Polyhedral Combinatorics and Neural Networks

(CUED/F-INFENG/TR 100)

Andrew H. Gee and Richard W. Prager [1]

Cambridge University Engineering Department
Trumpington Street
Cambridge CB2 1PZ
England

May 1992

Abstract

The often disappointing performance of optimizing neural networks can be partly attributed to the rather *ad hoc* manner in which problems are mapped onto them for solution. In this paper a rigorous mapping is described for quadratic 0-1 programming problems with linear equality and inequality constraints, this being the most general class of problem such networks can solve. The problem's constraints define a polyhedron P containing all the valid solution points, and the mapping guarantees strict confinement of the network's state vector to P. However, forcing convergence to a 0-1 point within P is shown to be generally intractable, rendering the Hopfield and similar models inapplicable to the vast majority of problems. Some alternative dynamics, based on the principle of tabu search, are presented as a more coherent approach to general problem solving. When tested on a large database of solved problems, the alternative dynamics produced some very encouraging results.

1 Introduction and Outline

While feedback neural networks have been studied for some time as a means of approximately solving combinatorial optimization problems, many researchers have reported that such networks frequently fail to find valid solutions, let alone high quality ones [21, 36]. We suggest that a significant cause of such failures has been the rather *ad hoc* manner in which problems are mapped onto the network for solution. Accepting that such networks can perform nothing more sophisticated than continuous bounded descent on a quadratic energy function, it is quite possible to derive efficient mappings to exploit these simple dynamics to the full.

Throughout this paper we draw on results from traditional mathematical programming [29] to examine both problems and network dynamics from a polyhedral perspective. For a general quadratic 0-1 programming problem with linear equality and inequality constraints, it is possible to identify a bounded polyhedron, or polytope, within which the solution vector must lie if it is not to violate any of the aforementioned constraints. In this paper we describe a rigorous mapping for such problems, which guarantees confinement of the network's state vector to the constraint polytope; hence the possibility of finding invalid solutions is eliminated. The mapping is a natural development of the 'valid subspace' approach presented in [2, 4]. However, confinement to the correct polytope is not the final goal: to find a valid solution we require convergence to a 0-1 point within this polytope. We demonstrate that forcing the network to converge to such a point is probably impossible, since finding such a point is generally an $\mathcal{NP}$-complete problem. In particular, this means that for correctly mapped problems, the popular 'annealing' techniques, often employed to force convergence to a 0-1 point, are quite ineffective.

For the small class of *integral* polytopes [29], convergence to a 0-1 point *is* feasible, though such polytopes are the exception and not the rule. It follows that conventional neural network techniques

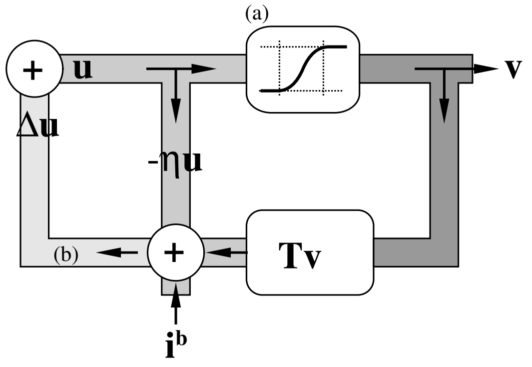
---

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

are quite unsuited to optimization over most polytopes. In response, we present modified network dynamics, based on the principle of tabu search, which, though by no means the only solution to the problem, constitute a more coherent approach to optimization over non-integral polytopes. Tabu search was first married with Hopfield-style dynamics in [5, 6] as a means of local minimum escape. Here we adapt the tabu dynamics to produce a viable neural *search* technique, which should be contrasted with the one-shot descent offered by the Hopfield and similar models. The dynamics are tested on a large database of solved problems, with very encouraging results.

The organization of the paper is as follows. In Section 2 we review continuous descent dynamics, including the Hopfield model, and their application to the solution of 0-1 programming problems. We proceed to describe in Section 3 a rigorous mapping for the most general quadratic 0-1 programming problems with linear equality and inequality constraints. In the light of this mapping we go on to address the issue of efficient simulation of descent dynamics on discrete-time machines. In Section 4 we examine the convergence properties of such dynamics, concluding that it is probably impossible to force convergence to a 0-1 point for most problems; in so doing we discuss the deficiencies of the various annealing techniques in this context. A more coherent approach is offered by tabu search dynamics, which we develop in Section 5 and apply to a large database of solved problems, with very encouraging results. Finally, in Section 6 we discuss the key issues raised in the paper and present our conclusions.

2 Bounded Descent Techniques



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

(b) The change in $\mathbf{u}$ is specified by the differential equation (1).

Figure 1: Schematic diagram of the continuous Hopfield network.

While the arguments of this paper are of relevance to any method attempting to solve 0-1 programming problems by continuous descent, of particular interest is the Hopfield model technique, since it has been the subject of significant research in the recent past, and highlights the potential advantages of fast, analogue implementations. A schematic diagram of the continuous Hopfield network [18] is shown in Figure 1. 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:

$$\begin{aligned} \dot{\mathbf{u}} &= -\eta\mathbf{u} + \mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}} && (1) \\ v_i &= g(u_i) && (2) \end{aligned}$$

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

$$g(u_i) = \frac{1}{1+\exp(-u_i/T)} \qquad (3)$$

If $\mathbf{T}$ is symmetric, and either $\eta = 0$ or the gain of the transfer functions (3) is high (as we shall assume throughout this paper), the system has a Liapunov function [18]

$$E(\mathbf{v}) = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{b}} \tag{4}$$

The Hopfield network therefore provides a continuous method of performing bounded descent minimization of functions of the form (4) with symmetric $\mathbf{T}$. By 'bounded', we mean that the descent is limited to the unit hypercube, at the vertices of which lie the 0-1 points; the network is therefore potentially suited to the solution of 0-1 programming problems. Should we wish to solve any particular problem, we must first decide how to set the network parameters $\mathbf{T}$ and $\mathbf{i}^{\mathrm{b}}$, so that minimization of the Liapunov function (4) coincides with minimization of the problem's objective function and enforces satisfaction of the problem's constraints; this process is termed 'mapping' the problem onto the network. The particular attraction of the network lies in the existence of an analogue electrical circuit with dynamics corresponding to (1) and (2) [18]. It is also possible to use the network for cases where $\mathbf{T}$ is not symmetric, by noting that

$$E = \tfrac{1}{2}(E + E^T) = -\tfrac{1}{2}\mathbf{v}^T\left[\tfrac{1}{2}(\mathbf{T} + \mathbf{T}^T)\right]\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{b}} \tag{5}$$

Hence, should we wish to use the network to minimize a function E with a non-symmetric $\mathbf{T}$, 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.

Also of interest are steepest descent dynamics

$$\dot{v}_i = \begin{cases} 0 & \text{if } \left[\mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}}\right]_i < 0 \text{ and } v_i = 0 \\ 0 & \text{if } \left[\mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}}\right]_i > 0 \text{ and } v_i = 1 \\ \left[\mathbf{T}\mathbf{v} + \mathbf{i}^{\mathrm{b}}\right]_i & \text{otherwise} \end{cases} \tag{6}$$

The dynamics (6) share the same Liapunov function (4) as the Hopfield network, and the $\mathbf{v}$ variables are also limited to the range $0 \leq v_i \leq 1$. Thus (6) provides an alternative method of minimizing E within the unit hypercube, and has the advantage of a more efficient simulation on digital computers, as we shall see in Section 3.3. Throughout this paper we shall refer to bounded, quadratic continuous descent systems (like the Hopfield and steepest descent models) as *descent networks*.

3 Quadratic 0-1 Programming on Descent Networks

3.1 Problem statement

.

In this paper, we study quadratic 0-1 programming problems with linear constraints, this being the most general class of problem which can be mapped onto a descent network for solution. The mathematical statement of such a problem is

$$\begin{aligned} \text{minimize} \quad & E^{\mathrm{op}}(\mathbf{v}) = -\tfrac{1}{2}\mathbf{v}^T\mathbf{T}^{\mathrm{op}}\mathbf{v} - \mathbf{v}^T\mathbf{i}^{\mathrm{op}} && (7) \\ \text{subject to} \quad & \mathbf{A}^{\mathrm{eq}}\mathbf{v} = \mathbf{b}^{\mathrm{eq}} && (8) \\ \text{and} \quad & \mathbf{A}^{\mathrm{in}}\mathbf{v} \leq \mathbf{b}^{\mathrm{in}} && (9) \\ \text{and} \quad & 0 \leq v_i \leq 1, \quad i \in \{1, \ldots, n\} && (10) \\ \text{and} \quad & \mathbf{v} \text{ integral} && (11) \end{aligned}$$

where $\mathbf{v} \in \mathbb{R}^n$, $\mathbf{b}^{\mathrm{eq}} \in \mathbb{R}^{m^{\mathrm{eq}}}$ and $\mathbf{b}^{\mathrm{in}} \in \mathbb{R}^{m^{\mathrm{in}}}$. With reference to condition (11), we define an integral vector to be one whose elements are all integers. The system (7)–(11) describes an optimization problem in which it is required to minimize the quadratic objective function (7) subject to a set of linear equality (8) and inequality (9) constraints on the elements of $\mathbf{v}$. The final two conditions (10) and (11) ensure that $v_i \in \{0, 1\}$.

From a geometrical point of view, we note that the conditions (8)–(10), 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. When attempting to solve such problems with descent networks, we strive towards discovering a mapping so that $\mathbf{v}$ remains strictly within P while at the same time performing some sort of descent on the objective function (7). It is often also necessary to employ an annealing process to free $\mathbf{v}$ from local minima of E^{op} and to drive $\mathbf{v}$ towards a hypercube corner, where the condition (11) is satisfied. However, it is often the case that problems are mapped onto descent networks in a rather *ad hoc* manner, and strict confinement to P is rarely achieved.

3.2 A rigorous mapping for quadratic 0-1 programming problems

In this section we describe a rigorous mapping for problems of the form (7)–(11) which achieves all the goals mentioned in Section 3.1. In particular, the mapping ensures that $\mathbf{v}$ remains strictly confined within P, thus ruling out the possibility of the network finding an invalid solution. The mapping is a natural extension of the 'valid subspace' technique presented in [2, 4].

We begin by considering the simple case in which there are no inequality constraints, leaving only equality constraints of the form (8). The equality constraints constitute a system of linear simultaneous 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

$$\begin{aligned} \mathbf{v} &= \mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s} && (12) \\ \text{where} \quad \mathbf{T}^{\mathrm{val}} &= \mathbf{I} - \mathbf{A}^{\mathrm{eq}T}(\mathbf{A}^{\mathrm{eq}}\mathbf{A}^{\mathrm{eq}T})^{-1}\mathbf{A}^{\mathrm{eq}} && (13) \\ \text{and} \quad \mathbf{s} &= \mathbf{A}^{\mathrm{eq}T}(\mathbf{A}^{\mathrm{eq}}\mathbf{A}^{\mathrm{eq}T})^{-1}\mathbf{b}^{\mathrm{eq}} && (14) \end{aligned}$$

If we now set the network's Liapunov function to be

$$E = E^{\mathrm{op}} + \tfrac{1}{2}c_0\|\mathbf{v} - (\mathbf{T}^{\mathrm{val}}\mathbf{v} + \mathbf{s})\|^2 \tag{15}$$

then, in the limit of large c_0, $\mathbf{v}$ will satisfy (12) and therefore (8) throughout convergence. In this way all the equality constraints have been combined into a single penalty term in the network's Liapunov function; moreover, this term does not interfere with the objective E^{op} at any point within the polytope P. The Liapunov function (15) may be achieved by setting the network's parameters as follows [2, 14]:

$$\begin{aligned} \mathbf{T} &= \mathbf{T}^{\mathrm{op}} + c_0(\mathbf{T}^{\mathrm{val}} - \mathbf{I}) && (16) \\ \mathbf{i}^{\mathrm{b}} &= \mathbf{i}^{\mathrm{op}} + c_0\mathbf{s} && (17) \end{aligned}$$

It now remains only to point out that the network's transfer functions naturally enforce conditions (10), and so confinement of $\mathbf{v}$ within P is guaranteed.

The technique can be extended to cope with inequality constraints of the form (9). It is first necessary to convert all the inequality constraints to equality constraints by introducing one slack variable for each inequality constraint (a similar approach has been previously proposed for a specific problem in [30]). For example, a typical constraint

$$A^{\mathrm{in}}_{i1}v_1 + A^{\mathrm{in}}_{i2}v_2 + \ldots + A^{\mathrm{in}}_{in}v_n \leq b^{\mathrm{in}}_i \tag{18}$$

becomes

$$A^{\mathrm{in}}_{i1}v_1 + A^{\mathrm{in}}_{i2}v_2 + \ldots + A^{\mathrm{in}}_{in}v_n + k_i w_i = b^{\mathrm{in}}_i, \quad k_i w_i \geq 0 \tag{19}$$

Here 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[2]. Thus we

---

[2] This can be achieved by setting k_i as follows:

$$k_i = \begin{cases} b^{\mathrm{in}}_i - \sum_{\{j|A^{\mathrm{in}}_{ij}<0\}} A^{\mathrm{in}}_{ij} & \text{if } b^{\mathrm{in}}_i > 0 \\ -\sum_{\{j|A^{\mathrm{in}}_{ij}<0\}} A^{\mathrm{in}}_{ij} & \text{if } b^{\mathrm{in}}_i \leq 0 \end{cases} \tag{20}$$

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 $[w_1\ w_2\ \ldots\ w_{m^{\text{in}}}]^T$: the size of $\mathbf{v}^+$ is therefore $n^+ = n + m^{\text{in}}$. The optimization problem may now be expressed as

$$\begin{aligned} &\text{minimize} && E^{\text{op}}(\mathbf{v}) = -\tfrac{1}{2}{\mathbf{v}^+}^T \mathbf{T}^{\text{op}}\mathbf{v}^+ - {\mathbf{v}^+}^T \mathbf{i}^{\text{op}} && (21)\\ &\text{subject to} && \mathbf{A}^+\mathbf{v}^+ = \mathbf{b}^+ && (22)\\ &\text{and} && 0 \le v_i^+ \le 1, \quad i \in \{1, \ldots, n^+\} && (23)\\ &\text{and} && \mathbf{v} \text{ integral} && (24) \end{aligned}$$

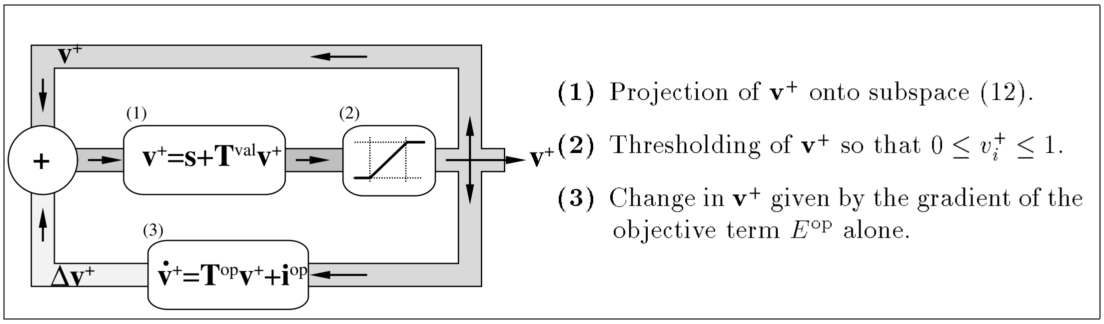
where $\mathbf{v}^+ \in \mathbb{R}^{n^+}$ and $\mathbf{b}^+ \in \mathbb{R}^{(m^{\text{eq}}+m^{\text{in}})}$. Note that E^{op} has no dependency on the slack variables $\mathbf{w}$, and the equality constraints (22) embody both the original equality and inequality constraints (8) and (9). The formulation (21)–(24) contains only equality constraints, and so the mapping technique described above is applicable. We can therefore find $n^+ \times n^+$ matrices $\mathbf{T}^{\text{op}}$ and $\mathbf{T}^{\text{val}}$, and n^+-element vectors $\mathbf{i}^{\text{op}}$ and $\mathbf{s}$, such that if the network's parameters are set as in (16)–(17), the state vector $\mathbf{v}^+$ will perform a descent on E^{op} while remaining strictly within the polytope P^+ defined by (22) and (23). If we observe only the variables $\mathbf{v}$ 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.

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. Thus it is not necessary (and indeed incorrect) to force the slack variables towards 0 or 1 using an annealing technique. To find a valid solution we need only ensure that $\mathbf{v}$ converges to an integral point within P, which is the subject of Section 4.

3.3 Efficient simulation on digital computers

It is the norm to study continuous descent techniques indirectly, through the use of simulations on digital computers, and yet there is precious little in the literature on this topic. The simulation of continuous systems on discrete machines is a tricky and time-consuming process, the precise details of which can have a significant bearing on any experimental results.

In this section we shall consider discrete-time simulation of descent dynamics on Liapunov functions of the form (15), as encountered in proper mappings of 0-1 programming problems. A standard Euler approximation of the dynamic equations, though feasible, would require a very small time step at each iteration, and would therefore be very slow to converge. This is because c_0 must be very 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 P^+.



(1) Projection of $\mathbf{v}^+$ onto subspace (12).

(2) Thresholding of $\mathbf{v}^+$ so that $0 \le v_i^+ \le 1$.

(3) Change in $\mathbf{v}^+$ given by the gradient of the objective term E^{op} alone.

Figure 2: Schematic diagram of the efficient descent algorithm.

Figure 2 shows a schematic illustration of a far more efficient simulation algorithm (previously presented in [2]) for the steepest descent dynamics (6). Each iteration has two distinct stages. First, as depicted in box (3), $\mathbf{v}^+$ is updated over a finite time step Δt using the gradient of the objective term $E^{\rm op}$ alone. The time step can be far larger than that for the standard Euler approximation, since the problematic c_0 term is not implemented. 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 through boxes (1) and (2), in which $\mathbf{v}^+$ is first orthogonally projected onto the subspace given by (12), 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 [1, 11, 20, 24]. The resulting algorithm is highly efficient and has general applicability to any quadratic 0-1 programming problem.

4 Polyhedral Issues in Convergence

4.1 Convergence of v in P

Having seen how it is possible to achieve descent dynamics on $E^{\rm op}$ while remaining within the polytope P, we now need to investigate the convergence properties of such a process. This is best illustrated by an example. Consider the following knapsack problem in two variables

$$\begin{aligned} \text{minimize} \quad & E^{\rm op}(\mathbf{v}) = -(v_1 + 2v_2) && (25)\\ \text{subject to} \quad & v_1 + v_2 \leq 1.7 && (26)\\ \text{and} \quad & 0 \leq v_i \leq 1, \quad i \in \{1,2\} && (27)\\ \text{and} \quad & \mathbf{v} \text{ integral} && (28) \end{aligned}$$

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 just how useful that item is. We wish to decide which items to pack to maximize the profit, while at the same time not overfilling the knapsack. For knapsack problems the objective function $E^{\rm op}$ is linear and therefore has no local minima. The problem is easily mapped onto a descent network for solution, through the introduction of one slack variable to cope with the single inequality constraint (26). It is clear that the optimal solution to our simple example problem is $\mathbf{v} = [0\ 1]^T$, though the general knapsack problem is $\mathcal{NP}$-complete [13, 29].

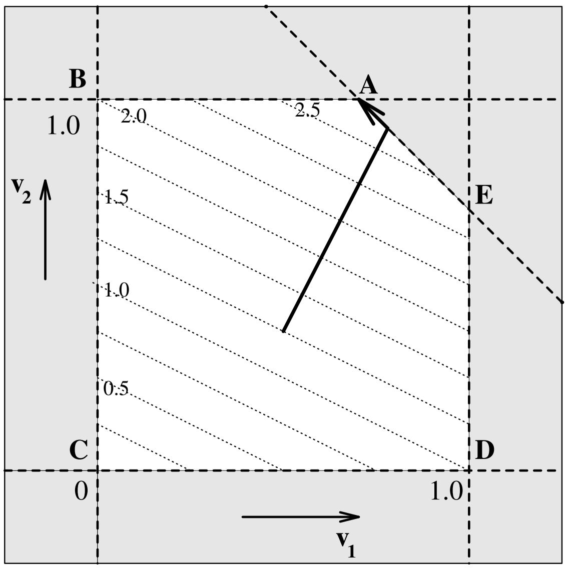


Figure 3: The polytope P, energy contours (shown in dotted lines) and a typical descent path for the example knapsack problem.

Figure 3 shows the polytope P for this problem. Also marked are contours of $E^{\rm op}$ 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. This vertex is in fact the optimal solution to the problem (25)–(27), which is the *LP-relaxation* (linear programming relaxation) of the knapsack problem. The LP-relaxation is a standard linear programming problem, involving the minimization of a linear objective subject to linear constraints. Linear programming is solvable in polynomial time using Khachiyan's method, though in practice the (non-polynomial) simplex method is usually more efficient. It is the integrality condition (28) which makes the knapsack problem $\mathcal{NP}$-complete; indeed, while linear programming is solvable in polynomial time, the general integer linear programming problem is $\mathcal{NP}$-complete [29].

It is apparent that the correct mapping of the knapsack problem leads the network to solve the LP-relaxation problem accurately[3], though a valid 0-1 solution is not found. So how can we force the descent to converge to B, C or D, one of the 0-1 points in P? The usual approach is to use some kind of annealing process, and we will examine the use of hysteretic annealing [12] (variants have been reported as convex relaxation [25] and matrix graduated non-convexity [2]) with this problem. The idea behind hysteretic annealing is to add a term of the form

$$E^{\rm ann} = -\gamma \sum_{i=1}^{n} (v_i - 0.5)^2 \tag{29}$$

to the objective function $E^{\rm op}$. The function $E^{\rm ann}$ is either convex or concave, depending on the sign of γ, and has full spherical symmetry around the point $\mathbf{v}^T = [0.5\ 0.5\ \ldots\ 0.5]$. A consequence is that $E^{\rm ann}$ has the same value at all 0-1 points, and therefore does not invalidate the objective for 0-1 programming problems. The annealing process is usually initialized with a large negative value of γ, in which case the Liapunov function E is convex and $\mathbf{v}$ converges to a point within the interior of P. Subsequently, the value of γ is gradually increased, eventually E 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 $E^{\rm op}$ and can help guide $\mathbf{v}$ towards good solutions, though this latter advantage is somewhat problem dependent [14].

Applied to the knapsack problem in Figure 3, however, hysteretic annealing has no useful effect. For $\mathbf{v}$ trapped at the vertex A, no amount of increasing γ will drive $\mathbf{v}$ towards a 0-1 point, at least until the magnitude of γ becomes comparable with that of c_0 in (16) and (17), at which point $\mathbf{v}$ will depart from the polytope 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[4].

---

[3] Indeed, a descent network correctly set up using the mapping in Section 3.2 will reliably solve linear programming problems, without the need for any modifications as proposed in [32, 34].

[4] We might well ask whether any other form of annealing might be more successful. For the Hopfield network, it is important to realize that, when $\eta = 0$, increasing the gain of the neuron transfer functions (3) has absolutely no effect, since the Liapunov function remains unchanged. Taking the knapsack problem in Figure 3 as an example, when $\mathbf{v}$ is at the vertex A, any direction away from the vertex is 'uphill' in the sense of the Liapunov function E. Since $\mathbf{v}$ changes in such a way that E is non-increasing, it is clear that $\mathbf{v}$ will not move from the vertex, however steep the transfer functions become. As a form of annealing, varying the neuron gains is therefore ineffective. The only other form of annealing to consider is mean field annealing (MFA) with neuron normalization, as proposed for the travelling salesman and graph partitioning problems in [28, 33]. In this technique a constraint of the form

$$\sum_{i \in S} v_i = 1 \tag{30}$$

is implicitly enforced for some set of neurons S using an update rule

$$v_i = \frac{\exp(-\frac{\partial E}{\partial v_i}/T)}{\sum_{k \in S} \exp(-\frac{\partial E}{\partial v_k}/T)} \tag{31}$$

As the annealing progresses, the 'temperature' T is gradually lowered until, when fully converged, $\mathbf{v}$ represents a 0-1 solution to the constraint equation (30). The annealing prevents $\mathbf{v}$ from getting trapped in local minima of $E^{\rm op}$ and enforces convergence to a 0-1 point which satisfies (30). However, the general problem of finding a 0-1 solution to the equation

$$\mathbf{a}^T\mathbf{v} = \beta \quad (\mathbf{a}\text{ being a rational vector and } \beta \text{ a rational scalar}) \tag{32}$$

In fact, the failure of annealing techniques to find a 0-1 point within P is hardly surprising, since finding such points is generally an $\mathcal{NP}$-complete problem [29]. The best we can expect from annealing procedures is to force convergence to some vertex of P. To see this, first note that increasing γ in hysteretic annealing will eventually make E^{op} concave. Furthermore, it can be easily proved (and is intuitively obvious) that if the objective 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* [29].

Integral polytopes constitute an important concept in the field of mathematical programming. For example, it follows that integer linear programming over integral polytopes can be solved in polynomial time using Khachiyan's method [29], though the general integer linear programming problem in $\mathcal{NP}$-complete. Likewise, we now see that integral polytopes are highly relevant to continuous descent solution techniques, which will reliably converge to a valid solution point only if the optimization is over an integral polytope. Neural network techniques, in their conventional form, are quite unsuited to the solution of 0-1 programming problems over non-integral polytopes.

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 [26]. However, it *is* possible to recognize some special classes of integrality, covering some of the polytopes implicitly studied by the neural network community. While it is beyond the scope of this paper to discuss the recognition of integral polytopes in detail (a good summary may be found in [29]), it is worth remarking that the polytope associated with the traditional Hopfield-Tank mapping of the travelling salesman problem [19, etc] is indeed integral [7, 23, 35]. This polytope, which contains all doubly stochastic matrices and has the permutation matrices at its vertices, has been well studied in [8, 9, 10]. Unfortunately, the objective function associated with the Hopfield-Tank mapping is not linear, and so while convergence to a valid solution point can be guaranteed, convergence to the optimal solution point cannot. This should be contrasted with the mappings of the travelling salesman problem traditionally studied by the mathematical programming community, in which a linear objective over a non-integral polytope is considered [22, 29, etc].

The strength of the neural network technique therefore lies in the use of annealing over integral polytopes. However, as we have seen with the knapsack problem, most problems do not present themselves naturally in this form. On the other hand, we have seen that the travelling salesman problem can be mapped either with a quadratic objective over an integral polytope, or with a linear objective over a non-integral polytope. This suggests that it might be possible to systematically map arbitrary problems onto either framework. Failing this, we could investigate alternative network dynamics for non-integral polytopes, which is the subject of Section 5 .

4.2 Convergence of $\mathbf{v}^+$ in P^+

To complete our understanding of the convergence process, we need to examine the trajectory of $\mathbf{v}^+$ in P^+ which gives rise to the trajectory of $\mathbf{v}$ in P. Taking the knapsack problem of Figure 3 as an example, in order to solve the problem with Hopfield or steepest descent dynamics, we need to introduce one slack variable w to cope with the inequality constraint (26). The problem, now defined over an extended variable set $\mathbf{v}^+ = [v_1\ v_2\ w]^T$, becomes

$$\text{minimize} \quad E^{\mathrm{op}}(\mathbf{v}) = -(v_1 + 2v_2) \tag{33}$$

$$\text{subject to} \quad v_1 + v_2 + 1.7w = 1.7 \tag{34}$$

$$\text{and} \quad 0 \leq v_i^+ \leq 1, \quad i \in \{1,2,3\} \tag{35}$$

$$\text{and} \quad \mathbf{v} \text{ integral} \tag{36}$$

---

is $\mathcal{NP}$-complete [29], and so we would not expect to be able to find update rules like (31) to enforce more general linear constraints of the form (32). We therefore conclude that MFA is not relevant to this paper's discussion of problems with general linear constraints.

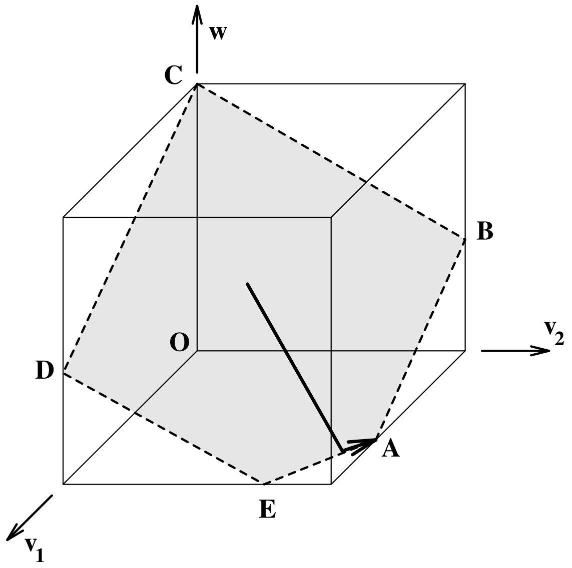


Figure 4: The polytope P^+ and a typical descent path for the example knapsack problem.

Equations (34)–(35) define a polytope P^+, which is shown in Figure 4. We see that 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 (26) into the constraint $w \geq 0$, which is directly enforced by the network's transfer functions. Furthermore, the trajectory of $\mathbf{v}$ is the projection of the trajectory of $\mathbf{v}^+$ onto the v_1–v_2 plane. Thus, when we speak of $\mathbf{v}$ being trapped at a vertex of P, we automatically imply that $\mathbf{v}^+$ is trapped at a vertex of P^+. It follows that all the arguments of the previous section are still applicable in the context of the slack variable mapping. In particular, to free $\mathbf{v}$ from a vertex of P we necessarily have to free $\mathbf{v}^+$ from the corresponding vertex of P^+; in the next section we shall examine some modified network dynamics designed to achieve this goal.

5 Tabu Search Networks for Non-Integral Polytopes

5.1 System definition

While conventional neural optimization techniques are completely unsuited to optimization over non-integral polytopes, a minor modification to their dynamics results in a far more coherent approach to the problem. In this section we describe how tabu search dynamics, previously presented in [5, 6] as a means of local minimum escape, can be modified to ensure that $\mathbf{v}^+$ escapes from vertices of P^+.

Tabu search [16] was originally presented 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 [16].

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 the Hopfield and similar models.

The implementation of the search on feedback networks is particularly elegant. In what follows we build on the approach in [5, 6], adapting it where necessary to achieve our specific goal of escape from vertices of P^+. We consider a time varying objective function

$$E_t^{\mathrm{op}}(\mathbf{v}^+, t) = E^{\mathrm{op}}(\mathbf{v}) + F_t(\mathbf{v}^+, t) \tag{37}$$

where

$$F_t(\mathbf{v}^+, t) = \beta \int_0^t e^{\alpha(s-t)} p\left(\mathbf{v}^+, \mathbf{f}(\mathbf{v}^+(s))\right)\, ds \tag{38}$$

In (38), α and β are positive constants, $p(\mathbf{a}, \mathbf{b})$ measures the proximity of the vectors $\mathbf{a}$ and $\mathbf{b}$, 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

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

It is apparent that 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

$$\mathbf{f}(\mathbf{a}) = \mathbf{a} + \epsilon(\mathbf{a} - \mathbf{c}) \tag{39}$$

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

$$\begin{aligned} F_t(\mathbf{v}^+, t) &= -\tfrac{1}{2}\theta \mathbf{v}^{+T}\mathbf{v}^+ - \mathbf{h}^T\mathbf{v}^+ + g(t) && (40)\\ \text{where} \quad \theta(t) &= 2\beta \int_0^t e^{\alpha(s-t)}\, ds && (41)\\ \text{and} \quad \mathbf{h}(t) &= -2\beta \int_0^t e^{\alpha(s-t)} \mathbf{f}(\mathbf{v}^+(s))\, ds && (42) \end{aligned}$$

Equations (41) and (42) can be differentiated to obtain the following dynamic update equations for θ and $\mathbf{h}$

$$\begin{aligned} \dot{\theta} &= 2\beta - \alpha\theta && (43)\\ \dot{\mathbf{h}} &= -2\beta\mathbf{f}(\mathbf{v}^+) - \alpha\mathbf{h} && (44) \end{aligned}$$

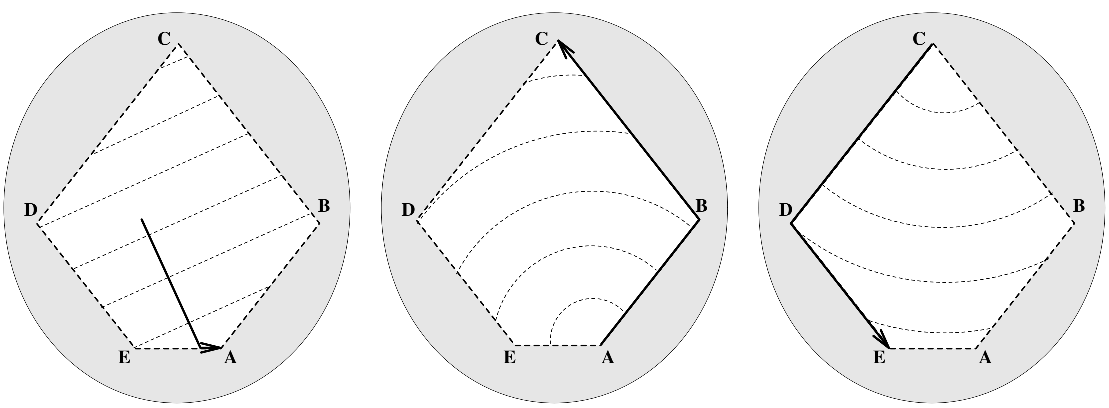
with initial conditions $\theta(0) = 0$ and $\mathbf{h}(0) = \mathbf{0}$. The overall tabu search system is obtained by arranging for the Liapunov function to be

$$E(\mathbf{v}^+) = -\tfrac{1}{2}\mathbf{v}^{+T}\mathbf{T}\mathbf{v}^+ - \mathbf{v}^{+T}\mathbf{i}^{\mathrm{b}} = E_t^{\mathrm{op}}(\mathbf{v}^+, t) - g(t) + \tfrac{1}{2}c_0||\mathbf{v}^+ - (\mathbf{T}^{\mathrm{val}}\mathbf{v}^+ + \mathbf{s})||^2 \tag{45}$$

where we have discarded $g(t)$, the part of F_t which was not dependent on $\mathbf{v}^+$. Remember that the $\mathbf{T}^{\mathrm{val}}$ and $\mathbf{s}$ terms enforce confinement to the polytope P^+, as discussed in Section 3.2. The Liapunov function (45) is achieved by setting

$$\begin{aligned} \mathbf{T} &= \mathbf{T}^{\mathrm{op}} + c_0(\mathbf{T}^{\mathrm{val}} - \mathbf{I}) + \theta(t)\mathbf{I} && (46)\\ \mathbf{i}^{\mathrm{b}} &= \mathbf{i}^{\mathrm{op}} + c_0\mathbf{s} + \mathbf{h}(t) && (47) \end{aligned}$$

$\mathbf{v}^+$ may 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 θ and $\mathbf{h}$, given in (43) and (44), depend on $\mathbf{v}^+$. The system of coupled first order differential equations is surprisingly simple and quite possibly retains the attraction of being implementable on analogue electrical networks.



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

Figure 5: Tabu search with steepest descent dynamics for the example knapsack problem. Parameters are $\alpha = 0.01, \beta = 2.0, \epsilon = 0.1$. Contours of E are shown in dotted lines.

The tabu search dynamics are intended for use over non-integral polytopes with linear objectives E^{op}. Indeed, if E^{op} is linear, it is straightforward to show that E_t^{op} is 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 by appropriate settings of the parameters α, β and ϵ (see Appendix A). For moderate β, the trajectory of $\mathbf{v}^+$ is initially dominated by the gradient of E^{op}, until $\mathbf{v}^+$ gets trapped in 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}. It is quite possible that the search might exhibit limit cycles, though these would typically be of a long time period and would include visits to many vertices. It is a matter of future research to develop continuous dynamics which search *every* vertex of a polytope, given an exponential amount of time.

5.2 Illustration and experiments

We implemented the tabu search system with steepest descent dynamics (6). 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 (3), and subsequently take a long time to return and effect significant changes in $\mathbf{v}$. Figure 5 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 3. 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 very 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 for larger values of α (see Appendix A).

To demonstrate the usefulness of tabu search with more challenging problems, we applied the technique to a database of 1000 randomly generated 10-item knapsack problems, each of which had been solved exactly by exhaustive search. The size of the problems was dictated by the ability to exhaustively search for the optimal solution in reasonable time, and not by any limitation of the tabu solution technique. 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

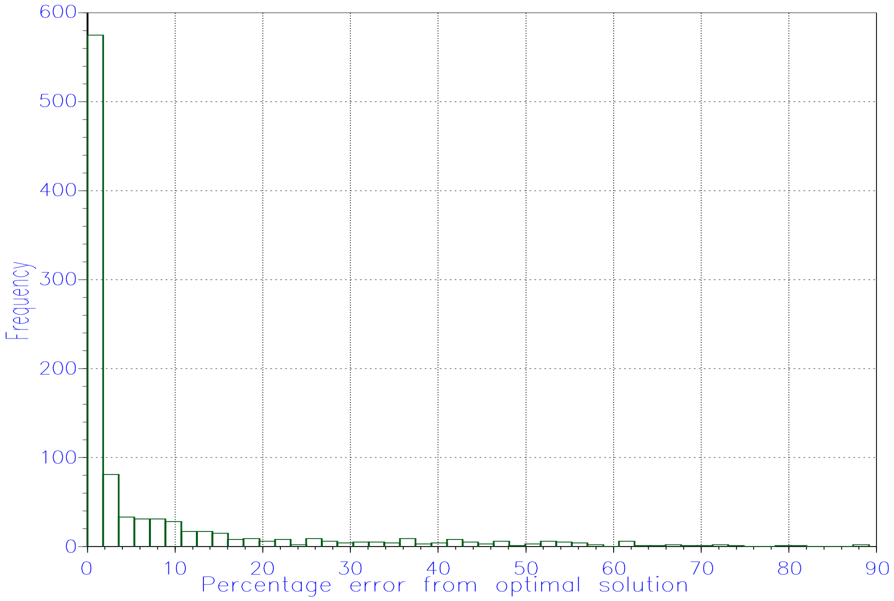


Figure 6: Suboptimality of the 972 valid solutions found by tabu search for the 1000 solved knapsack problems. Parameters are $\alpha = 0.01, \beta = 1.0, \epsilon = 0.1$.

of the items. For all 1000 problems, steepest descent dynamics were used on the time-varying Liapunov function (45), with the dynamics simulated using the algorithm described in Section 3.3 (see Appendix A for comments on the initialization of $\mathbf{v}^+$). 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 6 reflect the best solution found within the iteration limit. With valid solutions found over 97% of the time, the results are extremely encouraging, considering that conventional descent dynamics never find a valid solution, converging consistently to the optimal LP-relaxation solution. Moreover, mean solution errors around the 7% 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 (although the knapsack problem can be solved in pseudo-polynomial time using dynamic programming [13], this is generally not the case for $\mathcal{NP}$-complete problems). 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 A). Limit cycles were observed in the state vector trajectory, though these were of a long time period and typically included visits to many valid solution points.

6 Discussion and Conclusions

In Table 1 we see a classification of quadratic 0-1 programming problems, in which the problem complexity and suggested continuous solution technique are governed more by the nature of the polytope than by the order of the objective function. The popular deterministic annealing techniques are recommended for only one class of problem, that being quadratic optimization over integral polytopes. When the objective is linear or altogether absent (the latter being the case for pure constraint satisfaction problems), a simple descent technique will reliably find an optimal solution within an integral polytope. For non-integral polytopes the picture is very different. Even the simplest problems, in which we desire to find *any* 0-1 solution, are $\mathcal{NP}$-complete, and one-shot descent or annealing solution techniques are quite inappropriate. Tabu search dynamics offer one

Polytope	Integral		
Objective	Quadratic	Linear	None
Complexity	$\mathcal{NP}$-complete	$\mathcal{P}$-solvable	$\mathcal{P}$-solvable
Solution	Deterministic annealing	Simple descent	Simple descent[5]
Examples	TSP[6] [19, etc] GPP[7] [27, 28, 33]	Assigment [12, 30]	Crossbar switching [31]

Polytope	Non-integral		
Objective	Quadratic	Linear	None
Complexity	$\mathcal{NP}$-complete	$\mathcal{NP}$-complete	$\mathcal{NP}$-complete
Solution	Tabu search	Tabu search	Tabu search
Examples		TSP[6] [22, 29, etc] Knapsack [17]	'Teachers and classes' [15][8]

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

feasible continuous approach to tackling these problems, though no doubt other approaches will emerge with time; perhaps the most attractive prospect lies in somehow remapping these problems onto integral polytopes, as can be accomplished for the travelling salesman problem. Meanwhile, however, tabu search dynamics have achieved encouraging results with small knapsack problems, and certainly deserve further investigation, especially in relation to any possible analogue circuit implementation.

In this paper we have sought to reasses the field of 'neural' optimization from a refreshing viewpoint, and have demonstrated that, in the light of a rigorous problem mapping, the current techniques are unsuited to the solution of the vast majority of 0-1 programming problems. However, with appropriate modifications reflecting the polyhedral nature of the problem at hand, such techniques show great promise for the future, especially when the potential for fast, analogue implementations is realized.

---

[5] For pure constraint satisfaction problems over integral polytopes, simple descent on *any* linear or concave objective function will find a valid solution point.

[6] Travelling Salesman Problem.

[7] Graph Partioning Problem.

[8] This is in fact an example of resource-constrained multi-processor scheduling [13, p. 239].

A Tabu dynamics for stationary $\mathbf{v}^+$

In this appendix we consider $\mathbf{v}^+$ trapped at a vertex $\mathbf{v}_\text{o}^+$ of P^+ from time t_0. In these circumstances it is possible to analytically integrate the tabu dynamic equations, the results of which allow 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 (43) and (44) gives

$$\theta(t) = \frac{2\beta}{\alpha} - e^{-\alpha(t-t_0)}\left(\frac{2\beta}{\alpha} - \theta_0\right) \tag{48}$$

$$\mathbf{h}(t) = -\frac{2\beta}{\alpha}\mathbf{f}(\mathbf{v}_\text{o}^+) + e^{-\alpha(t-t_0)}\left(\frac{2\beta}{\alpha}\mathbf{f}(\mathbf{v}_\text{o}^+) + \mathbf{h}_0\right) \tag{49}$$

The resulting gradient of E at $\mathbf{v}_\text{o}^+$ is

$$-\nabla E = \mathbf{T}^\text{op}\mathbf{v}_\text{o}^+ + \mathbf{i}^\text{op} + \frac{2\beta\epsilon}{\alpha}(\mathbf{c} - \mathbf{v}_\text{o}^+) + e^{-\alpha(t-t_0)}\left(\mathbf{h}_0 + \theta_0\mathbf{v}_\text{o}^+ + \frac{2\beta\epsilon}{\alpha}(\mathbf{v}_\text{o}^+ - \mathbf{c})\right) \tag{50}$$

where we have substituted the expression for $\mathbf{f}(\mathbf{v}_\text{o}^+)$ from equation (39). We see that $-\nabla E$ has a limiting value as $t \rightarrow \infty$, specifically

$$-\nabla E_\infty = \mathbf{T}^\text{op}\mathbf{v}_\text{o}^+ + \mathbf{i}^\text{op} + \frac{2\beta\epsilon}{\alpha}(\mathbf{c} - \mathbf{v}_\text{o}^+) \tag{51}$$

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 $(\mathbf{c} - \mathbf{v}_\text{o}^+)$, since $\mathbf{c}$ is defined as being within P^+ and P^+ is convex. Assuming a linear objective E^op, so that $\mathbf{T}^\text{op} = \mathbf{0}$, we require

$$(\mathbf{c} - \mathbf{v}_\text{o}^+)^T\mathbf{i}^\text{op} + \frac{2\beta\epsilon}{\alpha}||\mathbf{c} - \mathbf{v}_\text{o}^+||^2 > 0 \tag{52}$$

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

$$\frac{\beta\epsilon}{\alpha} > \frac{||\mathbf{i}^\text{op}||}{2||\mathbf{c} - \mathbf{v}_\text{o}^+||} \tag{53}$$

Hence the relevant quantity governing the ability to escape from a vertex is the ratio $\beta\epsilon/\alpha$. Equation (53) indicates that it is desirable to locate the vector $\mathbf{c}$ as far from any vertex as possible. Some vector $\mathbf{c}$ within P^+ may be found in polynomial time using Khachiyan's method [29], or more practically one of the projection techniques [1, 11, 20, 24]; 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 (12)), 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

$$\frac{\beta\epsilon}{\alpha} > \frac{||\mathbf{i}^\text{op}||}{2} \tag{54}$$

Finally, a note about the speed of the overall search process. The system spends most time in situations examined in this appendix, that is with $\mathbf{v}^+$ trapped at a vertex and the tabu variables θ and $\mathbf{h}$ changing to free $\mathbf{v}^+$ from the vertex. Equation (50) 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.

References

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

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

[3] 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.

[4] 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), June 1990.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

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

[14] A. H. Gee and R. W. Prager. Alternative energy functions for optimizing neural networks. Technical Report CUED/F-INFENG/TR 95, Cambridge University Engineering Department, March 1992.

[15] L. Gislen, C. Peterson, and Söderberg B. 'Teachers and Classes' with neural networks. *International Journal of Neural Systems*, 1(2), 1989.

[16] F. Glover. Tabu search, a tutorial. Technical report, Center for Applied Artificial Intelligence, University of Colorado, November 1989. Revised February 1990.

[17] B. J. Hellstrom and L. V. Kanal. Knapsack packing networks. *IEEE Transactions on Neural Networks*, 3(2), March 1992.

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

[19] J. J. Hopfield and D. W. Tank. 'Neural' computation of decisions in optimization problems. *Biological Cybernetics*, 52:141–152, 1985.

[20] 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.

[21] B. Kamgar-Parsi and B. Kamgar-Parsi. On problem solving with Hopfield neural networks. *Biological Cybernetics*, 62:415–423, 1990.

[22] A. Langevin, F. Soumis, and J. Desrosiers. Classification of travelling salesman problem formulations. *Operations Research Letters*, 9(2):127–132, March 1990.

[23] M. Marcus and H. Mink. *A Survey of Matrix Theory and Matrix Inequalities*, pages 93–101. Allyn and Bacon, Boston, 1964.

[24] T. S. Motzkin and I. J. Schoenberg. The relaxation method for linear inequalities. *Canadian Journal of Mathematcs*, 6:393–404, 1954.

[25] 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*, San Diego, 1990.

[26] C. H. Papadimitriou and M. Yannakakis. Note on recognizing integer polyhedra. *Combinatorica*, 10(1):107–109, 1990.

[27] 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), 1988.

[28] C. Peterson and B. Söderberg. A new method for mapping optimization problems onto neural networks. *International Journal of Neural Systems*, 1(1), 1989.

[29] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, Chichester, 1986.

[30] G. A. Tagliarini and E. W. Page. A neural-network solution to the concentrator assigment problem. In D. Z. Anderson, editor, *Neural Information Processing Systems*, pages 775–782. American Institue of Physics, New York, 1988.

[31] 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.

[32] 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), May 1986.

[33] D. E. Van den Bout and T. K. Miller III. Graph partitioning using annealed neural networks. *IEEE Transactions on Neural Networks*, 1(2), June 1990.

[34] M. M. Van Hulle. A goal programming network for linear programming. *Biological Cybernetics*, 65:243–252, 1991.

[35] 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.

[36] V. Wilson and G. S. Pawley. On the stability of the TSP problem algorithm of Hopfield and Tank. *Biological Cybernetics*, 58:63–70, 1988.