perform extremely well. A good example is *local search*. Starting with some feasible solution, a subroutine is called to search for improved solutions within a small neighbourhood of the initial solution. The best improvement is stored, and the subroutine is called again to attempt to improve this new solution. When no further iterative improvement is possible, the algorithm halts. The skill in designing local search algorithms lies in identifying suitable small neighbourhoods in which to search for improvements. With a good choice of neighbourhood, local search can perform amazingly well. For example, Lin [97] experimented with the following *3-change* neighbourhood for the travelling salesman problem:

> Tour B is within the 3-change neighbourhood of tour A if it can be obtained by removing 3 edges from tour A and replacing them with 3 other edges.

The resulting algorithm, which was further refined in [98], is a much cited benchmark for the efficient solution of the travelling salesman problem [92]. Similar local search techniques also perform well on graph partitioning problems [84].

A.2 Novel approaches to combinatorial optimization

A.2.1 Simulated annealing

The *simulated annealing* algorithm [1, 36], first presented by Kirkpatrick et al. in 1983 [86], is inspired by phenomena in statistical physics. The physical process of annealing describes how a substance can be *slowly* cooled from a high temperature to a low one, resulting in a low energy distribution of the atoms or molecules within the substance. The rate of cooling is important: a rapid quench will leave the substance stranded in a high energy state. An analogy can be drawn with the process of solving combinatorial optimization problems, where it is required to find a low energy state of a set of variables. In [86] it is argued that iterative improvement techniques for solving optimization problems (like the local search procedures described in Section A.1) are similar in nature to the microscopic rearrangement processes taking place within any physical substance above absolute zero temperature, with the cost function mirroring the physical system's energy. The analogy is extended to argue that accepting only rearrangements which lower the cost function is like a rapid quench of a physical substance, leading most probably to poor solutions. The Metropolis algorithm [104] provides a means of allowing controlled uphill steps, with a nonzero probability P, in the search for a better solution. P is set to be a particular function of a *temperature* parameter T, with the result that the system evolves into a Boltzmann distribution at each temperature. As the simulated annealing progresses, T is gradually lowered until, at zero temperature, P is zero and some local minimum of the cost function is arrived at. In the (impractical) limit of an infinitely slow cooling schedule, simulated annealing is guaranteed to find the optimal solution to any problem.

Simulated annealing and neural network techniques can be combined to give a stochastic, parallel solution algorithm for 0-1 quadratic programming problems. The standard network dynamics are modified to allow for asynchronous operation, so that each neuron updates itself periodically with a nonzero probability P of making an uphill move. As with standard simulated annealing, P is related to a temperature parameter which is slowly lowered as the network converges. Networks operating in this manner are most often referred to as *Boltzmann Machines* [1, 154]. An alternative interpretation of this procedure is that an ever decreasing amount of noise is superimposed on top of the conventional network dynamics [10].

Simulated annealing procedures have the drawback that it often takes a very slow cooling schedule, and a correspondingly long amount of time, to obtain good solutions. This is a direct consequence of the stochastic nature of the algorithms: the system requires enough time at each temperature to acquire the correct statistical distribution through purely random rearrangements.

A.2.2 Genetic algorithms

Genetic algorithms [35, 36, 60] are another important, novel optimization technique. They are inspired by the theory of Darwinian evolution, in which the principle of *survival of the fittest* is used to decide which species survive into the next generation. As time progresses, the evolving generations of species become ever more suited to survival in their habitat. This principle can be extended to the solution of combinatorial optimization problems. The genetic algorithm starts with a randomly generated population of feasible solutions. These are all tested against the cost function, and the best solutions are kept and used to generate a new population by *reproduction*. The reproduction process typically involves combining features from two or more parent solutions to produce a child solution. *Mutations* of individual solutions are also allowed. The key to a good genetic algorithm lies in the design of good reproduction rules. With effective rules, the next generation will typically contain better solutions than the previous one. The best of these solutions then reproduce to form a new generation, and so on, until a solution of acceptable quality is found, or successive generations no longer show significant improvements. A typical evolutionary approach to the travelling salesman problem is described in [11].

A.2.3 Tabu search

Tabu search [59] is often referred to as a *meta-heuristic*. It is used in conjunction with traditional search techniques to enhance their performance. The idea is that during the search, parts of the search space are designated *tabu* for a while, and the search is prohibited from visiting them. For example, if recently visited states are marked tabu, then the search will be directed towards previously unvisited feasible solutions, and not waste time revisiting the same parts of the search space. This very simple idea has led to some impressive results in the field of combinatorial optimization [59].

A.2.4 Novel algorithms for the travelling salesman problem

The novel techniques discussed so far all have general applicability to a wide variety of combinatorial problems. However, it is often the case (as with the traditional solution methods) that tailoring an algorithm to a particular problem produces benchmark performance. This is particularly true for the Euclidean travelling salesman problem, where geometric properties can be exploited to obtain better solutions. Perhaps the most famous of these algorithms is Durbin and Wilshaw's *elastic net* [39, 40, 127]. The algorithm is inspired by the following thought experiment: a small circular path is placed on a map containing the cities to be visited, and is subsequently elongated under the influence of two types of force. The first pulls the path towards the nearest cities, the second pulls neighbouring points on the path together. The elastic net algorithm describes a set of first order difference equations to implement this scheme, which are also shown to be performing gradient descent on a particular Liapunov function. The algorithm performs very well even on large problems, and lends itself to implementation in parallel hardware. Simic [125] presents an interesting derivation of the elastic net algorithm from statistical mechanics foundations, revealing similarities with the mean field annealing and Hopfield network methods. He later elaborates on this work, presenting a generalized elastic net for the solution of quadratic assignment problems [126].

A similar approach to the Euclidean travelling salesman problem stems from Kohonen's work on self-organizing networks [88]. Here, the positions of a set of nodes in a ring are updated until the ring represents a valid tour. At each update a single city is considered: the node nearest the city moves towards the city, dragging its neighbouring nodes with it [15, 42]. Fritzke and Wilke [46] discovered how several approximations could be made to give the algorithm linear time and space complexity, something which had never been achieved previously. The result is a highly efficient algorithm, which can solve a 2392-city problem to within 10% of optimality in 16 minutes on a typical, modern computer workstation [46]. This must surely replace Lin's local search heuristic [97] as the benchmark for the solution of the Euclidean travelling salesman problem.

A.3 Hopfield networks

The network described by Hopfield in his seminal 1982 paper [69] is in fact a special case of the *additive model* developed by Grossberg in the 1960's (see [63] for a historical survey). Hopfield reviewed the network's application as a content addressable memory with both binary [69] and continuous valued [70] outputs. However, after detailed investigations into the network's performance (see for example [145]), it is now clear that other types of content addressable memory are far more efficient [136].

The Hopfield network was first proposed for combinatorial optimization applications in [71, 72], where a penalty function mapping for the travelling salesman problem was given. The mapping was of a rather *ad hoc* nature, with separately weighted penalty functions for each of the hard constraints. The weights were set by trial and error, resulting, not surprisingly, in poor performance [81, 149]. Nevertheless, some researchers persevered with this approach [2, 80], sometimes proposing modified networks to correct what is essentially a sloppy mapping [24, 68].

Real progress in the field can be identified with three main movements: rigorous investigation of network convergence, the development of more sensible mapping techniques, and the emergence of powerful annealing procedures.

Convergence Generalized feedback neural networks can exhibit oscillatory [34] or even chaotic [12, 95] behaviour. It is therefore important to determine under what conditions a network will admit a Liapunov function for optimization applications. The most comprehensive study of the Hopfield network's convergence is presented by Vidyasagar [146], whose analysis does not resort to energy based arguments which typically require the interconnections to be symmetric. Since any analogue implementation of the network cannot possibly exhibit perfect symmetry, Vidyasagar's stability results are of considerable importance. For simulated networks, convergence properties depend on whether the nodes are updated in serial or parallel: such issues are discussed (for discrete-valued networks) in [29, 124].

Mapping The development of rigorous mappings was delayed by the popularity of the original, *ad hoc* mapping of the travelling salesman problem. Researchers concentrated on finding better weights for the penalty functions, often by way of an eigenvector analysis of the Liapunov function [8, 9, 78]: this, though correct, is unduly complicated. Simpler, more reliable mappings began to appear for specific problems in [5, 6, 7]; some also dealt with inequality constraints [131]. A rigorous mapping for the most general 0-1 quadratic programming problems onto standard Hopfield network dynamics was first presented in [53].

Annealing In parallel with the development of rigorous mappings, modified dynamics were proposed as a way of improving the quality of any valid solutions found by the network. The most effective of these modifications typically represented some sort of annealing procedure. Mean field annealing was developed as an approximation to simulated annealing, and will be discussed from that viewpoint in the next section. More direct annealing procedures include hysteretic annealing [41], convex relaxation [112] and matrix graduated non-convexity [5, 6].

A.4 Mean field annealing

The simulated annealing algorithm, described in Section A.2.1, has the undesirable property that it is often very slow in operation. For applications where speed of operation is more important than quality of solutions, mean field annealing (MFA) provides a deterministic approximation to simulated annealing with good speed characteristics. Furthermore, the MFA algorithm maps readily onto parallel architectures for further speed improvements.

The derivation of the MFA algorithm, like that of simulated annealing, draws heavily on ideas from statistical physics. The principle theme is to investigate the *average* statistics of the simulated annealing process, instead of performing Monte Carlo simulations of the process itself. However, the analysis is far too complicated unless several simplifying assumptions are made. It is necessary to make the *mean field approximation*, which concerns the average statistics of two coupled spins, and to replace a complex function with its truncated Taylor expansion around a saddle point. It is these approximations which differentiate mean field annealing from simulated annealing: the latter, in the (impractical) limit of an infinitely slow cooling schedule, is guaranteed to find the globally optimum solution to any properly mapped problem.

Rigorous derivations of the MFA algorithm [5, 117] typically result in a pair of temperature dependent, coupled, nonlinear equations which are termed the *saddle point equations*. The tracking of solutions to these equations through a series of progressively lower temperatures constitutes the MFA algorithm. The solutions are exactly the stable states of a Hopfield network with hyperbolic tangent transfer functions of appropriate gain, so a Hopfield network can be used to solve the equations at each temperature: the resulting system is usually referred to as a *temperature annealed Hopfield network*. More common, however, is to use an iterative replacement algorithm [22, 117, 118, 143] to solve the equations: this is often faster than simulation of the continuous Hopfield dynamics, but is not necessarily stable. The algorithm is equivalent to an Euler approximation of the Hopfield dynamics with time-step $\Delta t = 1$ [5, 117]. Even though MFA has its roots in statistical physics, the more revealing analyses of its performance have drawn on its similarity with Hopfield networks. In particular, it has been demonstrated that MFA and hysteretic annealing have many functional similarities [5, 52].

The performance of MFA on certain problems can be improved using *neuron normalization* [118, 143]: this was originally inspired by an analogy with Potts glasses in statistical physics [82]. Neuron normalization modifies the MFA algorithm so that constraints of the form $\sum_{i \in S} v_i = 1$ are explicitly enforced, without the need for a penalty term. This is particularly useful for mapping travelling salesman and graph partitioning type problems, though the technique does not have general applicability.

A.5 Combinatorial applications of optimization networks

In addition to the classic benchmarks like the travelling salesman problem, optimization networks have also been proposed for the solution of many more practical problems. Although some of the papers cited below do not describe proper mappings or effective annealing procedures, they nonetheless provide a useful index of possible combinatorial applications of optimization networks. These include decoding error correcting codes [30], image segmentation [57], stereo correspondence [138], load balancing [31, 43], classroom scheduling [58], multiprocessor scheduling [67], synthesis of digital circuits [141], invariant pattern recognition [21, 96], analogue-to-digital conversion [94, 135], data rearrangement [74], assignment [41, 131, 132], communications link scheduling [112] and implementing the Viterbi algorithm for Hidden Markov Models [7].

A.6 Continuous applications of optimization networks

Optimization networks can be used for continuous optimization under linear, quadratic or entropic objectives. The most clumsy way to achieve this is to use the 0-1 network output to represent real numbers, to a desired precision, using some coding scheme [133, 152]. Far more straightforward is to allow the continuous network output to represent the solution, and not use an annealing procedure to force convergence to a 0-1 point. Since optimization networks traditionally operate within the unit hypercube, it may be necessary to rescale the problem to be sure that the region of interest lies within this hypercube.

A.6.1 Continuous applications with no hard constraints

The particularly trivial case of a convex quadratic objective is studied in [129]. For convex objectives there is a unique minimum that can be found by any descent procedure. A discussion of scaling the problem to ensure that the minimum lies in the network's range of operation, and a proof of exponential stability of the network's equilibrium point, are presented in [129].

A more sophisticated application is the maximum entropy reconstruction of noisy and blurred images [76, 102]. Here the neuron outputs are used to represent grey scale values in the image. An objective function is then constructed with a quadratic term (which ensures that the reconstructed image is not completely unrelated to the noisy image) and a logarithmic term (which maximizes the entropy of the reconstructed image, promoting smoothness). A Hopfield network, running with a nonzero decay parameter and low gain hyperbolic tangent transfer functions, has a suitable logarithmic term in its Liapunov function, as well as the usual quadratic term. The relative weighting of the two terms is a critical factor in performance and has been well studied in regularization theory [137]. Smoothness can also be enforced using alternative regularizers built around differential operators [152].

In its basic form, regularized image restoration tends to produce over-smoothed images. This can be overcome by solving the continuous image restoration problem in parallel with a discrete *line process* problem. The line process identifies edges in the image and interacts with the restoration problem so that smoothing is not promoted across edges. The edges can be conveniently represented by a set of 0-1 variables covering the image in the same way as the continuous variables do. The whole problem can be elegantly packaged for solution using a modified Hopfield network or MFA: the $\mathbf{u}$ variables converge to give the restored image, while the $\mathbf{v}$ variables are forced to 0-1 values and represent the line process. The technique has its roots in Geman and Geman's seminal paper [56], and has been developed in many directions since then — see for example [25, 54]. A good unification of the various techniques which emerged is presented in [55].

Optimization networks have also been proposed for the deconvolution of seismic reflectivity sequences [148].

A.6.2 Continuous applications with hard constraints

Optimization networks can be used for linear or quadratic programming under linear constraints, with many useful applications. As discussed in Section A.3, penalty functions provide one way of ensuring constraint satisfaction. Inequality constraints are the hardest to deal with, requiring either modified networks [33, 99, 135, 144] or slack variables [131, 132]. For linear programming problems, an optimization network will necessarily find the globally optimal solution. For quadratic programming, however, the network will generally converge to only a locally optimal solution, unless the objective function is convex. Lagrangian networks, described in the next section, provide an alternative connectionist approach to continuous optimization with hard constraints.

A.7 Lagrangian networks

Related to standard optimization networks are the so-called *Lagrangian networks*, which converge to saddle points of the Liapunov function instead of local minima. In [89] an equivalence is established between the optimal solutions to a constrained problem and saddle points of an associated *Lagrangian* function. The constrained problem can therefore be solved by continuous gradient techniques which search for saddle points, providing a viable alternative to penalty function approaches. It was pointed out that Lagrangian gradient search could be implemented on connectionist networks as long ago as 1958 [17]. More up-to-date approaches can be found in [100, 107, 108, 139, 150, 151]. Lagrangian techniques have been successfully applied to several problems in vision [130, 140, 150].

Appendix B

Derivations, Proofs and Details

B.1 Linearized analysis of network dynamics

In this section we derive the linearized network dynamics, finishing with the result stated in equation (5.11). Similar derivations can be found in [5, 51]. We start with the dynamic equation (5.5):

$$\dot{\mathbf{v}}^{\mathrm{val}} = \left(\frac{\phi}{T^p}\mathbf{T}^{\mathrm{opr}} - \eta\mathbf{I}\right)\mathbf{v}^{\mathrm{val}} + \frac{\phi}{T^p}\mathbf{i}^{\mathrm{opr}}$$

Expressing $\mathbf{v}^{\mathrm{val}}$ and $\mathbf{i}^{\mathrm{opr}}$ in terms of their decompositions ((5.8) and (5.10)) along the eigenvectors of $\mathbf{T}^{\mathrm{opr}}$, we obtain

$$\begin{aligned}
\sum_{i=1}^{N}\dot{\alpha}_i\mathbf{x}^i &= \sum_{i=1}^{N}\alpha_i\left(\frac{\phi}{T^p}\mathbf{T}^{\mathrm{opr}} - \eta\mathbf{I}\right)\mathbf{x}^i + \frac{\phi}{T^p}\sum_{i=1}^{N}\beta_i\mathbf{x}^i \\
&= \sum_{i=1}^{N}\alpha_i\left(\frac{\phi\lambda_i}{T^p} - \eta\right)\mathbf{x}^i + \frac{\phi}{T^p}\sum_{i=1}^{N}\beta_i\mathbf{x}^i
\end{aligned}$$

So

$$\dot{\alpha}_i = \left(\frac{\phi\lambda_i}{T^p} - \eta\right)\alpha_i + \frac{\phi}{T^p}\beta_i = \tilde{\lambda}_i\alpha_i + \frac{\phi}{T^p}\beta_i \tag{B.1}$$

where $\tilde{\lambda}_i \equiv (\phi\lambda_i/T^p - \eta)$. We can now integrate (B.1) to obtain the linearized dynamics in equations (B.2) and (B.3). We have to consider the cases $\tilde{\lambda}_i = 0$ and $\tilde{\lambda}_i \neq 0$ separately.

$$\begin{aligned}
\text{(i) } \tilde{\lambda}_i = 0 \qquad \Rightarrow \dot{\alpha}_i &= \frac{\phi}{T^p}\beta_i \\
\Leftrightarrow \int_{\alpha_i^\circ}^{\alpha_i(t)} d\alpha_i &= \frac{\phi}{T^p}\beta_i\int_0^t dt \\
\Leftrightarrow \alpha_i(t) &= \alpha_i^\circ + \frac{\phi}{T^p}\beta_i t \qquad \text{(B.2)} \\
\text{(ii) } \tilde{\lambda}_i \neq 0 \qquad \Rightarrow \dot{\alpha}_i &= \tilde{\lambda}_i\alpha_i + \frac{\phi}{T^p}\beta_i \\
\Leftrightarrow \int_{\alpha_i^\circ}^{\alpha_i(t)} \frac{d\alpha_i}{\tilde{\lambda}_i\alpha_i + (\phi/T^p)\beta_i} &= \int_0^t dt \\
\Leftrightarrow \alpha_i(t) &= \left(\alpha_i^\circ + \frac{\phi\beta_i}{\tilde{\lambda}_i T^p}\right)\exp(\tilde{\lambda}_i t) - \frac{\phi\beta_i}{\tilde{\lambda}_i T^p} \qquad \text{(B.3)}
\end{aligned}$$

B.2 $\mathbf{i}^{\mathrm{opr}}$ for the travelling salesman problem

In Chapter 3 we derived a mapping for the travelling salesman problem, concluding that

$$\begin{aligned}\mathbf{T}^{\mathrm{op}} &= \mathbf{P}\otimes\mathbf{Q}\\ \mathbf{i}^{\mathrm{op}} &= \mathbf{0}\\ \mathbf{T}^{\mathrm{val}} &= \mathbf{R}\otimes\mathbf{R}\\ \mathbf{s} &= \tfrac{1}{n}(\mathbf{o}\otimes\mathbf{o})\end{aligned}$$

Substituting these expressions into the expression for $\mathbf{i}^{\mathrm{opr}}$ (5.4), and drawing on the definitions of $\mathbf{R}$ (3.41) and $\mathbf{Q}$ (3.11), we obtain

$$\begin{aligned}\mathbf{i}^{\mathrm{opr}} &= \tfrac{1}{n}(\mathbf{R}\otimes\mathbf{R})(\mathbf{P}\otimes\mathbf{Q})(\mathbf{o}\otimes\mathbf{o})\\ &= \tfrac{1}{n}(\mathbf{RPo}\otimes\mathbf{RQo})\\ &= \tfrac{1}{n}(\mathbf{RPo}\otimes 2\mathbf{Ro})\\ &= \tfrac{1}{n}(\mathbf{RPo}\otimes 2(\mathbf{I}-\tfrac{1}{n}\mathbf{O})\mathbf{o})\\ &= \tfrac{1}{n}(\mathbf{RPo}\otimes 2(\mathbf{o}-\mathbf{o}))\\ &= \mathbf{0}\end{aligned}\tag{B.4}$$

Hence $\mathbf{i}^{\mathrm{opr}}=\mathbf{0}$ for the travelling salesman problem. A similar proof can be found in [5].

B.3 $\mathbf{i}^{\mathrm{opr}}$ and $\mathbf{x}^{\mathrm{max}}$ for the Hamilton path problem

In Section 3.5 we derived expressions for $\mathbf{T}^{\mathrm{op}}$, $\mathbf{i}^{\mathrm{op}}$, $\mathbf{T}^{\mathrm{val}}$ and $\mathbf{s}$ for the Hamilton path problem. The expressions were identical to those for the travelling salesman problem (see Appendix B.2), with the exception of a slightly modified matrix $\mathbf{Q}$ (3.44). We begin our analysis by obtaining Kronecker product expressions for $\mathbf{i}^{\mathrm{opr}}$ and $\mathbf{T}^{\mathrm{opr}}$:

$$\begin{aligned}\mathbf{i}^{\mathrm{opr}} &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{s}+\mathbf{T}^{\mathrm{val}}\mathbf{i}^{\mathrm{op}}\\ &= \tfrac{1}{n}(\mathbf{R}\otimes\mathbf{R})(\mathbf{P}\otimes\mathbf{Q})(\mathbf{o}\otimes\mathbf{o})\\ &= \tfrac{1}{n}(\mathbf{RPo}\otimes\mathbf{RQo})\end{aligned}\tag{B.5}$$

$$\begin{aligned}\mathbf{T}^{\mathrm{opr}} &= \mathbf{T}^{\mathrm{val}}\mathbf{T}^{\mathrm{op}}\mathbf{T}^{\mathrm{val}}\\ &= (\mathbf{R}\otimes\mathbf{R})(\mathbf{P}\otimes\mathbf{Q})(\mathbf{R}\otimes\mathbf{R})\\ &= (\mathbf{RPR}\otimes\mathbf{RQR})\end{aligned}\tag{B.6}$$

It is possible to use the properties of Kronecker products to relate the eigenvectors of $\mathbf{T}^{\mathrm{opr}}$ to those of $\mathbf{RPR}$ and $\mathbf{RQR}$. We shall find it useful to start by looking at the eigenvectors $\mathbf{h}_k$ of $\mathbf{Q}$, which are easily verified to be

$$[\mathbf{h}_k]_i=\sin\left(\frac{k\pi i}{n+1}\right),\qquad i,k\in\{1\ldots n\}\tag{B.7}$$

with corresponding eigenvalues

$$\mu_k=2\cos\left(\frac{k\pi}{n+1}\right),\qquad k\in\{1\ldots n\}\tag{B.8}$$

Now, those $\mathbf{h}_k$ with even k (ie. those with $k = 2j$ for some integer j) exhibit the property $\sum_{i=1}^{n} [\mathbf{h}_k]_i = 0$, so

$$\begin{aligned} \mathbf{h}_{2j}^T\mathbf{o} &= 0 \quad &\text{(B.9)}\\ \text{and } \mathbf{R}\mathbf{h}_{2j} &= \mathbf{h}_{2j} \quad &\text{(B.10)} \end{aligned}$$

It follows that $\mathbf{h}_{2j}$ is also an eigenvector of $\mathbf{RQR}$, with the same eigenvalue μ_{2j}, since

$$\mathbf{RQR}\mathbf{h}_{2j} = \mathbf{RQ}\mathbf{h}_{2j} = \mu_{2j}\mathbf{R}\mathbf{h}_{2j} = \mu_{2j}\mathbf{h}_{2j}$$

This accounts for about half of the eigenvectors of $\mathbf{RQR}$. The others seem to be more difficult to derive analytically, though experiments reveal that the eigenvector of $\mathbf{RQR}$ with 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 $\mu_{\max}$. There is also strong experimental evidence that all the eigenvalues of $\mathbf{RPR}$ 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

$$\mathbf{x}^{\max} = \phi(\mathbf{w}_{\max} \otimes \mathbf{h}_{\max}) \tag{B.11}$$

where $\mathbf{w}_{\max}$ is the eigenvector of $\mathbf{RPR}$ with the largest positive eigenvalue, and ϕ is a normalizing constant. It transpires that, for this problem, $\mathbf{i}^{\mathrm{opr}}$ and $\mathbf{x}^{\max}$ are mutually orthogonal, since

$$\begin{aligned} \mathbf{x}^{\max T}\mathbf{i}^{\mathrm{opr}} &= \frac{\phi}{n}(\mathbf{w}_{\max} \otimes \mathbf{h}_{\max})^T(\mathbf{RPo} \otimes \mathbf{RQo})\\ &= \frac{\phi}{n}(\mathbf{w}_{\max}{}^T\mathbf{RPo})(\mathbf{h}_{\max}{}^T\mathbf{RQo})\\ &= \frac{\phi}{n}(\mathbf{w}_{\max}{}^T\mathbf{RPo})(\mathbf{h}_{\max}{}^T\mathbf{Qo})\\ &= \frac{\phi\mu_{\max}}{n}(\mathbf{w}_{\max}{}^T\mathbf{RPo})(\mathbf{h}_{\max}{}^T\mathbf{o})\\ &= 0 \end{aligned} \tag{B.12}$$

B.4 E^{alp} for the Hamilton path problem

Recall that the auxiliary linear problem objective is given by $E^{\mathrm{alp}} = -\mathbf{i}^{\mathrm{opr}T}\mathbf{v}$ (5.18). For the Hamilton path problem, we substitute for $\mathbf{i}^{\mathrm{opr}}$ using equation (B.5):

$$\begin{aligned} E^{\mathrm{alp}} &= -\tfrac{1}{n}(\mathbf{o}^T\mathbf{PR} \otimes \mathbf{o}^T\mathbf{QR})\mathbf{v}\\ &= -\tfrac{1}{n}(\mathbf{o}^T\mathbf{P} \otimes \mathbf{o}^T\mathbf{Q})\mathbf{T}^{\mathrm{val}}\mathbf{v}\\ &= -\tfrac{1}{n}(\mathbf{o}^T\mathbf{P} \otimes \mathbf{o}^T\mathbf{Q})(\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s}) + \text{terms independent of } \mathbf{v} \end{aligned}$$

For $\mathbf{v}$ lying within P, we can substitute $\mathbf{v}$ for $(\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s})$. Doing this, and dropping the terms independent of $\mathbf{v}$, we obtain

$$\begin{aligned} E^{\mathrm{alp}} &= -\tfrac{1}{n}(\mathbf{o}^T\mathbf{P} \otimes \mathbf{o}^T\mathbf{Q})\mathbf{v}\\ &= -\tfrac{1}{n}\,\mathrm{vec}(\mathbf{o}^T\mathbf{QVPo}) \qquad \text{where } \mathbf{v} = \mathrm{vec}(\mathbf{V})\\ &= \tfrac{1}{n}\left[(\mathbf{Qo})^T\mathbf{V}(-\mathbf{Po})\right] \end{aligned}$$

B.5 Details for travelling salesman experiments

The experiments reported in Section 6.6 were conducted with the aim of obtaining the best possible solution quality at the expense of execution time. To this end, slow annealing schedules and small time-steps were used in the network simulations. The details are summarized below.

Steepest descent network

The algorithm described in Section 3.4 was used to simulate the steepest descent dynamics running under the mapping for the travelling salesman problem given in Section 3.5. The time-step at each iteration was chosen to be sufficiently small so that no further reduction of the time-step affected the perceived dynamics; in this way, the simulation is assumed to be modelling the continuous dynamics reasonably well. In practice, this was achieved by arranging for $\|\Delta \mathbf{v}^{\mathrm{val}}\|$ to be less than $0.05\|\mathbf{v}\|$ at each iteration. Hysteretic annealing was used to improve the solution quality. The annealing parameter γ was initialized with a large negative number, so that $\mathbf{v}$ stabilized near the point $\mathbf{v}^{\mathrm{val}} = \mathbf{0}$. γ was subsequently increased by a small amount (typically 0.01) at each time-step, until motion of $\mathbf{v}$ was detected. The motion criterion was that $\|\Delta\mathbf{v}\| > \epsilon$ for each iteration, where $\epsilon = 0.005$ typically. With $\mathbf{v}$ moving, γ was held constant until $\|\Delta\mathbf{v}\| < \epsilon$ again, at which point γ was incremented to produce new motion in $\mathbf{v}$. The simulation was stopped when all of the elements of $\mathbf{v}$ were within 0.1 of 0 or 1, and the corresponding matrix $\mathbf{V}$ was a valid permutation matrix.

Mean field annealing

The neuron normalization operation [118, 142] was used to enhance the performance of the MFA algorithm. With neuron normalization, row constraints for a permutation matrix

$$\sum_{j=1}^{n} V_{ij} = 1, \qquad i \in \{1 \ldots n\} \tag{B.13}$$

are implicitly enforced using an update rule

$$V_{ij}\big|_{t+1} = \frac{\exp(-\left.\frac{\partial E}{\partial V_{ij}}\right|_t / T^p)}{\sum_{k=1}^{n} \exp(-\left.\frac{\partial E}{\partial V_{ik}}\right|_t / T^p)} \qquad i, j \in \{1 \ldots n\} \tag{B.14}$$

The remaining column constraints must be enforced using a penalty term built into the objective function E. For the experiments reported in Section 6.6, the objective function used was

$$E = E^{\mathrm{op}} + \tfrac{1}{2}c_1 \sum_{j=1}^{n} \left(\sum_{i=1}^{n} V_{ij} - 1 \right)^2 - \tfrac{1}{2}c_2 \sum_{i=1}^{n}\sum_{j=1}^{n} V_{ij}^2 \tag{B.15}$$

where E^{op} is set for the travelling salesman problem as in Section 3.5, the c_1 term is the penalty function to force the column sums of $\mathbf{V}$ towards 1, and the c_2 term is a fixed hysteretic annealing term[1] which stabilizes the algorithm and makes E more concave, thus promoting convergence to a hypercube corner. For the Euclidean problems,

[1]Because $\|\mathbf{V}\|$ has the same value at all valid solution points, the more general hysteretic annealing term $\sum_{i=1}^{n}\sum_{j=1}^{n}(V_{ij} - 0.5)^2$ is not required.

setting $c_1 = 1.2$ and $c_2 = 0.8$ was found to give good quality solutions with relatively few invalid tours; for the random problems, the weightings used were $c_1 = 1.5$ and $c_2 = 1.0$.

The rows of $\mathbf{V}$ were updated using equation (B.14) *asynchronously*: that is, for each iteration, a row was selected at random, and then all the elements in that row updated using (B.14). It has been noted in [117] that MFA is much easier to stabilize using asynchronous update. As with the steepest descent network, a cautious annealing schedule was used. T^p was initialized with a sufficiently large positive number so that $\mathbf{V}$ stabilized near its starting position. T^p was subsequently reduced by a small amount (typically 0.001) at each time-step, until motion of $\mathbf{V}$ was detected. The motion criterion was that $\|\Delta\mathbf{V}\| > \epsilon$ at each iteration, where $\epsilon = 0.005$ typically. With $\mathbf{V}$ in motion, T^p was held constant until $\|\Delta\mathbf{V}\| < \epsilon$ again, at which point T^p was reduced to produce new motion in $\mathbf{V}$. The simulation was stopped when all of the elements of $\mathbf{V}$ were within 0.1 of 0 or 1, with $\mathbf{V}$ being a valid permutation matrix.

Simulated annealing

The simulated annealing code for the travelling salesman problem was taken directly from [119], with a slight modification to allow problem specification in terms of a distance matrix instead of a coordinate list: this allows random problems to be solved, as well as Euclidean ones. The annealing schedule is described in [119, p. 346], and is sufficiently cautious to obtain excellent results over the test problems.

3-opt search

The 3-opt search algorithm was implemented directly from the pseudo-code in [97, p. 2266], without the tour reduction procedure, which makes little difference for small problems. The results in Tables 6.1 and 6.2 were obtained using a single pass through the algorithm for each problem, starting from random tours. For small problems, it is suggested in [97] that there is little point in performing multiple passes through the algorithm, starting from different random tours, since the 3-opt tours so obtained are likely to be identical.

B.6 The vertices of knapsack polytopes

In this section we prove that the vertices of knapsack polytopes have at most one non-integral coordinate. A knapsack polytope P is defined by $P = \{\mathbf{v} \mid \mathbf{v}^T\mathbf{x} \leq C,\ 0 \leq v_i \leq 1\}$. Consider a point $\mathbf{v}$ within P with at least two non-integral coordinates v_k and v_l. Defining the vector $\delta\mathbf{v}$ to have zero elements except for $\delta v_k = \epsilon$ and $\delta v_l = (-\epsilon x_k / x_l)$, where ϵ is an arbitrarily small scalar, it is clear that adding $\delta\mathbf{v}$ to $\mathbf{v}$ does not change the value of $\mathbf{v}^T\mathbf{x}$, and does not take either v_k or v_l out of their allowed range of 0 to 1, since both originally lie between 0 and 1 and ϵ is arbitrarily small. Hence $(\mathbf{v} + \delta\mathbf{v})$ also lies within P. Since ϵ was not constrained to take any particular sign, it is also clear that $(\mathbf{v} - \delta\mathbf{v})$ lies within P. Hence $\mathbf{v}$ lies on at least a one-dimensional face of P, with direction vector $\delta\mathbf{v}$. Since a vertex of P is necessarily a zero-dimensional face of P, it follows that $\mathbf{v}$ cannot be a vertex of P. Hence any vertex of P has at most one non-integral coordinate.

B.7 Tabu dynamics for stationary $\mathbf{v}^+$

Here we consider $\mathbf{v}^+$ trapped at a vertex $\mathbf{v}_o^+$ of P^+ from time t_0. In these circumstances, it is possible to analytically integrate the tabu dynamic equations, allowing us to judge the speed of the dynamics and place bounds on the variables α, β and ϵ to guarantee escape from the vertex.

Suppose at time t_0 the tabu variables have values $\theta = \theta_0$ and $\mathbf{h} = \mathbf{h}_0$. Then integrating equations (7.10) and (7.11) gives

$$\theta(t) = \frac{2\beta}{\alpha} - e^{-\alpha(t-t_0)}\left(\frac{2\beta}{\alpha} - \theta_0\right) \tag{B.16}$$

$$\mathbf{h}(t) = -\frac{2\beta}{\alpha}\mathbf{f}(\mathbf{v}_o^+) + e^{-\alpha(t-t_0)}\left(\frac{2\beta}{\alpha}\mathbf{f}(\mathbf{v}_o^+) + \mathbf{h}_0\right) \tag{B.17}$$

The resulting gradient of E at $\mathbf{v}_o^+$ is

$$-\nabla E = \mathbf{T}^{\mathrm{op}+}\mathbf{v}_o^+ + \mathbf{i}^{\mathrm{op}+} + \frac{2\beta\epsilon}{\alpha}(\mathbf{c} - \mathbf{v}_o^+) + e^{-\alpha(t-t_0)}\left(\mathbf{h}_0 + \theta_0\mathbf{v}_o^+ + \frac{2\beta\epsilon}{\alpha}(\mathbf{v}_o^+ - \mathbf{c})\right) \tag{B.18}$$

where we have substituted the expression for $\mathbf{f}(\mathbf{v}_o^+)$ from equation (7.6). We see that $-\nabla E$ has a limiting value as $t \to \infty$, specifically

$$-\nabla E_\infty = \mathbf{T}^{\mathrm{op}+}\mathbf{v}_o^+ + \mathbf{i}^{\mathrm{op}+} + \frac{2\beta\epsilon}{\alpha}(\mathbf{c} - \mathbf{v}_o^+) \tag{B.19}$$

To guarantee escape from the vertex, we require that $-\mathbf{a}^T\nabla E_\infty > 0$ for some direction $\mathbf{a}$ which does not lead out of P^+. The only such direction we can reliably identify is $(\mathbf{c} - \mathbf{v}_o^+)$, since $\mathbf{c}$ is defined as being within P^+ and P^+ is convex. Assuming a linear objective E^{op}, so that $\mathbf{T}^{\mathrm{op}+} = \mathbf{0}$, we require

$$(\mathbf{c} - \mathbf{v}_o^+)^T\mathbf{i}^{\mathrm{op}+} + \frac{2\beta\epsilon}{\alpha}\|\mathbf{c} - \mathbf{v}_o^+\|^2 > 0 \tag{B.20}$$

The worst-case scenario is when $\mathbf{i}^{\mathrm{op}+}$ is in the direction $-(\mathbf{c} - \mathbf{v}_o^+)$, in which case the escape condition becomes

$$\frac{\beta\epsilon}{\alpha} > \frac{\|\mathbf{i}^{\mathrm{op}+}\|}{2\|\mathbf{c} - \mathbf{v}_o^+\|} \tag{B.21}$$

Hence the relevant quantity governing the ability to escape from a vertex is the ratio $\beta\epsilon/\alpha$. Equation (B.21) indicates that it is desirable to locate the vector $\mathbf{c}$ as far from any vertex as possible. Some vector $\mathbf{c}$ within P^+ can be found in polynomial time using Khachiyan's method [123], or more practically one of the projection techniques [4, 37, 77, 109]. Once located, $\mathbf{c}$ can also serve as a starting position for $\mathbf{v}^+$. Often, however, the most convenient choice of $\mathbf{c}$ is the vector $\mathbf{s}$ (see equation (3.20)), which is a natural product of the mapping process and lies within P^+ for many classes of problem. For the knapsack problem, using $\mathbf{c} = \mathbf{s}$, it is possible to demonstrate that the worst-case escape condition is approximately

$$\frac{\beta\epsilon}{\alpha} > \frac{\|\mathbf{i}^{\mathrm{op}+}\|}{2} \tag{B.22}$$

Finally, a note about the speed of the tabu search dynamics. For low values of α, the illustration in Figure 7.3 indicates that most of the search time is spent with $\mathbf{v}^+$ trapped at a vertex and the tabu variables θ and $\mathbf{h}$ changing to free $\mathbf{v}^+$ from the vertex. Equation (B.18) indicates that this process takes a characteristic time of $1/\alpha$, and so we conclude that the overall speed of the system is approximately proportional to α.

B.8 Details for knapsack experiments

For the knapsack experiments described in Section 7.2, a steepest descent network was used to find the LP-relaxation solution $\mathbf{v}^*$. This is a fairly easy case to simulate. Since the objective is linear, annealing is neither required nor beneficial. The algorithm in Section 3.4 was run under the mapping for the knapsack problem given in Section 3.5, using a sufficiently small time-step Δt, such that reducing Δt further did not change the convergence properties of the system. $\mathbf{v}^+$ was initialized at the point $\mathbf{v}^+ = \mathbf{s}$, which always lies within P^+ for knapsack problems. When $\|\Delta\mathbf{v}^+\| < 0.001$ at any iteration, it was assumed that $\mathbf{v}^+$ had converged to $\mathbf{v}^*$.

The tabu search experiments in Section 7.3 were much more difficult to perform. Simulation of the coupled equations (2.10) and (7.10)–(7.11) is not straightforward. In particular, a simple Euler approximation is not likely to mimic the behaviour of the continuous-time system. A proper simulation would have to be developed through a rigorous discrete-time analysis of the system. To avoid this, our simulations deliberately employed a low value of α, which causes the dynamics of θ and $\mathbf{h}$ to be orders of magnitude slower than those of $\mathbf{v}^+$ (see Appendix B.7). We can therefore regard θ and $\mathbf{h}$ as constant when $\mathbf{v}^+$ is moving, and employ the standard simulation algorithm for steepest descent dynamics described in Section 3.4. As before, $\mathbf{v}^+$ was initialized at the point $\mathbf{v}^+ = \mathbf{s}$, and a small time-step was used. When $\mathbf{v}^+$ came to a halt, that is when $\|\Delta\mathbf{v}^+\| < 0.001$ at any iteration, it was assumed that $\mathbf{v}^+$ was trapped at a vertex of P^+. Equations (B.16)–(B.17) were then used to predict future values of θ and $\mathbf{h}$, using a coarse-to-fine search for the time t when $\mathbf{v}^+$ escapes from the vertex. θ and $\mathbf{h}$ were then updated to their new values at time t, and the steepest descent algorithm was resumed to advance $\mathbf{v}^+$ to the next vertex of P^+. This provides an accurate and efficient simulation algorithm for tabu search with small α, though a hardware implementation would require a larger value of α to run at reasonable speed. The tabu search parameters used for the knapsack experiments were $\alpha = 0.01$, $\beta = 1.0$ and $\epsilon = 0.1$. The network's output was monitored throughout to spot any 0-1 points visited. Owing to the discrete nature of the simulation it was necessary to set a margin ϕ, such that v_i is considered integral if it lies within ϕ of 0 or 1. For the experiments in this thesis ϕ was set to be 0.05.

Appendix C

Kronecker Products

At several points in this thesis we make use of Kronecker products to conveniently map matrix representations of problems into vector representations. In this appendix we briefly review Kronecker product notation, and present some results which are used in the mathematical analysis of problem mappings. For more on Kronecker products, along with proofs of the results, see [62].

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 the $nm \times nm$ matrix given by:

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

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

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

Then the following identities hold:

$$\begin{aligned} \mathrm{trace}(\mathbf{AB}) &= \mathrm{trace}(\mathbf{BA}) \quad \text{(for } \mathbf{AB} \text{ and } \mathbf{BA} \text{ both square)} && \text{(C.3)} \\ (\mathbf{A} \otimes \mathbf{B})^T &= \mathbf{A}^T \otimes \mathbf{B}^T && \text{(C.4)} \\ (\mathbf{A} \otimes \mathbf{B})(\mathbf{X} \otimes \mathbf{Y}) &= (\mathbf{AX} \otimes \mathbf{BY}) && \text{(C.5)} \\ \mathrm{trace}(\mathbf{AB}) &= \left[\mathrm{vec}(\mathbf{A}^T)\right]^T \mathrm{vec}(\mathbf{B}) \quad \text{(for } \mathbf{A} \text{ and } \mathbf{B} \text{ both } n \times n\text{)} && \text{(C.6)} \\ \mathrm{vec}(\mathbf{AYB}) &= (\mathbf{B}^T \otimes \mathbf{A})\mathrm{vec}(\mathbf{Y}) && \text{(C.7)} \end{aligned}$$

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

$$(\mathbf{w} \otimes \mathbf{h})^T(\mathbf{x} \otimes \mathbf{g}) = (\mathbf{w}^T\mathbf{x})(\mathbf{h}^T\mathbf{g}) \tag{C.8}$$

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

Bibliography

[1] E. H. L. Aarts. *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing.* Wiley-Interscience series in discrete mathematics and optimization. John Wiley & Sons, Chichester, 1989.

[2] S. Abe. Theories on the Hopfield neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages I:557–I:564, 1989.

[3] S. Abe, J. Kawakami, and K. Hirasawa. Solving inequality constrained combinatorial optimization problems by the Hopfield neural networks. *Neural Networks*, 5(4):663–670, 1992.

[4] S. Agmon. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:382–392, 1954.

[5] S. V. B. Aiyer. Solving combinatorial optimization problems using neural networks. Technical Report CUED/F-INFENG/TR 89, Cambridge University Department of Engineering, October 1991.

[6] S. V. B. Aiyer and F. Fallside. A subspace approach to solving combinatorial optimization problems with Hopfield networks. Technical Report CUED/F-INFENG/TR 55, Cambridge University Department of Engineering, December 1990.

[7] S. V. B. Aiyer and F. Fallside. A Hopfield network implementation of the Viterbi algorithm for Hidden Markov Models. In *Proceedings of the International Joint Conference on Neural Networks*, pages 827–832, Seattle, July 1991.

[8] S. V. B. Aiyer, M. Niranjan, and F. Fallside. On the performance of the Hopfield model. In *Proceedings of the INNS*, Paris, July 1990.

[9] S. V. B. Aiyer, M. Niranjan, and F. Fallside. A theoretical investigation into the performance of the Hopfield model. *IEEE Transactions on Neural Networks*, 1(2):204–215, June 1990.

[10] Y. Akiyama, A. Yamashita, M. Kajiura, and H. Aiso. Combinatorial optimization with Gaussian machines. In *Proceedings of the International Joint Conference on Neural Networks*, pages 533–540, June 1989.

[11] B. K. Ambati, J. Ambati, and M. M. Mokhtar. Heuristic combinatorial optimization by simulated Darwinian evolution: a polynomial time algorithm for the travelling salesman problem. *Biological Cybernetics*, 65:31–35, 1991.

[12] D. J. Amit. *Modeling Brain Function. The World of Attractor Neural Networks.* Cambridge University Press, Cambridge, 1989.

[13] J. R. Anderson. A mean field computational model for PDP. In *Proceedings of the 1988 Connectionist Models Summer School*, pages 217–223, Carnegie Mellon University, 1988.

[14] J. R. Anderson and C. Peterson. Applicability of mean field theory neural network methods to the graph partitioning problem. Technical Report ACA-ST-064-88, Microelectronics and Computing Technology Corporation, Austin, Texas, 1988.

[15] B. Angéniol, G. de la Croix Vaubois, and J. le Texier. Self-organizing feature maps and the travelling salesman problem. *Neural Networks*, 1:289–293, 1988.

[16] M. Aourid and B. Kaminska. Neural networks for solving the quadratic 0–1 programming problem under linear constraints. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1092–1095, San Francisco, 1993.

[17] K. J. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Nonlinear Programming.* Stanford University Press, Stanford, 1958.

[18] M. L. Balinksi. *Approaches to Integer Programming.* Mathematical Programming Studies. North Holland Publishing Company, Amsterdam, 1974.

[19] D. A. Beyer and R. G. Ogier. The tabu learning neural network search method applied to the traveling salesman problem. Technical report, SRI International, Menlo Park, California, December 1990.

[20] D. A. Beyer and R. G. Ogier. Tabu learning: A neural network search method for solving nonconvex optimization problems. In *Proceedings of the International Joint Conference on Neural Networks*, Singapore, November 1991.

[21] E. Bienenstock and R. Doursat. Elastic matching and pattern recognition in neural networks. In L. Personnaz and G. Dreyfus, editors, *Neural Networks: From Models to Applications.* IDSET, Paris, 1989.

[22] G. Bilbro, R. Mann, T. K. Miller III, W. E. Snyder, D. E. Van den Bout, and M. White. Optimization by mean field annealing. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 91–98. Morgan Kaufman, 1989.

[23] G. Birkhoff. Tres observaciones sobre al algebra lineal. *Revista Facultad de Ciencias Exactas, Puras y Aplicadas Universidad Nacional de Tacuman, Serie A (Matematicas y Fisca Teorica)*, 5:147–151, 1946.

[24] A. R. Bizzarri. Convergence properties of a modified Hopfield-Tank model. *Biological Cybernetics*, 64:293–300, 1991.

[25] A. Blake and A. Zisserman. *Visual Reconstruction.* MIT Press, Cambridge MA, 1987.

[26] R. A. Brualdi and P. M. Gibson. Convex polyhedra of doubly stochastic matrices. I, Applications of the permanent function. *Journal of Combinatorial Theory A*, 22:194–230, 1977.

[27] R. A. Brualdi and P. M. Gibson. Convex polyhedra of doubly stochastic matrices. II, The graph of Ω^n. *Journal of Combinatorial Theory B*, 22:175–198, 1977.

[28] R. A. Brualdi and P. M. Gibson. Convex polyhedra of doubly stochastic matrices. III, Affine and combinatorial properties of Ω^n. *Journal of Combinatorial Theory A*, 22:338–351, 1977.

[29] J. Bruck. On the convergence properties of the Hopfield model. *Proceedings of the IEEE*, 78(10):1579–1585, October 1990.

[30] J. Bruck and M. Blaum. Neural networks, error-correcting codes and polynomials over the binary n-cube. *IEEE Transactions on Information Theory*, 35(5):976–987, September 1989.

[31] T. Bultan and C. Aykanat. A new mapping heuristic based on mean field annealing. *Journal of Parallel and Distributed Computing*, 16:292–305, 1992.

[32] V. Chvátal. A note on the traveling salesman problem. *Operations Research Letters*, 8(2):77–78, April 1989.

[33] A. Cichocki and R. Unbehauen. *Neural Networks for Optimization and Signal Processing*. John Wiley & Sons, Chichester, 1993.

[34] M. A. Cohen and S. Grossberg. Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man and Cybernetics*, 13(5):813–825, September/October 1983.

[35] Y. Davidor. An intuitive introduction to genetic algorithms as adaptive optimizing procedures. Technical Report CS90-07, Department of Applied Mathematics and Computer Science, Weizmann Institute of Science, Israel, April 1990.

[36] L. Davis. *Genetic algorithms and simulated annealing*. Research notes in artificial intelligence. Pitman, London, 1987.

[37] A. R. De Pierro and A. N. Iusem. A simultaneous projections method for linear inequalities. *Linear Algebra and its Applications*, 64:243–253, 1985.

[38] S. E. Dreyfus and A. M. Law. *The Art and Theory of Dynamic Programming*. Academic Press, 1977.

[39] R. Durbin, R. Szeliski, and A. Yuille. An analysis of the elastic net approach to the traveling salesman problem. *Neural Computation*, 1:348–358, 1989.

[40] R. Durbin and D. Wilshaw. An analogue approach to the travelling salesman problem using an elastic net method. *Nature*, 326(6114):689–691, April 1987.

[41] S. P. Eberhart, D. Daud, D. A. Kerns, T. X. Brown, and A. P. Thakoor. Competitive neural architecture for hardware solution to the assignment problem. *Neural Networks*, 4:431–442, 1991.

[42] F. Favata and R. Walker. A study of the application of Kohonen-type neural networks to the travelling salesman problem. *Biological Cybernetics*, 64:463–468, 1991.

[43] C. G. Fox and W. Furmanski. Load balancing loosely synchronous problems with a neural network. In *Proceedings of the 3rd Conference on Hypercube Concurrent Computers and Applications*, pages 241–278, Pasedena, 1988.

[44] L. Fox and D. F. Mayers. *Computing Methods for Scientists and Engineers.* Monographs on Numerical Analysis. Clarendon Press, Oxford, 1968.

[45] J. B. Fraleigh and R. A. Beauregard. *Linear Algebra.* Addison-Wesley, Reading MA, 1987.

[46] B. Fritzke and P. Wilke. FLEXMAP — a neural network for the traveling salesman problem with linear time and space complexity. In *Proceedings of the International Joint Conference on Neural Networks*, Singapore, November 1991.

[47] M. R. Garey and D. S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, San Francisco, 1979.

[48] R. S. Garfinkel and G. L. Nemhauser. *Integer Programming.* John Wiley & Sons, New York, 1972.

[49] A. H. Gee, S. V. B. Aiyer, and R. W. Prager. Neural networks and combinatorial optimization problems — the key to a successful mapping. Technical Report CUED/F-INFENG/TR 77, Cambridge University Department of Engineering, July 1991.

[50] A. H. Gee, S. V. B. Aiyer, and R. W. Prager. A subspace approach to invariant pattern recognition using Hopfield networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 795–800, Singapore, November 1991.

[51] A. H. Gee, S. V. B. Aiyer, and R. W. Prager. An analytical framework for optimizing neural networks. *Neural Networks*, 6(1):79–97, 1993.

[52] A. H. Gee and R. W. Prager. Alternative energy functions for optimizing neural networks. Technical Report CUED/F-INFENG/TR 95, Cambridge University Department of Engineering, March 1992.

[53] A. H. Gee and R. W. Prager. Polyhedral combinatorics and neural networks. Technical Report CUED/F-INFENG/TR 100, Cambridge University Department of Engineering, May 1992. To appear in *Neural Computation.*

[54] D. Geiger and F. Girosi. Parallel and deterministic algorithms from MRF's: Surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):401–412, May 1991.

[55] D. Geiger and A. Yuille. A common framework for image segmentation. *International Journal of Computer Vision*, 6(3):227–243, 1991.

[56] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[57] A. Ghosh, N. R. Pal, and S. K. Pal. Image segmentation using a neural network. *Biological Cybernetics*, 66:151–158, 1991.

[58] L. Gislen, C. Peterson, and B. Söderberg. 'Teachers and Classes' with neural networks. *International Journal of Neural Systems*, 1(2):167–176, 1989.

[59] F. Glover. Tabu search, a tutorial. Technical report, Center for Applied Artificial Intelligence, University of Colorado, November 1989. Revised February 1990.

[60] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading MA, 1989.

[61] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64:275–278, 1958.

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

[63] S. Grossberg. Nonlinear neural networks: Principles, mechanisms and architectures. *Neural Networks*, 1(1):17–61, 1988.

[64] C. M. Haberman. *Use of Digital Computers for Engineering Applications.* Charles E. Merril Books, Columbus, Ohio, 1966.

[65] A. Hamilton, A. F. Murray, D. J. Baxter, S. Churcher, H. M. Reekie, and L. Tarassenko. Integrated pulse stream neural networks: Results, issues and pointers. *IEEE Transactions on Neural Networks*, 3(3):385–393, May 1992.

[66] G. L. Heileman, G. M. Papdourakis, and M Georgiopoulos. A neural net associative memory for real-time applications. *Neural Computation*, 2:107–115, 1990.

[67] B. J. Hellstrom and L. V. Kanal. Asymmetric mean-field neural networks for multiprocessor scheduling. *Neural Networks*, 5(4):671–686, 1992.

[68] B. J. Hellstrom and L. V. Kanal. Knapsack packing networks. *IEEE Transactions on Neural Networks*, 3(2):302–307, March 1992.

[69] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences USA*, 79:2554–2558, April 1982.

[70] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences USA*, 81:3088–3092, May 1984.

[71] J. J. Hopfield and D. W. Tank. 'Neural' computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[72] J. J. Hopfield and D. W. Tank. Computing with neural circuits: a model. *Science*, 233:625–633, 1986.

[73] T. C. Hu. *Integer Programming and Network Flows.* Addison-Wesley, Reading MA, 1969.

[74] A. Imiya and M. Nozaka. Data rearrangement by neural network. In *Proceedings of the INNC*, Paris, 1990.

[75] D. Ingman and Y. Merlis. Local minimum escape using thermodynamic properties of neural networks. *Neural Networks*, 4:395–404, 1991.

[76] D. Ingman and Y. Merlis. Maximum entropy signal reconstruction with neural networks. *IEEE Transactions on Neural Networks*, 3(2):195–201, March 1992.

[77] A. D. Iusem and A. R de Pierro. A simultaneous iterative method for computing projections on polyhedra. *SIAM Journal on Control and Optimization*, 25(1):231–243, January 1987.

[78] M. Izumida, K. Murakami, and T. Aibara. Analysis of neural network energy functions using standard forms. In *Electronics, Information and Communication in Japan*. Scripta Technica, Silver Spring, Maryland, in press. Translated from the Japanese by Thomas H. Hildebrandt. The original appeared in Denshi Joohoo Tsuushin Gakkai Ronbunshi (Transactions of the Institute of Electronics, Information and Communications Engineers (of Japan)), volume J74-D-II, number 6, pages 804–811, June 1991.

[79] A. Johanet, L. Personnaz, G. Dreyfus, J-D Gascuel, and M. Weinfeld. Specification and implementation of a digital Hopfield-type associative memory with on-chip training. *IEEE Transactions on Neural Networks*, 3(4):529–539, July 1992.

[80] A. B. Kahng. Travelling salesman heuristics and embedding dimension in the Hopfield model. In *Proceedings of the International Joint Conference on Neural Networks*, pages I:513–I:520, Washington DC, 1989.

[81] B. Kamgar-Parsi and B. Kamgar-Parsi. On problem solving with Hopfield neural networks. *Biological Cybernetics*, 62:415–423, 1990.

[82] I. Kanter and H. Sompolinsky. Graph optimization problems and the Potts glass. *Journal of Physics A*, 20:L673–L679, 1987.

[83] A. Kaufman and A. Henry-Labordère. *Integer and Mixed Programming: Theory and Applications*. Mathematics in Science and Engineering Vol. 137. Academic Press, New York, 1977.

[84] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, pages L673–L679, February 1970.

[85] D. B. Khang and O. Fujiwara. A new algorithm for finding all vertices of a polytope. *Operations Research Letters*, 8(5):261–264, October 1989.

[86] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[87] S. Kirkpatrick and G. Toulouse. Configuration space analysis of travelling salesman problems. *Journal de Physique*, 46:1277–1292, August 1985.

[88] T. Kohonen. *Self-Organization and Associative Memory*. Springer Series on Information Sciences. Springer-Verlag, 1989.

[89] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley and Los Angeles, 1951. University of California Press.

[90] National Physical Laboratory. *Modern Computing Methods*. Notes on Applied Science No. 16. HMSO, London, 1961.

[91] A. Langevin, F. Soumis, and J. Desrosiers. Classification of travelling salesman problem formulations. *Operations Research Letters*, 9(2):127–132, March 1990.

[92] E. L. Lawler et al., editors. *The traveling salesman problem: a guided tour of combinatorial optimization*. Wiley-Interscience series in discrete mathematics. John Wiley & Sons, Chichester, 1985.

[93] E. L. Lawler and D. E. Wood. Branch-and-bound methods: a survey. *Operations Research*, 14:699–719, 1966.

[94] B. W. Lee and B. J. Sheu. Modified Hopfield neural networks for retrieving the optimal solution. *IEEE Transactions on Neural Networks*, 2(1):137–142, January 1991.

[95] J. E. Lewis and L. Glass. Nonlinear dynamics and symbolic dynamics of neural networks. *Neural Computation*, 4(5):621–642, 1992.

[96] W. Li and N. M. Nasrabadi. Object recognition based on graph matching implemented by a Hopfield-style neural network. In *Proceedings of the International Joint Conference on Neural Networks*, pages II:287–II:290, Washington DC, 1989.

[97] S. Lin. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, 44(10):2245–2269, December 1965.

[98] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[99] C-Y Maa and M. A. Shanblatt. Linear and quadratic programming neural network analysis. *IEEE Transactions on Neural Networks*, 3(4):580–592, July 1992.

[100] C-Y Maa and M. A. Shanblatt. A two-phase optimization neural network. *IEEE Transactions on Neural Networks*, 3(6):1003–1009, November 1992.

[101] M. Marcus and H. Mink. *A Survey of Matrix Theory and Matrix Inequalities*, pages 93–101. Allyn and Bacon, Boston, 1964.

[102] C. R. K. Marrian and M. C. Peckerar. Electronic 'neural' net algorithm for maximum entropy solutions of ill-posed problems. *IEEE Transactions on Circuits and Systems*, 36(2):288–294, February 1989.

[103] T. H. Matheiss and D. S. Rubin. A survey and comparison of methods for finding all vertices of convex polyhedral sets. *Mathematics of Operations Research*, 5(2):167–185, May 1980.

[104] N. Metropolis, A. W. Rosenbluth, and M. N. Rosenbluth. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087, 1953.

[105] D. A. Miller and S. W. Zucker. Efficient simplex-like methods for equilibria of nonsymmetric analog networks. *Neural Computation*, 4:167–190, 1992.

[106] E. Minieka. *Optimization Algorithms for Networks and Graphs.* Industrial Engineering. Marcel Dekker, New York, 1978.

[107] E. Mjolsness and C. Garrett. Algebraic transformations of objective functions. *Neural Networks*, 3:651–669, 1990.

[108] E. Mjolsness and W. L. Miranker. A Lagrangian approach to fixed points. In P. Lippmann et al., editors, *Advances in Neural Information Processing Systems 3*, pages 77–83. Morgan Kaufman, 1991.

[109] T. S. Motzkin and I. J. Schoenberg. The relaxation method for linear inequalities. *Canadian Journal of Mathematics*, 6:393–404, 1954.

[110] A. F. Murray, D. Del Corso, and L. Tarassenko. Pulse-stream VLSI neural networks mixing analog and digital techniques. *IEEE Transactions on Neural Networks*, 2(2):193–204, March 1991.

[111] R. M. Nauss. *Parametric Integer Programming.* University of Missouri Studies. University of Missouri Press, Columbia, 1979.

[112] R. G. Ogier and D. A. Beyer. Neural network solution to the link scheduling problem using convex relaxation. In *Proceedings of the IEEE Global Telecommunications Conference*, pages 1371–1376, San Diego, 1990.

[113] M. Ohlsson, C. Peterson, and B. Söderberg. Neural networks for optimization problems with inequality constraints — the knapsack problem. Technical Report LU TP 92-11, Department of Theoretical Physics, University of Lund, Sweden, March 1992.

[114] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization — Algorithms and Complexity.* Prentice-Hall, New Jersey, 1982.

[115] C. H. Papadimitriou and M. Yannakakis. Note on recognizing integer polyhedra. *Combinatorica*, 10(1):107–109, 1990.

[116] C. Peterson. Parallel distributed approaches to combinatorial optimization: Benchmark studies on the traveling salesman problem. *Neural Computation*, 2:261–269, 1990.

[117] C. Peterson and J. R. Anderson. Neural networks and NP-complete optimization problems; a performance study on the graph bisection problem. *Complex Systems*, 2(1):59–89, 1988.

[118] C. Peterson and B. Söderberg. A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 1(1):3–22, 1989.

[119] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C — The Art of Scientific Computing.* Cambridge University Press, Cambridge, 1988.

[120] E. M. Reingold, J. Nevergelt, and N. Deo. *Combinatorial Algorithms — Theory and Practice.* Prentice-Hall, New Jersey, 1977.

[121] T. L. Saaty. The number of vertices of a polyhedron. *The American Mathematical Monthly*, 62:326–331, 1955.

[122] H. M. Salkin. *Integer Programming.* Addison-Wesley, Reading MA, 1975.

[123] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Chichester, 1986.

[124] Y. Shrivastava, S. Dasgupta, and S. M. Reddy. Guaranteed convergence in a class of Hopfield networks. *IEEE Transactions on Neural Networks*, 3(6):951–961, November 1992.

[125] P. D. Simić. Statistical mechanics as the underlying theory of 'elastic' and 'neural' optimisations. *Network*, 1:89–104, 1990.

[126] P. D. Simić. Constrained nets for graph matching and other quadratic assignment problems. *Neural Computation*, 3:268–281, 1991.

[127] M. W. Simmen. Parameter sensitivity of the elastic net approach to the traveling salesman problem. *Neural Computation*, 3:363–374, 1991.

[128] M. J. S. Smith and C. L. Portmann. Practical design and analysis of a simple "neural" optimization circuit. *IEEE Transactions on Circuits and Systems*, 36(1):42–50, January 1989.

[129] S. I. Sudharsanan and M. K. Sundareshan. Exponential stability and a systematic synthesis of a neural network for quadratic minimization. *Neural Networks*, 4:599–613, 1991.

[130] D. Suter. Constraint networks in vision. *IEEE Transactions on Computers*, 40(12):1359–1367, December 1991.

[131] G. A. Tagliarini, J. Fury Christ, and E. W. Page. Optimization using neural networks. *IEEE Transactions on Computers*, 40(12):1347–1358, December 1991.

[132] G. A. Tagliarini and E. W. Page. A neural-network solution to the concentrator assignment problem. In D. Z. Anderson, editor, *Neural Information Processing Systems*, pages 775–782. American Institute of Physics, New York, 1988.

[133] M. Takeda and J. W. Goodman. Neural networks for computation: number representations and programming complexity. *Applied Optics*, 25(18):3033–3046, September 1986.

[134] Y. Takefuji and K-C. Lee. An artificial hysteresis binary neuron: a model suppressing the oscillatory behaviors of neural dynamics. *Biological Cybernetics*, 64:353–356, 1991.

[135] D. W. Tank and J. J. Hopfield. Simple 'neural' optimization networks: An A/D converter, signal decision circuit, and a linear programming circuit. *IEEE Transactions on Circuits and Systems*, 33(5):533–541, May 1986.

[136] L. Tarassenko, J. N. Tombs, and J. H. Reynolds. Neural network architectures for content-addressable memory. *IEE Proceedings, Series F*, 138(1):33–39, February 1991.

[137] A. M. Thompson, J. C. Brown, J. W. Kay, and D. M. Titterington. A study of methods of choosing the smoothing parameter in image restoration by regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(4):326–339, April 1991.

[138] V. Tresp. A neural network approach for three-dimensional object recognition. In P. Lippmann et al., editors, *Advances in Neural Information Processing Systems 3*, pages 306–312. Morgan Kaufman, 1991.

[139] A. G. Tsirukis, G. V. Reklaitis, and M. F. Tenorio. Nonlinear optimization using generalized Hopfield networks. *Neural Computation*, 1:511–521, 1989.

[140] S. Ullman. *The Interpretation of Visual Motion.* MIT Press, Cambridge MA, 1979.

[141] M. K. Unaltuna, M. E. Dalkilic, and V. Pitchumani. Solving the scheduling problem in high level synthesis using a normalized mean field neural network. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 275–280, San Francisco, 1993.

[142] D. E. Van den Bout and T. K. Miller III. Improving the performance of the Hopfield-Tank neural network through normalization and annealing. *Biological Cybernetics*, 62:129–139, 1989.

[143] D. E. Van den Bout and T. K. Miller III. Graph partitioning using annealed neural networks. *IEEE Transactions on Neural Networks*, 1(2):192–203, June 1990.

[144] M. M. Van Hulle. A goal programming network for linear programming. *Biological Cybernetics*, 65:243–252, 1991.

[145] S. S. Venkatesh and D. Psaltis. Linear and logarithmic capacities in associative neural networks. *IEEE Transactions on Information Theory*, 35(3):558–568, May 1989.

[146] M. Vidyasagar. Location and stability of the equilibria of nonlinear neural networks. Technical Report 91–01, Centre for Artificial Intelligence and Robotics, Bangalore, India, February 1991.

[147] J. von Neumann. A certain zero-sum two-person game equivalent to the optimal assignment problem. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games, II.* Annals of Mathematics Studies 28, Princeton University Press, Princeton, New Jersey, 1953.

[148] L-X. Wang. A neural detector for seismic reflectivity sequences. *IEEE Transactions on Neural Networks*, 3(2):338–340, March 1992.

[149] V. Wilson and G. S. Pawley. On the stability of the TSP problem algorithm of Hopfield and Tank. *Biological Cybernetics*, 58:63–70, 1988.

[150] S. Zhang and A. G. Constantinides. Lagrange programming neural networks. *IEEE Transactions on Circuits and Systems, II: Analog and Digital Signal Processing*, 39(7):441–452, July 1992.

[151] S. Zhang, X. Zhu, and L-H. Zou. Second-order neural nets for constrained optimization. *IEEE Transactions on Neural Networks*, 3(6):1021–1024, November 1992.

[152] Y. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins. Image restoration using a neural network. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 36(7):1141–1151, July 1988.

[153] S. Zionts. *Linear and Integer Programming.* Prentice-Hall, New Jersey, 1974.

[154] V. Zissimopoulos, V. T. Paschos, and F. Pekergin. On the approximation of NP-complete problems by using Boltzmann machine method: the cases of some covering and packing problems. Technical report, Laboratoire de Recherche en Informatique, Université Paris Sud, 91405 Orsay, France, 1990.