Training minimal artificial neural network architectures for subsoil object detection

Kenneth D. Boese, Donald E. Franklin[†], and Andrew B. Kahng

UCLA Dept. of Computer Science,
Los Angeles, CA 90024-1596

[†]U. S. Army Night Vision and Electronic Sensors Directorate,
Ft. Belvoir, VA 22060

ABSTRACT

We cast the training of minimal artificial neural network architectures as a problem of global optimization, and study the simulated annealing (SA) global optimization heuristic under a "best-so-far" model. Our testbed consists of separated-aperture radar data for subsoil mine detection. In previous analyses, we have found that the traditional SA "cooling" paradigm can be suboptimal for small instances of combinatorial global optimizations. Here, we demonstrate that traditional cooling is also suboptimal for training minimal neural networks for mine detection. Related issues include (i) how to find minimal network architectures; (ii) considering tradeoffs between minimality and trainability; (iii) the question of whether multi-start/parallel implementations of SA can be superior to a single long SA run; and (iv) adaptive annealing strategies based on the best-so-far objective.

Keywords: neural networks, simulated annealing, subsoil detection, separated-aperture radar

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 of the connection weights, and is both theoretically intractable [32] and difficult in practice. Network training heuristics reflect the global optimization literature (see [15] for a survey); common approaches include gradient-based methods (back-propagation [31], or conjugate gradient with restarts [29]), stochastic hill-climbing, and other methods such as variable-metric [13] and Kalman filter-based [6] techniques.

In this paper, we study the *simulated annealing* (SA) heuristic [11][20]. For neural network learning, SA has been used in "Boltzmann learning" for Boltzmann machines [1] and in the mean-field sense to optimize other connectionist architectures [7][28]. 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. We demonstrate that the traditional "cooling" implementation of SA is not necessarily optimal, and has furthermore

been incorrectly motivated, both by the theoretical analysis of the algorithm and by the thermodynamic analogy implied by the term "annealing". Our testbed consists of minimal neural network architectures for a subsoil detection task. We find that changing network architectures can have a significant effect on training time, with *larger* networks training much more quickly. Our simulations also indicate that multi-start annealing can be the best strategy for efficient training of an object detection network. Based on our observations, future work is aimed at developing an adaptive SA variant using on an autoregressive (AR) model of annealing.

2 NETWORK TRAINING AS GLOBAL OPTIMIZATION

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] Iterative methods of global optimization require 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. The neighborhood structure and the cost function induce a *cost surface* over S; for neural network training, we call this the *error surface* since the cost function is typically the sum of squared errors (SSE) in classification.

2.1 Simulated Annealing for Network Training

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. $\Delta = f(s') - f(s_i)$
4. **if** $f(s') \leq f(s_i)$
5. $s_{i+1} \leftarrow s'$
6. **else**
7. $s_{i+1} \leftarrow s'$ with probability $e^{-[f(s')-f(s_i)]/T_{i+1}}$
8. otherwise $s_{i+1} \leftarrow s_i$
9a. Return s_M
9b. Return s_i, $0 \leq i \leq M$, such that $f(s_i)$ is minimum.

Figure 1: The simulated annealing (SA) algorithm for a given bound of M time steps. Lines 9a and 9b show the difference between the "where-you-are" and "best-so-far" SA variants.

Simulated annealing (described by Figure 1) is an iterative algorithm that 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. 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. 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 [27], i.e.,

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

where $R \subset S$ denotes the set of all globally optimal solutions. In other words, SA is "optimal" in the limit of infinite time [21].

[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.

2.2 Annealing: Theory Vs. Practice

The central motivation of our work is practical: the ideal optimization algorithm should return a *good* solution within a *prescribed*, *finite* amount of time. This motivation reveals a fundamental inconsistency in the usual implementation of simulated annealing as it stems from the theoretical, infinite-time analysis found in the literature. Specifically, the "optimality" of SA (Equation 1) has always been analyzed with respect to what we have called [9][10] a "where-you-are" (WYA) implementation of the algorithm, in which SA simply returns the last solution seen s_M (Line 9a in Figure 1). It is this WYA solution that in the limit of $M \rightarrow \infty$ has probability 1 of being optimal. In *practice*, however, SA can always store the best of solutions $s_0, s_1, \ldots, s_{M-1}$, and return this "best-so-far" (BSF) solution (Line 9b of Figure 1). With respect to traditional convergence proofs, this distinction is moot since "optimality" of the WYA solution trivially implies "optimality" of the BSF solution. However, the distinction can have powerful implications for finite-time annealing strategies.

2.3 BSF-Optimal Finite-Time Schedules

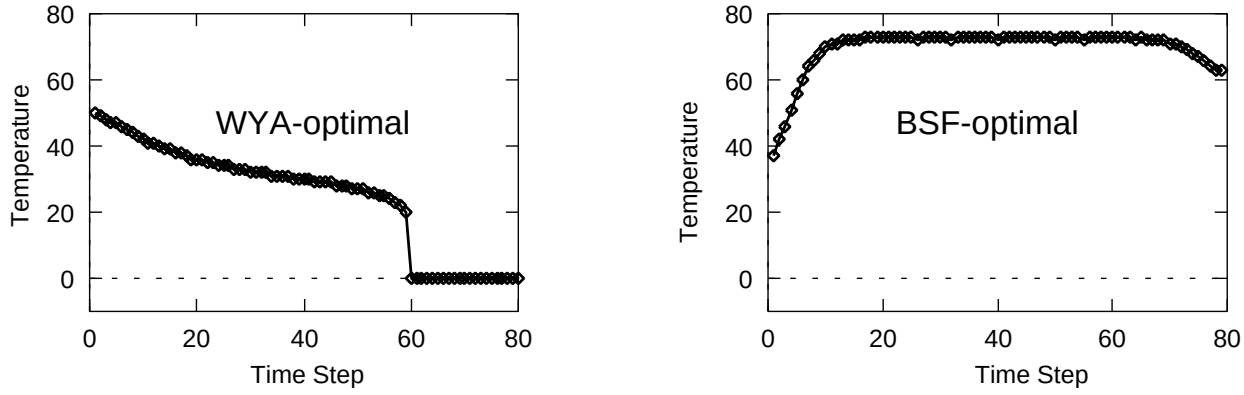

Figure 2: Locally optimal annealing schedules with 80 steps for a 6-city TSP instance using (left) the traditional WYA objective and (right) the practical BSF objective.

In [9,10] we computed BSF-optimal finite-time annealing schedules for three small global optimization instances. All instances showed dramatic differences between optimal schedules based on the BSF and WYA objectives. For example, Figure 2 displays the WYA- and BSF-optimal schedules with $M = 80$ steps for a 6-city instance of the traveling salesman problem (TSP). The optimal 80-step WYA schedule is monotone decreasing, as would be expected given the existing SA literature. However, the optimal 80-step BSF schedule is at first *monotone increasing*, then has a long flat region in the middle and a short decreasing region at the end. This BSF-optimal schedule is clearly superior to the optimal WYA schedule when judged by the "real-life" criterion of the expected BSF cost (0.2% above optimal versus 1.0% above optimal). Our results for schedules with different lengths show that a BSF-optimal schedule of approximately 42 time steps can achieve the same expected solution quality as a WYA-optimal schedule with 80 time steps, i.e., almost a 50% speed-up. We have obtained similar results for small instances of other classic combinatorial problems, including graph bisection and graph placement in VLSI [10][9].

3 NETWORKS FOR SUBSOIL OBJECT DETECTION

We study feedforward perceptron networks that are to be trained to discriminate buried nylon and wood targets in a highly cluttered background; we use a set of 29 training cases corresponding to "windows" of amplitude-only microwave sensor returns.[2] Each training window consists of an array of 15x15 sensor readings containing either a whole target or no target. We reduce the complexity of the network architectures by defining a single output with value of 0.2 for "background", 0.5 for a wood target, and 0.8 for a nylon target. We use the Karhunen-Loeve (KL) transform to reduce the 225 data points in each window to 4 input values; this data-reduction technique was used in [4][5][25] for the same data. The network architectures we study have one hidden layer and connections between adjacent layers only. 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. All hidden and output nodes have real-valued activations between 0 and 1, while the input nodes have unrestricted real values.

Our research indicates that a 4x4x1 network (i.e., 4 inputs, 4 hidden units, and 1 output, for a total of 25 connection weights) can be reliably trained by SA to achieve nearly zero cost (sum of squared error (SSE) over the 29 training cases) for this classification task. From a sample of 500 annealing runs with 64,000 steps each, the best run produced .00261 SSE, or a standard deviation from the desired output of only .0095. Thus, our experiments will concentrate on this 4x4x1 architecture. Our experiments in Section 3.3 indicate that it is likely to be the minimal architecture for this training task. In all of our annealing runs, we start with random initial weights with small absolute value, chosen uniformly from the interval $[-0.1, +0.1]$. A new solution s' is generated from s by perturbing each weight by a uniformly random value from the interval $[-0.05, +0.05]$.

3.1 Optimal Exponential Schedules

Because the space of connection weights is very large, it is practically impossible to compute optimal annealing schedules for our object detection neural networks. However, it is feasible to estimate an optimal schedule if we restrict ourselves to the class of *exponential* schedules which are commonly used in practice. An exponential schedule has the form

$$T_{i+1} = \alpha * T_i \tag{2}$$

for all i and a constant α, $\alpha > 0$. Exponential schedules were used by Kirkpatrick et al. in their original SA paper [20] and have been studied extensively by Johnson et al. [18,19]. For any fixed run length M, the space of exponential schedules is defined by exactly two parameters (e.g., T_1 and T_M). Searching over only two parameters to find a locally optimal exponential schedule is practical for our small object-detection networks.[3] This section extends our previous work [8] using *linear* annealing schedules on the same data.

Tables 1 and 2 report the BSF SSE of training runs with 16,384 steps and 32,768 steps, respectively. The values are averages over 500 different SA runs for each schedule.[4] The best schedule for 16,384 steps has $T_1 = 0.012$ and $T_M = 0.006$, a slowly cooling schedule that does not end near zero. For 32,768 steps the best schedule is constant (!), with $T_i = .009$ for all $i = 1, \ldots, M$. The most striking observation from the two tables is the relative similarity between the average BSF values, e.g., the differences in average BSF quality between most schedules are much less than the standard deviations in BSF quality. The tables also indicate that cooling schedules are not necessary for obtaining good results over finite length annealing runs, and that the most important factor affecting solution quality is the range of temperatures used. Schedules outside the range $[0.005, 0.012]$ for a majority of steps are significantly worse than schedules with temperatures exclusively within this range.

[2] The ground sensor data are obtained via the wave guide beyond cutoff (WGBCO), or separated aperture, microwave technique [30] and were provided by the U.S. Army Belvoir RD&E Center. The sensor apparatus and provenance of the data set are described in [4] [5] [17] [30]. The 29 training cases are printed in [17].

[3] Our intuition suggests that the expected BSF SSE is a convex function of T_1 and T_M, which would imply that any local minimum should also be a global minimum.

[4] Each run of 16,384 steps requires approximately ten seconds of CPU on an HP Apollo 735.

Average (StdDev) of SSE Over 500 Runs						
Final Temp. Factor	Initial Temperature T_1					
	.003	.005	.007	.009	.012	.015
0.1	.283 (.073)	.257 (.079)	.236 (.077)	.223 (.078)	.212 (.077)	.214 (.071)
0.5	.261 (.080)	.217 (.080)	.205 (.076)	.206 (.078)	.197 (.071)	.211 (.074)
1.0	.242 (.083)	.212 (.079)	.205 (.075)	.200 (.070)	.213 (.073)	.230 (.070)
1.5	.228 (.082)	.205 (.075)	.204 (.074)	.213 (.074)	.227 (.068)	.244 (.070)
2.0	.221 (.083)	.207 (.077)	.205 (.074)	.216 (.076)	.235 (.067)	.253 (.063)

Table 1: Comparison of 16,384 step exponential schedules for training a 4x4x1 subsoil object-detection network. Values shown are the average BSF SSE for the 29 training cases (standard deviations in parentheses). The final temperature of each exponential schedule is its initial temperature multiplied by "Final Temp. Factor". Sample size = 500. The standard error of each mean is approximately $.075/\sqrt{500} \approx .0035$. Thus, the best schedule in the table ($T_1 = 0.012$ and $T_M = 0.006$) is significantly better at the 95% confidence level than any schedule with average SSE .207 or greater (i.e., approximately three standard errors greater than .197).

Average (StdDev) of SSE Over 500 Runs						
Final Temp. Factor	Initial Temperature T_1					
	.003	.005	.007	.009	.012	.015
0.1	.251 (.082)	.218 (.085)	.200 (.083)	.195 (.080)	.180 (.072)	.187 (.071)
0.5	.218 (.088)	.187 (.081)	.183 (.081)	.178 (.073)	.179 (.070)	.190 (.070)
1.0	.205 (.088)	.178 (.077)	.179 (.075)	.173 (.069)	.188 (.065)	.204 (.062)
1.5	.196 (.087)	.181 (.080)	.176 (.069)	.183 (.072)	.198 (.065)	.222 (.066)
2.0	.186 (.082)	.179 (.075)	.181 (.070)	.190 (.069)	.207 (.070)	.235 (.066)

Table 2: Comparison of 32,768 step exponential schedules for training the same 4x4x1 network as in Table 1. Again, sample size = 500.

In an attempt to estimate the optimal exponential schedule more precisely, we may estimate a second-order model of the "response surface" induced by the expected BSF qualities of exponential schedules. Let t_1 and t_2 be the two parameters of a schedule. A second order model of expected SSE is an equation of the form

$$SSE = c + b_1 t_1 + b_2 t_2 + a_1 t_1 t_1 + a_2 t_2 t_2 + a_3 t_1 t_2 + \epsilon \tag{3}$$

where ϵ is zero-mean random noise. We estimate the parameters c, b_1, b_2, a_1, a_2, and a_3 in Equation 3 by a least-squares regression; the parameters that minimize Equation 3 give an estimate of the optimal exponential schedule. Based on intuition and experience, we use schedule parameters $t_1 = \log(T_1)$ and $t_2 = \log(\sqrt{T_1 * T_M}) = \log(T_{M/2})$.[5] We use data points from a square lattice of values for t_1 and t_2.

[5] We use log values to capture our belief that temperatures scale exponentially, e.g., that $T = 1.0$ is the same distance from $T = 0.5$ as from $T = 2.0$. We use $T_{M/2}$ in order to maximize the estimates of a_1 and a_2. If we use T_M or T_M/T_1 instead, the correct value of a_2 appears to be small, and our estimate is sometimes negative which results in Equation 3 having a saddle point rather than a minimum point.

Number of Steps	range of T_1	range of $T_{M/2}$	# data points	Parameter Estimates (T-Statistics)						Est. Opt. Schedules	
				c	b_1	b_2	a_1	a_2	a_3	T_1	T_M
16,384	.003 - .02	.003 - .01	1,024 (32x32)	1.302	.155 (1.5)	.309 (1.4)	.0179 (2.1)	.0333 (1.6)	-.0025 (-0.2)	.00943	.00710
16,384	.004 - .015	.004 - .009	10,000 (100x100)	1.321	.148 (2.1)	.311 (1.9)	.0183 (3.2)	.0335 (2.2)	-.0048 (-0.6)	.00922	.00518
32,768	.003 - .02	.003 - .01	2,500 (50x50)	1.382	.242 (3.6)	.256 (1.7)	.0189 (3.4)	.0208 (1.5)	.0116 (1.5)	.00721	.00963
32,768	.004 - .014	.004 - .014	5,041 (71x71)	1.334	.128 (1.4)	.339 (3.8)	.0080 (0.9)	.0286 (3.4)	.0105 (1.4)	.00904	.00444

Table 3: Results of regressions to fit second-order models to simulations for exponential schedules. Parameter estimates are given along with their t-statistics. (T-statistics with absolute value > 1.96 are significant at the 95% confidence level.)

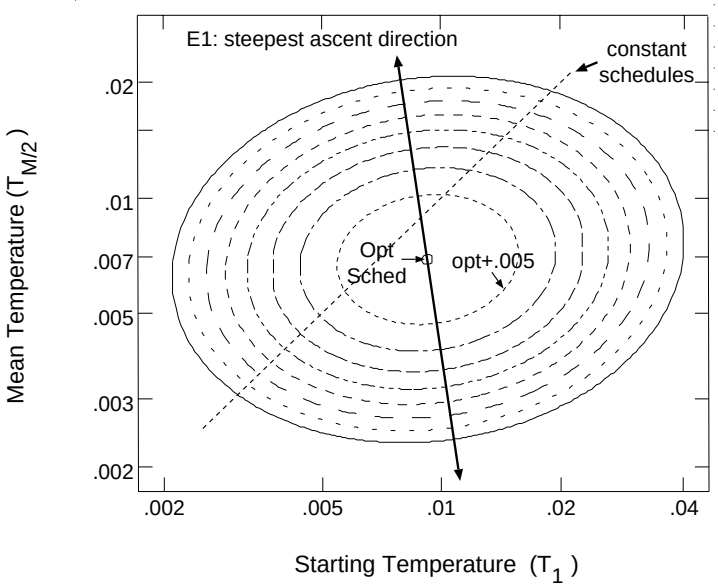

Figure 3: Equal-cost curves for exponential schedules with 16,384 steps based on a second-order model of BSF SSE from simulations of 10,000 different exponential schedules.

Results from four regression experiments are shown in Table 3. For 16,384-step schedules, we first simulated 1,024 different schedules for t_1 between log(.003) and log(.02) and t_2 between log(.003) and log(.01). The estimated optimal schedule from this experiment has $T_1 = .00943$, $T_{M/2} = .00818$, and $T_M = .00710$. To verify this estimated schedule, we reran the experiment with 10,000 data points and a smaller range of schedules (t_1 from log(.003) to log(.02) and t_2 from log(.003) to log(.01)). The new estimated best schedule has $T_1 = .00922$ and $T_M = .00518$. Figure 3 displays lines of equal SSE for the model estimated by this second regression. Notice that there is a relatively large area with expected SSE within 0.005 of optimal. Moreover, a relatively large range of schedules with constant temperature can be found near the optimal schedule. Note that the most important factor in the quality of the exponential schedules is captured by the direction of steepest descent (corresponding to the eigenvector of the largest eigenvalue of the Hessian matrix). This steepest descent direction is nearly vertical (in fact, only about 9 degrees from vertical), indicating that the most important parameter of schedule quality is the geometric mean temperature, $T_{M/2}$. The third and fourth models in Table 3 are for schedules with 32,768 steps; the third model estimates the optimal schedule to be a *warming* schedule with $T_1 = .00721$ and $T_M = .00963$, while the fourth model estimates the optimal schedule to have $T_1 = .00904$ and $T_M = .00444$.

We observe that the second-order models for 16,384 and 32,768 steps give estimates of optimal schedules that are similar to those derived from Tables 1 and 2. The models support the conclusions that (i) some constant-

temperature schedules are very nearly optimal; (ii) the most important factor is finding the correct range of temperatures (more specifically, the mean temperature $T_{M/2}$ in an exponential schedule).

3.2 BSF-Optimal Vs. Traditional Schedules

Our previous comparisons of BSF-optimal and WYA-optimal schedules in [9,10] (see Section 2.3) were confined to very small instances, and consequently, short annealing runs. One assertion that has been made in response to these studies is that for larger instances and longer runs, the qualities of the BSF and WYA solutions are nearly identical. If this were the case optimizing the WYA quality would still be the correct strategy for problems of practical size. However, we believe that practitioners' observations of similarity between BSF and WYA solutions occur because traditional cooling schedules are used. For *non-traditional* schedules, we believe that the difference between the BSF and WYA objectives will remain significant for large instances.

Figure 4 plots the average BSF and WYA costs at each step in a "traditional" annealing schedule and in the non-traditional BSF-optimal schedule for $M = 32,000$ steps. The BSF-optimal schedule is derived from our experimental results in Table 2, which estimate that a constant schedule with $T_i = .009$ is nearly optimal within the class of exponential schedules. We have designed a "traditional" exponential schedule based on the traditional wisdom (see e.g., [21]) that T_1 should be very high, so that initially between 90 and 95 percent of all candidate moves s' are accepted. The value of T_M should be low, so that less than two percent of moves are accepted at the end of the run.[6] The "traditional" schedule used for Figure 4 has $T_1 = 0.3$ with approximately a 90 percent acceptance rate, and $T_M = .001$, which at random starting solutions allows approximately one percent of the disimproving moves to be accepted. (This assumes that the distribution of $|f(s') - f(s_i)|$ is approximately the same at the beginning and at the end of the run.)

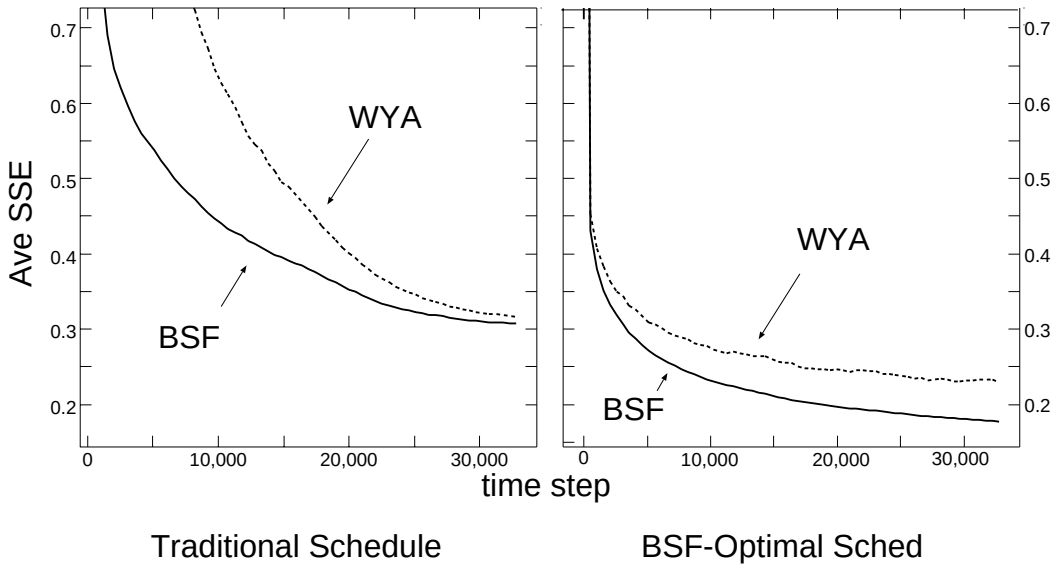

Figure 4: BSF and WYA curves for two different exponential schedules on the 4x4x1 object detection network. (a) A "traditional" schedule with $T_1 = 0.3$ and $T_M = 0.001$; (b) the optimal exponential schedule from Table 2, with constant temperature $T_i = 0.009$. BSF and WYA SSE are averaged over 250 runs and plotted every 128 time steps.

[6] Aarts and van Laarhoven [2,21] propose a more complicated stopping criterion. We adopt the two percent criterion from Johnson et al. [18,19] because of its greater simplicity.

The results in Figure 4 show a dramatic difference between the qualities of a traditional schedule and our BSF-optimal schedule. It also supports our hypothesis that the BSF and WYA costs converge at the end of the run only when traditional schedules are used. For the BSF-optimal schedule, the average WYA and BSF qualities have diverged because of the higher temperatures at the end of the run. These higher temperatures result in worse average WYA solution qualities, but allow greater mobility within the solution space, and consequently better BSF solutions.

3.3 Minimal Training-Time Architectures

A number of different articles have discussed the approach of constructing networks architectures simultaneously with their training (e.g., recently [12,33]). These works have observed that smaller networks require greater training time but usually generalize better to non-training data because they are less likely to overfit the training data. Motivated by these studies, we have set up the following experiment to test which architecture is minimal, and how much faster larger networks can be trained. We assume that the time required by each annealing step is approximately proportional to the number of weights in the network[7] and allocate the equivalent of 32,000 training steps for a 4x4x1 network to the different architectures. Each architecture is trained using a constant-temperature annealing schedule with $T_i = 0.007$; based on our results in Section 3.1 and experiments on other architectures, this appears to be close to optimal within the class of exponential schedules. Table 4 shows our experimental results using 400 training runs on network architectures ranging in size from 2x2x1 to 8x8x1.

Our results confirm the traditional wisdom that larger architectures are easier to train, although there is a limit beyond which very large architectures will not train efficiently: the average SSE of an 8x8x1 architecture is greater than an 8x6x1 architecture.[8] Our experiments also indicate that, in terms of trainability to zero SSE, the 4x4x1 architecture is either a minimal or nearly minimal architecture: no strictly smaller architecture trained to less than 0.01 SSE in any of its 400 training runs. Moreover, there appears to be a significant jump in the minimum SSE at the next smaller architectures: from 0.010 to 0.068 for the 4x3x1 architecture and to 0.058 for the 3x4x1 architecture. Nevertheless, our results are not conclusive; it is possible that 4x3x1 and 3x4x1 architectures can be trained to nearly zero SSE.

Average and Minimum BSF SSE of 400 runs										
	Num Hidden Nodes									
Num	2		3		4		6		8	
Inputs	ave	min	ave	min	ave	min	ave	min	ave	min
2	.45	.41	.36	.21	.31	.20	.24	.089	.20	.084
3	.41	.26	.30	.13	.192	.058	.141	.024	.116	.0054
4	.39	.21	.28	.068	.170	.0099	.084	.0003	.049	.0016
6	.36	.15	.163	.0001	.091	.0004	.034	.0008	.035	.0028
8	.26	.068	.085	.0000	.045	.0001	.018	.0004	.024	.0029

Table 4: Results of training runs on different architectures using single-temperature annealing runs. The number of steps in each run is normalized to the number of weights in the network (32,000 steps for the 4x4x1 architecture). Sample size = 400 runs for each architecture.

[7] If i is the number of inputs and h is the number of hidden nodes, then the number of weights equals $(i+2)*h+1$. (Recall that we include threshold weights.)

[8] Kahng in [17] ran comparisons on even larger architectures that confirm that very large architectures do train more slowly.

4 CONTINUING RESEARCH

4.1 Ideas I: Multi-Start Annealing

A number of works on simulated annealing have asked whether a single long annealing run is preferred to a number of shorter annealing runs or greedy descents (i.e., with $T_i = 0$ throughout each run). For instance, Ferreira and Zerovnik [14] showed that as M approaches infinity, an equivalent number of greedy descents will be preferred to a single annealing run. Lasserre et al. [24] showed that for several small problem instances, SA is frequently outperformed by multi-start greed, random search, and "steepest ascent descent". Our preliminary experiments with multi-start greed for network training had very little success. However, the large standard deviations reported in Tables 1 and 2 suggest that multiple runs of simulated annealing may be very successful as an alternative to one long annealing run. Multi-start annealing is also appealing because of its trivial parallelizability: if k processors are available, then k different runs can be performed on the different processors with no loss of efficiency.

Thus, we have set out to test the efficacy of multi-start annealing on the nearly-minimal 4x4x1 network architecture for our subsoil mine detection training cases. Given a CPU budget of M steps, what is the best way to divide the M steps among n different runs of length M/n steps? We conjecture that as M becomes larger and larger, the optimal run length may remain nearly constant. Thus, for any particular instance there may in effect be an optimal annealing run length that should be applied repeatedly until the CPU resource is exhausted. Alternatively, for any given instance there may be some constant number of runs n that is optimal or nearly optimal for any CPU budget M.

Ave. SSE from best of n runs				
Run Lengths (M/n)	Step Budget M			
	60,000	120,000	180,000	240,000
10,000	.136	.113	.102	.094
15,000	.132	.107	.094	.086
20,000	.132	.106	.094	.085
24,000		.104		.080
30,000	.135	.105	.091	.081
40,000		.111		.082
60,000	.157	.117	.099	.089

Table 5: Average BSF SSE of different allocations of 60,000 to 240,000 steps to multiple annealing runs for training our 4x4x1 network. Average SSE values are obtained as follows: (i) 250 runs of each run length M/n are simulated, and then (ii) 500 random sets of n BSF SSE values are chosen from the 250 runs.

Table 5 shows the results of our multi-start experiments. Based on our results in Section 3.1, we assume that constant annealing schedules with $T_i = 0.007$ are nearly optimal and use this schedule for run lengths from $10,000$ to $60,000$ steps and total training budgets of from $60,000$ to $240,000$ steps. For each run length M/n, our test sample contained 250 different runs. Then for each training budget M, we chose 500 different random subsets of size n, from which we obtained an average value for the best SSE from n runs. Our results indicate that multiple runs of annealing can indeed outperform a single long run (!). For instance, with a budget of $M = 240,000$ steps, ten runs with $24,000$ steps give an average SSE of .080 compared to .089 average SSE for one run with $240,000$ steps. This speed-up is significant, first because it allows the total step budget to be split among different processors, and second because it can save the total number of time steps required to achieve a given solution quality. For instance, splitting a CPU budget $M = 180,000$ into six runs gives nearly the same average SSE as a single run with $M = 240,000$.

4.2 Ideas II: Adaptive Schedules

From our experiments in Section 3.1 it is clear that for our present application, using annealing at a good constant temperature (i.e., the original Metropolis algorithm) is nearly as effective as the best exponential schedule. Thus, a good objective for an adaptive annealing schedule would be to search periodically during the run for the best temperature under the BSF criterion given the number of steps remaining in the run. For such an adaptive scheme to work, we need to estimate the "quality" of a single temperature. To this end we make use of a simple autoregressive (AR(1)) model of the Markov chain formed by running at a single temperature that was used by Lam and Delosme [22,23] to derive and justify their adaptive schedules (based on a WYA objective). For a given temperature T, the $AR(1)$ model is

$$f(s_{i+1}) = \mu + r * (f(s_i) - \mu) + \epsilon_i \tag{4}$$

where ϵ_i is a random variable with mean 0 and variance $\sigma^2(\epsilon)$, and μ, r, and $\sigma^2(\epsilon)$ depend on T. Note that the parameters in Equation 4 can be estimated by an ordinary least-squares regression. In the equation, μ represents the average cost at stationarity for temperature T, and can be used as an estimate for the WYA quality of temperature T. For the BSF quality of T at step k, we need to estimate $\min_{i=k}^{M} f(s_i)$, and to this end we are investigating the following general approach:

- Periodically during the run, estimate the BSF qualities of several (e.g., three) different values of T.
- For each T, run a number of trial steps starting from the current solution s_k.
- Estimate the AR(1) model of Equation 4 from the trial steps for each T; use the estimated parameters to compute μ, the average cost at stationarity for each T.
- Derive an estimate of $\sigma^2(f(s_i))$, the variance at stationarity of $f(s_i)$.
- Estimate the BSF quality of each tested T based on μ, $\sigma^2(f_i)$, and the number of remaining steps, $M-k$.
- Choose the next temperature based on these estimates.

Figure 5 shows a possible implementation of a procedure **FindNextTemp** which would be inserted into the annealing schedule periodically to determine the next temperature.

A result of Loynes [26] shows that for computing $\min_{i=k}^{M} f(s_i)$, we can assume that each of the $f(s_i)$ are independent if $M-k$ is sufficiently large. We also assume that the $f(s_i)$ are normally distributed with mean μ and variance $\sigma^2(f(s_i))$. An elementary result of order statistics [3] can then be used to compute the *median* of the distribution of $\min_{i=k}^{M} f(s_i)$ as follows. Let F be the cumulative distribution of $f(s_i)$ and let G be the cumulative distribution of $\min_{i=k}^{M} f(s_i)$. The probability that all $f(s_i)$ are greater than some value X is equal to $(1-F(X))^{M-k}$. Thus, we have that $G(X) = 1-(1-F(X))^{M-k}$. Substituting $G(X) = 1/2$ then gives us the median value for $\min_{i=k}^{M} X_i$, which is $F^{-1}(1-2^{-1/(M-k)})$.

5 SUMMARY

In this paper we have applied simulated annealing to the training of feedforward neural networks for detection of subsoil objects using separated-aperture radar. We investigate the use of non-traditional temperature schedules for SA, motivated by our previous research on small instances of global optimization problems. Our previous research suggested that traditional cooling schedules can be improved by optimizing the "best-so-far" solution rather than the "where-you-are" solution. Here, our simulations show that for training a (nearly) minimal architecture on 29 training cases, the optimal exponential schedule is in fact non-traditional. Indeed, the most effective way to improve annealing schedules appears to be to use temperatures from the correct range of values; even constant-temperature schedules from within this range are nearly optimal.

Procedure FindNextTemp(T,s,i)
Inputs: T = current temperature; s = current solution; i = current step
Output: next estimate T' of best temperature for the run
1. $j = i$ / chainLength
2. stepSize = $\sqrt{2/\log(2+j)}$
3. est1 = estimateBSF(T,s,testLength)
4. est2 = estimateBSF(T*stepSize,s,testLength)
5. est3 = estimateBSF(T/stepSize,s,testLength)
6. T' = minOfQuadratic(T,stepSize,est1,est2,est3)
7. **return** T'

Figure 5: Template of procedure **FindNextTemp** to change the temperature in an adaptive annealing run based on the BSF objective. Global variable "chainLength" represents the number of steps between changes in T. The procedure **estimateBSF**(T,s,l) estimates the quality of temperature T by running a Markov chain of length l, starting from solution s, and using temperature T. Procedure **minOfQuadratic** fits a quadratic equation $a * T^2 + b * T + c$ to the three points (T,est1), (T*stepSize,est2), and (T/stepSize,est3) (using a log scale for temperatures) and returns temperature T' at the minimum point of this quadratic equation.

We give experimental evidence that the network architecture we use is minimal among those that can be trained to zero error for our training cases. We also present experiments which indicate that multiple simulated annealing runs can be more effective than one long annealing run; multi-start annealing shows particular promise for training by parallel processors because of its trivial parallelizability. Finally, we outline a new adaptive annealing strategy based on a simple autoregressive model of simulated annealing and on order statistics. The strategy searches for the best annealing temperature for the best-so-far objective and a given number of steps remaining in the run. Other future research directions include (i) possible application of more complicated AR(k) or ARMA models to adaptive BSF annealing; (ii) adaptive methods of finding the best run lengths for multi-start annealing; (iii) fast methods to predict network architecture trainability; and (iv) experiments on BSF-optimal exponential schedules for other global optimization problems.

6 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. H. L. Aarts and P. J. M. van Laarhoven, "A New Polynomial-Time Cooling Schedule", in *Proc. IEEE Intl. Conf. on Computer-Aided Design*, Nov. 1985, pp. 206-8.

[3] B. C. Arnold, N. Balakrishnan and H. N. Nagaraja, *A First Course in Order Statistics*, Wiley, 1992, p. 12.

[4] M. R. Azimi-Sadjadi, D. E. Poole, S. Sheedvash, K. D. Sherbondy and S. A. Stricker, "Detection and Classification of Buried Dielectric Anomalies Using a Separated Aperture Sensor and a Neural Network Discriminator", *IEEE Trans. on Instrumentation and Measurement* 41(1) (1992), pp. 137-143.

[5] M. R. Azimi-Sadjadi and S. A. Stricker, "Detection and Classification of Buried Dielectric Anomalies Using Neural Networks–Further Results", *IEEE Trans. on Instrumentation and Measurement* 43(1) (1994), pp. 34-39.

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

[7] 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.

[8] K. D. Boese and A. B. Kahng, "Simulated Annealing of Neural Networks: the 'Cooling' Strategy Reconsidered", in *Proc. IEEE Int. Symp. on Circuits and Systems*, May 1993, pp. 2572-2575.

[9] K. D. Boese, A. B. Kahng and C.-W. A. Tsao, "Best-So-Far vs. Where-You-Are: New Perspectives on Simulated Annealing for CAD", in *Proc. European Design Automation Conf.*, Sept. 1993, pp. 78-83.

[10] K. D. Boese and A. B. Kahng, "Best-So-Far vs. Where-You-Are: Implications for Optimal Finite-Time Annealing", *Systems and Control Letters*, 22(1), Jan. 1994, pp. 71-8.

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

[12] P. Courrieu, "A Convergent Generator of Neural Networks," *Neural Networks* 6 (6) (1993), pp. 835-44.

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

[14] A. G. Ferreira and J. Zerovnik, "Bounding the Probability of Success of Stochastic Methods for Global Optimization", *Computers and Mathematics with Applications* 25(10/11) (1993), pp. 1-8.

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

[16] 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.

[17] 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.

[18] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by Simulated Annealing: an Experimental Evaluation; Part I, Graph Partitioning", *Operations Research* 27 (6) (1989), pp. 865-892.

[19] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by Simulated Annealing: an Experimental Evaluation; Part II, Graph Coloring and Number Partitioning", *Operations Research* 39 (3) (1991), pp. 378-406.

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

[21] P. J. M. van Laarhoven, *Theoretical and Computational Aspects of Simulated Annealing*, Amsterdam: Stichting Mathematisch Centrum, 1988.

[22] J. K.-C. Lam, "An Efficient Simulated Annealing Schedule", Ph.D. Thesis, Yale University, Dec. 1988.

[23] J. Lam and J.-M. Delosme, "Performance of a New Annealing Schedule", in *Proc. ACM/IEEE Design Automation Conf.*, June 1988, pp. 306-11.

[24] J. B. Lasserre, P. P. Varaiya, and J. Walrand, "Simulated Annealing, Random Search, Multistart or SAD?", *Systems and Control Letters* **8**, 297-301 (1987).

[25] 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.

[26] R. M. Loynes, "Extreme Values in Uniformly Mixing Stationary Stochastic Processes", *Annals Math. Stats.* 36 (1965), pp. 993-999.

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

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

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

[30] 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.

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

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

[33] B.-T. Zhang and H. Mühlenbein, "Genetic programming of minimal neural nets using Occam's razor", in *Proc. ICGA-93: Fifth Intl. Conf. on Genetic Algorithms*, July 1993, pp. 342-9.