Simulated Annealing of Neural Networks: the "Cooling" Strategy Reconsidered*

Kenneth D. Boese and Andrew B. Kahng
UCLA Computer Science Dept., Los Angeles, CA 90024-1596

Abstract

The **simulated annealing** *(SA) algorithm [12] [5] has been widely used to address intractable global optimizations in many fields, including training of artificial neural networks. Implementations of annealing universally use a* **monotone decreasing**, *or "cooling", temperature schedule which is motivated by the algorithm's proof of optimality as well as analogies with statistical thermodynamics. In this paper, we challenge this motivation: the fact that cooling schedules are "optimal"* **in theory** *is not related to the* **practical** *performance of the algorithm. Our finding is based on a new "best-so-far" criterion for measuring the quality of annealing schedules. Motivated by studies of optimal schedules for small problems, we study highly nonstandard annealing schedules for training of feedforward perceptron networks on a real-world sensor classification benchmark. We find clear evidence that optimal schedules do not necessarily decrease monotonically to zero.*

1 Introduction

A typical application of neural networks is to emulate a given input-output (classification) function for which no closed-form representation is known. A given network is *trained* via a sequence of small changes to its real-valued connection weights to decrease, e.g., the mean squared error (MSE) in the classification of a finite set of training examples. This process is equivalent to global optimization of a multimodal error function defined over the space of connection weights, and is both theoretically intractable [25] and difficult in practice. Network training heuristics reflect the global optimization literature (see [8] for a survey); common approaches include gradient-based methods (back-propagation [21], or conjugate gradient with restarts [19]), stochastic hill-climbing, and other methods such as the variable-metric technique [6] and Kalman filter-based methods (e.g.,[2]).

In this paper, we study the *simulated annealing* (SA) method [1][5][9] [12]. In neural network learning, SA has been used in "Boltzmann learning" for Boltzmann machines [1] [9] and in the mean-field sense to optimize other connectionist architectures [3][18][26]. While simulated annealing is slower than other training algorithms, it outperforms greedy methods and remains popular when minimum error in the trained network is critical. Moreover, the long run times associated with SA may soon be ameliorated by special-purpose hardware [15] as well as improved strategic variants. Our main result is that the traditional "cooling" implementation of SA is not necessarily optimal and has been incorrectly motivated, both by the theoretical analysis of the algorithm and by the thermodynamic analogy implied by the term "annealing".

2 SA for Neural Network Training

Given a (finite) set S of feasible solutions and a real-valued cost function $f : S \rightarrow \Re$, global optimization may be formulated as the search for $s \in S$ such that $f(s) \leq f(s')$ $\forall s' \in S$. For neural networks, S is the space of connection weight vectors and f is the error function associated with the desired classification task.[1] SA requires the notion of a *neighborhood structure* over S, where the neighborhood $N(s_i)$ of the current solution $s_i \in S$ is the set of new solutions s' that can be generated from s_i. Typically, $N(s)$ consists of slight perturbations of s, e.g., a weight vector can be perturbed by adding a random vector in $[-\epsilon, \epsilon]^n$, where n is the number of connection weights.

SA is motivated by analogies between the solution space of an optimization instance and the microstates of a statistical thermodynamic ensemble. The algorithm (Figure 1) allows escape from local minima in the error surface by probabilistically accepting disimprovements, or "uphill moves". Over the M steps for which the SA algorithm is executed, a *temperature schedule* $T_1, T_2, \ldots, T_M$, $T_i \geq 0$ $\forall i$, guides the optimization.

SA Algorithm Template
0. $s_0 \leftarrow$ random solution in S
1. For $i = 0$ to $M - 1$
2. Choose $s' \leftarrow$ a random element from $N(s_i)$
3. **if** $f(s') \leq f(s_i)$
4. $s_{i+1} \leftarrow s'$
5. **else**
6. $s_{i+1} \leftarrow s'$ with probability $e^{-[f(s')-f(s_i)]/T_{i+1}}$
7. otherwise $s_{i+1} \leftarrow s_i$
8a. Return s_M
8b. Return s_i, $0 \leq i \leq M$, such that $f(s_i)$ is minimum.

Figure 1: The simulated annealing algorithm for a given bound of M time steps. Lines 8a and 8b show the difference between the "where-you-are" and "best-so-far" SA variants.

*Partial support for this work was provided by a GTE Graduate Fellowship and by NSF MIP-9110696, NSF Young Investigator Award MIP-9257982, ARO DAAK-70-92-K-0001, and ARO DAAL-03-92-G-0050.

[1] Note that S can be assumed finite even for neural network training, if we impose limits on the size and precision of each connection weight in the network.

Typical SA practice uses a large initial temperature and a final temperature of zero, with T_i decreasing monotonically either according to a fixed schedule or in order to maintain some measure of "thermodynamic equilibration" at each temperature value. SA enjoys certain theoretical attractions: using Markov chain arguments and basic aspects of Gibbs-Boltzmann statistics, one can show that for any finite S, SA will converge to a globally optimal solution given infinitely large M and a temperature schedule that converges to 0 sufficiently slowly [17], i.e.,

$$Pr(s_M \in R) \rightarrow 1 \quad as \quad M \rightarrow \infty \tag{1}$$

where $R \subset S$ denotes the set of all globally optimal solutions. In other words, SA is "optimal" in the limit of infinite time [13]. Sorkin [22] showed that certain classes of geometric cooling schedules are efficient on one-dimensional, deterministically fractal, error surfaces; this result is interesting because our recent work [10] [11] has shown that neural network error surfaces indeed resemble a class of high-dimensional statistical fractals.

3 Cooling Reconsidered

The central motivation of our work is practical: the ideal optimization algorithm should return a *good* solution within a *prescribed, finite* amount of time.[2] Through this motivation, we have discovered a fundamental inconsistency in the simulated annealing methodology as it stems from the theoretical, infinite-time Markov analysis.

Throughout its vast literature, the SA algorithm is universally implemented with *monotone decreasing* temperature schedules: hundreds of papers have been written on various "cooling" and "equilibration" approaches [13] [27]. The thermodynamic analogy suggests that monotone temperature schedules allow SA to explore "large features" of the cost surface at high T, then perform finer optimization at lower T. However, theoretical analysis has *incorrectly* led researchers to concentrate on the "cooling" paradigm.

In particular, the "optimality" of SA (Equation 1) has always been analyzed with respect to what we have termed [4] a "where-you-are" (WYA) implementation of the algorithm. According to the theoretical analysis, at time step M the SA algorithm simply returns the last solution seen, i.e., s_M, and it is this single solution that in the limit of $M \rightarrow \infty$ has probability 1 of being optimal. This theoretical model is at odds with common sense: *in practice*, no implementation will simply ignore all of the solutions $s_0, s_1, \ldots, s_{M-1}$. Certainly, we can remember the best solution seen so far, and return it after M steps rather than s_M. Indeed, as we have indicated by Line 8b of Figure 1, any SA implementation will in practice return this "best-so-far" (BSF) solution. With respect to traditional convergence proofs, this distinction is moot since "optimality" of a WYA algorithm trivially implies "optimality" of its BSF variant. However, the distinction between BSF and WYA can have tremendous implications for annealing strategies.[3]

An Illustrative Example

Consider a six-city instance of the traveling salesman problem (TSP) in the Manhattan plane with city coordinates $A = (0, 0)$, $B = (100, 0)$, $C = (100, 200)$, $D = (0, 200)$, $E = (40, 95)$ and $F = (40, 105)$. (The TSP is extensively studied in both the global optimization and neural network fields, e.g., [14].) In this instance, $|S| = 5!/2 = 60$, and there is one globally optimal solution (ABCDEF(A)), along with three other locally optimal solutions: (ABFECD(A)), (ABECDF(A)), and (ABFCDE(A)). We use the Lin 2-opt neighborhood operator that is usual in studies of the TSP [14]: a 2-opt move deletes two non-adjacent edges of the current solution s_i and then reconnects the two resulting paths into a new tour.

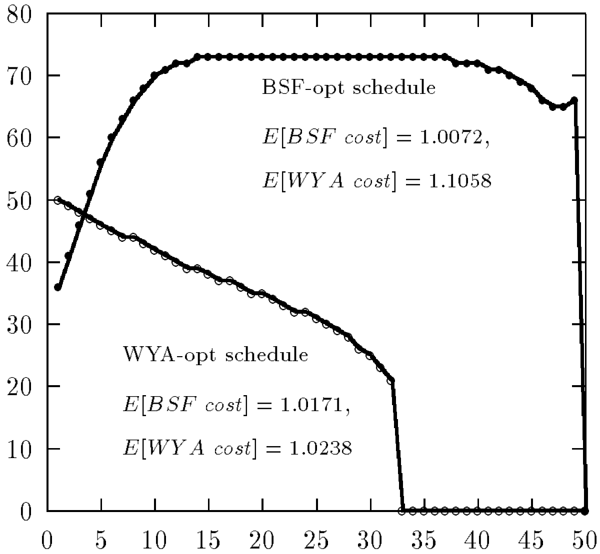

Figure 2: Locally optimal annealing schedules for the 6-city TSP example, determined by BSF and WYA measures. The vertical axis represents temperature and the horizontal axis represents the number of time steps. Expected solution costs are normalized to optimal cost. In BSF annealing, the last temperature T_M is irrelevant because an improving move s' will always be accepted no matter what the value of T_M. We break ties lexicographically, and so report $T_M = 0$ in the BSF-optimal schedule.

We have numerically solved for locally optimal annealing schedules of lengths up to $M = 80$, using both the BSF and WYA criteria.[4] The results in Figure 2

[2] Notice that the "infinite-time optimality" of SA is not a very useful or unique result: even exhaustive search and random search are "optimal" in the limit of infinite CPU resources.

[3] Given infinite time, for example, a schedule that converges to zero sufficiently slowly will be WYA-optimal, while any schedule that is bounded *away* from zero will be BSF-optimal.

[4] To solve for an optimal schedule, M distinct $|S| \times |S|$ matrices of transition probabilities are constructed over the solution space; the i^{th} matrix containing all one-step transition proba-

for 50 steps are dramatic. The optimal 50-step WYA schedule is monotone decreasing, as would be expected from the body of results in the current literature. However, the optimal 50-step BSF schedule is nearly *monotone increasing*, and even though it is worse in terms of the WYA objective (1.106 versus 1.024), it is clearly superior to the optimal WYA schedule (1.007 versus 1.017) when judged by the "real-life" BSF criterion.

We have obtained similar results [4] for small instances of several classic combinatorial problems, including graph bisection and graph placement in VLSI. Thus, we now examine the performance of non-traditional, *warming* temperature schedules for bounded-time training of a perceptron network architecture.

4 Annealing Schedules for a Real-World Perceptron Network

We study a feedforward perceptron network [21] with one hidden layer and connections only between adjacent layers. In addition to the connection weights, each hidden or output node has a "bias" weight. A standard logistic function is used as a squashing function on the activation of all non-input nodes.

The network is trained to distinguish buried nylon and wood targets from a highly-cluttered background; we use a set of 29 training examples corresponding to "windows" of amplitude-only microwave sensor returns.[5] Each training window consists of an array of 25x25 sensor readings containing either a whole target or no target at all. The complexity of the network architecture is reduced by defining a single output with value of 0.2 for "background" windows, 0.5 for wood target windows, and 0.8 for nylon target windows. We use the Karhunen-Loeve (KL) transform to reduce the 225 data points in each window to the 4 input values of the network; this data-reduction technique was used in [16] [24] for the same data.

Our previous research [11] has shown that a network with 4 hidden units can quickly (e.g., within 64K time steps) be trained by the SA algorithm to learn this classification task with nearly zero sum of squared error (SSE) over the 29 training cases. Thus, our experiments use an architecture with 4 input units, 4 hidden units, and 1 output unit.[6]

bilities for a given temperature T_i. Then, for a given probability distribution of the starting solution s_0, the appropriate matrix chain product can be computed to yield the WYA probability of each possible ending solution s_M (see [23]). A variation of this methodology computes BSF probabilities; see[4] for details. We minimize $E[f(s_M)]$ to determine WYA-optimal schedules, and $E[\min_{i=0}^{M} f(s_i)]$ to determine BSF-optimal schedules.

[5] The ground sensor data are obtained via the wave guide beyond cutoff (WGBCO), or separated aperture, microwave technique [20] and were provided by the U.S. Army Belvoir RD&E Center. The sensor apparatus and provenance of the data set are described in [11] [20] [24].

[6] Interestingly, our perceptron networks are almost two orders of magnitude smaller than those used by Widrow and Lehr at Stanford [16] and by Stricker and Azimi at Colorado State University [24] for exactly the same data sets, yet can be trained to achieve the same perfect classification performance.

To reduce the space of temperature schedules, we consider only linear schedules. For each schedule, we compute 50 separate annealing runs with initial weights chosen uniformly from $[-0.1, +0.1]$.A new solution s' is generated from s by perturbing each weight by a uniform random value in the range $[-0.05, +0.05]$.

Final Temp. Factor	Initial Temperature					
	.002	.005	.007	.009	.0125	.015
0.0	.292 (.079)	.224 (.081)	.210 (.070)	.231 (.089)	.190 (.070)	.206 (.072)
0.5	.274 (.076)	.218 (.075)	.188 (.079)	.184 (.069)	.211 (.077)	.202 (.063)
1.0	.254 (.085)	.206 (.069)	.206 (.074)	.198 (.086)	.218 (.072)	.230 (.076)
1.5	.247 (.087)	.209 (.081)	.194 (.082)	.214 (.078)	.225 (.071)	.252 (.059)
2.0	.238 (.087)	.203 (.068)	.197 (.058)	.234 (.084)	.247 (.069)	.259 (.061)

Table 1: Training of a neural network (4 input nodes, 4 hidden nodes, and 1 output node), showing BSF SSE for the 29 training cases, averaged over 50 runs of 16,384 steps each (standard deviations in parentheses). Final temperature of each linear schedule is initial temperature multiplied by "Final Temp. Factor".

Initial Temp.	Average	Standard Deviation	Initial Temp.	Average	Standard Deviation
0.0	.355	(.069)	0.15	.368	(.067)
.001	.324	(.071)	0.5	.500	(.167)
.0175	.211	(.078)	1.5	.711	(.231)
.02	.198	(.064)	15	1.333	(.339)
.03	.247	(.070)	150	1.108	(.324)
.05	.302	(.075)			

Table 2: BSF results of training runs of length 16,384 steps using linear temperature schedules that decrease to zero (averages of 50 runs).

Tables 1 and 2 contain average BSF results for a number of linear schedules of 16,384 steps. Solution qualities are measured in terms of the SSE over the 29 training cases. Also included in the tables are standard deviations of the solution quality for each schedule. Table 1 shows that for some initial temperatures, the best linear schedule is either constant or increasing! The most successful of these schedules starts at temperature .009 and cools to .0045, but not to the standard zero temperature. A striking result from this table is the relative similarity between most of the BSF values. In fact, the difference in the average quality of most of the schedules reported is much less than the standard deviation in the expected BSF quality. The table indicates that cooling schedules are not necessary for obtaining good results over finite length annealing runs. The most important factor affecting solution quality appears to be the range of temperatures used. Those schedules with temperatures below 0.002 or above 0.015 for a majority of steps are significantly worse than schedules in the middle of this range.

Table 2 explores a wide range of linear schedules ending at zero. It indicates again that the best schedules have temperatures in the range between 0.002 and 0.015 for most steps. Raising or lowering the initial temperature outside this range significantly worsens the final solution quality. Tables 3 and 4 contain results for the same set of initial and final temperatures, using a reduced schedule length of 4,096 steps. These two tables reinforce the qualitative results of Tables 1 and 2, and show that our main observations regarding the utility of non-decreasing schedules are not affected by schedule length.

Final Temp.	Initial Temperature					
Factor	.002	.005	.007	.009	.0125	.015
0.0	.352 (.056)	.305 (.065)	.300 (.065)	.280 (.083)	.253 (.074)	.295 (.073)
0.5	.340 (.058)	.286 (.071)	.284 (.060)	.279 (.073)	.279 (.067)	.289 (.066)
1.0	.322 (.067)	.275 (.068)	.285 (.072)	.275 (.078)	.291 (.074)	.304 (.070)
1.5	.316 (.041)	.296 (.073)	.262 (.065)	.291 (.079)	.310 (.062)	.331 (.056)
2.0	.322 (.062)	.281 (.070)	.264 (.066)	.276 (.073)	.327 (.057)	.322 (.074)

Table 3: BSF results for training runs of 4,096 steps using linear temperature schedules (averages of 50 runs).

Initial Temp.	Average	Standard Deviation	Initial Temp.	Average	Standard Deviation
0.0	.379	(.054)	.15	.396	(.080)
.001	.365	(.061)	.5	.515	(.137)
.0175	.284	(.082)	1.5	.728	(.234)
.02	.288	(.082)	15	1.113	(.330)
.03	.307	(.063)	150	1.126	(.316)
.05	.323	(.075)			

Table 4: BSF results of training runs of length 4,096 steps using linear temperature schedules that decrease to zero (averages of 50 runs).

5 Conclusion

Practical implementations of the simulated annealing algorithm for global optimization reflect a "best-so-far" (BSF) criterion. However, SA temperature schedules that "cool" monotonically to zero have been justified by theoretical analysis of a "where-you-are" (WYA) implementation. We have presented experimental evidence that challenges the appropriateness of cooling schedules for BSF annealing.

We have estimated numerically the *optimal* schedules for some very small combinatorial problems, e.g., the six-city TSP instance described above. These examples all have BSF-optimal schedules that are non-cooling. Our simulations to train a medium-sized perceptron neural network indicate that the best linear cooling schedules are not significantly better or worse than constant or "warming" schedules. In fact, it seems that the most important requirement for devising a good temperature schedule is to find the appropriate temperature range. Hence, methods for adaptively determining this optimal range during the actual SA run will be most effective in improving the performance of SA and other stochastic hill-climbing variants. Certainly, such adaptive procedures should not rule out the use of non-cooling strategies.

References

[1] E. H. L. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: a Stochastic Approach to Combinatorial Optimization and Neural Computing*, Wiley, 1989.

[2] E. Barnard, "Optimization for Training Neural Nets", *IEEE Trans. on Neural Networks* 3(2) (1992), pp. 232-240.

[3] G. L. Bilbro, W. E. Snyder, S. J. Garnier and J. W. Gault, "Mean Field Annealing: A Formalism for Constructing GNC-Like Algorithms", *IEEE Trans. on Neural Networks* 3(1) (1992), pp. 131-138.

[4] K. D. Boese and A. B. Kahng, "Best-So-Far vs. Where-You-Are: New Directions in Simulated Annealing for CAD", *technical report* UCLA CSD TR-920050, 1992.

[5] V. Cerny, "Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm", *J. Optimization Theory and Applications* 45(1) (1985), pp. 41-51.

[6] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[7] B. Hajek, "Cooling Schedules for Optimal Annealing", *Mathematics of Operations Research* 13 (1988), pp. 311-329.

[8] R. Hecht-Neilsen, *Neurocomputing*, Reading, MA: Addison-Wesley, 1990.

[9] G. E. Hinton, "Deterministic Boltzmann Learning Performs Steepest Descent in Weight Space", *technical report* CRG-TR-89-1, Univ. of Toronto, 1989.

[10] A. B. Kahng, "Exploiting Fractalness in Error Surfaces: New Methods for Neural Network Learning", *Proc. IEEE Intl. Symp. on Circuits and Systems*, San Diego, 1992, pp. 41-44.

[11] A. B. Kahng,"Random Structure of Error Surfaces: New Stochastic Learning Methods", invited paper, *Proc. SPIE Conf. on Neural Networks and Optimization*, Orlando, April 1992.

[12] S. Kirkpatrick, Jr. C. D. Gelatt, and M. Vecchi, "Optimization by Simulated Annealing", *Science* 220(4598) (1983), pp. 671-680.

[13] P. J. M. Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, Boston: D. Reidel, 1987.

[14] E. L. Lawler, J. K. Lenstra, A. Rinnooy-Kan and D. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Chichester: Wiley, 1985.

[15] B. W. Lee and B. J. Sheu, "Electronic Neural Circuits with Simulated Annealing", *Proc. IEEE/INNS Intl. Joint Conf. on Neural Networks*, Washington, 1989, pp. II-615 - II-618.

[16] M. A. Lehr and B. Widrow, "Adaptive Multisource Decision-Making: Detecting Land Mines with Neural Networks Using Separated Aperture Sensor Data Collected at Fort Belvoir", *report* BRDE-ISL/TR-1/1, April 1992, Stanford University Department of Electrical Engineering.

[17] M. Lundy and A. Mees, "Convergence of an Annealing Algorithm", *Math. Programming* 34 (1986), pp. 111-124.

[18] C. Peterson and J. R. Anderson, "A Mean Field Theory Learning Algorithm for Neural Networks", *Complex Systems* 1 (1987), pp. 995-1019.

[19] M. J. D. Powell, "Restart Procedures for the Conjugate Gradient Method", *Mathematical Programming* 12 (1977), pp. 241-254.

[20] L. S. Riggs and C. A. Amazeen, "Research Efforts with the Waveguide Beyond Cutoff or Separated Aperture Dielectric Anomaly Detection Scheme", *report*, U. S. Army Belvoir RD&E Center, December 1989.

[21] D. E. Rumelhart, J. L. McClelland et al., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* , Cambridge, MIT Press, 1986.

[22] G. Sorkin, "Efficient Simulated Annealing on Fractal Energy Landscapes", *Algorithmica* 6 (1991), pp. 367-418.

[23] P. Strenski and S. Kirkpatrick, "Analysis of Finite Length Annealing Schedules", *Algorithmica* 6 (1991), pp. 346-366.

[24] S. A. Stricker, M. R. Azimi-Sadjadi, and D. E. Poole, "Application of the Karhunen-Loeve Transform for Target Detection and Classification Using Neural Networks", draft, 1992.

[25] A. Torn and A. Zilinskas, *Global Optimization* , Lecture Notes in Computer Science 350, G. Goos and J. Hartmanis, eds., Springer-Verlag, 1987.

[26] D. E. Van den Bout and T. K. Miller, "Graph Partitioning Using Annealed Networks", *IEEE Trans. on Neural Networks* 1(2) (1990), pp. 192-203.

[27] D. F. Wong, H. W. Leong and C. L. Liu, *Simulated Annealing for VLSI Design*, Boston: Kluwer Academic, 1988.