DRAFT: September 3, 1994

SIMULATED ANNEALING: FOLKLORE, FACTS, AND DIRECTIONS

Bennett L. Fox
Mathematics Department
Campus Box 170
University of Colorado
P.O. Box 173364
Denver, CO 80217-3364

E-Mail: bfox@cudenver.bitnet

ABSTRACT:

Simulated annealing is a (sometimes-maligned) probabilistic search method used in global optimization. Folklore has it that simulated annealing must use static (generally-large) neighborhoods, that it must blindly propose moves independently of objective-function values, that it must waste a lot of (computer) time rejecting moves, that tabu search and genetic algorithms as well as random restarting and deterministic methods are disjoint from simulated annealing, and that the canonical cooling schedule is worthwhile. All this is false. After exploding these myths, the role of search diversification in eliminating pathologies and in setting the stage for guaranteed linear speedup of parallel search is pointed out. Next we sketch a way to use low-discrepancy point sets in a preprocessor. Finally, we outline methods to handle noisy objective functions and argue that convergence results for simulated annealing have strong practical relevance in that setting.

1. INTRODUCTION

Simulated annealing generally is implemented naively. When it is, many of the criticisms of simulated annealing are valid. Here we outline a sophisticated implementation of simulated annealing and argue that the negative folklore about the naive version becomes irrelevant.
We also point out directions for research.

1.1 THE PROBLEM

We seek to minimize {c(x): x in S}.
The objective function c is real-valued, but the set S can be quite general. Most often, S is either finite or a compact subset of finite-dimensional Euclidean space.

1.2 WHAT THIS PAPER DOES

Both state spaces mentioned above are treated

here. Section 5.4.4 discusses a hybrid state space. Much of what we say applies to general state spaces, including our discussion of sophisticated simulated annealing in section 2. Speedup on parallel processors in discussed in section 3.2.
We assume that c is known everywhere on S, except in section 5. There we assume that c(x) can be estimated dynamically by observing certain random variables, say by Monte Carlo simulation. For S finite, a straightforward simulated-annealing algorithm discussed in sections 5.1 and 5.2.2 converges when the range of c is restricted to a finite set, a condition which we think is weaker than others which have been proposed. We adapt our results to get asymptotically-optimal work-normalized estimators in Section 5.4.3. For S infinite, we propose new simulated-annealing algorithms in sections 4 and 5.3. These handle noiseless and noisy observations, respectively. They use the framework of section 2, with c evaluated on a low-discrepancy point set as a preprocessor.
No prior knowledge of Quasi-Monte Carlo is needed to understand our description of this preprocessor.
Thus, simulated annealing globally optimizes over strategies whose respective performances are simulated by Monte Carlo as well as in problems where all data are known.
Our methods for the former are direct counterparts of our methods for the latter.
This paper is accessible to those unfamiliar with simulated annealing. For those unfamiliar with Quasi-Monte Carlo, we try to convey some of its flavor in section 2.3.4 and cite references for more detail in section 4.1.

1.3 NAIVE SIMULATED ANNEALING

In the naive version:

* there is a one-to-one correspondence between elements of S and states
* the neighborhoods are static and, generally, large
* moves are proposed blindly, independently of objective-function values
* many proposed moves are explicitly rejected
* there is no preprocessing.

None of these restrictions are necessary, as the next section details. The (standard) terms neighborhood and proposed are defined in the algorithm SLOW below. No preprocessing means that the initial state is chosen arbitrarily.
Let t(k) be the temperature at time k. The sequence {t(i): i = 1,2,...} is called the cooling schedule. When at state x at time k, the naive move mechanism is

```
SLOW(x,k)
  Until exit, repeat
    Set k <-- k + 1              {update simulated time}
    Generate a proposed move to a state y in the
```

```
            neighborhood N(x) of x, according to some
            distribution (generally, the uniform
            distribution over N(x)).  {tentative move}
         If y is downhill from x relative to c, then
                Set x <-- y           {accept the move}
                Exit
            Else
                Generate a uniform random number U.
                If U < exp{[c(x)-c(y)]/t(k-1)}, then
                     Set x <-- y      {accept the move}
                     Exit
      End.
```

The explicit input parameters (x,k) are respectively the current state and (simulated) time. Implicit inputs are c( ), t( ), N( ), and the tentative-move probabilities. The neighborhood N(x) of x is often specified implicitly; for example, it could consist of those y accessible from x by a fixed number of iterations of a given interchange heuristic. At exit, the generic ouptput state is y with time updated as indicated. The acceptance probability is the right side of the inequality above, when y is uphill from x. Any iteration of the until-repeat loop in which rejection occurs (i.e., no exit) is part of an explicit self-loop sequence of the form x --> x --> ... --> x.
When some state has all its neighbors strictly uphill and the temperatures converge to zero, the naive method is not practical because of increasingly-long self loops -- though many papers study exactly that situation; likewise, when (if S is not discrete) near local minimizers the probability that the proposed move is strictly uphill is near one.
In the next section, we greatly improve this scheme.

2. SOPHISTICATED SIMULATED ANNEALING

In sophisticated simulated annealing:

* the neighborhoods are dynamic and of modest size, while guaranteeing in a problem-independent way irreducibility.

* the original chain X(k) is replaced by a pruned chain Y(k) = X(J(k)), where the J(k)'s are the successive jump epochs between pairwise distinct states [i.e., X(i) is constant for i = J(k),J(k)+1,..., J(k+1)-1].

* now a state is a collection of feasible solutions, together with respective recent histories (temporal memory) in the pruned chain.

* we add a nonnegative penalty function to c to account for recent history, for example to inhibit revisiting a recently-visited state when S is finite or a recently-visited region of S when S is infinite, producing an intermediate pseudo-objective function c'.

* the final pseudo-objective function c* for a state is now the minimum of the c'-values corresponding to the feasible solutions (with their respective histories) it contains.

* when at state x at time k (i.e., simulated time in the original chain), the proposed move depends on {c*(y): y in F(x,k)}, where F(x,k) is the current neighborhood of x and remains unchanged during (implicit) self loops (of the form x --> x --> ... --> x in in simulated time).

Among these features, there is significant synergy.
This will become apparent below.
The penalty function above can also reflect ranking of the importance of the variables and the respective directions for altering their values.
Often, F(x,k) is a superset of a nominal neighborhood much smaller than (and generally contained in) the naive neighborhood N(x).
The additional elements of F(x,k) are generated randomly from the entire state space. Normally, we choose F(x,k) much smaller than N(x). Proposed moves can depend on aspects of history not reflected in c*, as section 4.2 illustrates. Making proposed moves depend on objective-function values and on recent history is taken for granted by the tabu-search community. That explains, in part, why tabu search empirically beats naive simulated annealing. How it fares against sophisticated simulated annealing remains to be seen.

2.1 MACROSTATES AND LOOK-AHEAD

There is a mapping from from our generic (macro)state v to S that lets us recover the corresponding best feasible solution, with respect to c', "imbedded" in v. Among the components of v, it picks out the one with minimum c'-value, breaking ties arbitrarily. Under weak conditions, the minimum of all possible c'-values equals min{c(x): x in S}. See Fox (1993) for this and the link to tabu search and genetic algorithms, via the perspective of probabilistic search schemes as Markov chains on state spaces rich enough to accommodate ideas previously seen as disparate. As Fox (1993) points out, the search is much more than a concatenation of individual searches corresponding to the respective solutions in v. For example, crossover (in the genetic-algorithm sense) can be used to generate candidates for F(x,k).
Another reason to have more than one feasible solution imbedded in v is to profit more from a random-restarting preprocessor, as section 2.3.1 explains.

Making the size of F(x,k) small makes intelligent tentative moves practical. One feature of such smart moves is that the tentative-move probabilities depend on all c*-values in F(x,k). A second feature is (limited) look-ahead as follows. The distribution used to generate (some) elements of F(x,k) can be implicit. For example, after generating an element u uniformly from S, a descent path from u can be followed to pick a u' for F(x,k) or, if c is differentiable, a search can be started from u for an approximate stationary point u' of c. Section 4 refines the latter idea. For the former, the number of descent steps (each chosen from a small set of local candidates) allowed can be decreasing function of c(u) -- on the principle that the amount of look-ahead should be higher for better u's. If u is significantly above x, perhaps there should be no look-ahead for that u unless all (tentative) elements of F(x,k) are strictly uphill from x. It probably makes sense to impose an upper bound on the overall number of descent steps from all the u's generated. There is no requirement that u be picked uniformly. Instead, its distribution may make it likely that it is picked from what might have been the neighborhood of x with naive simulated annealing.

Look-ahead meshes well with having multiple solutions in a state, because look-ahead from the current state can supply several solutions for the next state. Thus, because of the spatial memory thereby induced in macrostates, no solution found by look-ahead needs to be forgotten. Likewise, memory of selected x in S in valleys traversed (at least the deeper ones) lingers.

Another ingredient of smart simulated annealing is search diversification. One aspect comes from the multiple solutions in each state. Enforcing diversification requires that no two of these are in the same "region" of S. Fox (1993) gives a specific method to achieve such spacing. Ideally, each solution in a given state should be in a distinct basin of attraction -- ignoring minor, local roughness. A second aspect enriches F(x,k) with elements of S that can be far from x. When S is finite, each element of S has a positive probability to enrich F(x,k). When S is the unit cube, the density for putting elements in F(x,k) is bounded away from zero.

There are memory-work tradeoffs in deciding how much recent history to record. Generally, one explictly records only the (dynamically-defined) oldest solution in this history with more recent history implicitly defined by recording only the necessary updates. This saves memory and work directly and makes it easier to check whether a proposed move would be to a recently-visited state, at least when that move would be to a "nearby" state. Thus, though F(x,k) can potentially contain elements far from x, one should arrange that the number of such elements is likely to be small when there is significantly more work or memory associated with processing those elements. No memory problem arises in section 4.2, where S is the unit cube. Instead of recording recently-visited states, one merely records recently-visited boxes in a certain partition of S; that is easy.

One of the reasons that N(x) is generally rather large in naive simulated annealing is to assure irreducibility. In this paragraph only, denote by E the elaborate state space incorporating temporal and spatial memory and denote by M(x) the counterpart to N(x) on E. The random elements added to (the generally much smaller) M(x) in sophisticated simulated annealing to form F(x,k) do more than assure irreducibility on E. They also assure that weak reversibility holds on E, also needed to assure the convergence properties in section 6. {Weak reversibility means that: there is a path from

x to y with maximal objective-function height h <==> there is a path from y to x with maximal height at most h.} Without precautions, generally E does not inherit weak reversibility from S. For this, the random elements we put in F(x,k) are enough.
Even if neighborhood symmetry {x in N(y) <==> y in N(x)} holds on S, generally not even weak reversibility holds on E unless the random elements are added to M(x). See Fox (1993) for a precise description of these random elements and how dynamic neighborhoods are generated using them and other candidates.

2.2 MOVE MECHANISM

2.2.1 THE ALGORITHM AND ITS MAIN SUBROUTINE

Consider a generic search algorithm of the form:

```
SEARCH(x,k)

  Generate a finite set F(x,k) of candidates for
    the next state in the pruned chain.
    {See section 2.1.}

  Optional:  if F(x,k) contains a state at least
    eta better {with respect to c} than any yet seen,
    move to it -- breaking ties arbitrarily.  [If S is
    finite, replace "at least eta" by "strictly";
    otherwise, set eta to a positive tolerance.]

  Execute QUICKER(x,k,F(x,k),j) to generate L, the
    transition number in the original chain when a
    state different from x is next visited [see below].

  Execute NEXT(x,k,F(x,k),L) to generate that state.

  End.
```

Let a(x,i) = P{X(i+1) not equal x | X(i) = x}. Given a(x,L) and the c*-values of all elements in F(x,L), implementing NEXT is easy: just generate from a discrete distribution conditioned on acceptance at L.
The work to compute a(x,i) is essentially free, given that we want smart tentative moves -- depending on all c*-values in F(x,i). In turn, such smart moves are practical because the random elements in F(x,i) make it possible to have |F(x,i)| small while assuring irreducibility. Thus, QUICKER and our dynamic neighborhoods dovetail. Fox and Heine (1994a) show, among other things, that the optional step in SEARCH preserves the convergence properties in Section 6. Some merely set L equal k+1, thus avoiding QUICKER. The work thus saved is negligible.
Such (random) slowing of the cooling is arguably bad in terms of first hitting time to an optimal state.
The non-trivial subroutine follows:

```
QUICKER(x,k,F(x,k),j)
  Set i <-- k
  Set h <-- 0              *
  Until exit, repeat
```

```
      Set  h <-- h + 1       *
      Generate a geometric r.v. G with parameter a(x,i)
              {Thus, G is the trial number of the first
              success in iid Bernoulli trials with
              success probability a(x,i).}
      Set  L <-- i + G - 1
      Generate a uniform r.v. V
      If
           V < a(x,L)/a(x,i)
                or
           h > j             *
        Then
           Exit with L
        Else
           Set i <-- L
   End.
```

The original version with j equal infinity is obtained by deleting all statements marked with an asterisk. In this case, L is stochastically the same as it would have been with the naive algorithm as Fox [(1993),(1995)] proves. Under mild conditions on the cooling schedule, Fox (1995) shows that the number of iterations of the until-repeat loop converges in quadratic mean to one as k goes to infinity, for all fixed x. A geometric variate can be generated in O(1) time, regardless of its parameter; e.g., see Bratley, Fox, and Schrage (1987), section 5.4.5. Thus each execution of the until-repeat loop requires essentially constant time, so that the computer time to generate r accepted moves is essentially proportional to r.

If nothing is excluded from F(x,k), as above, then irreducibility and weak reversibility follow immediately. When S is finite, F(x,k) is typically a superset of a fixed, nominal neighborhood N(x). With our setup, we can choose N(x) selectively without worrying about irreducibility. The latter we get from the random elements added to N(x) to form F(x,k) at step k, which depend on k and possibly on x. Tailoring the method to construct F(x,k), along with specifying the tentative-move probabilities and (tabu) penalty function, to particular problems is the art of simulated annealing. The rest of SEARCH and QUICKER is problem-independent, provided that problems are encoded in a uniform way (partly to avoid problem-dependent data structures). We see that SEARCH is a top-level procedure, with the construction of F(x,k) at the bottom. The look-ahead and diversification built in that construction strengthen the analogy with multigrid.

2.2.2 COMMENTS ON QUICKER

While QUICKER is the part of SEARCH most amenable to mathematical analysis (Fox (1995), Heine (1994a), and the remarks below), we think that its combination with the ideas set forth at the outset of section 2 and in section 2.1 is what makes our overall approach so attractive. As presented, QUICKER assumes that the temperatures are nonincreasing. While that condition could be relaxed, the corresponding modification of QUICKER slows it -- generally significantly.

Heine (1994a) shows how to

choose j finite, depending on the cooling schedule and neighborhood structure, while guaranteeing certain convergence properties of the original and pruned chains, respectively. Section 6 briefly discusses these properties.
If the elaborate state space is finite and every element of it has a positve probability of being in F(x,k) and if a "subcanonical" cooling schedule (defined below) is used, then we can choose j equal one. If S is a compact subset of finite-dimensional Euclidean space and the density for choosing elements of F(x,k) is bounded away from zero, then we can choose j equal one; results of Belisle (1992) extend if F(x,k) is forced to contain an element not uphill from x. If $|F(x,k)|$ reaches a certain threshold with no such element, this can be done by putting x in F(x,k) with corresponding small tentative-move probability. In that case, the speedup from QUICKER is significant.
If nothing is excluded from F(x,k), as above, then irreducibility and weak reversibility follow immediately.

Heine (1994b) sketches Richard Holley's proof (communicated personally) that is never necessary to take j greater than 2, regardless of the contents of F(x,k) relative to x. Heine also gives an example where taking j equal 2 is needed to assure convergence in probability to the set of optimal states. He conjectures that j equal 2 is always necessary if every state is visited infinitely often and there is a state with all neighbors strictly uphill.

Without QUICKER, simulated annealing would always get beat by random restarting in the sense that, for all k large enough, the probability of visiting an optimal state by (simulated) time k is at least as large using random restarting. This is not true when using QUICKER and counting only accepted moves, as Fox (1994) shows by example. It is true that enumeration is unbeatable according to this criterion, but generally enumeration is absurd whereas random restarting and simulated annealing each have a reasonable chance to succeed.

We do not know whether the final L output from QUICKER can be replaced by its expectation, rounded up, and still have all the convergence properties in section 6 hold. If L can be replaced by a random variable with the same expectation, then the final L can be replaced by its conditional expectation, rounded up, given the respective parameters of the geometric variates generated during the execution of QUICKER. This inhibits (anomalous) rapid cooling in the pruned chain. Using expectations this way is called (extended) conditional Monte Carlo in the simulation literature; e.g., see Bratley, Fox, and Schrage (1987), section 2.6.

2.3 PREPROCESSORS

We view preprocessors as a key part of our approach, not a frill. They generate the initial state X(0), intelligently. This initializes SEARCH with a pair (X(0),0), which we write simply as (x,0). With all the preprocessors, except the one described in section 2.3.1, their effects persist throughout the entire simulated-annealing process.
In a well-designed implementation, there is a problem-dependent

(pre)pre/postprocessor that converts a given problem formulation to some problem-independent format and feeds it to a preprocessor below. When annealing proper terminates, its output is converted back to a format easily understood by the user.

2.3.1 RANDOM RESTARTING

No matter what S is, we can use random restarting in tandem with descent to generate an initial state for (smart) simulated annealing proper. Run this pair thousands of times, stratifying S as in Fox (1993). Choose the best solution from each stratum to go into an initial state, subject to the following constraint on its size: no two solutions in it are close to each other in a suitable metric. Since this state contains multiple feasible solutions, the work for random restarting is thereby amortized. The heuristic reasoning for this step is that, because simulated annealing has signficant overhead and can make uphill moves, the initial state should have low height above the minimal value of c. Under mild conditions, the (ordinal) rank of the c*-value of the starting state puts it close to that of the best states. This holds even without the descent phase of each random restart. The restricted, local neighborhoods used here can be the same as those used for look-ahead.

2.3.2 BRANCH AND CUT OR LAGRANGIAN RELAXATION

Branch-and-cut algorithms or Lagrangian relaxation (e.g., see Ahuja, Magnanti, and Orlin (1993), chapter 16, or Nemhauser and Wolsey (1988), section II.3.6) can generate lower bounds on c (hence, a stopping criterion), suggest a starting solution (perhaps via rounding), and indicate qualitatively good proposed moves. In some settings, it may pay to use simulated annealing to generate good cuts via maximizing a dual objective function. It is also possible that simulated annealing may be a good method to use on subproblems generated by Lagrangian relaxation (say, when the running time of deterministic algorithms for these subproblems is not polynomially bounded).
This reduces the number of ad hoc subroutines needed for probabilistic search. Instead, one writes subroutines for frequently-occurring subproblems and lets Lagrangian relaxation deal with troublesome side constraints. One could turn this procedure around, by using probabilistic search to find feasible solutions for various branch-and-bound subtrees. Still, we think, probabilistic search (as a stand-alone method or part of a hybrid) is viable only on structured problems where it is feasible to construct intelligent neighborhoods.

2.3.3 GRADIENT-BASED POINTS

When S is infinite and algorithms are at hand that converge to a stationary (though possibly not optimal) point relative to c, they can be stopped short of convergence and the corresponding point(s) can be added to the initial state.

Alternatively and perhaps better, such points can be generated within the simulated annealing algorithm proper as section 4 details.

2.3.4 LOW-DISPERSION POINT SETS

Suppose that S is is a compact subset of finite-dimensional Euclidean space. Most people unfamiliar with Quasi-Monte Carlo do not realize that initially sampling c over a grid imbedded in S is a bad choice. If c depends strongly on only a few variables, then clearly most grid points are wasted. Even if c depends strongly on all its variables, grids leave unnecessarily large "holes" in S. See Bratley, Fox, and Schrage (1987), problem 6.2.2, for example. Likewise, a point set selected using coordinates that try to mimic genuinely independent and uniform random numbers leaves unnecessarily large holes. More precisely, low-dispersion sets D have low values of sup {min {dist(x,y): y in D}: x in S}.

At least when S has modest dimension, a low-dispersion point set D is a good choice. It can form the core of a preprocessor as section 4 details. If c is evaluated on D and an upper bound on its modulus of continuity or on a Lipschitz constant is known, lower bounds on c over various regions of S follow. This induces a termination criterion. Without such lower bounds, the problem is arguably ill-posed. One has to rule out pathologies such as a flat c except for a downward spike. Likewise, if S is unbounded, the problem is ill-posed unless every local minimizer is a global minimizer (and the already-mentioned provisos hold).

2.3.5 MARKOV-CHAIN MONTE CARLO

When S is infinite and there is no simple mapping M from the unit cube to S, the only practical procedure seems to be based on Markov-chain Monte Carlo. The latter constructs a Markov chain whose stationary distribution is the desired sampling distribution over S. For example, see Bratley, Fox, and Schrage (1987), problem 1.9.16, for the case where S is a polytope defined by linear inequalities and the target distribution is uniform. In such cases, especially if the dimension of S is more than a dozen say, only a crude partitioning (stratification) of S seems practical. Without a convenient M, using low-discrepancy point sets seems out of the question. Markov-chain Monte Carlo (MCMC) can be combined with random restarting as well as used in simulated annealing proper. To recast the latter combination in standard simulated-annealing format, we use (macro)states of the form (x,y) where x corresponds to c* and y is the state of the MCMC subroutine.

If MCMC is used in simulated annealing proper, then look-ahead (via gradients say) generally is essential. Otherwise, the rate of convergence typically will be very slow. This bad behavior would be aggravated if SLOW were used, because SLOW gets slower as the probability of a tentative uphill move increases.

3. DIVERSIFICATION

We show by examples pathologies that can occur when "long" jumps across S are forbidden. Next, we indicate that, when such jumps are allowed, linear speedup occurs when independent copies of the pruned chain are run on respective parallel processors.

3.1 PATHOLOGIES

Neighborhoods which are purely "local" lead to myopic search. To see what can go wrong when F(x,k) does not (potentially) contain elements from diverse parts of S, consider the following four-state example using the naive state space with static, local neighborhoods:

EXAMPLE 1 (Fox (1995)):

state	c	neighborhood
w	2	x
x	1	w,y
y	3	x,z
z	0	y

For a fixed starting state, let N be the time to first hit z, relative to the pruned chain.
With any cooling schedule where the temperatures converge to zero, it is easy to check that because of likely long sequences of x <--> w oscillations:

$$-k + E[N \mid N > k] \longrightarrow \text{infinity}$$

as k --> infinity. Thus, roughly, the longer the search has been unsuccessful, the longer the expected remaining time to hit the sole globally-optimal state z.

Similar pathologies can be constructed for the more elaborate state spaces, incorporating temporal and spatial memory, indicated above. Such examples are by no means contrived. They scale up to huge state spaces, where the search is likely to be unsuccessful for a long time and thus likely to be unsuccessful for an even much longer time. However, such examples cannot occur with our construction of F(x,k) above.

One might argue that the pathology is an artifact of letting the temperatures converge to zero. However, there are (at least) two arguments in favor of letting the temperatures converge to zero. First, this is necessary for the convergence results of Section 6. For those, cooling must not be too fast. Second, heuristically, the longer the search has gone on, the more we want to drive it downwards -- even in the short run, because our neighborhood construction prevents local "stalling". Reheating would merely stimulate a (generally slow) climb out of the local valley, whereas lowering the temperature stimulates long jumps when all local elements of F(x,k) are uphill. The latter is especially desirable, intuitively, when those elements reflect look-ahead.
Having multiple solutions in each state prevents the search from degenerating merely to random restarting when seemingly

stalled at a local minimum at very low temperatures. A balance with local search is maintained.
Even if the temperatures do not converge to zero but rather to a small positive number, the pathologies above give insight into the drawbacks of myopic search and suggest that diversification is worthwhile in that setting as well. Some implement simulated annealing with adaptive cooling in which the temperatures do not converge. In that case, the speedup result of Section 3.2 does not extend.

An even more startling pathology occurs in

EXAMPLE 2 (folklore):

> Delete state w from example 1 and use the corresponding canonical cooling schedule, where the temperature at move k in simulated time is 2/log(k+1).

Starting at state x and letting T be the first time to hit z relative to the original chain, it is routine to show that ET = infinity -- even though (by Borel-Cantelli) X(k) --> z almost surely.

Subcanonical schedules replace the numerator 2 above (in general, the maximal "depth" d* of local, nonglobal minima) by a some larger number. The speed of QUICKER decreases discontinuously at 2. If the numerator is near 2, QUICKER will have underflow problems.

3.2 SPEEDUP WITH PARALLEL PROCESSORS

Now let N(m) be the first time in the pruned chain when some optimal state is hit, looking globally across m independent copies of the chain. The initial state X(0) is the same, stochastically, in all copies. Thus, X(0) has the same distribution on all m processors but its value may differ across processors. It follows easily from results of Fox (1995) that, when

* S is finite
* our neighborhood construction is used
* certain (mild) technical conditions hold,

$$\{E[N(1) \mid N(1) > k\}/\{E[N(m) \mid N(m) > k]\} \to m,$$

i.e., the speedup is (in a well-defined sense) linear.
>From section 3.1, it is clear that (vacuously) no such speedup is possible without some sort of neighborhood diversification. It seems likely that the condition that S be finite is not essential, but a counterpart for general S would require sophisticated technical tools (though probably nothing really new conceptually).
The key to the proof in Fox (1995) is recasting, for theoretical purposes only, the dynamic neighborhoods as static neighborhoods on an expanded state space and using the spectral representation of the move matrices, conditioned on accepting all proposed moves, on this expanded state space. One of the technical conditions, alluded to above, says roughly that asymptotically the current time k is negligible relative to the expected remaining time to first hit the set S* of optimal states. A more precise statement relates k to the limiting

spectral radius of the move matrices, conditioned as above, with all rows and columns corresponding to S^* deleted. The speedup result above is obtained by letting the limiting spectral radius approach one and applying L'Hopital's rule.

An analogous result holds if we replace S^* by any nonempty set $Q(b)$ of the form $\{x: c^*(x) < b\}$. One would ordinarily choose b to make $Q(b)$ be a set of nearly-optimal and optimal states. When S is infinite, consideration of $Q(b)$ generally becomes not only practical but essential because S^* then typically has zero probability of being hit in any move. Heuristically, if

* c is smooth

* the transition kernel $K(x,A)$ is smooth at almost all x

* S is partitioned into a finite number of pieces each with a small diameter,

one can define a process on the corresponding finite-state "reduction" that is "virtually" indistinguishable from the original process -- with a corresponding implication for linear speedup.

Our work on parallel speedup was stimulated by a preprint of Shonkwiler and Van Vleck (1994). There are major differences between our work and theirs. Among these, we measure speedup relative to computer time (by considering only accepted moves, because of QUICKER's speed) but they measure it relative to simulated time -- far from the same thing.

In SIMD computers, each processor executes same instruction on any given clock tick -- but on processor-dependent data. Synchronization of the processors to be compatible with this architecture is impossible when using SLOW but is feasible when using QUICKER, especially with j equal one (legitimate with our construction of $F(x,k)$) or at worst j equal two.

For random restarting, a parallel-speedup result is given in Fox (1993), based on the Palm-Khintchine theorem. That result assumes that the number of processors gets large, but does not assume that the search has been unsuccessful for a long time; in both respects, this contrasts with what we assume for our result about parallel speedup with simulated annealing.

Laursen (1994) studies branch and bound without communication among (say) m processors. He first decomposes the original problem into m disjoint subproblems and then runs these independently on the respective processors. His measure of effectiveness is processor utilization: (apparently) the fraction of time up to solution of the original problem that all processors are busy. Processor utilization is positively correlated with speedup, but is not a well-defined function of it. Laursen tries to make each subproblem roughly equal in difficulty (as measured by computer time to solve it) via a heuristic, but there is no guarantee of success. He has encouraging empirical results for m equal 17. He has no empirical results for massively-parallel simulation (that is, a large number of processors). For that case, he proposes a method to create the m subproblems which he claims the m processors working together in a future computer architecture can do in order log m time. All this contrasts with our guaranteed linear speedup. In general, no theoretical analysis seems possible for any algorithm in which the processors are allowed to communicate.

4. USING LOW-DISPERSION POINT SETS

4.1 BACKGROUND

In this section, we sketch how low-dispersion point sets can be used in a preprocessor to simulated annealing. We freely adopt the definitions and results in Niederreiter (1992). Low-dispersion point sets on the unit cube can, in principle, be mapped into arbitrary compact (connected) subsets of Euclidean space. When S is merely a rescaling of the unit cube, the mapping is trivial. However, the general case is a hard, unsolved problem. In any case, we assume that a suitable mapping is at hand so that Theorem 6.4 in Niederreiter (1992) can be invoked. Therefore, we consider just the case where S is the unit cube. From Theorem 6.6 in Niederreiter (1992), it follows that low-discrepancy point sets have low dispersion; so we consider only the former, because good constructions for low-discrepancy point sets are known. We can use any construction which gives a (t,m,s)-net D, per definition 4.1 in Niederreiter (1992). Section 4.2.1 below summarizes the properties of such nets. Their appeal is clear without reference to dispersion or discrepancy.

Bratley, Fox, and Niederreiter (1992) point out that techniques for quickly generating base-2 Niederreiter (low-discrepancy) sequences also speed up the generation of the (low-discrepancy) base-2 (t,m,s)-nets in Hansen, Mullen, and Niederreiter (1993). See also Larcher, Lauss, Niederreiter, and Schmid (1994).

4.2 THE ALGORITHM

Define, dynamically, the attractiveness of an elementary interval (informally: a box with faces parallel to the respective faces of the unit cube) as any given decreasing function of the minimum of the c-values so far computed on that interval. The current attractiveness measure summarizes relevant history. Together with the current point in S, it forms the current state -- which can be augmented and redefined as in section 4.3.

4.2.1 PREPROCESSOR

To initialize the attractiveness measure, compute c for all points in D and then, for each elementary interval J in a class C defined below, take the minimum of those c-values for the points in the intersection of D and J. Assuming that the (t,m,s)-net D uses base b {and hence that D contains b^m points}, we choose C so that its members have the following properties:

* each has volume b^(t-m)

* each (therefore) contains b^t points of D {by definition of the net}

* the edges of the boxes parallel to (say) the i-th axis of the unit cube $[0,1]^s$ all have length $b^{d(i)}$ and partition that axis

* $d(i)$ is a nonpositive integer

* $d(1) + \ldots + d(s) = t - m$.

These properties hold for all solutions to the preceding equation. This is where the net gets its power, at least when t is small. That equation generally does not uniquely specify the d(j)'s. We get the partition C of the unit cube $[0,1]^s$ by fixing $\{d(1),...,d(s)\}$, thus making the boxes disjoint.
Perhaps it is good to take them as nearly equal as possible, getting a partition into congruent cubes if s divides $t - m$.
In any case, the i-th axis of the unit cube is partitioned into $b^{(-d(i))}$ equal-sized pieces. Thus, it is tempting to choose $|d(i)|$ as an increasing function of the a priori importance of variable i -- getting a partition into congruent non-cubic boxes.
The properties above make it easy to enumerate the $b^{(m-t)}$ boxes in the partition and to generate a variate uniformly from any given box by rescaling depending only on the s fixed values $d(1), \ldots, d(s)$.
Likewise, given any point in the unit cube, it is easy to find the box in the partition which contains it.
These last three operations can be (much) harder to execute if the unit cube is partitioned into boxes from the net which are not congruent.

4.2.2 PROVISIONAL MOVE MECHANISM

Normalize (implicitly) the attractiveness measure across all elementary intervals (boxes) in C to get a probability mass function. Next, generate a variate with that pmf (see section 4.3) and then, given the corresponding box, generate a point z uniformly from it. Optionally, look ahead: initiate a search for a stationary point of c {projecting the gradient back to the unit cube if currently at its boundary} starting from z and stopping after a finite number of iterations. Repeat this whole process a given number of times to generate F(x,k), invoking SEARCH. Right after accepting a move, update the attractiveness measure (recursively). And so on.

4.3 COMMENTS

The convergence of the algorithm follows as a special case from a theorem of Belisle (1992) -- even though he does not mention remembering history.
Thus, unlike the method of localization of search (e.g., see Niederreiter (1992), pp. 151-152), the scheme above has the global-convergence properties listed in section 6. The price is that, with low probability, substantial time is wasted exploring

uninteresting regions of S. Such wasted time is likely to be orders of magnitude larger with crude search (e.g., see Niederreiter (1992), p. 150, where it is noted that crude search generally is impractical).

There are many variations to this procedure. Some can be used to bring the provisional algorithm above into final form. For example, it is probably worth inhibiting visits to recently-visited boxes via a (tabu) penalty function on an augmented state space that remembers the recent history of the boxes visited. Another modification lets the number of iterations of the gradient-based subroutine be an increasing function of the move number in the routine calling it. With a certain probability strictly between zero and one, if that subroutine was previously used in the current box, it can be initiated from where it previously left off. Generally, this makes the probability that the proposed move is strictly uphill significantly less than one. If Lagrangian relaxation decouples subproblems, crossover can be used to try to create additional solutions to the original problem.

To generate efficiently the variate that selects the box in the partition to examine next, the algorithms of Fox (1990) and Fox and Young (1991) can be used. They exploit the fact that the respective (unnormalized) attractiveness measures at successive moves differ on at most a few boxes, though generally there is no convenient majorizing vector for a pure rejection method. Probably, the pmf used to select the next box should be updated only when the attractiveness measure changes significantly on at least one box. In any case, it should never go below a user-selected positive threshold.

Even if the number of stationary points is not large, a nontrivial global-optimization algorithm such as the one above is needed when the global minima are in small basins of attraction to gradient search that looks for local stationary points.

5. NOISY OBJECTIVE FUNCTIONS

There is a large literature on sampling plans when the observations are noisy. The bandit setup ignores possible structure of the set S and the objective function c. Roughly, that setup has N(x) = S (including x itself) for x in S and each move can depend on the entire history of the process. So far, bandit analysis (Yakowitz and Lowe (1991)] for example) looks more refined than annealing analysis. Perhaps a synthesis is possible. As with noiseless observations, simulated annealing makes sense when the state space has structure exploitable by clever selection of the dynamic neighborhoods F(x,k).

Our approach has nothing in common with the papers that study global optimization over a subset (sometimes unbounded) of Euclidean space by adding successively-more damped Gaussian noise to a stochastic gradient algorithm. That too has been called (continuous) simulated annealing. Very loosely, it does have that flavor even though it accepts all moves. However, we see no mathematical link.

Up to section 5.4.4, regardless of the cardinality of S, we assume that

* $c(x)$ is a known function of $d(x)$, say, and that $d(x)$ is the expectation of a random variable $Z(x)$

* c is unknown

* when at x, iid copies of $Z(x)$ can be generated (say, by simulation)

* a consistent estimator of $d(x)$ is available, at least when x is visited infinitely often.

The state space is expanded to keep track of enough information to update d. To have the corresponding process with hypothetically-known c live on the same state space, record the same information but ignore it.

5.1 STRAIGHTFORWARD APPROACH

No matter what the cardinality of S is, a natural scheme bases its current move on the current estimates of {c(x): x in S} -- pretending that these estimates are the respective true values.

Assuming that the estimator of c(x) is consistent for all x in a finite set T say, for any positive tau all estimates of c(x) for x in T will be within a band of width tau around the respective true values after some finite time N almost surely. However, N is not a stopping time and, without restricting the range of c to a finite set, the process never couples (exactly) to one in which c is known.

5.2 WHEN S IS FINITE

5.2.1 PREVIOUS WORK

Gelfand and Mitter (1989) and Yan and Mukai (1992) propose convergent search schemes for noisy measurements of c when S is finite, without restricting c to a finite set.

THE GELFAND-MITTER PAPER

Roughly, Gelfand and Mitter assume that

* the tentative-move probabilities are independent of objective-function values

* $c(x) - c(y)$ is measured at time k only up to an additive Gaussian noise $W(k)$ independent of x and y

* the variance of $W(k)$ has lower order than the temperature $t(k)$ at simulated time k, for large k.

While their first condition may be reasonable in some special situations, generally it goes counter to our conception of smart simulated annealing. It is unclear how to check their second two conditions. Normally, one expects that in practice the error in measuring $c(x) - c(y)$

depends on x and y. But because some states may be visited only rarely, it is then hard to get a bound on the error variance that is valid uniformly over S, unless perhaps when all unknown parameters of c can be estimated at each state. The third Gelfand-Mitter condition guarantees that, aymptotically, the respective acceptance probabilities with and without noise differ negligibly.

The algorithm of Gelfand and Mitter is (implicitly) the straightforward approach. To see this, observe that c(x) - c(y) + noise at move k is the estimated difference of c(x) and c(y) based on all observations of Z(x) and Z(y) up to move k. In turn, that estimated difference is the numerator in the exponent of the acceptance test in SLOW at move k. The rest of SLOW (and hence QUICKER) is unaffected. To analyze the resulting scheme, Gelfand and Mitter assume that the (Gaussian) noise at move k has mean zero and known variance depending on k and then compute the acceptance probability by integrating with respect to the noise distribution. The assumptions they use probably indicate that analyzing the straightforward scheme without them is hard -- at least without assuming something else.

THE YAN-MUKAI PAPER

Yan and Mukai note that their scheme has no clear link to simulated annealing. It does not estimate c(x) via averaging copies of Z(x). Every time it needs an estimate of c(x) it uses a fresh copy of Z(x), ignoring previous copies. Even if the proposed move is to a downhill state, accepting it becomes increasingly hard. The number of copies of Z(x) generated at visit j to x grows without bound as j increases. No counterpart of QUICKER is possible. They assume that the tentative-move probabilities are symmetric, running counter to what we think a good search scheme should do.

5.2.2 OUR PROPOSAL

This critique of the Gelfand-Mitter and Yan-Mukai papers motivates our use of the straightforward approach, with none of the Gelfand-Mitter hypotheses. The price is restriction of the range of c to a known finite set H say. See section 5.4.2 for a discussion of that restriction.

Without loss of generality (by rescaling, if needed), we can assume that the set H above is a set of consecutive integers and that c(x) is the nearest integer to d(x) -- rounding up in case of ties. A naive scheme rounds the average of the Z(x)'s to the nearest integer, rounding up in case of ties. This works when d(x) is not a half-integer, although it may not work well when d(x) is near a half-integer. When d(x) is a half-integer, the rounded average of the Z(x)'s will not converge, instead fluctuating between the integers neighboring d(x). Fox and Heine [(1994a),(1994b)] propose two estimators which, almost surely, converge to c(x), assuming that d(x) is estimated consistently. One is based on the law of the iterated logarithm. Proofs are in Fox and Heine (1994b).

From some time N onwards, finite a.s., the estimates of c(x) will be exactly correct for all x in S. Assuming that the convergence properties in section 6 hold starting from all initial states and times with the correct $c(x)$'s, Fox and Heine (1994a) show that they still hold with the above scheme -- even though N is not a stopping time. In fact, this follows from their much more general result about overriding strategies for Monte Carlo simulation and coupling after some finite time to a process in which no overrides occur.

The finiteness of N follows from the finiteness of S, the only place where the latter hypothesis is used.

A variation of the straightforward scheme above modifies the objective function at x while the sample variance at x exceeds a given positive threshold, making it more attractive to move to states x that have a high sample variance in their estimator of $c(x)$. Since these estimators are consistent, their sample variances converge to zero almost surely. Thus, again, we couple to a process with known c from some finite time onwards. We can likewise stimulate visits to states that have been visited less than some positive threshold.

5.3 WHEN S IS INFINITE

5.3.1 THE ALGORITHM

In this case, we mimic the procedure of section 4. We find this attractive, at least heuristically. Now gradients are estimated, say, by the likelihood-ratio method or by infinitesimal perturbation analysis (e.g., see Glasserman (1991) and L'Ecuyer [(1990),(1991)]) and stationary points of d are estimated by stochastic approximation -- generally, by some variant of Robbins-Munro. As before, we stop stochastic approximation short of convergence -- bounding the number of iterations. If it terminates at a point v in a cube V, say, belonging to the class C defined in section 4, we may choose to let the estimator of $d(v)$ depend not only on (copies of) $Z(v)$ but also, via likelihood ratios (when mutual absolute continuity and other regularity conditions hold), on all copies of $Z(w)$ for all w in the intersection of V and D. Perhaps $Z(w)$, multiplied by the appropriate likelihood ratio, should be weighted by some decreasing function of the distance between w and v, because one suspects that the corresponding variance is an increasing function of that distance.

5.3.2 ITS CONVERGENCE

We try to pattern a convergence analysis after the one we use for finite S. To start, assume that all minimizers of d occur at stationary points and (hardly without loss of generality) that the number of its stationary points is finite. The first condition can be assured by imposing a penalty function which is large in a thin layer abutting the faces of the unit cube. Since the number of cubes used in our partition of the unit cube is also finite, the number of points x where we need a consistent estimator of $d(x)$ is finite. With our algorithm, it seems likely that consistency at these points can be shown; however, we have not worked out the details. Introducing the set H as above, all induced estimates of {c(x): x is a stationary point of d} are then exactly correct from some finite time N onwards almost surely. Assume that the updates to the attractiveness measure are affected only by updates of estimates of the respective objective-function values at stationary points. This remains true if the attractiveness measure now also depends on the respective sample variances of these estimates, while those variances exceed a given positive threshold. Now the convergence of the scheme in section 4 from any starting state and time implies convergence of the algorithm in section 5.3.1 by the main result in Fox and Heine (1994a). The reduction above to considering consistency at only a finite number of points is crucial, because the results in Fox and Heine [(1994a),(1994b)] would not apply otherwise.

5.3.3 CONVERGENCE RATES

Recently, Yakowitz (1993) gave a globally-convergent extension of Kiefer-Wolfowitz stochastic approximation -- preserving the n^(1/3) convergence rate of the Cesaro average. Thus, it is an alternative to our proposal in Section 5.3.1. Because of the forced choices it makes on a deterministic subsequence of moves, that extension cannot converge in probability except in trivial cases. The Robbins-Munro stochastic-approximation subroutine of our proposal has the (better) convergence rate n^(1/2), but finding the overall convergence rate for our scheme (relative to the pruned chain) is an open problem. The convergence rate for Robbins-Munro stated above assumes that gradient estimators are unbiased. They are with the likelihood-ratio method or infinitesimal-perturbation analysis but not with finite differences. If the data come from a real system rather than a simulation, then only finite differences are available.

5.4 COMMENTS

5.4.1 CONSISTENT ESTIMATORS

For S finite, suppose that each state x is visited infinitely often. See Fox and Heine (1994a) for ways to assure this when c is dynamically estimated. When S is infinite, suppose that for every positive epsilon a ball of radius epsilon centered at x is visited infinitely often, The algorithm of section 5.3.1 assures this, when x is a stationary point of d. Information about the sample path off x or about copies of Z(y) observed for y not equal to x prior to time t generally biases any Z(x) observed before t as an estimator of d(x). One way to get a consistent estimator of d(x) ignores such information and averages Z(x) over all visits to x. Otherwise, even if consistency holds, proving it probably requires a martingale argument that considers a sum with j-th summand corresponding to the j-th visit to x. That summand is the average of the copies of Z(x) for that visit minus the conditional expectation of that average given the sigma-field generated by everything up to the j-th visit. Under weak moment conditions, consistency of the natural estimator of d(x) follows from a standard martingale convergence theorem (e.g., see Chung (1974), Theorem 9.4.4). In turn, this implies that the induced estimators of c(x) of Fox and Heine [(1994a),(1994b)] are exactly correct from some finite point onwards, almost surely. There are settings where we must or would like to take account of the additional information in the sigma-field above. This occurs when the function c depends on a set of parameters much smaller than $|S|$. It also occurs when the number of copies of Z(x) generated at the current visit to x depends on everything observed up to that point. Some restriction on that number is generally needed; a counterpart of the algorithm GENERATE in Section 5.4.3 may be appropriate. Remarkably, all our results require is that the estimator of d(x) be consistent for a suitable finite set T. When S is finite, we take T equal S. In the setting of section 5.3.2, we take T equal to the set of stationary points of d. A "visit" to a stationary point x of d must (somehow) be defined dynamically in terms of a visit of to a ball of suitably-decreasing radius centered at x.

5.4.2 THE FINITE-RANGE ASSUMPTION

The assumption that the range of c is a finite set H is key to coupling to a process with c known after some finite time. This set can consist, for example, of a subset of all computer-representable numbers, in which case it is merely a finite-precision assumption. It is actually more realistic than assuming c(x) is simply in the reals and that Z(x) is measured to infinite precision. From the viewpoint of traditional mathematics, however, it is somewhat unsatisfying. Fox and Heine (1994a) also consider an alternative to the finite-range condition, relaxing it and instead imposing a condition roughly analogous to the third Gelfand-Mitter condition and, like the latter, generally hard to check.

5.4.3 EXPERIMENTAL DESIGN

Our setup is broad. It includes cases where x indexes simulation strategies in a set S and c(x) is the variance of the limiting distribution of a work-normalized estimator, in the sense of Fox and Glynn (1990) for example, of a parameter alpha of interest. Suppose that when using simulation strategy x the work-normalized estimator (defined below) converges weakly to a normal distribution with variance c(x) and the convergence properties of section 6 hold, extended to noisy observations. For this subsection only, we generate the Z(x)'s as follows:

Let S(n) be the set of estimated optimal states at move n in the pruned chain Y, i.e., those for which the estimated c(x) is minimal. Pick epsilon as a small positive number. Let A(n) be the set of Z(x)'s generated up to move n. Let B(n) [resp., C(n)] be the sum of those Z(x)'s in the subset of A(n) corresponding to x not in [resp., in] S(n).

```
GENERATE(x,n,epsilon)

  Generate a copy of Z(Y(n)).

  If Y(n) in S(n), then generate just enough {possibly zero}
     additional iid copies of Z(Y(n)) so that

        C(n) >  n^(1/2 + epsilon).

End.
```

Now let $N(t)$ be the number of moves in the chain Y generated up to computer time t.
The conditions in the next paragraph imply that

$$B(N(t))t^{-1/2} \to 0$$

in probability as $t \to \infty$, because

* given that $Y(N(t)) \in S(N(t))$, step two of GENERATE forces the right thing to happen
* $S(n)$ contains exactly the optimal states for all large n {assuming the conditions of Fox and Heine [(1994a), 1994b)]}
* $P\{Y(n) \in S(n)\} \to 1$.

The factor $t^{-1/2}$ corresponds to work-normalization.
In general $B(N(t))t^{-1/2}$ does not converge to zero almost surely. It follows from the converging-together lemma (Billingsley (1968), p. 25, for example) that with simulated annealing the work-normalized estimator

$$\{[B(N(t)) + C(N(t))]/N(t) - \alpha\}t^{1/2}$$

$$\Rightarrow N(0,\min \{c(x): x \in S).$$

Thus, at least when S is finite, what might have been a hard convergence argument gets easily piggybacked onto our results for simulated annealing.
The additional conditions, referred to above, assumed in this subsection are that uniformly in x and ω

(i) $EZ(x) = \alpha$

(ii) $0 < k_1 < Z(x) < k_2 < \infty$

(iii) $0 < k_3 < T(x) < k_4 < \infty$,

where $T(x)$ is the (random) computer time to generate a copy of $Z(x)$. Condition (ii) implies that $B(n) + C(n)$ has order n and hence that $B(n)n^{-1/2} \to 0$ in probability as $n \to \infty$. Condition (iii) implies that $N(t)$ has order t. Hence $B(N(t))t^{-1/2} \to 0$ in probability as asserted above. Note that $c(x) = ET(x)\mathrm{Var}Z(x)$.

REMARKS. The upper bound in condition (ii) follows from the fact that computers have only a finite number of representable numbers and, more fundamentally, from the fact that virtually any model which literally allows unbounded output random variables is unrealistic. In addition, when the model must be handled numerically (as here) using unbounded output random variables makes statistical anomalies more likely. The lower bound in condition (ii) implies that O(n^(1/2 + epsilon) additional copies in step two of GENERATE are enough, while the exact order may be less. If Z(x) > k_5 > - infinity, the data can be rescaled to satisfy the lower bound in (ii). Thus, condition (ii) is reasonable. The upper bound in condition (iii) is needed to make the computer program well defined, while the lower bound is a practical necessity. From a mathematical viewpoint, these conditions probably can be weakened at the expense of a much more sophisticated mathematical argument.

We can tailor a search scheme if there is a convenient formula for arg min {c(x): x in S} when all auxiliary parameters are known. The problem is nontrivial if not all of those parameters can be observed at every x or when the variance of the estimators of those parameters depends on x. Fox and Heine (1994b) illustrate this situation in a case where arg min {c(x): x in S} is known to be the nearest positive integer to a given real-valued expression involving the auxiliary parameters, rounding up in case of ties.

5.4.4 HYBRID STATE SPACES

Another indication of the scope of our setup follows. Suppose that c(x) = min {f(x,y): y in B}. A trite example is mixed-integer programs where B and S correspond to the continuous and discrete variables, respectively. A more interesting example occurs when finding c(x) is a nontrivial global-optimization problem in its own right. In such cases, we can consider a two-level hierarchy of simulated annealing. At its top is our original problem: min {c(x): x in S}. At its bottom is a family of subproblems: min {f(x,y): y in B} for x in S. Temporarily, we assume that f(x,y) is evaluated without noise. At the k-th (top-level) visit to x, restart (bottom-level) simulated annealing to estimate c(x), running it for g(x,k) accepted moves where g is an increasing function of k. We estimate c(x) by the smallest of the f(x,y)-values seen so far. Suppose that the range of f(x,y) is a finite set. Under weak conditions, the set on which f(x,y) is minimized has positive measure and then our estimator of c(x) is exactly correct from some finite time onwards almost surely. Thus, sections 5.2.2 and 5.3 apply. When f(x,y) is estimated with noise, we have a subproblem of the form considered at the outset of section 5. Alternatively, if f(x,y) is estimated by steady-state simulation, we can generally estimate c(x) consistently by increasing the run length on successive visits to x -- perhaps restarting where the preceding run left off.

6. CONVERGENCE WITHOUT NOISE

Let S* be the set on which c* is minimized. The set S' below is closely related to S* and, with our neighborhood construction which prevents all local minimizers from having only uphill neighbors, equals S*. Four convergence properties follow:

(i) P{X(k) in S*} --> 1

(ii) [I{X(1) in S*} + ... + I{X(n) in S*}]/n --> 1 a.s.

(i') P{Y(k) in S'} --> 1

(ii') [I{Y(1) in S'} + ... + I{Y(n) in S'}]/n --> 1 a.s.

The primes signal the pruned chain. Heine (1994a) gives weak conditions on the cooling schedule with corresponding selection of the parameter j in QUICKER which imply that these four parameters hold. From properties (i) and (i') respectively, expected-value counterparts of (ii) and (ii') follow. The expected-value counterpart of property (ii) may seem to say more than it does, in view of example 2 in section 3.1. Properties (i') and (ii') show that convergence is not merely an artifact of increasingly long self-loop sequences of rejected proposed moves. The point of QUICKER is to generate these sequences implicitly, not explicitly.

With our neighborhood construction, property (i) follows from Belisle (1992) even if NEXT is used without QUICKER, just setting L equal k+1. The resulting adaptive cooling is generally much slower than the nominal schedule. Because in this case the chains X and Y coincide, property (i') follows.

7. CONCLUSIONS

7.1 THE POWER OF SOPHISTICATED SIMULATED ANNEALING

We have shown that simulated annealing is a much more flexible algorithm than naive implementations have led many to believe. Key features of our approach, missing in naive versions, are macrostates, look-ahead, loop-skipping, and diversification as well as proposed moves depending on objective-function values. Contrary to what partisians of other techniques often assert or tacitly imply, virtually every optimization method can be combined profitably with simulated annealing. Thus, tabu penalities (in part, to inhibit short-run oscillation) and the most important feature of genetic algorithms, having multiple solutions in each state, can be imbedded in simulated annealing proper by expanding the state space. This Markov-chain setting is vital for unification and a complete analysis that is not ad hoc. Other techniques can serve as a front end to simulated annealing proper. Such hybrids are illustrated by our use of low-discrepancy point sets in a preprocessor. This appears to be the first contact between simulated annealing and Quasi-Monte Carlo. Even with noisy observations, all the preprocessors indicated in section 2.3 apply -- except branch and cut.

7.2 THE RELEVANCE OF CONVERGENCE

Though the relevance of the convergence results in section 6 is arguable when c is known, they can be extended to noisy observations via the results in Fox and Heine [(1994a),(1994b)] as indicated in sections 5.2.2 and 5.3.2. When c at x must be estimated by (Monte Carlo) simulation at x and greater accuracy is desired at optimal and near-optimal states, the practical relevance of this Fox-Heine extension is clearcut. On the other hand, if the only parameters on which c depends can be estimated at every state, relevance is again arguable.
The four properties in section 6 seem to say that the search is "attracted" to S*, but it is hard to pin down the link (if any) with the first hitting time of S*. All we can say for now is that cooling more slowly than the canonical schedule and allowing every state to be potentially in F(x,k), as we do above, eliminates the pathologies of section 3.1, implies the speedup of section 3.2, and is (more than) enough for the four properties in section 6.

Properties (ii) and (ii') reflect what is actually observed on a single (long) run of simulated annealing. With massively-parallel runs, properties (i) and (i') become directly observable. These latter two properties are essential for the experimental-design application in Section 5.4.3.

7.3 THE RELEVANCE OF THE INFINITE-STATE CASE

It might be argued that the case S infinite is a mathematical fiction, analogously to our rationale in section 5.4.2 for the assumption in sections 5.2.2 and 5.3.2 that the range of c is a finite set. However, unlike the latter situation we believe that considering S infinite leads to important practical insights. It makes gradient-based subroutines feasible. Our algorithm in Sections 4.2 and 5.1 applies to a general class of problems, with no problem-specific subroutines other than those to compute or estimate c and its gradient. This is an incentive to write programs and, with few exceptions, contrasts with the ad hoc way discrete problems are generally attacked by probabilistic search.

7.4 DIRECTIONS FOR RESEARCH

We have identified three major directions for research, namely to find reasonable conditions (if possible) which imply

* an expected-value version of QUICKER preserves the convergence properties in section 6

* a linear speedup of simulated annealing with parallel processing for general state spaces

* convergence of simulated annealing for general state spaces with noisy observations and, perhaps less important, without restricting the range of c to a finite set.

For the first direction above, Heine [unpublished] has solved the problem for properties (ii) and (ii') which relate to almost-sure Cesaro convergence. The second direction points to linear speedup for general state spaces. We know from section 3.2 that, for finite S, a satisfactory treatment exists. Promise of linear speedup adds to the attractiveness of Monte Carlo methods, especially relative to determinstic alternatives. The latter have hardly any chance for guaranteed linear speedup, because the computations generally cannot be decoupled across processors -- other than by a heuristic partitioning of the original problem into disjoint subproblems. The third direction points to global optimization in simulation, which much of the literature skirts. When S is finite, we know from section 5.2.2 that simulated annealing provides a sturdy framework. Extension to general state spaces is sketched in section 5.3.2, but fleshing out our proof skeleton by proving consistency of the estimators at the stationary points of d is an outstanding problem.

Finding convergence rates corresponding to (i') and (ii') and getting reasonable, easily-computed bounds on expected first hitting time to S* relative to the chain Y are open problems. The discovery of QUICKER makes them important. It also makes analyses of the corresponding X-chain properties irrelevant.

ACKNOWLEDGMENTS:

The impetus to write this paper came from an invitation to give a featured talk at the Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing at Las Vegas, Nevada, June 23-25, 1994.

REFERENCES:

R. K. AHUJA, T. L. MAGNANTI, and J. B. ORLIN (1993). Network Flows: Theory, Algorithms, and Applications, Prentice-Hall, Englewood, Cliffs, New Jersey.

C. J. P. BELISLE (1992). Convergence Theorems for a Class of Simulated Annealing Algorithms on R^d. Journal of Applied Probability 29 , 885-895.

P. BILLINGSLEY (1968). Convergence of Probability Measures, Wiley, New York.

P. BRATLEY, B. L. FOX, and H. NIEDERREITER (1992). Implementation and Tests of Low-Discrepancy Sequences, ACM Transactions on Modeling and Computer Simulation 2, 195-213.

P. BRATLEY, B. L. FOX, and L. E. SCHRAGE (1987). A Guide to Simulation, Springer-Verlag, New York, second edition.

K. L. CHUNG (1974). A Course in Probability Theory, Academic Press, New York, second edition.

B. L. FOX (1990). Generating Markov-chain Transitions Quickly: I, ORSA Journal on Computing 2, 126-135.

B. L. FOX (1993). Integrating and Accelerating Tabu Search, Simulated Annealing, and Genetic Algorithms, Annals of Operations Research 41, 47–67.

B. L. FOX (1993). Random Restarting Versus Simulated Annealing, Computers and Mathematics with Applications 27, No. 6, 33–35.

B. L. FOX (1995). Faster Simulated Annealing, SIAM Journal on Optimization, to appear.

B. L. FOX and P. W. GLYNN (1990). Discrete-time Conversion for Simulating Finite-horizon Markov Processes, SIAM Journal on Applied Mathematics 50, 1457–1473.

B. L. FOX and G. W. HEINE (1994a). Probabilistic Search with Overrides, technical report, University of Colorado, Denver.

B. L. FOX and G. W. HEINE (1994b). Stable Estimators for Self-Adjusting Simulations, technical report, University of Colorado, Denver.

B. L. FOX and A. R. YOUNG (1991). Generating Markov-chain Transitions Quickly: II, ORSA Journal on Computing 2, 3–11.

S. B. GELFAND and S. K. MITTER (1989). Simulated Annealing with Noisy or Imprecise Measurements, Journal of Optimization Theory and Applications 62, 49–62.

P. GLASSERMAN (1991). Gradient Estimation Via Perturbation Analysis, Kluwer Academic Publishers, Norwalk, Massachusetts.

T. HANSEN, G. L. MULLEN, and H. NIEDERREITER (1992). Good Parameters for a Class of Node Sets in Quasi-Monte Carlo Integration, Mathematics of Computation 61, 225–234.

G. W. HEINE (1994a). Smart Simulated Annealing, PhD Dissertation, University of Colorado, Denver.

G. W. HEINE (1994b). Convergence Properties of a Loop-Skipping Algorithm for Simulated Annealing, in preparation.

G. LARCHER, A. LAUSS, H. NIEDERREITER, and W. Ch. SCHMID (1994). Optimal Formulas for (t,m,s)-Nets and Numerical Integration of Multivariate Walsh Series, technical report, University of Salzburg, Austria.

P. S. LAURSEN (1994). Can Parallel Branch and Bound Without Communication Be Effective?, SIAM Journal on Optimization 4, 288–296.

P. L'ECUYER (1990). A Unified View of the IPA, SF, and and LR Gradient Estimation Techniques, Management Science 36, 1364–1383.

P. L'ECUYER (1991). An Overview of Derivative Estimation, Proceedings of the 1991 Winter Simulation Conference, 207-217.

G. L. NEMHAUSER and L. A. WOLSEY (1988). Integer and Combinatorial Optimization, Wiley, New York.

H. NIEDERREITER (1992). Random Number Generation and Quasi-Monte Carlo Methods, CBMS-NSF Regional Conference Series in Applied Mathematics #63, Society for Industrial and Applied Mathematics, Philadelphia.

R. SHONKWILER and E. VAN VLECK (1994). Parallel Speed-Up of Monte Carlo Methods for Global Optimization, Journal of Complexity 10, 64-95.

S. YAKOWITZ (1993). A Globally Convergent Stochastic Approximation, SIAM Journal on Control and Optimization 31 (1993), 30-40.

S. YAKOWITZ and W. LOWE (1991). Nonparametric Bandit Methods, Annals of Operations Research 28, 297-312.

D. I. YAN and H. MUKAI (1992). Stochastic Discrete Optimization, SIAM Journal on Control and Optimization 30, 594-612.