Global Minimization on a Quantum Computer

Willard Miranker
Department of Computer Science
Yale University

TR1321

Global Minimization on a Quantum Computer

Willard Miranker
Department of Computer Science
Yale University

Abstract
Recursive methods for finding a local minimum are multiplexed with a diffusion process to produce a technique for global minimization. Application to neural nets and to quantum computing is made. The diffusion supplies the latter with a probability concentration feature for favoring the extraction of the global minimum from among the local minima.

Keywords - Annealing, averaging, diffusion, energy function, global minimization, Hopfield nets, quantum computing

1. INTRODUCTION

Suppose we apply a recursive method for finding a local minimum of a function $H(V)$, such as a method of steepest descent or the input-output dynamics of a Hopfield net in the case that $H(V)$ is the energy function of the net. We propose to replace $H(V)$ by a diffused version (in discretized form) of itself recursively, weaving the diffusion process into the minimization recursion (possibly alternating between both procedures). Diffusion will cause minima to flatten. In optimal circumstances the shallowest will disappear, and only the most robust minimum will remain as an attractor of the multiplexed recursion. This method is not universal, producing the global minimum only in some cases. It is of interest because of the role the diffusion plays in a quantum computer implementation. Minimization with diffusion could be contrasted to its opposite, annealing (Kirkpatrick, Gelatt, Vecchi, 1983).

We shall show how such a multiplexed recursion can be performed as a parallel quantum computation on the superposition of all possible minimization paths. Moreover the diffusion supplies a probability concentration process for the quantum computation, so that the culminating spin measurements for extracting the solution have increased likelihood of producing the global minimum.

In the general case we expect the multiplexed recursion to deliver, if not the smallest, then a smaller minimum than would have been found without the ancillary diffusion. The interleaved diffusion/minimization-dynamics will cause both the location and the value of the minimum found to be drifted away approximations to their values sought. A way to accommodate this drifting is to use the multiplexed process as a preprocessing phase, and to restart the minimization dynamics at the location delivered by the preprocessing. To avoid drifting, a final phase should be performed without the diffusion. Other strategies recommend themselves.

In Sections 2 we describe the diffusion process in a simple case. Sections 3 and 4 treat the application to neural nets. The general diffusion process is described in Sections 5 and 6. Finally in Section 7, the application to a parallel quantum computation is made.

2. DIFFUSION

We shall use a discrete approximation to the diffusion process. So let ΔV be a vector of increments in V. Then a step of diffusion replaces $H(V)$ by $H_{diff}(V)$, where

(2.1) $$H_{diff}(V) = \lambda(H(V+\Delta V) + H(V-\Delta V)) + (1-2\lambda)H(V).$$

Here λ is an arbitrary parameter. (λ may be thought of as equal to $D\Delta t/\|\Delta V\|^2$, the familiar ratio of increments arising in the finite difference approximation to the diffusion equation, where D is the diffusion constant.) For convenience we shall write ΔV as Δ.

Next define the shift operators $\mathbf{S}^{\pm}$ (with $\mathbf{S}^{+}= \mathbf{S}$) as

(2.2) $$\mathbf{S}^{\pm} H(V) = H(V \pm \Delta).$$

Let $\boldsymbol{M} = \frac{1}{2}[\,\mathbf{S} + \mathbf{S}^{-}]$ denote the averaging operator, and let $\boldsymbol{D}_{iff} = \boldsymbol{D}_{iff}(\lambda)$ denote the following discrete diffusion operator.

(2.4) $$\boldsymbol{D}_{iff} = \lambda\,[2\boldsymbol{M} + \frac{(1-2\lambda)}{\lambda} I],$$

Then

(2.5) $$H_{diff}(V) = \boldsymbol{D}_{iff} H(V).$$

Note that diffusion reduces to averaging for $\lambda = 1/2$ in which case we write H_{diff} as H_{av}.

3. HOPFIELD NETS

The dynamics of a Hopfield net with N neurons are given by (Haykin, 1999)

(3.1) $$\tau_i \frac{du_i}{dt} = -u_i + \sum_{j=1}^{N} w_{ij} g(u_j), \quad i = 1,\cdots,N.$$

Here u_i is the total weighted input to the i-th neuron whose output is denoted by V_i. g is the neuronal gain function, and the (N-vectors of) inputs and outputs are related by

(3.2)
$$V = g(u) \square \; V_i = g(u_i), \quad \forall i.$$

Under appropriate conditions on the weight matrix $W = \left(w_{ij}\right)$ (namely, $w_{ij} = w_{ji}$, $\forall i, j$) and on g (namely, $g' > 0$), the net's dynamics (3.1) drives the associated energy (Lyapunov) function

(3.3)
$$H(V) = -\frac{1}{2}\sum_{i,j} w_{ij} V_i V_j + \sum_i \int_0^{V_i} g^{-1}(\xi)d\xi$$

to a relative minimum.

3.1 The replacement energy $H_{av}(V)$ and the replacement gain function $h(u)$

Let the displacement vector be $\Delta = (\Delta_1, \ldots, \Delta_N)$. Then using (3.3) and specializing diffusion to the case of averaging (the general case is treated in Section 5), we define

(3.4)
$$\begin{aligned}
2H_{av}(V) &= 2\boldsymbol{M}H(V) \\
&= -\frac{1}{2}\sum_{ij} w_{ij}(V_i + \Delta_i)(V_j + \Delta_j) - \frac{1}{2}\sum_{ij} w_{ij}(V_i - \Delta_i)(V_j - \Delta_j) \\
&\qquad + \sum_j \left[\int_0^{V_j+\Delta_j} + \int_0^{V_j-\Delta_j}\right] g^{-1}(\xi)d\xi \\
&= -\sum_{ij} w_{ij}V_iV_j - \sum_{ij} w_{ij}\Delta_i\Delta_j \\
&\qquad + \sum_j \left[\int_{-\Delta_j}^{0} g^{-1}(\xi + \Delta_j)d\xi + \int_{-\Delta_j}^{0} g^{-1}(\xi - \Delta_j)d\xi\right] \\
&\qquad\qquad + \sum_j \left[\int_0^{V_j} g^{-1}(\xi + \Delta_j)d\xi + \int_0^{V_j} g^{-1}(\xi - \Delta_j)d\xi\right]. \\
&= 2\overline{H}(V) + 2J(\Delta).
\end{aligned}$$

Here

(3.5)
$$\overline{H}(V) = -\frac{1}{2}\sum_{ij} w_{ij}V_iV_j + \sum_j \int_0^{V_j} h_j^{-1}(\xi)d\xi,$$

where

(3.6)
$$h_j^{-1}(\xi) = \frac{1}{2}\left[g^{-1}(\xi + \Delta_j) + g^{-1}(\xi - \Delta_j)\right].$$

$J(\Delta)$ is a constant independent of V. In particular,

(3.7)
$$J(\Delta) = \frac{1}{2}\sum_{ij} w_{ij}\Delta_i\Delta_j + \frac{1}{2}\sum_j \left[\int_{-\Delta_j}^{0} g^{-1}(\xi + \Delta_j)d\xi + \int_{-\Delta_j}^{0} g^{-1}(\xi - \Delta_j)d\xi\right].$$

For purposes of minimization, this constant may be neglected in (3.4). Then comparing (3.3) and (3.4), we see that the Hopfield net corresponding to $\overline{H}$ is the same as the original net that corresponds to H except that the gain function g is replaced by the neuron dependent gain function h_j given in (3.6). These observations allow us to perform each diffusion step in a repetitive manner.

For convenience, we shall take all of the displacements Δ_j to be equal. Then we may drop the subscripts on the h_j and on the Δ_j. Note that if g is linear in its argument, averaging changes nothing since then h and g are identical.

4. SOLVING FOR THE REPLACEMENT GAIN FUNCTION

Since $u = g^{-1}(V)$, we have $\frac{d}{dV} g^{-1}(V) = \frac{1}{g'(u)}$, and $\frac{d^2}{dV^2} g^{-1}(V) = -\frac{g''(u)}{(g'(u))^3}$. Then employing Taylor's theorem, we find

(4.1) $$g^{-1}(V \pm \Delta) = g^{-1}(V) \pm \frac{\Delta}{g'(u)} - \frac{g''(u)}{(g'(u))^3}\frac{\Delta^2}{2} + \cdots.$$

Inserting this into (3.6) gives

(4.2) $$h^{-1}(V) = g^{-1}(V) - \frac{g''(u)}{(g'(u))^3}\frac{\Delta^2}{2} + \cdots.$$

This specifies an equation for determining the input-output relation

(4.3) $$V = h(u)$$

for the averaged Hopfield net. In particular, combining (4.2) and (4.3) gives

(4.4) $$u = g^{-1}(V) - \frac{g''(u)}{(g'(u))^3}\frac{\Delta^2}{2} + \cdots.$$

From this, we find the gain function h of the averaged net in terms of the gain function g of the original net. Namely,

(4.5) $$h(u) = g\left(u + \frac{g''(u)}{(g'(u))^3}\frac{\Delta^2}{2} + \cdots\right)$$

Since we are to repeat this net averaging process (equivalently, the net's gain function replacement process) recursively, we index the successive stages with n. Let us denote the gain functions for the succession of averaged neural nets as $g_n(u), n = 0,1...$ Then $g_0(u) = g(u)$, the original gain function and $g_1(u) = h(u)$, the gain function of the first averaged net. Let us also denote with $\Delta(n)$ the displacement Δ used at the n-th stage ($\Delta(0) = 0$), and with $V(n)$ the output of the neural net relevant to the n-th stage. Then from (4.5) (using Taylor's theorem), we may derive the following approximation to the successive input-output relations $V_n = g_n(u), n \geq 0$, all in terms of the original gain function $g(u)$. Namely,

(4.6) $$g_n(u) = g(u) + 2^{n-2}\frac{g''(u)}{(g'(u))^2}\Delta^2(n) + \cdots, \quad n \geq 0.$$

To avoid divergence of this expression as $n \to \infty$, the displacement $\Delta(n)$ must be made to tend to zero as n increases. A convenient choice for achieving this is $\Delta(n) = 2^{1-\frac{n}{2}}\delta$, where δ is a constant chosen suitably small, namely so that expressions like $\frac{g''(u)}{(g'(u))}\delta^2$ are small for all values of u that may arise in the course of running the computation. Notice that these choices of the $\Delta(n)$ may be interpreted as requiring that the average of the original energy function $H(V)$ (see (3.3)) be taken over ever smaller displacements in the independent variable V. This is a common feature of recursive minimization methods wherein the displacements toward the minimum sought (are made to) tend to zero as the location of that minimum is approached.

5. DIFFUSION

It is a simple matter to translate the development for averaging a Hopfield net to the general case of diffusion. Referring to Sections 2 and 3 (and writing $h(u)$ as is $h_{diff}(u)$), we see that

(5.1) $$H_{diff}(V) = 4\lambda(1-\lambda)\overline{H}_{diff}(V) + 2\lambda J(\Delta).$$

Here $J(\Delta)$ is given in (3.5), and

(5.2) $$\overline{H}_{diff}(V) = -\frac{1}{2}\sum_{ij} w_{ij}V_iV_j + \sum_j \int_0^{V_j} h_{diff}^{-1}(\xi)d\xi,$$

where

(5.3) $$h_{diff}^{-1}(\xi) = \frac{\lambda g^{-1}(\xi+\Delta) + (1-2\lambda)g^{-1}(\xi) + \lambda g^{-1}(\xi-\Delta)}{4\lambda(1-\lambda)}.$$

The analog of (4.2) is

(5.4) $$h_{diff}^{-1}(V) = \frac{1}{2(1-\lambda)}\left[g^{-1}(V) - \frac{g''(u)}{(g'(u))^3}\frac{\Delta^2}{2} + \cdots\right].$$

The input-output relation (4.3) becomes $V = h_{diff}(u)$, where

(5.5) $$h_{diff}(u) = g\left[4\lambda(1-\lambda)u + \frac{g''(u)}{2(1-\lambda)(g'(u))^3}\frac{\Delta^2}{2} + \cdots\right],$$

the latter being the analog of (4.5). Finally the analog of the input-output relation (4.6) is

(5.6) $$V(n) = 4\lambda(1-\lambda)g(u) + \frac{2^{n-2}g''(u)}{2(1-\lambda)(g'(u))^2}\Delta^2(n) + \cdots, \quad n > 0.$$

6. AVERAGING IN *N* DIMENSIONS

In this section we describe some formalism for specifying and computing averages in N dimensions ($V = (V_1, \cdots, V_N)$). To proceed, define the shift operators $\mathbf{S}_j$, $j = 1, \cdots N$ appropriate to each coordinate ($\mathbf{S}_j H(V) = H(V_1, \cdots, V_{j-1}, V_j + \Delta_j, V_{j+1}, \cdots, V_N)$). Next define a grid of lattice points in N dimensions, G^N, where

(6.1) $$G = \{0, \pm 1, \cdots, \pm N\},$$

and a corresponding grid $V(G)$ in V space. Namely,

(6.2) $$V(G) = \prod_{j=1}^{N} V_j, \; V_j = \{0, \pm\Delta_j, \cdots, \pm N\Delta_j\}, \quad j = 1, \cdots, N.$$

We say that the grid $V(G)$ is centered at the origin, $V = 0$. Then

(6.3) $$\mathbf{S}^{\mathrm{N}}(\mathrm{G}) = \prod_{j=1}^{\mathrm{N}} \sum_{i \in \mathrm{G}} \mathbf{S}_j^i$$

is a shift operator, which when applied to $H(V)$delivers the sum of its values over the grid $V(G)$.

Let A be an array of real numbers that sum to unity. A has the configuration of the grid G^N. That is, A is a $(2N+1)^N$ array in N dimensions. Let

(6.4) $$\boldsymbol{M}_A = A \otimes \mathbf{S}^{\mathrm{N}}(\mathrm{G}),$$

where $\otimes$ denotes the direct (element-wise) product. Then $\boldsymbol{M}_A$ is a generalized averaging operator in N dimensions. Note that by inserting zeros appropriately into the array A (maintaining the sum to unity property), the grid in question may be shaped into a desired stencil of points. Application of $\boldsymbol{M}_A$ to $H(V)$ delivers a generalized average of $H(V)$ over a grid of the form $V(G)$ that is centered at V (more precisely, over a stencil specified by A). Discretized diffusion operators in N dimensions correspond to particular choices of A, i.e., they are special cases of $\boldsymbol{M}_A$.

7. THE MULTIPLEXED RECURSION ON A QUANTUM COMPUTER

We use the parallelism of quantum computing to accelerate the global minimization iteration. Suppose that the computation is conducted on the binary integers $\mathbf{N} = \{0,1,\cdots,2^N - 1\}$. In particular the gain function g is replaced by a rounded version itself (that is, g followed by rounding into **N**, say, rounding to nearest). So g is an endomorphism of **N**, the latter being a group with addition mod 2^N. Similarly, suppose that the entries of the matrix W are also rounded into **N**. Suppose also in what follows that products of the form $W\psi_0$ (where the components of the vector ψ_0 are in **N**) denote the product also rounded into **N**. These arrangements assure that the net's dynamics are defined on the vertices of the binary N-cube. Correspondingly, the minima of the energy function are located at the cube vertices. (For examples, see Hertz, Krogh, Palmer, 1991.)

Any vertex v of the cube may be encoded as a quantum state $|v\rangle$, that state decoded into a binary integer by N spin measurements, as is well known. This is conveniently written as

(7.1) $$v \underset{\substack{\text{decoded via}\\ \text{measurement}}}{\overset{\text{encoded}}{\rightleftarrows}} |v\rangle, \quad \forall i,\ v \in \mathbf{N}.$$

Since $2^N = 1\underbrace{0\cdots0}_{N}$ (in binary), the kets in (7.1) are specified as tensor products of qubits:

(7.2) $$|0\rangle = \underbrace{|0\rangle\cdots|0\rangle}_{N} = \underbrace{|0\cdots0\rangle}_{N}, \quad |1\rangle = \underbrace{|0\cdots01\rangle}_{N}, \quad \cdots \quad , \quad |2^N - 1\rangle = \underbrace{|1\cdots1\rangle}_{N}.$$

Let $|\psi_0\rangle$ be the quantum state consisting of superposition of all of the vertices of the N-cube (i.e., of all of the kets in (7.2)).

(7.3) $$|\psi_0\rangle = \frac{1}{2^{N/2}} \sum_{x=0}^{2^N-1} |x\rangle$$
$$= \underbrace{(H \otimes \cdots \otimes H)}_{N} \underbrace{|0\cdots0\rangle}_{N}.$$

As indicated, this superposition of 2^N states can be generated by N applications of the unitary Hadamard transformation $H = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$, as is well known.

Note according to (7.1), ψ_0 is a sum of integers in **N**. Now consider the quantum circuit in Figure 7.1. The third output register contains the vector-valued ket that encodes the function gW applied to the superposition of states $|\psi_0\rangle$.

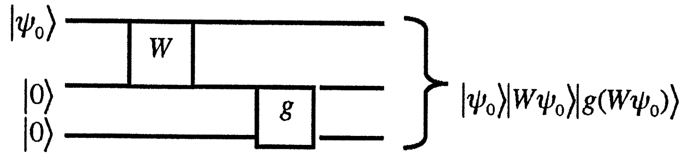

Figure 7.1 Quantum evaluation of multiplication of W followed by the evaluation of g.

For convenience, we shall omit the parentheses surrounding $W\psi_0$ in Figure 7.1, since confusion should not occur. Then iterating this quantum circuit k times, as indicated in Figure 7.2, delivers the quantum state

$$|\psi_k\rangle = |\psi_0\rangle \prod_{j=1}^{k} |W[gW]^{j-1}\psi_0\rangle |[gW]^j\psi_0\rangle. \tag{7.4}$$

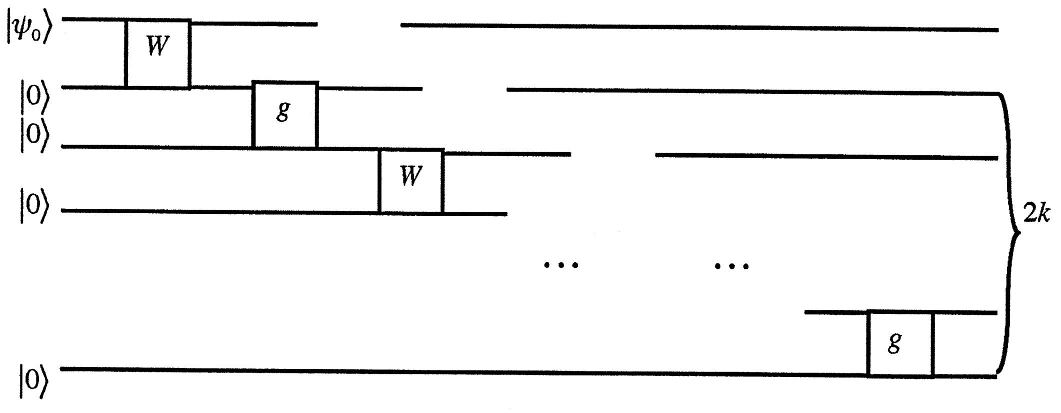

Figure 7.2 Quantum circuit for the k-fold evaluation of the circuit of Figure 1.

The final value of the last output register (say after m applications of the circuit) is the quantum state $|[gW]^m|\psi_0\rangle\rangle$. N spin measurements of this register deliver a vertex of the N-cube as a candidate for the global minimum.

7.1 Concentration of probability

To be effective, the quantum computation must concentrate the probability so that the likelihood of delivery of the global minimum is increased. To accomplish this, we employ the technique of Sections 3 and 4. Namely, we replace the gain function g used in each stage of the quantum circuits by g_j, where g_j is the round into N of the g_j given in (4.6). In particular, the expression $[gW]^j$ in (7.4) is replaced by g_jW. A recursive circuit corresponding to Figure 7.1 and 7.2, showing the use of g_j, is displayed in Figure 7.3.

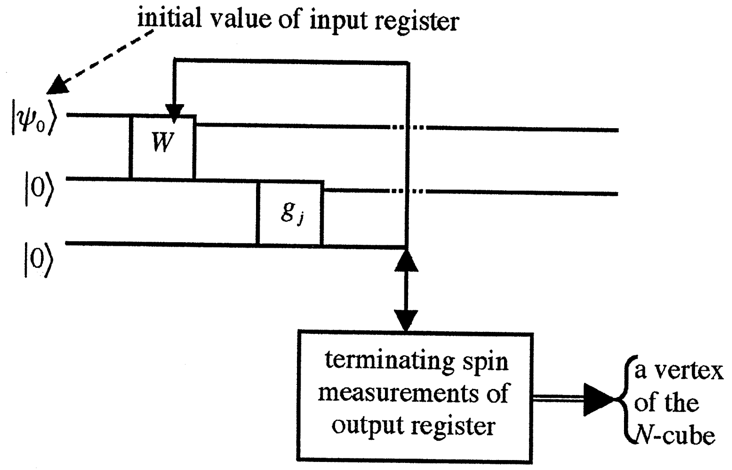
initial value of input register
$|\psi_0\rangle$
$|0\rangle$
$|0\rangle$
W
g_j
terminating spin measurements of output register
a vertex of the N-cube

Figure 7.3 A recursive quantum circuit for performing the multiplexed recursion followed by the N terminating spin measurements that yield a vertex of the unit N-cube.

7.2 Interpretation

The 2^N vertices of the N-cube are represented by the initial state $|\psi_0\rangle$ (N successive qubits per vertex). As the recursion commences, there emanates from each such vertex a path along N-cube edges. Paths grow, one edge per recursion. After the j-th recursion, the output register of the circuit in Figure 7.3 contains a state (denoted $|\psi_j\rangle$), the last $N \times 2^N$ qubits of which represent the concatenation of all the leading vertices of these paths. Since the energy function H decreases as each path is developed, N-cube vertices are steadily being discarded from $|\psi_j\rangle$. Of course, $|\psi_j\rangle$ always contains 2^N vertices, so there are an increasing number of repetitions (i.e., paths begin to overlay; equivalently, probabilities begin to concentrate). Each path will approach a relative minimum of the diffused energy function. Because of the diffusion, the larger relative minima tend to disappear as j increases, so increasingly more of the lead vertices of each of the 2^N paths will approach the location of the global minimum. Thus the probability of the terminating measurement being the location of the global minimum increases with j. (In optimal circumstances this probability will approach unity.) As noted in the penultimate paragraph of

Section 1, to accommodate drifting a second stage of processing is suggested. The discreteness of the domain of H may enable that second stage to be replaced by an examination of H values.

These arguments are not restricted to the minimizing dynamics (3.4) of the Hopfield net. They are valid for descending recursions (such as steepest descent...), generally.

7.3 The case[1] of unitary *W*

If the matrix W is unitary, the evaluation of the function W (i.e., multiplication by W) allows us to eliminate the second input register of W, thereby saving approximately one third of the circuitry.

In the case of basic Hopfield nets, W is symmetric with zero entries on the diagonal. The symmetry implies that the eigenvalues of W are real. The unitarity implies that these eigenvalues have modulus unity. Then all eigenvalues equal ± 1. Since $tr\, W = 0$, the positive and negative eigenvalue have the same multiplicity. Then W can be written in the form

$$(7.5) \qquad W = U \begin{bmatrix} I_m & 0 \\ 0 & -I_m \end{bmatrix} U^*,$$

where U is unitary and the order of W, $n = 2m$. Then writing $U = \begin{bmatrix} U_{11} & U_{12} \\ U_{21} & U_{22} \end{bmatrix}$ in terms of its $m \times m$ blocks, we have

$$(7.6) \qquad W = 2\begin{bmatrix} U_{11} \\ U_{21} \end{bmatrix}\begin{bmatrix} U_{11} \\ U_{21} \end{bmatrix}^* - I_n.$$

The columns of the $n \times m$ matrices here are orthonormal and the rows have norm 1/2.

As an example, take $\sqrt{2}U = \begin{bmatrix} X & X \\ Y & -Y \end{bmatrix}$ with X and Y unitary. Then $\begin{bmatrix} X \\ Y \end{bmatrix}\begin{bmatrix} X \\ Y \end{bmatrix}^* - I$ satisfies all conditions.

REFERENCES

Haykin, S., (1999), *Neural Networks, A Comp. Found.*, Prentice Hall, Upper Saddle River.
Hertz, J., Krogh, A., and Palmer, R., (1991), *Introduction to the Theory of Neural Computation* (Redwood City: Addison -Wesley).
Kirkpatrick, S., Gelatt, C., Vecchi, M., (1983), Science, **220**, 671.
Nielsen, M., Chuang, I., (2000), *Quantum Computation and Quantum Information,* Cambridge.

[1] The author is grateful to S. Eisenstat for the observations in this section.