JOURNAL OF COMPUTATIONAL PHYSICS **146,** 263–281 (1998)
ARTICLE NO. CP986065

A Feedback Algorithm for Determining Search Parameters for Monte Carlo Optimization

Chris Morey,* John Scales,† and Erik S. Van Vleck*

**Department of Mathematical and Computer Sciences, †Department of Geophysics and Center for Wave Phenomena, Colorado School of Mines, Golden, Colorado 80401*
E-mail: jscales@mines.edu, evanvlec@mines.edu

Received November 4, 1997; revised July 20, 1998

Monte Carlo methods have become popular for obtaining solutions to global optimization problems. One such Monte Carlo optimization technique is simulated annealing (SA). Typically in SA the parameters of the search are determined *a priori*. Using an aggregated, or *lumped*, version of SA's associated Markov chain and the concept of expected hitting time, we adjust the search parameters dynamically using information gained from the SA search process. We present an algorithm that varies the SA search parameters dynamically, and show that, on average, dynamic adjustment of the parameters attains better solutions on a set of test problems than those attained with a logarithmic cooling schedule. © 1998 Academic Press

1. INTRODUCTION

Researchers in mathematics, engineering, and the sciences frequently encounter global optimization problems of such magnitude that total enumeration of the state space or application of deterministic algorithms is computationally impractical. Because of the size of these problems, Monte Carlo methods—methods that employ some aspect of randomness—are used to search for extrema of the objective function. The Monte Carlo optimization technique that we will emphasize is simulated annealing (SA). SA is defined by search parameters that are typically determined *a priori*. Without knowledge of the behavior of the function being analyzed, poor selection of the search parameters can lead to a poor solution or to a prohibitive computational effort in attaining a solution.

Our solution to determining the search parameters is to do so dynamically using information obtained during the search process by perturbing the transition probability matrix associated with the underlying Markov chain. Our contributions include an algorithm that, on average, attains better solutions to nonlinear multimodal global optimization problems. We have tested the algorithm on established test problems from the literature to determine the viability of the method. The algorithm relies on the transition probability matrix associated with the Markov chain underlying SA. We give conditions for which the SA Markov chain is lumpable. Our criterion for dynamically adjusting the search parameters is the *expected hitting time*.

This paper is organized as follows. Section 2 provides background on SA and a diagnostic tool, the expected hitting time, that we use to adjust the search parameters. Section 3 gives background theory on the underlying Markov chain, its lumpability, and one set of conditions for which we can analytically calculate expected hitting time. Section 4 contains theoretical results in our application of expected hitting time to the lumped Markov chain. In Section 5, we present our algorithm using the lumped model and indicate the relationship between this model and the model that uses the whole state space. Section 6 contains our numerical results, and Section 7 has our conclusions and areas requiring further study.

2. BACKGROUND

Simulated annealing (SA) was introduced in [16] as a means of solving large combinatorial optimization problems. A scalar valued objective function f is defined over a suitable finite state space $S = \{1, 2, \ldots, n\}$, where each $s \in S$ corresponds to a feasible solution of the optimization problem. Typically we want to minimize the objective function, f. For each point, i, in the domain, define a neighborhood, $N(i)$, of i. Define $|N(i)|$ to be the number of states in $N(i)$.

It is convenient to think of the SA algorithm acting on a single particle moving through S. A sequence of states develops by randomly choosing a starting state, i. Next a state, $j \in N(i)$, is selected at random. If $f(j)$ is less than or equal to $f(i)$, then the particle moves to state j and the process is continued. If $f(j)$ is greater than $f(i)$, then the move to j is accepted with probability $\exp[(f(i) - f(j))/c]$, where c is a parameter (the temperature). The heart of SA is the move criterion $\exp[(f(i) - f(j))/c]$, where $f(j) > f(i)$ and was originally proposed in [18] as a means of simulating the equilibration of a gas in a heat bath. When c is sufficiently large (analogous to the gas in the heat bath being heated to a very high temperature), virtually every move will be accepted, including uphill moves; this allows the particle to escape from local minima. As c decreases, the chances of accepting an uphill move decrease so that at a low value of c, the particle will be trapped in a minimum. A *cooling schedule* is a specification of c as a function of time. It has been shown [10, 13] that, given an ideal cooling schedule that decrements c as well as determines how many moves to make at each value of c, the process will converge to the globally optimal solution with probability 1. But such an ideal schedule would require an impractical amount of computation.

Mathematically, SA can be viewed as a discrete time nonstationary Markov chain. The underlying transition probability matrix associated with the stationary (fixed c) SA Markov chain has the following transition probabilities for c given [1]:

$$p_{ij} = \begin{cases} \frac{1}{|N|}, & f(j) \le f(i),\ j \in N(i); \\ \frac{1}{|N|} \exp\left(\frac{f(i)-f(j)}{c}\right) \equiv \frac{1}{|N|} g_{ij}, & f(i) < f(j),\ j \in N(i); \\ 0, & j \notin N(i); \end{cases} \tag{1}$$

$$p_{ii} = 1 - \sum_{\substack{j=1 \\ j \ne i}}^{n} p_{ij}.$$

The elements p_{ij} give the probability of a transition to state j given that the process is in state i.

In the application of SA, many factors affect the algorithm's performance. Some of these factors have to do with the cooling schedule, and some with the neighborhood structure [1]. To reduce the time required to reach a good (although not necessarily optimal) solution, cooling schedules have received much attention [1, 3, 4, 10, 13, 20, 28].

Neighborhood structure is made up of two components. The first involves the method for generating the next state, j, to which a move is proposed from the current state, i. The second component is the cardinality of the set $N(i)$. Both the generation of a neighbor and neighborhood size have been addressed by various researchers [7, 11, 33, 34]. Usually, the neighborhood structure is determined *a priori*, although in [33] and [34] it is done using some information gained from the search. Yet, because of the size of the underlying stochastic matrix, use of the matrix itself for gaining information about the search process is typically avoided.

We refer to the cooling schedule and neighborhood structure as the *search parameters*. Our goal is to adjust these parameters dynamically to accelerate the convergence of the algorithm to the set of globally optimal solutions. We want to do this without *a priori* knowledge of the function being studied, aside from variable constraints. The tool we use is the *expected hitting time*.

Shonkwiler and Van Vleck [26] provide a concept of convergence that, given the Markov chain associated with a Monte Carlo search process, determines the expected time to reach a goal state, which in our case is a global minimum. If θ denotes the random variable equal to the first time t that the state of the search process reaches a goal state, then the expected hitting time (EHT) is given by

$$\mathrm{E}(\theta) = \sum_{t=1}^{\infty} t \,\Pr(\theta = t), \tag{2}$$

where $\Pr(\theta = t), t = 1, 2, \ldots,$ is the probability density function for θ. Suppose that $\hat{P}$, called the *deleted transition probability matrix*, is the submatrix that results from the deletion of the rows and columns of P that correspond to the states in which global minima occur. Define $|S|$ to be the cardinality of the set of all states, S. A formula for the EHT (see [26]) is

$$\mathrm{EHT} = 1 + \hat{\alpha}_0^T \left(\sum_{t=0}^{\infty} \hat{P}^t \right) \mathbf{1}, \tag{3}$$

where $\hat{\alpha}_0$, a vector of length $|S| - 1$, is the initial probability distribution on the states in $\hat{P}$, and $\mathbf{1}$ is an $|S| - 1$ length vector of ones. Because of the difficulty in evaluating (3) for large $\hat{P}$, Shonkwiler and Van Vleck provide an estimate for the expected hitting time. Let $\hat{\lambda}$ be the maximal eigenvalue of $\hat{P}$. Let χ be a right eigenvector of $\hat{P}$ associated with $\hat{\lambda}$, where χ has been normalized such that its inner product with the left eigenvector (which has been normalized to be a probability vector) is 1. If we define the parameter s as

$$s = \frac{\hat{\lambda}}{\hat{\alpha}_0^T \chi}, \tag{4}$$

then the expected hitting time, for a stationary, single process search, can be approximated by

$$E_1 = \frac{1}{s}\frac{1}{1-\hat{\lambda}}, \tag{5}$$

while for independent identical processes (IIP), that is, m parallel processes, the expected hitting time is approximated by

$$E_m = \frac{1}{s^m}\left(\frac{1}{1-\hat{\lambda}^m}\right). \tag{6}$$

The difficulty in using the estimate (5) or (6) is the need to calculate left and right eigenvectors of $\hat{P}$. In general, working with the underlying Markov matrix is unrealistic given its $|S| \times |S|$ dimension. In [4] a lumped model of the SA Markov chain is developed using theory that can be found in [15]. Their criterion for designing an annealing schedule is based on thermodynamic considerations; the temperature is decreased in such a way as to minimize entropy production. The equation governing the temperature decrease can be expressed in terms of the relaxation rate of the Markov chain, which they estimate from the lumped model. In the most general sense, lumping a Markov chain involves aggregating states into megastates. The result of the lumping is a Markov chain giving a coarse analog of the full chain. In this sense our use of lumping is an example of a more general class of methods known in operations research as aggregation and disaggregation techniques [23]. However, our strategy is not simply to reduce the size of the original optimization problem, but rather to use lumping as a means of estimating the expected hitting time of the full problem. In this sense our approach is rather different from standard aggregation/disaggregation techniques.

Typically, the full chain cannot be lumped arbitrarily without losing the Markov property, but given certain conditions, and matrices U and V defined in Section 3, the transition probability matrix P can be lumped to L using the transformation [15]

$$L = UPV. \tag{7}$$

We refer to the underlying Markov chain as the *state space chain* and the lumped chain as the *energy space chain*, where energy is meant to denote the value of the objective function. The lumping in [4] is a lumping by energies, i.e., states with similar energies are aggregated in the lumped model.

We conclude this section with notation and some of the assumptions that we use throughout the remainder of this paper. We assume that the transition probability matrix, P, has been ordered $\{1, 2, \ldots, |S|\}$ such that $f(1) \leq f(2) \leq \cdots \leq f(|S|)$. We also assume that the stationary matrix, P, is ergodic. The deleted transition probability matrix associated with P is $\hat{P}$. The lumped version is L and its associated deleted transition probability matrix is $\hat{L}$. The expected hitting time calculated using (3) for $\hat{P}$ is denoted by E_P while that associated with $\hat{L}$ is denoted by E_L. The values using the estimate of the expected hitting time in (5) are denoted by $\tilde{E}_P$ and $\tilde{E}_L$. Although it is possible to go from i at step t to i at $t+1$, for the following definitions, and throughout the paper, we assume $i \notin N(i)$ for the purpose of generating states to which a move is proposed from i. The vector of ones is denoted by $\mathbf{1}$, and the length of $\mathbf{1}$ will be clear from the context.

3. ALGORITHM

Our primary goal in developing the algorithm was to provide a robust approach, capable of attaining an optimal solution with no *a priori* knowledge of the function being optimized other than variable constraints. The algorithm automates the feedback loop associated with typical implementations of SA; that is, rather than run the SA several times to determine appropriate choices of c and N, the algorithm does so automatically. In addition, as we will see there is some theoretical justification for our choice of c and N. The top-level algorithm involves augmenting a search process such as SA with a diagnostic on the efficiency of the search that allows one to adjust the search parameters.

TOP LEVEL ALGORITHM

1. Given a fixed value of the cooling parameter and a structure for the neighbors of the states $i \in S$ begin the **search** process.
2. Gather information during the search process including function values and neighborhood structure to **approximate the Markov chain.**
3. Using the information found during the search and a **diagnostic** that predicts an acceleration in the search process, choose new values of c and the neighborhood structure.

This is a simple example of a feedback loop that uses the previously sampled states to determine values of c and the neighborhood structure that are predicted to converge to the optimum value faster. Our goal is to come up with a diagnostic that is efficient, easy to implement, and a good predictor. What we propose is to use the EHT of a lumped version of the approximate Markov chain we have constructed through sampling. We will then compare the lumped EHT for the current values of c and neighborhood structure with different values of c and different neighborhood structure.

Diagnostic. 1. Given pairs of function values and states and a neighborhood structure, build lumped Markov chains for different values of c and neighborhood structure.

2. Determine the lumped EHT for different values of c and neighborhood structure using (3), with $\hat{L}$ substituted for $\hat{P}$, (4), and (5) and an eigendecomposition of $\hat{L}$, or using (3), with $\hat{L}$ substituted for $\hat{P}$, directly by solving the linear system $(I - \hat{L})x = \mathbf{1}$.

3. Choose a new c and neighborhood structure by selecting the one with the minimum value of lumped EHT.

In our implementation we used uniform neighborhood size $|N|$ and the following simple method for determining the updated c and N. Using increments for c and N, δc and δN, respectively, we calculate E_L so that we have nine two-tuples: $(c + \delta c, N)$, $(c + \delta c, N - \delta N)$, $(c + \delta c, N + \delta N)$, $(c, N - \delta N)$, (c, N), $(c, N + \delta N)$, $(c - \delta c, N - \delta N)$, $(c - \delta c, N)$, and $(c - \delta c, N + \delta N)$. This list can evidently be expanded for different values of δ. Calculating E_L for the nine two-tuples has the effect of performing a local search on the (c, N) surface to find the point that minimizes E_L. The minimum value obtained for E_L during this process gives us (c, N) for the next k iterations of the SA search.

We have formed L by sampling along the search trajectory, considering the current state and all of its neighbors, and have updated L and the eight related forms of L using an insertion sort. As we explain in more detail in the next section, if L is an m by m matrix, then the first $m - 1$ states correspond to the $m - 1$ lowest function values encountered along our search trajectory, while the remaining state is an aggregate of all those states with higher function values. The only entries that are difficult to keep track of in L (see (1) and (8)) are the transition probabilities from any of the first $m-1$ states to the aggregated state. We accomplish this by updating the transition probabilities to the aggregated state as new states are encountered. Once our sampling is complete, we form $\hat{L}$ by deleting the first row and column of L.

4. THE SA MARKOV CHAIN

Given our ordering and a uniform neighborhood assumption, $N = |S| - 1$, the transition probability matrix has the structure:

$$P = \begin{pmatrix} 1-\sum_{j=2}^{|S|}\frac{g_{1,j}}{N} & \frac{g_{1,2}}{N} & \frac{g_{1,3}}{N} & \cdots & \frac{g_{1,|S|}}{N} \\ \frac{1}{N} & \frac{N-1}{N}-\sum_{j=3}^{|S|}\frac{g_{2,j}}{N} & \frac{g_{2,3}}{N} & \cdots & \vdots \\ \frac{1}{N} & \frac{1}{N} & \ddots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \frac{2}{N}-\frac{g_{|S|-1,|S|}}{N} & \frac{g_{|S|-1,|S|}}{N} \\ \frac{1}{N} & \cdots & \cdots & \frac{1}{N} & 0 \end{pmatrix}. \qquad (8)$$

We call this matrix the *canonical form* of the SA transition probability matrix. We can make any SA problem fit the canonical form by adjusting N and by making $p_{ij} = 0$ for those one-step transitions that are not permitted. The deleted transition probability matrix, $\hat{P}$, is simply the matrix P in (8) with the first row and first column deleted.

Remark 4.1. Not all of the structural information of P and $\hat{P}$ is readily evident from the form of the matrices. If $k > j > i$, $p_{ij} \neq 0$, and $p_{ik} \neq 0$, then $p_{ij} \geq p_{ik}$. Additionally, if $k < j < i$, $p_{ji} \neq 0$, and $p_{ki} \neq 0$, then $p_{ji} \geq p_{ki}$. Finally, $p_{ij} = 0 \Leftrightarrow p_{ji} = 0$.

Before we present our results on lumpability, we provide definitions and theorems from [15]. Assume that we want to lump the SA Markov chain with respect to the partition $\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_m\}$, $m \leq |S|$. For our purposes, $m \ll |S|$. Let $q_{iA_j} = \sum_{k\in\mathbf{A}_j} p_{ik}$; that is, q_{iA_j} is the sum of the transition probabilities from i to the states in lump $\mathbf{A}_j$. The next theorem gives the conditions under which a Markov chain can be lumped while guaranteeing that the Markov property is preserved.

THEOREM 4.1 [15, *Theorem* 6.3.2]. *A necessary and sufficient condition for a Markov chain to be lumpable with respect to a partition* $\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_m\}$ *is that, for every pair of sets* $\mathbf{A}_i$ *and* $\mathbf{A}_j$, q_{kA_j} *is the same for every state* s_k *in* $\mathbf{A}_i$.

The condition given in Theorem 4.1 is referred to as the *row sum* criterion [6].

DEFINITION 4.1. Let U be the $m \times |S|$ matrix whose ith row is a uniform probability vector over the states in $\mathbf{A}_i$, and 0 otherwise. Let V be the $|S| \times m$ matrix such that $v_{ij} = 1$ if $s_i \in \mathbf{A}_j$, and 0 otherwise.

If P is lumpable with respect to the partition $\mathbf{A}$, the lumped chain, L, is obtained from the transformation

$$L = UPV, \qquad (9)$$

with transition probabilities L_{ij} [2, 15].

THEOREM 4.2 [15, *Theorem* 6.3.4]. *If P is the transition matrix of a chain lumpable with respect to the partition* $\mathbf{A}$, *and if the matrices U and V are defined as in Definition* 4.1 *with respect to this partition, then*

$$VUPV = PV. \tag{10}$$

THEOREM 4.3 [15, *Theorem* 6.3.5]. *If P, $\mathbf{A}$, U, and V are as in Theorem* 4.2, *then condition* (10) *is equivalent to lumpability.*

Let n_i be the number of states in the partition $\mathbf{A}_i$, and let s_{kA_i} denote the kth state in partition $\mathbf{A}_i$, that is, state s_k is a member of the aggregated state A_i. With these definitions and the background we have presented from [15], we can show that the lumping cannot be done arbitrarily if the lumped chain is to maintain the Markov property.

THEOREM 4.4. *Given the state-space transition probability matrix P associated with an SA problem with full neighborhood size, the Markov chain is lumpable with respect to the partition*

$$\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_m\} \tag{11}$$

if and only if

$$\mathbf{A}_j = \{s_{1A_j}, s_{2A_j}, \ldots, s_{n_jA_j}\}, \quad j = 1, 2, \ldots, m-1, \tag{12}$$

where

$$f(s_{iA_j}) = f(s_{kA_j}) \quad \textit{for all } i, k \in \mathbf{A}_j. \tag{13}$$

Proof. Suppose P is lumpable with respect to the partition $\mathbf{A}$. By Theorem (4.3), $VUPV = PV$. Let $i, j \in \mathbf{A}_k$. Let e_i be the unit vector of length $|S|$ with 1 in the ith position, let e_j be a unit vector of the same length with 1 in the jth position, and let e_k be a unit vector of length m with 1 in the kth position. Then $e_i^T VUPV = e_i^T PV \Rightarrow e_k^T UPV = e_i^T PV \Rightarrow e_k^T L = e_i^T PV$. Similarly, $e_k^T L = e_j^T PV$. Hence, $e_i^T PV = e_j^T PV$. Partition V in the form $V = (\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_m})$, and choose arbitrary $s \in \{1, 2, \ldots, m-1\}$. Then $e_i^T P\mathbf{v_s} = e_j^T P\mathbf{v_s}$, where $P\mathbf{v_s}$ has the effect of summing the probabilities in P associated with the states in lump $\mathbf{A}_s$. The equation $e_i^T P\mathbf{v_s} = e_j^T P\mathbf{v_s}$ requires that the sums in rows i and j be the same for lump $\mathbf{A}_s$. Recalling the structure of P given in Remark 4.1, we see that this means that rows i and j are the same element-wise. Therefore, $f(s_i) = f(s_j)$ for all $i, j \in \mathbf{A}_k$. Conversely, suppose $\mathbf{A}_k = \{s_{1A_k}, s_{2A_k}, \ldots, s_{n_kA_k}\}$, $j = 1, 2, \ldots, m-1$, where $f(s_i) = f(s_j)$ for all $i, j \in \mathbf{A}_k$. By the structure of the matrix, the rows for the states in $\mathbf{A}_k$ are the same. The result follows directly from Theorem 4.1. ■

Theorem 4.4 allows us to lump the chain by energy level (or function value in our analogy) as long as the state that corresponds to the minimum function value is the only state in the first lump. More precisely, if we want m lumps, then the first $m-1$ lumps must satisfy $f(1) \leq f(2) \leq \cdots \leq f(m-1)$, while the remaining states can all be lumped into state $\mathbf{A}_m$, as long as $f(j) \geq f(i)$, $j \in \mathbf{A}_m$, $i \notin \mathbf{A}_m$. For the full neighborhood size case, we can have $2 \leq m \leq |S|$ and still maintain the Markov property. On the other hand, the conditions necessary for lumping a chain with other than full neighborhood size do not exist in general as can be seen from Theorem 4.1. However, in our numerical experiments we will form an aggregated transition probabity matrix L and associated aggregated deleted transition probabity matrix $\hat{L}$ by putting the $m-1$ states with lowest function values into aggregated states $A_1, \ldots, A_{m-1}$ and the remaining states into aggregated state A_m.

5. THEORETICAL RESULTS

We begin this section with theorems relating the deleted transition probability matrix, $\hat{P}$, to the lumped matrix, $\hat{L}$. Recall that $\hat{P}$ is derived from the transition probability matrix, P, by deleting the first row and column from P. The matrix $\hat{L}$ is similarly derived from the lumped matrix, L. Using the results we obtain on the relationship between $\hat{P}$ and $\hat{L}$ along with results on the inverses of certain M-matrices, we demonstrate the relationships between the expected hitting time for the state space model and the expected hitting time for the lumped model for the full neighborhood size case and the other than full neighborhood size case. As before, the state space matrix is $|S| \times |S|$, while the lumped matrix is $m \times m$. Let E_P and E_L be the expected hitting times obtained using (3) for the state space model and the lumped model, respectively. Let $\tilde{E}_P$ and $\tilde{E}_L$ be the estimated expected hitting times obtained using (5) for the state space model and the lumped model, respectively.

DEFINITION 5.1. If P is lumpable with respect to the partition $\mathbf{A} = \{\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_m\}$, and if U and V are as defined in Definition 4.1, then if $L = UPV$, P is lumpable to L.

Remark 5.1. From Theorem 4.4, an implicit part of Definition 5.1 is that P has full neighborhood size, since full neighborhood size is the only case in which P is lumpable to L (by the row sum criterion).

DEFINITION 5.2. $\hat{U}$ is the $(m-1) \times (|S|-1)$ matrix constructed by removing the first row and column of U. Similarly, $\hat{V}$ is the $(|S|-1) \times (m-1)$ matrix resulting from the deletion of the first row and column of V.

LEMMA 5.1. *If P is lumpable to L, then*

$$\hat{V}\hat{U}\hat{P}\hat{V} = \hat{P}\hat{V}. \tag{14}$$

Proof. From Theorem 4.2, we have $VUPV = PV$. As demonstrated in the proof of Theorem 6.3.4 on page 126 in [15], VU has the form

$$VU = \begin{pmatrix} W_1 & 0 & 0 & \cdots & 0 \\ 0 & W_2 & 0 & \cdots & \cdot \\ \cdot & & \ddots & & \cdot \\ \cdot & & & W_{m-1} & 0 \\ 0 & \cdot & \cdot & 0 & W_m \end{pmatrix}. \tag{15}$$

In (15), each W_i is square with the number of columns equal to the number of states in lump i. Additionally, each W_i is a rank one stochastic matrix with identical rows. By the definitions of $\hat{V}$, and $\hat{U}$, the matrix $\hat{V}\hat{U}$ is the same as VU, but without W_1 and its corresponding row(s) and column(s) of zeros. By our construction of P, W_1 multiplies only those rows and columns corresponding to the states where the global minima occur and hence has no effect on the remainder of P. Likewise, the remaining W_j multiply only those rows and columns corresponding to the states in lump j and have no effect on the rows and columns corresponding to the states in the first lump. Hence we get the desired result. ■

Paralleling the result in [15] that $L^n = UP^nV$, we also have $\hat{L}^n = \hat{U}\hat{P}^n\hat{V}$. Let β_0 be the initial distribution over the states in L. Construct β_0 so that the ith component is $1/|S|\sum_S I_{\mathbf{A}_i}$, $i = 2, 3, \ldots, m$, where $I_{\mathbf{A}_i}$ is the characteristic function indicating membership of a state in partition $\mathbf{A}_i$. In this case, if $|\mathbf{A}_i| = n$, choose the ith component of β_0 to be $n/|S|$. We let $\hat{\beta}_0$ be the vector of length $m-1$ whose components correspond to the initial distribution over those states in $\hat{L}$. This construction of β_0 is equivalent to the following definition.

DEFINITION 5.3. Given α_0, an initial distribution over the states in S, and $\hat{V}$ as defined in Definition 5.2, then

$$\hat{\beta}_0 = \hat{V}^{\mathrm{T}}\hat{\alpha}_0. \tag{16}$$

LEMMA 5.2. *If P is lumpable to L, α_0 is a uniform initial distribution over the states in S, and β_0 is as defined in Definition* 5.3, *then*

$$\hat{\beta}_0^{\mathrm{T}}\hat{L}^n\mathbf{1} = \hat{\alpha}_0^{\mathrm{T}}\hat{P}^n\mathbf{1}. \tag{17}$$

Proof. We start with the equality

$$\hat{\beta}_0^{\mathrm{T}}\hat{L}^n\mathbf{1} = \hat{\beta}_0^{\mathrm{T}}\hat{U}\hat{P}^n\hat{V}\mathbf{1}, \tag{18}$$

where $\hat{V}$ is $(|S|-1)\times(m-1)$ with only one 1 in each row. Hence, $\hat{V}\mathbf{1}$ gives an $|S|-1$ vector of ones. Similarly, $\hat{\beta}_0$ has ith component $n_i/|S|$, where n_i is the number of states in lump i. By Definition 5.2, the ith row of $\hat{U}$ has n_i elements, each of which is $1/n_i$. Consequently, $\hat{\beta}_0^{\mathrm{T}}\hat{U} = \hat{\alpha}_0^{\mathrm{T}}$. Making these substitutions on the right-hand side of (18) gives

$$\hat{\beta}_0^{T}\hat{L}^k\mathbf{1} = \hat{\alpha}_0^{T}\hat{P}^k\mathbf{1}, \tag{19}$$

the desired result. ■

When $\hat{P}$ is not obtained from a matrix with full neighborhood size, the analysis is not so straightforward because we do not have the necessary conditions for lumping $\hat{P}$. Yet, because of the size of $\hat{P}$, we still prefer to work with some sort of aggregation of the states. Let $\hat{V}$ and $\hat{U}$ be as in Definition 5.2, and let $\hat{L} = \hat{U}\hat{P}\hat{V}$. Let $E_L = 1 + \hat{\beta}_0^{\mathrm{T}}\mathbf{1} + \sum_{t=2}^{\infty}\hat{\beta}_0^{\mathrm{T}}\hat{L}^t\mathbf{1}$, be the expected hitting time for the lumped model using (3). We are interested in finding $|E_L - E_P|$. We choose to find $|E_L - E_P|$ rather than $|\tilde{E}_L - E_P|$. As will become evident, finding the former requires finding the inverse of a relatively small matrix, whereas the latter requires calculating the maximal eigenvalue and corresponding left and right eigenvectors of $\hat{U}\hat{P}\hat{V}$.

Beginning with $E_L - E_P$ and using (3), we can write

$$E_L - E_P = \sum_{t=0}^{\infty}\hat{\beta}_0^{\mathrm{T}}\hat{L}^t\mathbf{1} - \sum_{t=0}^{\infty}\hat{\alpha}_0^{\mathrm{T}}\hat{P}^t\mathbf{1}. \tag{20}$$

By the definition of $\hat{\beta}_0$, and recognizing that $\hat{\beta}_0^{\mathrm{T}}\hat{L}\mathbf{1} = \hat{\beta}_0^{\mathrm{T}}\hat{U}\hat{P}\hat{V}\mathbf{1} = \hat{\alpha}_0^{\mathrm{T}}\hat{P}\mathbf{1}$, (20) reduces to

$$E_L - E_P = \hat{\beta}_0^{\mathrm{T}}\left(\sum_{t=0}^{\infty}(\hat{U}\hat{P}\hat{V})^t - \hat{U}\left(\sum_{t=0}^{\infty}\hat{P}^t\right)\hat{V}\right)\mathbf{1}. \tag{21}$$

The matrices $\hat{U}\hat{P}\hat{V}$ and $\hat{P}$ have row sums that are less than or equal to one. Since each of the matrices is nonnegative and primitive, from Perron–Frobenius theory, each has maximal eigenvalue less than 1. Given our assumption of ergodicity, by Lemma 2.1 in [5], the infinite sums in (21) converge to $(I - \hat{U}\hat{P}\hat{V})^{-1}$ and $(I - \hat{P})^{-1}$, respectively, so that (21) becomes

$$E_L - E_P = \hat{\beta}_0^{\mathrm{T}}[(I - \hat{U}\hat{P}\hat{V})^{-1} - \hat{U}(I - \hat{P})^{-1}\hat{V}]\mathbf{1}. \tag{22}$$

In (22), we see the source of error in the estimate using the lumped model as opposed to the full model. The error matrix is obtained from $(I - \hat{U}\hat{P}\hat{V})^{-1} - \hat{U}(I - \hat{P})^{-1}\hat{V}$. Evaluating $(I - \hat{U}\hat{P}\hat{V})^{-1}$ and $(I - \hat{P})^{-1}$ is difficult for general $\hat{P}$. Fortunately, some theory exists for matrices with this structure. In some cases, the theory will allow us to bound the error $|E_L - E_P|$. The matrices $(I - \hat{P}\hat{V}\hat{U})$ and $(I - \hat{P})$ are nonsingular M-matrices. An M-matrix is a matrix that has nonpositive off-diagonal elements and positive diagonal elements (see, e.g. [5, p. 132]). By Theorem 3.11 of [29], $(I - \hat{P}\hat{V}\hat{U})$ and $(I - \hat{P})$ have inverses whose elements are strictly greater than zero, and we write $(I - \hat{P}\hat{V}\hat{U})^{-1} > 0$ and $(I - \hat{P})^{-1} > 0$.

Remark 5.2. Upper bounds on $\|(I - \hat{P}\hat{V}\hat{U})^{-1}\|_\infty$ and $\|(I - \hat{P})^{-1}\|_\infty$ give us upper bounds on E_P and E_L, respectively. For the case of $(I - \hat{P})^{-1}$, we have

$$\begin{aligned} E_P &= \alpha_0^{\mathrm{T}}(I - \hat{P})^{-1}\mathbf{1} && (23)\\ &\leq \|\alpha_0\|_\infty \, \|(I - \hat{P})^{-1}\|_\infty \, \|\mathbf{1}\|_\infty && (24)\\ &= \frac{|S| - 1}{|S|}\|(I - \hat{P})^{-1}\|_\infty && (25)\\ &< \|(I - \hat{P})^{-1}\|_\infty. && (26)\end{aligned}$$

Using $\hat{\beta}_0$ and $(I - \hat{U}\hat{P}\hat{V})^{-1}$ and the same approach, we obtain $E_L < \|(I - \hat{U}\hat{P}\hat{V})^{-1}\|_\infty$.

PROPOSITION 5.1. *Let $M = (I - \hat{P}) \in \mathbb{R}^{n\times n}$ be a nonsingular M-matrix, where $\hat{P}$ is obtained from a row-stochastic matrix $P \in \mathbb{R}^{(n+1)\times(n+1)}$. Suppose M has row sums equal to either zero or r. Let $C = M^{-1}$ and $L = \{l \mid \sum_{j=1}^{n} m_{lj} = r\}$. Then,*

$$\sum_{\substack{l=1\\ l\in L}}^{n} c_{il} = \frac{1}{r} \quad \text{for all } i. \tag{27}$$

Proof. Define the sets K and L such that $K = \{k \mid \sum_{j=1}^{n} m_{kj} = 0\}$ and $L = \{l \mid \sum_{j=1}^{n} m_{lj} = r\}$. Let $M\mathbf{1} = \mathbf{r}$, where $r_i = 0$ or $r_i = r$. Since $M^{-1} = C$ exists, we can write $\mathbf{1} = C\mathbf{r}$. Choose an arbitrary row, say i of C. We have, $\sum_{j=1}^{n} c_{ij}r_j = 1$, or $\sum_{k\in K} c_{ik}r_k + \sum_{l\in L} c_{il}r_l = 1$. Since $r_i = r$ for $l \in L$, and 0 otherwise, we can write $r\sum_{l\in L} c_{il} = 1$. Dividing both sides by r, we obtain the desired result. ■

The result in Proposition 5.1 gives us a crude lower bound on the expected hitting time in that it gives us a lower bound on $\|(I - \hat{P})^{-1}\|_\infty$. In an SA process, there will be N nonzero row sums of $I - \hat{P}$ corresponding to the states that are neighbors of the global minimum, and the row sums that are nonzero will all equal $1/N$. For $N < |S| - 1$, since $(I - \hat{P})^{-1}$ has positive elements, $\|(I - \hat{P})^{-1}\|_\infty > N$ and $\|(I - \hat{P})^{-1}\|_\infty = N$ for $N < |S| - 1$. Therefore, $\alpha_0^{\mathrm{T}}(I - \hat{P})^{-1}\mathbf{1} \geq N\alpha_0^{\mathrm{T}}\mathbf{1}$.

5.1. *Parallel Expected Hitting Time*

Thus far, we have been concerned with the sequential SA, that is, a single SA process. In [26], it is shown that running Monte Carlo optimization techniques in parallel can result in super-linear speedup for the process of finding global minima. The results they provide are applicable to our algorithm, but here we are concerned with the magnitude of the error between the expected hitting time of a state space model run in parallel and the expected hitting time of the lumped model run in parallel. The expected hitting time for a single process is $\sum_{t=1}^{\infty} \Pr(\theta \geq t)$. In addition to providing the estimate in (6) for the expected hitting time for parallel processes, it is shown in [26] that the expected hitting time for n independent identical processes is $\sum_{t=1}^{\infty}[\Pr(\theta \geq t)]^n$. Let $E_P = \sum_{t=1}^{\infty} \Pr(\theta_P \geq t)$ and $E_L = \sum_{t=1}^{\infty} \Pr(\theta_L \geq t)$ be the full and lumped expected hitting times for the single process, respectively, and let $E_P^{(n)} = \sum_{t=1}^{\infty}[\Pr(\theta_P \geq t)]^n$ and $E_L^{(n)} = \sum_{t=1}^{\infty}[\Pr(\theta_L \geq t)]^n$ be the full and lumped expected hitting times for n independent identical processes. Then,

$$E_L^{(n)} - E_P^{(n)} = \sum_{t=1}^{\infty}[\Pr(\theta_L \geq t)]^n - \sum_{t=1}^{\infty}[\Pr(\theta_P \geq t)]^n. \tag{28}$$

Choose the cooling parameter $c > 0$ such that $\lim_{t\to\infty}[\Pr(\theta_P \geq t)]^n = 0$ and $\lim_{t\to\infty}[\Pr(\theta_L \geq t)]^n = 0$. We impose this condition to maintain ergodicity. Without ergodicity, one or both of the sums on the right-hand side of (28) may not converge. We can write (28) as

$$\left|E_L^{(n)} - E_P^{(n)}\right| = \left|\sum_{t=1}^{\infty} \beta(t)\alpha(t)\right|, \tag{29}$$

where

$$\beta(t) = [\Pr(\theta_L \geq t) - \Pr(\theta_P \geq t)] \tag{30}$$

and

$$\begin{aligned}\alpha(t) = {} & \Pr(\theta_L \geq t)^{n-1} + \Pr(\theta_L \geq t)^{n-2}\Pr(\theta_P \geq t) + \cdots \\ & + \Pr(\theta_L \geq t)\Pr(\theta_P \geq t)^{n-2} + \Pr(\theta_P \geq t)^{n-1}.\end{aligned}$$

We also have

$$|E_L - E_P| = \left|\sum_{t=1}^{\infty} \beta(t)\right|. \tag{31}$$

From (29), it is clear that as $n \to \infty$, $\alpha(t) \to 0$ because $\Pr(\theta_P \geq t)$ and $\Pr(\theta_L \geq t)$ are both less than one. Consequently, $|E_L^{(n)} - E_P^{(n)}| \to 0$ as $n \to \infty$. If $\sum_{t=1}^{\infty} \alpha^2(t) < \infty$, then using the Cauchy–Schwarz inequality,

$$\left|E_L^{(n)} - E_P^{(n)}\right| \leq \left(\sum_{t=1}^{\infty} \alpha^2(t)\right)^{1/2} \left(\sum_{t=1}^{\infty} \beta^2(t)\right)^{1/2}, \tag{32}$$

where a crude lower bound on $\alpha(t)$ is $n[\max\{\Pr(\theta_L \geq t), \Pr(\theta_P \geq t)\}]^{n-1}$.

6. NUMERICAL RESULTS

To test the feedback algorithm we have chosen a suite of standard global optimization test functions. They are representative of the kinds of functions that arise in our work in geophysical optimization and seismic inversion. (See, for example, [19], [21], [22], [24], [30]). In Sections 6.1–6.3, we test our algorithm on the following functions (defined in the Appendix):

1. Branin's RCOS [27, p. 1027]
2. 2D six-hump camel back [27, p. 1027]
3. 2D Shubert [27, p. 1028]
4. Function F2 [31, p. 241]
5. Function F8 [31, p. 241].

Each of the above functions has variables x_i with $l_i \leq x_i \leq u_i$, where l and u are lower and upper bounds, respectively. The problems in this set exhibit varying degrees of nonconvexity. Branin's RCOS has three global minima with minimum function value 0.397887. The 2D six-hump camel back has two global minima with a minimum function value of -1.0316. The 2D Shubert has 760 local minima. Among these 760 local minima are 18 global minima giving a minimum function value of -186.73 [27]. Functions F2 and F8 both have minimum function value 0. F8 is highly multimodal and the number of extrema depends on the dimension of the problem.

We have formed the lumped transition probability matrix L and associated deleted transition probability matrix $\hat{L}$ by forming aggregated states $A_1, \ldots, A_m$, where $A_1, \ldots, A_{m-1}$ contain the $m-1$ states with the lowest function values and A_m contains all other states. We compute the eigenvalues of $\hat{L}$ to find the expected hitting time E_L using the EISPACK routines elmhes, eltran, hqr, and hqr2. The eigenvalue computation is a small cost because of the small size of the lumped transition probability matrix.

6.1. *E_L as an Estimate for E_P*

In (22) and the discussion following it, we illustrated that the error between E_L and E_P depends on the element-wise difference between $(I - \hat{U}\hat{P}\hat{V})^{-1}$ and $\hat{U}(I - \hat{P})^{-1}\hat{V}$. We then gave special cases where we can provide one-sided bounds for the error between E_L and E_P and indicated that E_L is a fair estimate of E_P, even though the requisite conditions for lumpability do not exist. Throughout the following discussion, we define g to be the state that has as its associated function value the global minimum.

In Fig. 1 we plot the relative error, $|E_P - E_L|/E_P$, against neighborhood size. In this particular plot, $c = 10$ and $\hat{P} \in \mathbb{R}^{100\times100}$ is lumped to $\hat{L} \in \mathbb{R}^{10\times10}$. We choose to plot a relatively small lump size here. As one would expect, the error decreases as the lump size increases. From the plot, we see that the error generally decreases with increasing neighborhood size. Of course, when the neighborhood size is full, regardless of the lump size, no error exists.

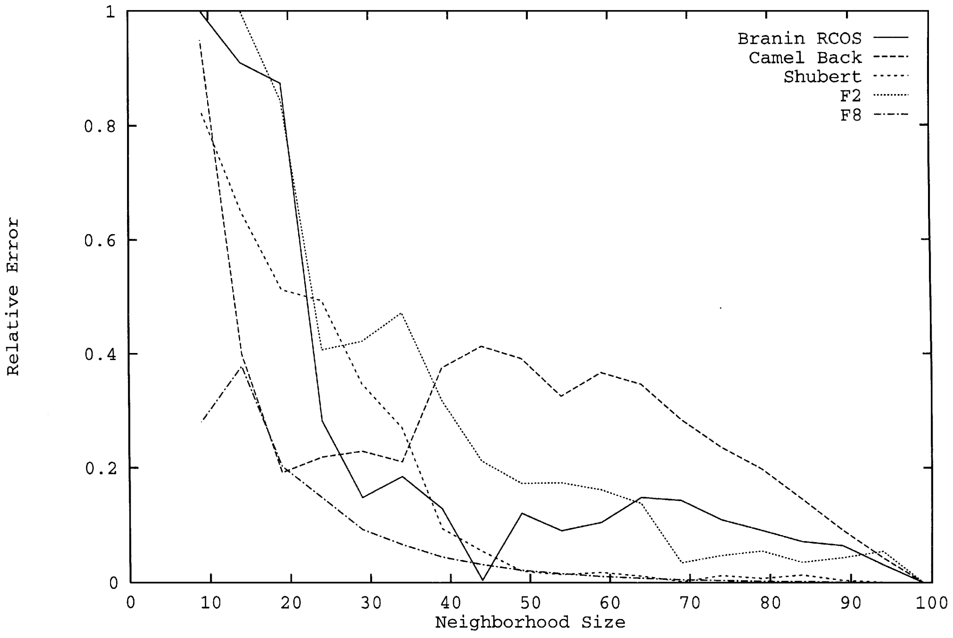

FIG. 1. Relative error of E_P, E_L vs neighborhood size, $c = 10$.

In Fig. 2 we have plotted the actual values for E_P and E_L as a function of neighborhood size for the 2D Shubert function. Here, we have held c fixed at 10. Evidence of the disparity between E_P and E_L is apparent, especially for small neighborhood sizes. The value of c in Fig. 2 gives a large expected hitting time. We see in Fig. 1 that the relative error between E_P and E_L for large N is generally largest for the 2D Shubert function.

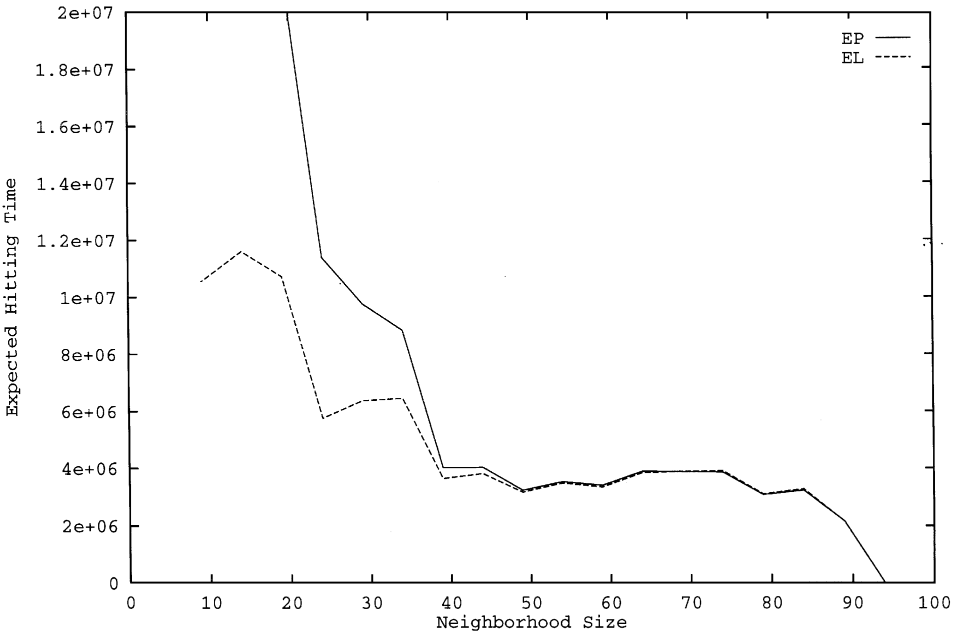

FIG. 2. E_P, E_L vs neighborhood size, 2D Shubert, $c = 10$.

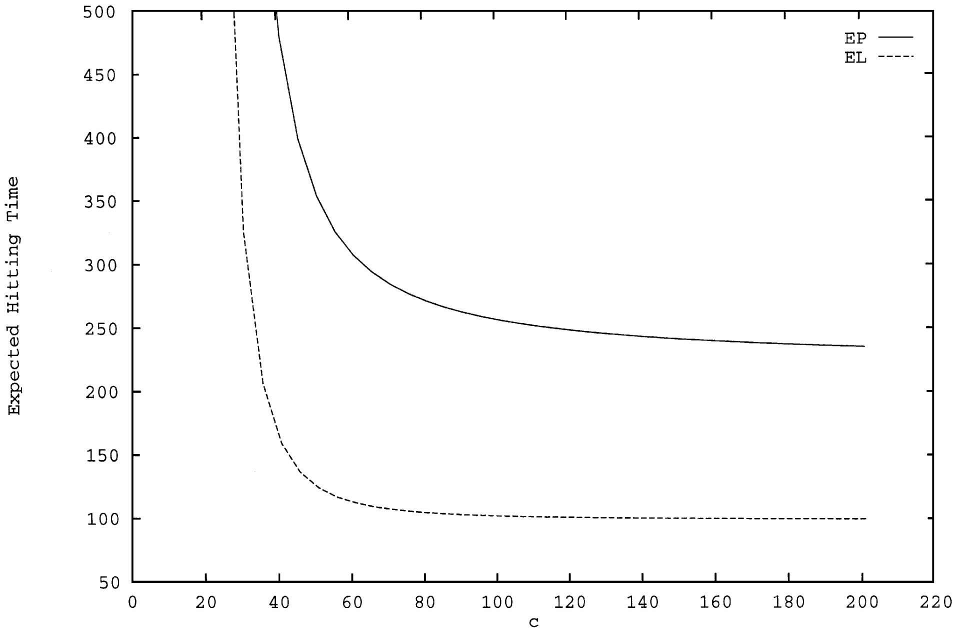

FIG. 3. E_P, E_L vs c, 2D Shubert, $N = 20$.

Figure 3 is similar to Fig. 2, except that we plot E_P and E_L as a function of c while fixing N at 20. The difference in these plots results only from the lumping procedure and the error is relatively constant for most neighborhood sizes. This result is not unexpected. The forms of each of the matrices have not changed. The zeros are in the same places in each of the matrices, and the minimum number of transitions required to get to the global minimum is the same. As c decreases, E_P and E_L increase since uphill transitions occur at a smaller rate. E_L is lower here and is in all cases, because there is always a positive probability of reaching g from m in $\hat{L}$. We see from Fig. 1 that for $N = 20$ there is large relative error between E_P and E_L for all the functions.

We implemented the algorithm on each of the test problems. Our approach was to run the algorithm first with either c or N held fixed while we allowed the algorithm to adjust the unfixed parameter and then to allow the algorithm to run, allowing both parameters to vary. In our case, keeping c fixed means keeping c to a fixed schedule and not allowing the algorithm to affect its value. All of our comparisons are in terms of independent trials with each trial over a fixed number of iterations (i.e., a fixed number of function evaluations). We do, however, neglect in our comparisons the overhead involved in computing the diagnostic. This involves constructing the relevant deleted lumped transition matrices and solving small eigenvalue problems, so the cost can be made minimal and could be accomplished in parallel by a processor that is devoted to the task of computing the diagnostic.

6.2. *The Algorithm with Fixed c or N*

We chose as the cooling schedule the standard logarithmic schedule that can be found in [27], namely,

$$c(k) = \frac{C}{\ln(1 + k)} \tag{33}$$

TABLE I
Test Problem Results with *c* Fixed, *N* Variable (10^6 Iterations)

	f from SA			f from algorithm		
Function	Avg	Min	Max	Avg	Min	Max
1	0.397887	0.397887	0.397888	0.397887	0.397887	0.397888
2	−0.93369	−1.03163	−0.21546	−1.0153	−1.03163	−0.21546
3	−32.564	−186.731	−7.723	−40.2522	−186.731	−7.72788
4	5.17×10^{-7}	0.0	1.65×10^{-6}	3.87×10^{-7}	0.0	1.44×10^{-6}
5	0.40768	0.043671	0.49968	0.18971	6.62×10^{-4}	0.442817

with $C = 1$. For each of the tests in this section, we randomly generated 50 starting points, and then for each of the starting points ran the algorithm for 10^6 iterations. We discretized the domain with a mesh size of 10^{-4}. For the five problems with this mesh size, the state spaces range in size from about 4×10^8 to 2×10^{10} so that between 0.01% and 1% of the state space is potentially visited during the search. We started the search with $c = 10.0$ and $N = 0.15$, the initial c and neighborhood size used in [27], and at the completion of $k = 250$ iterations, we allowed either c or N to vary 1%, depending on which parameter we held fixed.

For the first test, we used the logarithmic schedule in (33), that is, the algorithm did not adjust c, but did adjust N. Table I gives the average, minimum, and maximum objective function values obtained for the fifty trials as well as the corresponding values for the annealing algorithm run according to the logarithmic schedule with no change in the parameter N (typical SA). In each of the five cases, the average minimum found for the fifty trials using the algorithm is as good as or better than that found by SA.

Table II gives the results of the run with N held fixed at 0.15 and with c varied by the algorithm. As in Table I, we used $K = 10^6$ and $k = 250$. The results here are similar to those for the fixed neighborhood size, except that here the average function value obtained from SA for function 5 is lower than that obtained using the algorithm. Unlike SA, however, the algorithm found the minimum function value for one of the trials.

6.3. *The Algorithm with Variables c and N*

We next ran the algorithm allowing both c and N to vary. For this test, we again started with $c = 10$ and $N = 0.15$, but here, we performed 99 trials. Table III gives the results for

TABLE II
Test Problem Results with *N* Fixed, *c* Variable (10^6 Iterations)

	f from SA			f from algorithm		
Function	Avg	Min	Max	Avg	Min	Max
1	0.397887	0.397887	0.397888	0.397887	0.397887	0.397888
2	−0.93369	−1.03163	−0.21546	−0.95001	−1.03163	−0.21546
3	−32.564	−186.731	−7.723	−71.9194	−186.731	−38.296
4	5.17×10^{-7}	0.0	1.65×10^{-6}	1.67×10^{-7}	0.0	1.09×10^{-6}
5.	0.40768	0.043671	0.49968	0.4494	0.0	0.49999

TABLE III
Test Problems Run with 10^6 Iterations; *c*, *N* Variable

	f from SA			f from algorithm		
Function	Avg	Min	Max	Avg	Min	Max
1	0.397887	0.397887	0.397888	0.397887	0.397887	0.397888
2	−0.95743	−1.03163	−0.21546	−1.03163	−1.03163	−1.03163
3	−31.665	−186.731	−7.723	−67.286	−186.731	−38.286
4	5.06×10^{-7}	0.0	1.65×10^{-6}	1.95×10^{-7}	0.0	1.04×10^{-6}
5	0.405776	0.043671	0.499982	0.41006	1.14×10^{-4}	0.499985

10^6 iterations for both SA using the logarithmic schedule and for the algorithm. As in the case where N was held fixed (see Table II), SA had a smaller function value average for function 5, but the algorithm found a better solution in one of the trials. In every other case, however, the algorithm performed as well as, or better than, SA in terms of averages, minimum values, and maximum values. The performance of the algorithm is especially better for function 2, where the algorithm found the minimum on every trial. It is important to note here that the choice of $k = 250$ was arbitrary, yet even with the arbitrary choice, allowing c and N to vary gave better results than typical SA.

The behavior of c and N differs depending on the function being tested. Both behave nonmonotonically, although c generally decreases as the number of iterations increases. The neighborhood size does not exhibit any general upward or downward trend, but is affected more by the type of function.

6.4. *The Algorithm on Higher Dimension Problems*

For the final set of results we compare the algorithm against typical SA on the six and ten variable Griewank function [31]. These functions are highly nonconvex. Our approach here was the same as that given in the previous three sections; i.e., we conducted 50 trials, holding either c or N fixed, and then allowed both parameters to vary. We began with an initial neighborhood size of 1 for both problems, an initial c value of 150 for the six-variable problem, and an initial c value of 40 for the 10-variable case. We conducted 50 trials for obtaining the starting values for c and chose c such that the initial value for c was the next integer value greater than any c attained for the trials. We did not investigate the reason for the disparity between the two starting values of c. In each of the first two tests (fixed c and then fixed N), we ran the algorithm a total of $K = 10^6$ iterations and allowed the parameters to adjust 1% after $k = 750$ iterations.

Table IV contains the results for the algorithm run while varying N and adhering to the logarithmic cooling schedule. In this instance, the average of the trials, the minimum found

TABLE IV
Griewank Function with *c* Fixed, *N* Variable (10^6 Iterations)

	f from SA			f from algorithm		
Variables	Avg	Min	Max	Avg	Min	Max
6	1.01197	1.00195	1.02318	0.983326	0.974810	0.990232
10	1.120159	1.043276	1.940210	1.023717	1.003709	1.045056

TABLE V
Griewank Function with *N* Fixed, *c* Variable (10^6 Iterations)

	f from SA			f from algorithm		
Variables	Avg	Min	Max	Avg	Min	Max
6	1.01197	1.00195	1.02318	0.969481	0.964264	0.971445
10	1.120159	1.043276	1.940210	1.002638	1.000173	1.00413

over the trials, and the maximum found over the trials are all lower for the algorithm than for typical SA. For the 50 trials represented in Table IV the neighborhood sizes ranged from the starting value of 1.0 up to a maximum of 3.04.

Table V gives the results of the algorithm run while varying c and holding N fixed. Here again, each of the quantities is lower for the algorithm than it is for SA. For these trials, c had as its maximum its starting value of either 150 for the six-variable case or 40 for the 10-variable case, and a minimum of 0.01 in both cases. The results shown in Tables IV and V indicate that varying only one of the parameters while holding the other fixed leads to better solutions to large problems than does typical SA.

Table VI gives the results for the case where the algorithm adjusts both c and N. The average value obtained for both the 6- and 10-variable cases are lower than for SA, the algorithm with c fixed, and the algorithm with N fixed.

7. CONCLUSIONS

We have shown that dynamic adjustment of the SA parameters c and N can result in obtaining better solutions than using a typical SA-cooling schedule with a fixed neighborhood size. We have used a lumped form of the underlying transition probability matrix to determine these parameters and then adjusted the parameters after evaluating the expected hitting time. This approach provides an advantage over typical SA in that it maintains some information of the problem in the form of the lumped matrix. To implement our technique fully, several other areas require further study. We arbitrarily select the starting neighborhood size, N, the number of iterations k between the adjustment of the parameters, and the values for the incremental changes to c and N. An analytical approach to selecting these parameters could lead to an acceleration of the attainment of the global minima. We do not discuss stopping criteria but, instead, demonstrate that, over a fixed number of iterations, the algorithm obtains better solutions on average than SA.

TABLE VI
Griewank Function with *c*, *N* Variable (10^6 Iterations)

	f from SA			f from algorithm		
Variables	Avg	Min	Max	Avg	Min	Max
6	1.01197	1.00195	1.02318	0.969295	0.965595	0.972565
10	1.120159	1.043276	1.940210	1.002377	1.000672	1.00413

APPENDIX

1. Branin RCOS,

$$f(x_1, x_2) = a\left(x_2 - bx_1^2 + cx_1 - d\right)^2 + e(1 - f)\cos(x_1) + e,$$

where $-5 \le x_1 \le 10$, $0 \le x_2 \le 15$, $a = 1$, $b = 5.1/(4\pi)^2$, $c = 5/\pi$, $d = 6$, $e = 10$, $f = 1/(8\pi)$.

2. 2D six-hump camel back,

$$f(x_1, x_2) = \left(4 - 2.1x_1^2 + x_1^4/3\right)x_1^2 + x_1x_2 + \left(-4 + 4x_2^2\right)x_2^2,$$

where $-3 \le x_1 \le 3$, $-2 \le x_2 \le 2$.

3. 2D Shubert,

$$f(x_1, x_2) = \sum_{i=1}^{5} i\cos[(i+1)x_1 + i] \sum_{i=1}^{5} i\cos[(i+1)x_2 + i],$$

where $-10 \le x_1 \le 10$, $-10 \le x_2 \le 10$.

4. F2,

$$f(x_1, x_2) = 100\left(x_1^2 - x_2\right)^2 + (1 - x_1)^2,$$

where $-2.048 \le x_i \le 2.047$.

5. F8 (Griewank),

$$g(\mathbf{x}) = 1 + \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{N} \cos\left(\frac{x_i}{\sqrt{i}}\right),$$

where $-512 \le x_i \le 511$.

ACKNOWLEDGMENTS

We are grateful to the referees for helpful remarks on an earlier version of this paper. This work was partially supported by the sponsors of the Consortium Project on Seismic Inverse Methods for Complex Structures at the Center for Wave Phenomena and the National Science Foundation under Grant DMS-9506603. In addition, E.V.V. was partially supported by the National Science Foundation under Grant DMS-9505049.

REFERENCES

1. E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines* (Wiley, New York, 1989).
2. A. Abdel-Moneim and F. Leysieffer, Weak lumpability in finite Markov chains, *J. Appl. Prob.* **19**, 685 (1982).
3. B. Andresen and J. M. Gordon, Constant thermodynamic speed for minimizing entropy production in thermodynamic processes and simulated annealing, *Phys. Rev. E* **50**, 4346 (1994).
4. B. Andersen, K. H. Hoffmann, K. Mosegaard, J. Nulton, J. M. Pedersen, and P. Salamon, On lumped models for thermodynamic properties of simulated annealing problems, *J. Phys. France* **49**, 1485 (1988).
5. A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences* (SIAM, Philadelphia, 1994).
6. P. Buchholz, Exact and ordinary lumpability in finite Markov chains, *J. Appl. Prob.* **31**, 59 (1994).

7. K. M. Cheh, J. Goldberg, and R. Askin, A note on the effect of neighborhood structure in simulated annealing, *Comp. Ops. Res.* **18**, 537 (1991).
8. L. Eisner, I. Koltracht, M. Neumann, and D. Xiao, On accurate computations of the Perron root, *SIAM J. Matrix Anal. Appl.* **14**, 456 (1993).
9. W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1, 3rd ed., Wiley, New York, 1968.
10. B. Gidas, Nonstationary Markov chains and convergence of the annealing algorithm, *J. Stat. Phys.* **39**, 73 (1985).
11. L. Goldstein and M. Waterman, Neighborhood size in the simulated annealing algorithm, *Amer. J. Math. Management Sci.* **8**, 409 (1988).
12. O. Gross and U. G. Rothblum, Approximations of the spectral radius, corresponding eigenvector, and second largest modulus of an eigenvalue for square, nonnegative, irreducible matrices, *SIAM J. Matrix Anal. Appl.* **14**, 15 (1993).
13. B. Hajek, Cooling schedules for optimal annealing, *Math. Oper. Res.* **13**, 311 (1988).
14. D. L. Isaacson and R. W. Madsen, *Markov Chains: Theory and Applications* (Krieger, Melbourne, FL, 1985).
15. J. G. Kemeny and J. L. Snell, *Finite Markov Chains* (Van Nostrand, New York, 1960).
16. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Optimization by simulated annealing, *Science* **220**, 671 (1983).
17. J. J. McDonald, M. Neumann, H. Schneider, and M. J. Tsatsomeros, Inverse M-matrix inequalities and generalized ultrametric matrices, *Lin. Alg. App.* **220**, 321 (1995).
18. N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, Equations of state calculations by fast computing machines, *J. Chem. Phys.* **21**, 1087 (1953).
19. K. Mosegaard, and P. D. Vestergaard, A simulated annealing approach to seismic model optimization with sparse prior information, *Geophysical Prospecting* **39**, 599 (1991).
20. J. D. Nulton and P. Salamon, Statistical mechanics of combinatorial optimization, *Phys. Rev. A* **37**, 1351 (1988).
21. D. H. Rothman, Nonlinear inversion, statistical mechanics, and residual statics estimation, *Geophysics* **50**, 2784 (1985).
22. D. H. Rothman, Automatic estimation of large residual statics corrections, *Geophysics* **51**, 332 (1986).
23. D. F. Rogers, R. D. Plante, R. T. Wong, and J. R. Evans, Aggregation and disaggregation techniques and methodology in optimization, *Operations Research* **39**, 553 (1991).
24. J. A. Scales, M. L. Smith, and T. L. Fischer, Global optimization methods for multimodal inverse problems, *Journal of Computational Physics* **103**, 258 (1992).
25. E. Seneta, *Non-negative Matrices and Markov Chains*, 2nd ed. (Springer-Verlag, New York, 1981).
26. R. Shonkwiler and E. S. Van Vleck, Parallel speed-up of Monte Carlo methods for global optimization, *J. Complexity* **10**, 64 (1994).
27. B. E. Stuckman and E. E. Easom, A comparison of Bayesian/sampling global optimization techniques, *IEEE Trans. Syst. Man Cybern.* **22**, 1024 (1992).
28. J. Szu and R. Hartley, Fast simulated annealing, *Phys. Lett. A* **122**, 157 (1987).
29. R. S. Varga, *Matrix Iterative Analysis* (Prentice-Hall, Englewood Cliffs, NJ, 1962).
30. P. D. Vestergaard and K. Mosegaard, Inversion of post-stack seismic data using simulated annealing, *Geophysical Prospecting* **39**, 613 (1991).
31. D. Whitley, K. Mathias, S. Rana, and J. Dzubera, Building better test functions, in *Proc. Sixth Int. Conf. Gen. Alg., Univ. of Pittsburgh, 1995,* p. 239.
32. J. H. Wilkinson, *The Algebraic Eigenvalue Problem* (Oxford Univ. Press, London/New York, 1965).
33. X. Yao, Simulated annealing with extended neighbourhood, *Int. J. Comp. Math.* **40**, 169 (1991).
34. X. Yao, Comparison of different neighbourhood sizes in simulated annealing, in *Proc. Fourth Aust. Conf. Neural Net., 1993,* p. 216.