JOURNAL OF COMPUTATIONAL PHYSICS 110, 320-326 (1994)

# Seeking Global Minima

WAN AHMAD TAJUDDIN WAN ABDULLAH

*Physics Department,* Imperial College, London SW7 2BZ, United Kingdom and Jabatan Fizik,† Universiti Malaya, 59100 Kuala Lumpur, Malaysia*

Received November 20, 1991; revised March 4, 1993

Ease in finding the configuration at the global energy minimum in a symmetric neural network is important for combinatorial optimization problems. We carry out a comprehensive survey of available strategies for seeking global minima by comparing their performances in the binary representation problem. We recall our previous comparison of steepest descent with analog dynamics, genetic hill-climbing, *simulated diffusion, simulated annealing, threshold accepting and* simulated tunneling. To this, we add comparisons to other strategies including taboo search and one with field-ordered updating. © 1994 Academic Press, Inc.

## 1. INTRODUCTION

Certain classes of cooperative systems like spin glasses and neural networks possess dynamics which can be conveniently described in terms of traversing energy landscapes. Conversely, certain computational problems can be solved by comparison to the dynamics of such systems. One important objective in such problems is the location of the energy global minimum in configuration space.

In particular, the dynamics of a neural network with symmetric connections result in the decrement of a configurational energy function [1]. Such a network can be used to solve problems in combinatorial optimization [2] where a *minimum energy configuration, corresponding to the combination* with minimum cost function, is sought. However, the energy landscape of the neural network may contain local minima which result in non-optimal solutions. For some problems, it may be adequate to obtain near-optimal solutions, but for certain problems, like the logic interpretation of sets of clauses [3], it is important to find the global minimum.

There are available several proposed algorithms to avoid local minima. In this paper we present a comprehensive survey of these algorithms by comparing their performances for a specific problem. We chose the binary representation problem, where the configuration of the network is interpreted as a binary number, and a number is stored by having it to correspond to the configuration at minimum energy, thus allowing the global minimum to be known. We also propose several other algorithms based on the conjecture that stronger connections dominate in the selection of the global minimum, and we compare their performances as well. As the performances of the algorithms may vary depending on the nature of the problem to be solved, and thus an algorithm inefficient for the selected problem may yet be useful in some other, we have preferred breadth rather than depth in the survey.

## 2. SYMMETRIC NEURAL NETWORKS AND COMBINATORIAL OPTIMIZATION

We consider networks of binary neurons: $V_i \in \{0, 1\}$, $i = 1, ..., N$, with connections $T_{ij}$ and thresholds $U_i$, giving the local field at $i$ as:

$$h_i = \sum_j T_{ij} V_j - U_i. \tag{1}$$

If the network is symmetric and zero-diagonal such that $T_{ij} = T_{ji}$ and $T_{ii} = 0$, we can write an energy function [1],

$$E = -\tfrac{1}{2} \sum_i \sum_j T_{ij} V_i V_j + \sum_i U_i V_i \tag{2}$$

which monotone decreases for a dynamics where the neural states become aligned to the local fields, i.e., $V_i \to \theta(h_i)$ where $\theta$ is the step function.

Combinatorial optimization involves looking for the combination of choices from a discrete set which yields a minimum value for some associated cost function. If [2] we map neurons to choices and equate $E$ to the cost function concerned, we can carry out combinatorial optimization on the neural network. The values of the connections are obtained from the equation for $E$ and the network with these connections are relaxed with the appropriate dynamics to (hopefully) arrive at the minimum of $E$ and thus of the cost function. The resulting configuration then

* e-mail w_abdullah@uk.ac.ic.ph.v1

† Permanent address: e-mail wat@fileserver.cc.um.my

0021-9991/94 $6.00

yields the optimal choices. However, non-optimal solutions may be obtained when local minima, as opposed to the global minimum, are reached.

## 3. BINARY REPRESENTATION PROBLEM

In this paper, we use the binary representation problem as a benchmark. This problem is useful because the global minimum is known. It is also relevant as some problems, like track reconstruction [4], incorporate it in the choice for parameters in the minimization, and binary representation is one of the possible representations for these parameters.

The network configuration is taken as the binary representation of a number $X$:

$$\sum_i 2^{(i-1)} V_i = X. \tag{3}$$

To memorize this number, we can define a cost function

$$E = \left( \sum_i 2^{(i-1)} V_i - X \right)^2 \tag{4}$$

which is minimized when the configuration corresponds to this number. This yields

$$\begin{aligned} T_{ij} &= -2(2^{i-1})(2^{j-1}), \qquad i \neq j \\ U_i &= 2^{2i-2} - 2^i X \end{aligned} \tag{5}$$

for the connections and thresholds.

Given these connections, we investigate how often the correct minmum is found from an arbitrary starting configuration, using different algorithms.

## 4. MINIMUM-SEEKING ALGORITHMS

In principle, one can always arrive at the global minimum, by exploring all possible configurations. This is, however, inefficient; there is a payoff between correctness and computational input. An algorithm for minimum-seeking is basically a method of attempting to arrive at the global minimum, from some initial configuration, by looking at only a small number of configurations, typically along some trajectory leading to the minimum.

In a previous paper [5], we have compared seven algorithms for minimum-seeking which are in the literature: the default steepest descent which locally alters configurations to decrease energy, analog [2, 6] or mean field [7] dynamics, genetic algorithm [8], simulated diffusion [9], simulated annealing [10], threshold accepting [11], and simulated tunneling [12]. These are described schematically below (we refer the reader to the respective original papers for more detailed descriptions):

ALGORITHM 1. STEEPEST DESCENT.

given init config
OPT: choose new config: a stochastic small perturb of old config
compute $\Delta E = E(\text{new config}) - E(\text{old config}) = h_i \Delta V_i$
IF $\Delta E < 0$ then *old config* := *new config*
IF for some time no change then STOP
GO TO OPT

This is the steepest descent algorithm in the context of neural networks. The neurons are updated sequentially: one is chosen stochastically and its state is modified to move down the projection $h_i$ of the energy gradient.

ALGORITHM 2. MEAN FIELD/ANALOG DYNAMICS.

given init config
choose init gain $1/T > 0$
OPT: choose new config: a stochastic small perturb of old config
compute new config thru $V_i := \text{sigmoid}(h_i, \text{gain})$
IF for some time no change then STOP
increase gain (lower $T$)
GO TO OPT

This algorithm is also a steepest descent algorithm, but analog neurons are used ($V_i$ can have real values). The neural dynamics need then to be modified—the step function is replaced by a sigmoid function which is parametrized by a gain.

ALGORITHM 3. GENETIC ALGORITHM.

given init config
generate small population around init config
OPT: mutate each member of population and add to population
cross each pair in population and add to population
keep only members w lowest $E$
IF for some time no change in config w lowest $E$ then STOP
GO TO OPT

This algorithm is a simulation of biological evolution. We deal with populations of configurations in which mutation (random changes in some neural states) and crossovers (swapping of corresponding sets of neural states between pairs of configurations) can occur, like the corresponding processes in chromosomes during biological reproduction.

ALGORITHM 4. SIMULATED DIFFUSION.

given init config
choose init temperature $T>0$
OPT: calculate local fields
gradient descent using values of local fields
do random walk w no. of steps $\propto \sqrt{T}$
IF for some time no change then STOP
lower $T$
GO TO OPT

This algorithm is related to diffusion in physical systems. The "diffusion" of a configuration is carried out via a random walk, which, for the neural network configuration space, corresponds to state changes in random neurons the number of which is given by the number of steps.

ALGORITHM 5. SIMULATED ANNEALING.

given init config
choose init temperature $T>0$
OPT: choose new config: a stochastic small perturb of old config
compute $\Delta E = E(\text{new config}) - E(\text{old config})$
with prob $\propto \exp(-\Delta E/T)$
old config := new config
IF for some time no change then STOP
lower $T$
GO TO OPT

This algorithm simulates the physical process of annealing, where a substance is heated up and cooled, and the atoms slowly form ordered structures. With this algorithm, the system is able to climb over energy barriers in search of lower valleys, with the Boltzmann probability factor.

ALGORITHM 6. THRESHOLD ACCEPTING.

given init config
choose init threshold $T>0$
OPT: choose new config: a stochastic small perturb of old config
compute $\Delta E = E(\text{new config}) - E(\text{old config})$
IF $\Delta E < T$ then old config := new config
IF long time no significant change in $E$ or too many iter then lower $T$
IF for some time no change then STOP
GO TO OPT

This algorithm also allows energy barriers to be climbed, but without any probabilities involved; as long as the energy difference is acceptable, a particular change in configuration is allowed.

ALGORITHM 7. SIMULATED TUNNELING.

given init config
OPT: generate some population around init config
LOOP: FOR every member of population
choose new config: a stochastic small perturb of old config
compute $\Delta E = E(\text{new config}) - E(\text{old config})$
IF $\Delta E < 0$ then old config := new config
IF any w lower $E$ than current min
GO TO OPT
IF for some time no change in config w lowest $E$ then STOP
GO TO LOOP

This algorithm attempts to copy tunneling in quantum mechanics. The energies of configurations around but not necessarily adjacent to the old configuration are calculated and if any happen to be lower than that of the latter, the system "tunnels" to that new configuration.

The results of the tests with these algorithms, reproduced in Table I, favours threshold accepting (see the next section for details concerning parameters, etc.), which gives a higher rate of success, but at the expense of a larger number of average updates per neuron in order to arrive at the stable configurations.

We have now extended the study to include further runs of threshold accepting with different parameters in order to estimate its optimum performance, and other algorithms. In particular, there is taboo search [13, 14], whose algorithm follows.

ALGORITHM 8. TABOO SEARCH.

given init config
OPT: generate some population around init config
choose best solution for this population which is not in taboo list
old config := best solution
update taboo list
IF for some time no change in config then STOP
GO TO OPT

Here possible "tracks" to the global minimum are tried in turn, the best ones first, and an inventory is kept of the ones which do not really improve the search.

We also include several other algorithms, most of which are related to field-ordered neural state determination. It is plaussible to assume that in the coupled dynamics of the self-organization of the neural states, connections with greater magnitudes play more dominant roles. Thus it may be conjectured that more global solutions are obtained when importance is given to these connections in the

minimum-seeking algorithms. With this in mind, we have devised the following algorithms:

ALGORITHM 9. SUBNETWORK GROWING.

subnetwork := { }
FOR all neuron pairs:
find largest $|T_{ij}|$ w both $i, j \notin$ subnetwork
subnetwork := subnetwork $\cup \{i, j\}$
solve subnetwork using gradient descent w solution of previous subnetwork as part of init config

ALGORITHM 10. NETWORK CRYSTALIZATION.

find largest $|U_i|$
fix $V_i$ depending on $U_i$
done_list := $\{i\}$
LOOP: find $j$, $j \notin$ done_list s.t. $|h_j|$ is largest for subnetwork done_list
fix $V_j$ depending on $h_j$
done_list := done_list $\cup \{j\}$
IF $\exists k$ s.t. $k \notin$ done_list then GO TO LOOP
else STOP

ALGORITHM 11. WEIGHT-WEIGHTED RELAXATION.

given init config
choose init exponent $T > 0$
OPT: calculate connection-weighted local fields: $h_i = \sum_j T_{ij} |T_{ij}|^T V_j - U_i$
gradient descent using values of local fields
IF for some time no change then STOP
lower $T$
GO TO OPT

ALGORITHM 12. FIELD-ORDERED UPDATING.

given init config
OPT: calculate local fields $h_i$
update neuron with largest $|h_i|$
IF for some time no change in whole config then STOP
GO TO OPT

ALGORITHM 13. FIELD-WEIGHTED UPDATING.

given init config
OPT: calculate local fields $h_i$
choose neuron $i$ w prob $\propto |h_i|$
update neuron $i$
IF for some time no change in whole config then STOP
GO TO OPT

ALGORITHM 14. ENERGY-WEIGHTED UPDATING.

given init config
OPT: calculate local fields $h_i$
choose neuron $i$ w prob $\propto |\Delta E|$ $(= |h_i \Delta V_i|)$
update neuron $i$
IF for some time no change in whole config then STOP
GO TO OPT

In the subnetwork growing algorithm, we solve for the network in stages—at each stage the pair of remaining neurons with strongest absolute connections are added to a subnetwork and the configuration of the subnetwork is found using gradient descent with the solution to the previous subnetwork as part of its initial configuration. In this way, we hope to capture the influences of the stronger neurons first. The network crystalization algorithm is similar, but the addition of neurons to the subnetwork is based on the field rather than the connection strengths, and the neural values are immediately determined rather than through gradient descent.

For weight-weighted relaxation, the influence of the stronger connection magnitudes is included by the (decreasing) factor which enhances the effects of stronger connections. In field-ordered updating, the stronger influences are given preferences in terms of the updating order of neurons. Neurons experiencing larger absolute fields are updated first, then the fields are recalculated for the others. A variant to this is field-weighted updating, where the updating order follows the same philosophy, but is implemented stochastically. The probability for a neuron to be updated depends on the strength of the field that it feels. If this probability depends on the associated energy change instead, then we have energy-weighted updating.

In addition, there is an algorithm inspired by quantum tunneling. In this algorithm, the tunneling mechanism is through energy-borrowing, allowed by an "uncertainty principle": $\Delta E \, \Delta t \leqslant \hbar$, where $t$ is timestep. The algorithm is sketched below.

ALGORITHM 15. QUANTUM TUNNELING.

given init config
OPT: choose new config: a stochastic small perturb of old config
compute $\Delta E = E$(new config) $- E$(old config)
IF $\Delta E < 0$ then old config := new config
ELSE allow change provided $\Delta E \, \Delta t \leqslant \hbar$
IF for some time no change then STOP
GO TO OPT

To enable the climb over local energy barriers, the following algorithm has also been designed and included in the study:

ALGORITHM 16. ENERGY REVERSING.

given init config
OPT: do gradient descent to a minimum
do gradient ascent (by changing sign of $E$ for $N_{rev}$ updates/neuron)
IF for some time no change then STOP
reduce $N_{rev}$
GO TO OPT

This algorithm allows for climbing of energy barriers by occasionally allowing sign change in the energy. The details of specific algorithms are described in the next section.

## 5. NUMERICAL SIMULATIONS

A comparison of the performances of the algorithms in the previous section as applied on the binary representation problem is carried out through numerical simulations. In particular, a 20-neuron network is used. For each algorithm, 100 random values of $X$ are employed, each using 1000 random initial configurations; i.e., 100,000 trials are carried out for each algorithm.

For algorithms with the parameter $T$ (reciprocal gain, temperature, threshold, or weight exponent), the following prescription is taken:

$$T \propto \alpha^{-t}, \tag{6}$$

where $t$ is the neural processing time (timestep, or the number of updates per neuron) and $\alpha$ is a constant. $N_{rev}$ in energy reversing also follows a similar prescription. For reciprocal gain, temperature, and threshold, $T(t=0) = 10^6$ and two values for $\alpha$, $\alpha = 100$ and $\alpha = 10$, were tried. The sigmoidal function for analog neurons is $1/(1+\exp(-\Delta E/T))$ and the number of steps for the random walk in simulated diffusion is taken to be $0.001N\sqrt{T}$, where $N$ is the number of neurons.

Simulated tunneling looks at a random population of five configurations nearest to and including the configuration under current study. For the genetic algorithm, two other random nearest configurations are added to the initial configuration to give the set of three configurations studied at a time. From these three configurations, another three are obtained by mutating each of them, and from the resulting six, 15 more are obtained by crossing (swapping random equivalent sections) each with each of the others. From the toal of 21, the best three (with lowest $E$) are selected for the next generation.

The mutation rate (probability of change in neural value per neuron) in the genetic algorithm is taken as 0.2. For analog dynamics, neurons are considered to be of similar value if they differ by less than 0.01.

For taboo search, the neighbourhood population generated is a Hamming distance of $1/N$ away; i.e., the configurations are different from the original by only one neuron state. The new configuration is chosen based on the lowest energy. All others in the neighborhood are placed on the taboo list. The previous configuration, if higher in energy compared to the new configuration, is also placed on the taboo list. If it is lower, then, in Version 1, no action is taken, or in Version 2, it is temporarily placed on the taboo list (for the immediate update loop only). The taboo list is taken to be a FIFO stack of definite length; we used a length of 500.

In field-ordered updating, there can be two variations. In the non-restrictive version, the number of relative updates of a neuron compared to the others may be different from one, while in the restrictive case a neuron is only allowed to be updated once per timestep. We look at the results of the numerical studies in the following section.

## 6. RESULTS AND DISCUSSIONS

In Table I we reproduce results from the previous study [5] (success is the retrieval of the global minimum configuration with 100% correctness; the time taken to reach stable end configurations are reflected by the average number of updates per neuron taken). Except for simulated tunneling and the genetic algorithm (see Table I), there is generally no apparent dependence on $X$. There is also no apparent dependence on the Hamming distance (which is proportional to the number of neural states different) of the initial configuration to the one at the global minimum, except again, for simulated tunneling (see Table I).

**TABLE I**

Previous Numerical Results

| | Percentage of success | Average updates/neuron | Notes |
|---|---|---|---|
| Steepest descent | 2.78 | 4.06 | |
| Analog $\alpha = 100$ | 2.21 | 4.74 | |
| $\alpha = 10$ | 2.77 | 5.73 | |
| Genetic | 1.42 | 2.81[a] | Very poor for large $X$ |
| Sim. diffusion $\alpha = 100$ | 2.50 | 4.48 | |
| $\alpha = 10$ | 2.30 | 6.22 | |
| Sim. annealing $\alpha = 100$ | 2.86 | 4.29 | |
| $\alpha = 10$ | 2.92 | 4.72 | |
| Threshold acc. $\alpha = 100$ | 3.94 | 6.40 | |
| $\alpha = 10$ | 4.11 | 8.18 | |
| Sim. tunneling | 0.79 | 17.6 | Slightly poorer for larger $X$; quick for small init Hamming dist |

[a] Average no. of generations.

From this initial study, threshold accepting seems to yield the best rate of success, while simulated tunneling the worst. The rest are not much better, if at all, when compared to steepest descent. The smaller value of $\alpha$ seems to generally give better performance, and that with a perhaps surprisingly small expense in the number of updates to reach the minimum. The number of updates per neuron to reach the minimum is roughly of the same order, except for simulated tunneling, which is larger. Of course, this number does not correspond directly to computational time since some algorithms require more computations per update than others.

The superiority of threshold accepting over, say, simulated annealing reflects the terrain of the energy landscape for the particular problem. It suggests that the probability of locating the global energy valley behind a certain barrier is not efficiently represented by the height of the barrier (or the Boltzmann factor related to the height). The success of threshold accepting seems to suggest a landscape of scattered low "energy hillocks" behind some of which are hidden paths to the global valley. The multidimensional, but value-restricted, space of the neural network configuration has to be kept in mind. Ease in traversing these energy hillocks results in a higher rate of location of the energy minimum. This is also supported by the results from the initial and the extended study that for this problem, a lower rate of decrease of the threshold yields a better performance in terms of success rate.

The results of the extended study involving more runs

**TABLE II**

Numerical Results—Extended Study of Threshold Accepting

| | Percentage of success | Average updates/neuron | Notes |
|---|---|---|---|
| Threshold acc. | | | |
| $T(t=0)=10^6$, $\alpha=3.0$ | 4.61 | 12.5 | |
| $\alpha=1.7$ | 4.94 | 19.7 | Upd/neuron up to 68 |
| $\alpha=1.3$ | 6.61 | 37.0 | Max upd/neuron over 100 |
| $\alpha=1.1$ | 3.94 | ~80 | Max upd/neuron over 100 |
| $T(t=0)=10^4$, $\alpha=1.7$ | 3.87 | 7.04 | Upd/neuron up to ~50 |
| $\alpha=1.7$ | 7.09 | 40.7 | Upd/neuron up to ~85 |
| $\alpha=1.7$ | 8.19 | 60.3 | Upd/neuron up to 103 |

**TABLE III**

Numerical Results of Other Algorithms

| | Percentage of success | Average updates/neuron | Notes |
|---|---|---|---|
| Taboo search (Vers. 1) | 2.64 | 10.1 | Odd no. updates for odd no. of neural stattes diff & vv |
| (Vers. 2) | 1.73 | 11.9 | Odd no. updates for odd no. of neural states diff & vv |
| Subnetwork growing | 2.63 | 13.27[a] | Upd/neuron up to 33 |
| Network crystalization | 0.38 | 1.45 | v poor for large $X$ |
| Weight-weighted $T(t=0)=1$, $\alpha=2$ | 0.98 | 8.13 | |
| $\alpha=4$ | 3.01 | 6.37 | |
| $\alpha=8$ | 3.22 | 6.55 | |
| $\alpha=16$ | 3.09 | 4.70 | |
| $T(t=0)=2$, $\alpha=8$ | 3.08 | 5.70 | |
| Field-ordered (no restriction) | 0.0 | 0.32 | |
| *(restricted)* | 4.97 | 1.16 | |
| Field-weighted | 0.0 | 1.79 | |
| Energy-weighted | 3.2 | 1.00 | |
| Quantum tunneling $\hbar=10^6$ | 1.87 | 3.99 | |
| $\hbar=10^4$ | 2.14 | 3.96 | |
| $\hbar=10^2$ | 2.62 | 4.02 | |
| Energy rev. $N_{rev}(t=0)=20/N$, $\alpha=2$ | 2.99 | 24.2 | |
| $N_{rev}(t=0)=40/N$, $\alpha=2$ | 3.72 | 28.4 | |
| $N_{rev}(t=0)=80/N$, $\alpha=2$ | 2.87 | 33.4 | |
| $N_{rev}(t=0)=40/N$, $\alpha=4$ | 3.16 | 16.6 | |

[a] Iterations needed to show stability for each subnetwork not included.

with threshold accepting and involving the other algorithms are given in Tables II and III, respectively. The results underline the superiority of threshold accepting in terms of the success rate, although this algorithm requires a huge number of updates per neuron for convergence. The rate of success, as well as the number of updates per neuron for convergence, seem to increase as the rate of decrement of the threshold is decreased. The run with the decrement rate nearest to the limit of one yields a success rate that is still less than 10%. The success rate can perhaps be improved substantially, but at the expense of huge increases in the number of updates, while it is at this limit.

Despite the expectation of better performance with algorithms giving importance to connections of bigger magnitudes, the results do not exhibit very substantial improvement from the basic steepest descent algorithm. In fact, some of the proposed algorithms did especially poorly. The best exception is field-ordered updating, which, along with a high rate of success, possesses a very low number of converging timesteps. With respect to this latter characteristic, it is our opinion that despite its slightly lower rate of success, it is a better algorithm compared to threshold accepting, at least for the binary representation problem applied here.

## 7. CONCLUSIONS

We have compared eight algorithms for minimum-seeking that are available in the literature, and we have proposed and compared eight more, as applied to the binary representation problem. For this specific problem at least, and with the parameters used, it seems that, although threshold accepting (with optimised parameters) yields the best rate of success, a field-ordered updating scheme offers the best solution with a minimal amount of neural processing steps. Even so, only a small percentage of the time is the true minimum found. Although one may expect otherwise, simulated tunneling is a poor failure—somehow the local population is not large enough to guarantee the position on a trajectory to the global minimum. Most other schemes do not outperform steepest descent very much, if at all.

Looking at the best performances, there is still a lot of need for the design of better algorithms seeking the minimum-energy configuration.

## REFERENCES

1. J. J. Hopfield, *Proc. Nat. Acad. Sci. U.S.A.* **79**, 2554 (1979).
2. J. J. Hopfield and D. W. Tank, *Biol. Cybern.* **52**, 141 (1985).
3. W. A. T. Wan Abdullah, *J. Intell. Systems* **7**, 513 (1992).
4. W. A. T. Wan Abdullah, "Studies in Reconstructing Circular Tracks Using Neural Networks," in *Proc., Int. Conf. Computing in High Energy Physics, Tsukuba, Japan, March 1991* (Universal Academy Press, Tokyo, 1991).
5. W. A. T. Wan Abdullah, "Looking for the Global Minimum in a Symmetric Neural Network," in *Int. Conf. Complex Systems: Fractals, Spin Glasses and Neural Networks, Trieste, Italy, July 1991.*
6. T. Fukai and M. Shiino, *Phys. Rev. A* **42**, 7459 (1990).
7. C. Peterson and J. R. Anderson, *Complex Systems* **2**, 59 (1988).
8. J. H. Hill, *Adaptation in Natural and Artificial Systems* (Univ. Michigan Press, Ann Arbor, 1975).
9. R. G. Hoptroff and T. J. Hall, *Electron. Lett.* **25**, 531 (1989).
10. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Science* **220**, 671 (1983).
11. G. Dueck and T. Scheuer, *J. Comput. Phys.* **90**, 161 (1990).
12. P. Ruján, *Z. Phys. B—Condensed Matter* **73**, 391 (1988).
13. F. Glover, *Comput. Oper. Res.* (1986).
14. P. Hansen, "The Steepest Ascent Mildest Descent Heuristic for Combinatorial Programming," in *Congress on Numerical Methods in Combinatorial Optimization, Capri, Italy, 1986.*