An Efficient Monte Carlo Approach to Optimization

James Douglas Pulfer* and Clement Waine†

Department of Chemistry, University of Warwick, Coventry, CV4 7AL, U.K.

Received November 25, 1997

A highly efficient Monte Carlo algorithm for global optimization has been developed which accepts beneficial moves, rejects all detrimental ones, picks a new step size at random from a guided range, and samples a new region of the response surface using a randomly generated directional search technique. The step size guide is the modified Heaviside function: $r = r_0$ $0 < m \leq nn$), $r = r/2 + 1/r$ ($nn < m \leq 3nn$), and $r = \{r/2 + 1/r\} \sin(2r_0 r)$ ($m > 3nn$), where m is the number of frustrated steps and nn a decision parameter. This approach quickly steps through local optima. Two strategies have been evolved leading to solutions for both the highly intractable modified n-dimensional shekel function and COS_n, the "cosine function". The strategies are efficient: starting from the point (10,10), an average of only 187 steps (successful and otherwise) taking a total of 0.22 shekels of time were required to find the minimum of the COS_2 function to three significant figures with a success rate of 987/1000. In the case of COS_4 the current calculations started from $\{x_i = 10\}$ and found the global minimum to three significant figures in 864 steps and 1.80 shekels with a success rate of 999/1000. COS_{100} has also been solved. In almost all comparisons with currently accepted methods, considerable efficiencies are achieved, and, more importantly, global optimization could be implemented in all test cases.

1. INTRODUCTION

Simulated annealing belongs to a class of heuristic algorithms[1] used to find global optima.[2,3] It was developed by drawing analogies from metallurgical annealing where a solid is repeatedly heated under highly controlled conditions to at (or near) its melting point. This process anneals the material into a desired crystalline form, usually corresponding to the lowest energy ground state arrangement for the lattice structure. The advantage of the process lies in the repetitious heating and cooling cycles which help to rearrange trapped metastable deformities into more stable forms.

To simulate annealing, the spatial position of a particle is used to identify system behavior.[4] The probability, $P_{T(i)}$, that the system is in configuration i with energy $E(x_i)$ at temperature T is given by the Boltzmann distribution

$$P_{T(i)} = (1/Q_T) \exp(-E_i/kT)$$

where Q_T is the molecular partition function, k the Boltzmann constant, and T the thermodynamic temperature. The exponential is commonly referred to as the Boltzmann factor. The Boltzmann distribution inherently favors the lowest energy configuration because it predicts the correct maximum thermodynamic probability[5] and thus physically realistic populations for all occupied energy states.

A modeling algorithm which simulated the cooling of such a system to thermal equilibrium was first developed by Metropolis[6] by using a Monte Carlo method to iteratively generate sequential states of the system.[7] Simulated annealing (SA) was originally used by Kirkpatrick et al.[8] and Cerny[9] for combinatorial optimization of discreet events. Later, it was modified to optimize continuous phenomena.[10−16] However, there is an inherent weakness in continuous simulated annealing because it is forced to employ a discreet step size. This turns out to be a major concern because it has been shown that such protocols may never find the exact global optima or may only do so in the limit as the total number of steps increases.

Sutter and Kalivas[17] considered the problem of optimizing a multivariate n-dimensional function $E(x) = E(x_{i1},x_{i2},...,x_{in})$ where, for all i, the x_{ik} are independent variables, belonging to the real number space R_n. The method simulates the annealing process by choosing an initial set of starting positions $(x_{i1},x_{i2},...,x_{in})$, calculating the function value $E(x_i)$ and then giving the x_{in} random displacements along each of their coordinate axes, thus generating a new $E(\mathbf{x_j})$ on the surface. The move is "downhill" if $\Delta E_{ji} < 0$. Since the new configuration $\mathbf{x_j}$ is moving in the right direction for the minimization problem, the move is accepted unconditionally. Conversely, an "uphill" movement is assumed if $\Delta E_{ji} > 0$ and is accepted in accord with the Metropolis[6,7] acceptance criterion defined by p

$$p = P_{T(j)}/P_{T(i)} = \exp(-\Delta E_{ji}/kT)$$

where the acceptance of any detrimental move arises from an auxiliary sampling experiment whereby a random number q is generated on the range [0,1] and the move is accepted if $p \geq q$. Otherwise, the move is rejected, another set of coordinates is chosen, and the evaluation is repeated. This criterion clearly simulates the annealing process by making it possible to avoid being trapped in local minima, even though it introduces a fairly high degree of inefficiency.

* To whom all correspondence should to be addressed. E-mail: mssed@csv.warwick.ac.uk. Fax: +44 1203 524112.

† Centre for Drug Design and Development (3D Centre), University of Queensland, Brisbane, Queensland, 4072, Australia. E-mail: ddcwaine@dingo.uq.edu.au or C.Waine@mailbox.uq.edu.au.

S0095-2338(97)00103-0 CCC: $15.00
Published on Web 05/23/1998

Details of this fundamental algorithmic process have been succinctly summarized by Lipkowitz and Boyd,[18] Kalivas,[4] and others and will not be presented further except to say that any algorithm depending on Metropolis type criteria is bound to be relatively inefficient. Also, depending on the algorithm, several different termination criteria have been imposed. Collins et al.[19] have reviewed these strategies, and it appears that those which skillfully combine several criteria have the best chance of success.

2. APPROACH

The overall global optimization strategy is based on randomly orienting a stochastically picked step from a guided range. In this algorithmic approach, the general form of the simulated annealing optimization strategy has been kept but several important principles have been abandoned in favor of more effective heuristic methods for searching the range space. Three important criteria have been evolved for developing an efficient Monte Carlo search strategy. First, a stochastically chosen guided step size is randomly oriented for every move, which either remains stuck at a local optima or leads to less optimal values elsewhere. This effectively abandons the relatively inefficient Metropolis criterion. Overall, it has the desired effect of expeditiously sampling the entire space. Second, the problem of fixed step size is largely overcome by continuously varying it in ways which are advantageous to the search procedure. This is done by guiding the maximum possible step size using a modified Heaviside step function which systematically holds possible step sizes at quite large values for the first few frustrated moves and then gradually allows them to converge to $\sqrt{2}$. If the moves are still frustrated after a threshold number, then the step is modulated with a sine function which has the advantage of contracting and expanding the size of the move. This procedure has been found to efficiently step through local optima and, once the algorithm establishes that it is through and nearer the global one, brings the optimal step size back to its starting value once more, to repeat the process. Third, two different termination strategies are utilized, depending on the type of function and the behavior of its parameters in the neighborhood of their optimal positions. If a function has both dominant and relatively slack variables such as those encountered in protein fold simulations, where the coordinate variables are influenced by either strong electrostatic or weak van der Waals interactions, then the termination strategy required will differ substantially compared to functions which have numerous shallow minima near the optimal point. Experience quickly indicates which strategy is required.

2.1. Strategy 1. This approach starts by simultaneously optimizing all of the variable positions until the difference between successful moves passes a screening threshold, as does the function value itself. Then the parameters are shaken, one independent variable at a time, in order to move the slower varying ones until they all pass a two step screening: the first checks to see whether the difference between successful function values falls below a preset tolerance, and the second compares the corresponding parameter moves. All *n* variables are independently shaken in sequence, a total of (usually) three times. Then, to ensure the global optimum is found, all variables are simultaneously varied one final time in case there is a strong functional interdependence between two or more variables. For most functions this is sufficient. Without having to evaluate the first or second derivatives, this ensures that the optimal point is reached. The strategy of this algorithm is thus summarized:

(i) an *n*-dimensional function $E(\mathbf{x_i})$ is given an initial random or selectively chosen set of parameter positions, $\mathbf{x_i}$, in the range of interest.

(ii) An appropriate screening parameter *del0* is chosen along with a rocking integer iroc which keeps track of the total number of times the complete set of parameters has been optimized to pass threshold settings del$|(\epsilon_j - \epsilon_i)|$ and thresh$|x_j - x_i|$. *istep* is set to 0 so as to simultaneously vary all parameters. If $istep = i$, then the search is confined to the *i*th variable. Initialize the function screening parameter del and calculate both the screening parameter thresh and terminating parameter (scan) using *del0*. The very first E_j value is set arbitrarily high if minimizing or low if maximizing a function; otherwise it is calculated and employed in the search in the manner indicated below.

(iii) A suitable step size guide, *r*, is calculated using the modified Heaviside step function:

$$r = r_0 \qquad 0 < m \leq nn$$

$$r = r/2 + 1/r \qquad nn < m \leq 3nn$$

$$r = \{r/2 + 1/r\} \sin(2r_0 r) \qquad m > 3nn$$

where r_o is usually between 4 and 8 arbitrary units (i.e. 40−80% of the parameter range), *m* is a frustration index which increments by 1 each time the search is unsuccessful, and *nn* and 3*nn* are user defined integer parameters which control the calculation of *r*. The actual step size, *rx*, is then chosen at random on the range [0,*r*]. Usually *nn* is set to 5 unsuccessful moves and 3*nn* to 15. Thus, for the first several unsuccessful attempts, the guide is kept relatively large so as to provide a reasonable sampling of the entire space; then for the next 10 frustrated moves, the size is narrowed down to approximately $\sqrt{2}$, after which the modified sine function is used to offer an almost random variety of both large and very small numbers. Usually, by this stage, it does not take many attempts to find a better position. The step direction, *cx*, for each variable is generated by randomly picking a number on the interval [−1,1]. This is equivalent to orienting the step in a new random direction and has been found to improve algorithmic efficiency.

(iv) The new position $\mathbf{x_j}$ is calculated for all coordinates with functional value $E(\mathbf{x_j})$ using

$$x_j = x_i + rx*cx$$

(v) ΔE_{ji} is calculated.

(vi) If $\Delta E_{ji} < 0$ and if the global minimum is sought, then the move is successful and accepted (conversely when a global maximum is searched for, if $\Delta E_{ji} > 0$, then the move is accepted) and $\mathbf{x_j}$ becomes $\mathbf{x_i}$ and $E(\mathbf{x_j})$ becomes $E(\mathbf{x_i})$. If not, then the $\mathbf{x_j}$ are not changed, the frustration parameter *m* is updated to $m + 1$, and the thwarted optimization goes to (iii). If successful, *r* is reset to r_0 and the counter *m* is set back to 0.

Table 1. Test Functions Employed

identifying symbols	function	optimizing param
A	$TCMAX = 20 + 0.8x_1 + 0.8x_2 + 0.022x_1x_2 - 0.015x_1^2 - 0.015x_2^2$	max(100,100)
B	$COSX_1X_2 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)\cos(4\pi x_2) + 0.3$	min(0,0)
C	$COSX_1{+}X_2 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$	min(0,0)
D	$COS_n = \sum_{i=1}^{n}\{ix_i^2 - [(i+2)/100]\cos((i+2)\pi x_i) + (i+2)/100\}, \quad [-1,10]$	$\{\min \forall x_i = 0\}$
E	$SHEKEL_n = -\sum_{i=1}^{n}\{1/[(a_i - x_i)^2 + c_i]\}, \quad c_i = 10^{n-3-i}, [0,10]$	$\{\min \forall x_i = 2i\}$
F	$EXP_n = \exp(-0.5\sum_{i=1}^{n}\{x_i^2\}), \quad [-1,1]$	$\{\max \forall x_i = 0\}$
G	$BC\_COS_n = \sum_{i=1}^{n}\{0.1\cos(5\pi x_i) - x_i^2\}, \quad [-1,1]$	$\{\max \forall x_i = 0\}$
H	$RAS_n = \sum_{i=1}^{n}\{x_i^2 - \cos(18x_i)\}, \quad [-1,10]$	$\{\min \forall x_i = 0\}$
I	$TEST_n = \sum_{i=1}^{n}\{x_i^4 - 16x_i^2 + 5x_i\}, \quad [-3,3]$	$\{\min \forall x_i = -2.904\}$

(vii) The tolerance of $|(E_j - E_i)|$ is checked using del. If the function tolerance test fails, then the procedure moves to (iii); if it passes, all of the parameter tolerances $|x_j - x_i|$ are checked using thresh. If any fails, the procedure moves to (iii), otherwise, *istep* is incremented to *istep* + 1. This optimizes the next parameter variable x_{istep} during those passes where ispot, which counts the number of completed sets of individual parameter optimizations, is in the range 0 < ispot ≤ iroc. r0 is increased by a factor of 10 during this stage of the search. When istep becomes greater than *n*, it is reset to 1 and ispot is advanced by one to show that all the parameters have been individually shaken. If both del and thresh have been passed and if ispot is greater than (iroc-2), it is assumed that the calculation is still searching for the global optimum. However, if the previous best *E*, compared to the current *E* and the previous best x_i compared to the current ones pass scan (a termination screening parameter), then the calculation is considered successful and stops. If not, it goes to (iii). If ispot > iroc, then the search assumes that it is very near the global optimum.

(viii) At this stage istep is set to zero and all variables are simultaneously varied one more time. In this instance, convergence is screened by using slightly smaller tolerance values for del and thresh, where

$$\text{del} = \text{del}/2 + del0^2/(6.5*\text{del}); \quad \text{thresh} = \text{del}$$

When these are passed, the calculation assumes it is very close to the optimum position and terminates. If iroc is large enough and del0 small enough, this is a valid assumption.

2.2. Strategy 2. Test functions such as the highly intractable modified *n*-dimensional shekel function of the type

$$SHEKEL_n = -\sum_{i=1}^{n}\{1/[(a_i - x_i)^2 + c_i]\}, \quad \min \forall x_i = 2i$$

where $SHEKEL_n$ might have a_i of 2, 4, 6, 8, and 10 with corresponding c_i of of 0.1, 0.01, 0.001, 1, and 10, prove to be surprisingly difficult and require a different strategy (Table 4). This arises because they possess two divergent properties. Some of the minima vary rapidly in the immediate neighborhood of their optima (e.g. x_2 and x_3), whereas others are extremely shallow (x_4 and x_5). Consequently, what will work well for one type will fail for the others and vice versa. This situation can be overcome by making use of the prominence of the sharp minima first. During the course of simultaneously searching all of the parameters, those contributing most strongly to the emerging function optima will normally be moved more quickly to near their best position. Once there, the variable is frozen and the total number of parameters to be optimized is reduced by one. The remainder are then simultaneously searched until the second is found and so on. When all of the variables have been located in this manner, either the assumption is made that the global optimum has been reached or one final pass is performed, simultaneously varying all parameters one last time. In this manner, reasonably quick and efficient solutions have been found for $SHEKEL_n$ and others noted in Tables 2 and 3.

This strategy can be summarized as follows:

(i) As in the first case, an *n*-dimensional function $E(\mathbf{x_i})$ is given an initial random or selectively chosen set of parameter positions, $\mathbf{x_i}$ on the range of interest.

(ii) A suitable initial step size r_o is picked, usually between 4 and 8 arbitrary units (40−80%) of the range (however, see Table 4); *m*, an integer, is also picked. It determines the size of the difference matrix dif(*n*, *m* + 1) which holds the absolute value of the last *m* differences $|x_j - x_i|$ between successive parameter moves for the *n* parameters of the problem. The $(m + 1)^{st}$ location holds the sum of these differences, and thresh is set so as to be an appropriate tolerance parameter for screening such sums. Each location in the dif matrix is initialized to an arbitrarily large value

Table 2. Performance Summary under Optimal Algorithmic Strategies[a]

strategy/ function	significance[b]	no. f.e.	time shekels[c]	no. correct/ 1000 params[d]	sps[e]
1/TCMAX[f]	3SF	719	0.55	902	1318
2/COSX$_1$X$_2$	3SF	467	0.68	968	688
1/COSX$_1$+X$_2$	3SF	424	0.43	951	993
1/COS$_1$	4SF	168	0.14	984	1220
1/COS$_2$	3SF	187	0.22	987	849
1/COS$_3$	3SF	378	0.61	998	623
1/COS$_4$	3SF	864	1.80	999	479
2/SHEKEL$_5$[g]	3SF	8362	10.80	894	774
1/EXP$_4$	2SF	161	0.22	998	730
1/BC_COS$_4$[h]	3SF	447	0.98	972	457
1/BC$\bar{}$COS$_4$	3SF	662	1.29	968	513
1/RA$\bar{}$S$_2$	3SF	1052	1.24	906	850
1/TEST$_2$	3SF	1917	1.81	969	1061

[a] All calculations have been carried out using a Lahey Fortran F77L3 compiler on a 486 DX2/66 Canon Innova personal computer. [b] Parameter averages for 1000 runs came to at least the indicated number of significant figures and for many, much better. [c] Computer time to evaluate 1000 function calls of the five-dimensional shekel function.[29] [d] Parameter outlier criteria: each TCMAX parameter within 2 units of the optimal position; each COSX$_1$X$_2$ within 0.2; COSX$_1$+X$_2$ < 0.2; COS$_n$ < 0.2; SHEKEL$_5$ < 0.45; EXP$_4$ < 0.05 (for function only); BC_COS$_4$, RAS$_2$, and TEST$_2$ < 0.2. [e] Steps per shekel of time. [f] Starting coordinates for each function: TCMAX = (0,0); COSX$_1$X$_2$ = (10,10); COSX$_1$+X$_2$ = (10,10); COS$_n$ = {$\forall$ x_i = 10}; SHEKEL$_5$ = {$\forall$ x_i = 5}; EXP$_n$ = {$\forall$ x_i = 0.2}; BC_COS$_n$ = {$\forall$ x_i = 10}; RAS$_n$ = {$\forall$ x_i = 10}; and TEST$_n$ = {$\forall$ x_i = 3}. [g] Total possible steps limited to 9999, this number representing the highest case accuracy required. [h] BC_COS$_4$ = {$\forall$ x_i = 0.5} to compare with Breiman and Cutler's work.[28]

Table 3. Comparison with Other Methods

	Pulfer−Waine method[a]		comparison calcs	
functions	f.e.	time(shekels)	authors(s)	f.e.
A	473	0.36	Kalivas[b]	?[c]
B	227	0.32	Bohachevsky[d]	400
C	285	0.27	Bohachevsky[d]	400
COS$_2$	54[e]	0.08	Bohachevsky[d]	150[f]
COS$_2$	54[e]	0.08	Kalivas[b]	819
EXP$_4$	161	0.22	B and C[g]	234
BC_COS$_4$	447	0.98	B and C[g]	2784
RA$\bar{}$S$_2$	1052[h]	1.24	KG and V[i]	5917
TEST$_2$	1917[j]	1.81	S and H[k]	9228

[a] For comparison, whenever possible the Pulfer and Waine algorithm used the same initial coordinates and termination accuracy, except as noted below. [b] Variable step size generalized simulated annealing (VSGSA) algorithm;[4] optimum position of COS$_2$ found to five significant figure accuracy. [c] The total number of steps was not reported but, from the accompanying figure,[4] it looks like about 142, starting from (20,20) and achieving an accuracy of 3 SF. [d] Generalized simulated annealing (GSA) algorithm of Bohachevsky et al.[22] [e] Found using Bohachevsky's[22] starting conditions of (1,1) and accuracy of 2−3 SF. Using Kalivas'[4] initial conditions (0,−1), the current method took 513 steps and 0.69 shekels of time to achieve 4 SF accuracy and a prediction rate of 1000 correct for every 1000 parameter evaluations. [f] Bohachevsky's calculated accuracy is 2−3 SF in all test cases. [g] Deterministic vertexes pivot (DVP) algorithm of Breiman and Cutler.[28] They also report shekel times of 8.34 (ours 0.22) and 179.88 (cf. 0.98). [h] Initial coordinates, (10,10); termination accuracy, three significant figures or better. [i] Simulated annealing (SA) of Kirkpatrick et al.[8] [j] Initial coordinates, (3,3); termination accuracy, three significant figures or better. [k] Fast simulated annealing method (FSA) of Szu and Hartley.[13]

like 1. Also, a parameter labeling matrix, id(n), is initialized to hold the sequential integer numbers of each variable 1, 2, ..., n. As before, the very first E_j is set arbitrarily high if

Table 4. Optimal Algorithmic Settings

For Strategy 1[a]

function	del0	r_0	iroc	lbound	ubound
(a) Used in Table 2					
TCMAX	0.125	90.0	2	−1.0	110.0
COSX$_1$+X$_2$	0.020	8.0	3	−1.1	10.1
COS$_1$	0.085	4.0	3	−1.1	10.1
COS$_2$	0.085	4.0	3	−1.1	10.1
COS$_3$	0.045	4.0	3	−1.1	10.1
COS$_4$	0.0175	4.0	3	−1.1	10.1
EXP$_4$	0.350	4.0	3	−1.1	1.1
BC_COS$_4$[b]	0.040	4.0	3	−1.0	1.0
BC$\bar{}$COS$_4$	0.035	4.0	3	−1.1	10.1
RA$\bar{}$S$_2$	0.010	4.0	3	−1.0	11.0
TEST$_2$	5.000	90.0	5	−4.0	4.0
(b) Used in Table 3					
TCMAX	0.150	90.0	2	−1.0	110.0
COSX$_1$+X$_2$	0.045	4.0	3	−1.1	10.1
COS$_1$[c]	0.450	4.0	3	−1.1	1.1
COS$_2$[d]	0.012	4.0	3	−1.1	1.1

For Strategy 2[e]

function	m	thresh	rhold	rangel	rangeu
(a) Used in Table 2					
COSX$_1$+X$_2$	36	0.115	4.0	0.4	6.0
SHEKEL$_5$	36	0.100	8.0	0.4	8.0
(b) Used in Table 3					
COSX$_1$+X$_2$	24	0.450	4.0	0.8	6.0

[a] All other settings the same; nn = 5; 3nn = 15; del = thresh = scan. [b] Settings used for Brieman and Cutler[28] comparison in Table 3. [c] Bohachevsky[22] comparison. [d] Kalivas[4] comparison. [e] All other settings the same; the maximum number of function evaluations permitted is 9000 (except SHEKEL$_5$, which is 9999); if the calculated r_0 value falls below rangel, then $r_0 = \sqrt{2}/2$; if above rangeu, then r_0 = rhold; dif(n, m + 1) = 1 initially.

minimizing or low if maximizing a function; otherwise it is calculated in the manner indicated below.

(iii) Steps (iii)−(v) are exactly the same as those for strategy 1 and need not be reproduced again except to point out that in the cases where id(i) is zero, the corresponding x_i is frozen in the last successful position and the calculation skips to the next variable in the search.

(vi) If the move is successful, then dif(j,m) is dropped off and replaced by dif(j,m − 1). This is done for all m differences, ending with dif(j,1) containing the difference between the last two successful moves. As this is being done, the sum is updated and placed in dif(j,m + 1), and if larger than thresh, then id(j) is set to j, otherwise 0. A failure parameter, ident, is updated each time id(j) is set to j to keep track of the total number of variables still being simultaneously varied. ident is n at the beginning of the calculation and 0 when all variables have passed thresh in the above manner. If ident is zero, then either the calculation is terminated or every id(j) is set to j for one final pass, where all variables are simultaneously varied one final time. The calculation then goes to (iii).

3. RESULTS

The performance of the algorithm was evaluated by globally optimizing standard multivariate test functions having a large number of extrema and one global optimum position as shown in Table 1. For each, the initial positions, $\mathbf{x_i}$, were arbitrarily chosen to be as remote as possible from

the optimum in order to provide a reasonable test of the method, to be sure that there was adequate sampling of the response surface, and to check the general efficiency of the procedure. Starting from such extreme positions, the algorithm found the optimum parameters to reasonably high accuracy for eight of the nine test functions. Precision was more variable, with six of the nine having prediction rates better than 950 times out of 1000 in an economic number of moves (both accepted and rejected steps are counted) as exhibited in Table 2. The other three cases are acknowledged to be severe tests, yet when considered under similar stringent starting conditions and working to three significant figure accuracy, the prediction rate is still about 0.9 or better. The success over such a wide variety of functions clearly demonstrates that the performance of the algorithm does not depend on any special property of the object function. Also, although not shown here, functions which have multiply degenerate optima work just as well.

The other aspect of the algorithm which makes it useful is its ability to start at considerable distances from the global optimum of highly variable functions such as COS_4 and yet still locate the correct position with about the same speed and efficiency as for trials which start much closer. The COS_2 case serves to illustrate this attribute. The total number of function evaluations required to locate the global position was 187 steps to achieve 3 SF accuracy when starting from (10,10) and 54 steps to achieve between 2 and 3 SF when starting from (1,1). If the (10,10) calculation is adjusted to yield a solution between 2 and 3 SF, then the difference in the number of function evaluations narrows still further. Comparisons with other methods considered efficient, such as Bohachevsky's[22] work on the COS_2 function, show a 3:1 improvement (Table 3).

Starting at (0,−1) and working to four significant figures, the method took 513 steps to locate the optimum for COS_2. This compares very favorably with Kalivas' 819 steps[4] to achieve five SF accuracy. We believe the comparison to be fair for two reasons: first, the Monte Carlo prediction rate of 1000 correct out of 1000 parameter evaluations means that locating the optimum is a dead certainty for such a class of functions evaluated to that level of accuracy, and consequently there is no need to refine the coordinates further. Second, most of the effort of VSGSA goes into getting into the immediate neighborhood of the optimum because VSGSA tends to wander more than GSA until it gets there.[4] For all calculations reported in Tables 2 and 3, the optimal strategies are given in Table 4.

It should be mentioned that, for each test function, more than one initial position was chosen, at widely divergent sites, to ensure that the method is independent of starting coordinates. In every instance, the algorithm achieved the global optimum with a high prediction rate no matter where the starting points were (within reason) or what the basic step size, r_0, was, as long as it was in the range 4−90. Also, keeping all other factors the same for a given test, the total number of steps taken from any extreme initial point to the optimum were about the same. For instance, starting COS_2 from either (10,10) or (−10,−10) took about the same number of steps, as one would expect for a fully randomized process operating on a symmetric function.

Another general observation on this algorithm concerns the basic step size r_0. If chosen too small, it causes the search to be inefficient simply because it takes several tries to move out of any local minimum. If chosen too large, then once again the algorithm becomes inefficient because it begins to return to the same local minima over and over again as the search progresses. Some numerical experimentation is required to achieve an optimal result for any given problem. The selection of settings shown in Table 4 serves as a reference guide, not only for r_0 but for the other key settings such as del0, iroc, and the like. Experience will quickly reveal what the approximate correct settings will be for a given problem. For example, in strategy 1, the del0 setting should be kept as wide as possible so that the calculation does not get bogged down and moves rapidly and efficiently to the neighborhood of the optimum. Then, the refinement of the coordinates can be left to the smaller del and thresh screening parameters on the very last pass, when all of the coordinates are simultaneously varied one last time.

Also, iroc should be made as small as possible without sacrificing too much accuracy. A value of around 3 proves to be about right. As a rule of thumb, we start with the following standardized algorithmic settings at the beginning of each calculation: $r_0 = 8.0$, $nn = 5$, $del0 = 0.035$, iroc = 3, scan = *del0*, and del = *del0*. For every pass within the calculation where $istep = 0$, del = del/2 + $del0^2$/(6.5*del) and thresh = del. Then these are modified to achieve the best possible result. In practice we never change *nn* and 3*nn* and find that r_0, *del0*, and iroc are the only ones that are important. Indeed, if iroc were to be left at 3 and only the other two changed, an optimal strategy would quickly emerge. In strategy 2, the general principles guiding this method hinge on the tradeoff between thresh and *m*: in general, as the size of the difference matrix increases ($m + 1$) and the threshold sum of these differences goes down, the precision and accuracy improves while the time required deteriorates. However, in this strategy, some leverage can be obtained from the observation that as *m* increases, thresh can also increase somewhat because the sum of the absolute values of all *m* differences becomes larger even though each individual difference may be at or well below what is required. This leads to surprising speeds and accuracy when just the right combination is found. Low settings of *m* and thresh usually result in high speed, moderate accuracy, and moderate success prediction rates; medium settings of *m* (12−36) and low to medium settings of thresh (0.1−0.5) lead to reasonable speed, high accuracy, and good success rates. We have found that the effort required to optimize these parameters is highly rewarding and very worthwhile for solving similar problems on a regular basis.

The efficiency of the calculation, as measured in steps per shekel (sps), is useful as a guide or pointer, indicating the complexity or difficulty of the problem at hand. Efficiency generally declines from around 1200 sps for the simplest problems (COS_1) to around 450 sps for functions such as BC_COS_4. Given a high initial efficiency, an optimized strategy for that class of functions ought to ultimately yield good accuracy, high speed, and excellent prediction rates.

4. DISCUSSION

In this algorithm, the global minimum was adopted as the default search strategy. There are many ways to choose the initial starting positions, $\mathbf{x_i}$. If no prior knowledge is

available on the system of interest, then they might be picked at random. Otherwise, as familiarity with a given problem increases, the $\mathbf{x_i}$ may be specified in a highly detailed manner. For example, it is far too costly to start at a random position for the global energy minimization of a large peptide or protein molecule or even, for that matter, for a two to seven residue cell within such macromolecular structures. Where homologies exist and the general backbone coordinates of one of them is reasonably accurately resolved, then this information, coupled with known side-chain bond lengths, Ramachandran angles,[20] and preferred angle positionings,[21] would serve as suitable starting positions for optimizing the energy of the biomolecule. Strategy 2 shows particular promise in this regard, optimizing met-enkephalin (22 torsion parameters) in under a 1000 function evaluations, provided the starting backbone coordinates are threaded from a library of known functional shapes.

In both algorithmic strategies, after choosing a suitable set of initial positions, all of the parameters are simultaneously varied by randomizing the optimal guide setting r and then randomly orienting that step length by picking a direction cosine from the interval [−1,1]. Bohachevsky et al.[22] have demonstrated that this method is as effective as any other.

In strategy 1, after each successful move, the function tolerance parameter checks $|(E_j - E_i)|$ to see if it falls below the preset threshold (del), which is usually in the range 0.1−5 depending on the degree of precision and accuracy required and on the type of function. If this check fails, then the algorithm continues to simultaneously search all of the variables for increasingly better optima. If it happens to pass, then thresh is used to check on all of the coordinate tolerances $|x_j - x_i|$. Thresh is usually set to be the same as del and, if varied, done so in the same way. As before, if this check on the coordinate positions fails, then the optimization strategy continues. Otherwise, depending on the history of the search, when both succeed, then the optimization of the variables is either terminated or considered to be only a first approximation to the global optimum. In the latter case, to focus the calculation on those parameters which contribute little to the function optimum, i.e., cause the function minimum to vary in a shallow manner, each parameter is isolated in turn and optimized in a separate pass, with all the rest frozen in their current best positions. After each of the n variables has been individually optimized in this fashion, the process is repeated. This may be done a number of times, as it turns out to be quite a rapid procedure. It has the effect of not changing the positions of the dominant variables to any extent but moves the others to much more realistic sites. When it gets to within the last two complete such passes, the current function and coordinate values, compared to the previous best, are scanned to see if they pass a termination tolerance value called scan. After that, if the termination criteria has not yet been triggered, then del and thresh are slightly reduced and all the variables simultaneously varied one last time before terminating under the assumption that the search must be at the global optimum.

One other variation could be pursued during the latter stages of a search. It usually takes very few steps to reach the terminal screening stage of the algorithm. Then, instead of going on, an alternative might be to switch to a steepest decent algorithm[23,24] or a simplex routine[25−27] to refine the coordinates. We have not pursued this strategy, but it might prove highly worthwhile for some classes of functions.

The second strategy only needs to be invoked if the function has a mixture of both sharp, deep minima and a number of quite shallow ones, all contributing to the global optimum value. Its success lies in its ability to "remember" the last m absolute difference values of the parameter positions. During the global search, if a given parameter is not changing much and has not for the last m moves, then either it is quite close to its optimal value or it is one of the dominant contributors to the total function value, leading to it being located first; in either case, it is then frozen and the search continues with the remainder of the variables. In this way, the number of parameters gets reduced until there is no more to be analyzed. The parameters that contribute little to the global optima tend to be able to move widely about their optimal position without influencing the success of the next move so that if they are left within the simultaneous optimization of all the parameters, then termination criteria will be triggered which leave them quite far from the correct position. This strategy eliminates that possibility by turning their ability to move widely (slackness) into a sensitive optimization criteria which notes that when the sum of the differences remains high, the parameters cannot pass the threshold barrier easily and are kept within the search strategy longer than any of the others, and usually the very slackest variable of all is the one which is considered on its own, at the end of this process. Isolated in that way, it usually moves very quickly to its optimal position.

In Table 3, direct comparisons with several current methods are made, such as the best current modified simulated annealing considered efficient, namely, generalized simulated annealing (GSA)[22] and variable step size GSA (VSGSA)[4,17] which show that this method is highly competitive in the cases where relevant recent literature data are available; however, the real value of this algorithm lies in its ability to rapidly and efficiently locate the global optimum for *every* type of test function. For example, VSGSA cannot handle COS_4.

5. CONCLUSION

We believe that this algorithm is useful for those applications that require very high speed to find the global minimum with reasonable accuracy. If the accuracy demands are modest, such as at the two significant figure level, then the prediction rate can be made routinely high, like 999 out of 1000. However, as with all methods, speed and efficiency drop if one requires more than three or four significant figures of accuracy; also, if one wants to be dead certain of the result in every case, then this algorithm is not going to be helpful. Last, we report success with the COS_{100} test case, correctly predicting 95 out of the 100 coordinates using strategy 1. Work is now in progress on optimizing the algorithm for large parameter applications. From preliminary calculations, we think that it will be robust enough to handle large-scale optimizations in protein chemistry such as those required to elucidate mutant structure stability.

ACKNOWLEDGMENT

The authors would like to thank the University of Papua New Guinea Research and Publications Committee for

funding equipment and software purchases. J.D.P. would also like to thank the Chemistry Department, University of Warwick, for appointing him to be a Visiting Research Fellow.

REFERENCES AND NOTES

(1) Rajasekaran, S.; Reif, J. H. Nested Annealing: A Provable Improvement to Simulated Annealing. *Theor. Comput. Sci*. **1992**, *99* (1), 157−176.
(2) Törn, A. A.; Zilinskas, A. In *Lecture Notes in Computer Science (Global Optimization*); Goos, G., Hartmanis, J., Eds.; Springer-Verlag: Berlin, 1989; Vol. 350.
(3) DiGennaro, F. S.; Cowburn, D. Parametric-Estimation of Time Domain NMR Signals Using Simulated Annealing. *J. Magn. Reson.* **1992**, *96* (3), 582−588.
(4) Kalivas, J. H. Optimization Using Variations of Simulated Annealing. *Chemom. Intell. Lab. Syst*. **1992**, *15* (1), 1−12.
(5) Atkins, P. W. In *Physical Chemistry*, 5th ed.; Oxford University Press: Oxford, U.K., 1994; p 670.
(6) Metropolis, N. Monte Carlo−In the Beginning and Some Great Expectations. *Lect. Notes Phys.* **1985**, *240*, 62−70.
(7) Metropolis, N.; Nelson, E. C. Early Computing at Los Alamos. *Ann. Hist. Comput*. **1982**, 4, 348−357.
(8) Kirkpatrick, S.; Gelatt, C. D., Jr.; Vecchi, M. P. Optimization by Simulated Annealing. *Science* **1983**, *220* (4598), 671−680.
(9) Cerny, V. Thermodynamical Approach to the Traveling Salesman Problem−An Efficient Simulation Algorithm. *J. Optim. Theor. Appl*. **1985**, *45* (1), 41−51.
(10) Vanderbilt, D.; Louie, S. G. A Monte Carlo Simulated Annealing Approach to Optimization over Continuous Variables. *J. Comput. Phys*. **1984**, *56*, 259−271.
(11) Khachaturyan, A. Statistical-Mechanics Approach in Minimizing a Multivariable Function. *J. Math. Phys*. **1986**, *24*, 1834−1838.
(12) Wille, L. T. Searching Potential Energy Surfaces by Simulated Annealing. *Nature* **1986**, *324*, 46−48.
(13) Szu, H,; Hartley, R. Fast Simulated Annealing. *Phys. Lett. A* **1987**, *122* (3, 4), 157−162.
(14) van Laarhoven, P. J. M.; Aarts, E. H. L. In *Simulated Annealing: Theory and Applications*; Reidel: Dordrecht, The Netherlands, 1987; p 99.
(15) Nachtsheim, C. J.; Johnson, M. E. A New Family of Multivariate Distributions with Applications to Monte Carlo Studies. *J. Am. Stat. Assoc*. **1988**, *83* (404), 984−989.
(16) Romeo, F.; Sangiovanni-Veincentelli, A. A Theoretical Framework for Simulated Annealing. *Algorithmica* **1991**, *6* (3), 302−345.
(17) Sutter, J. M.; Kalivas, J. H. Convergence of Generalized Simulated Annealing with Variable Step Size with Application toward Parameter Estimations of Linear and Nonlinear Models. *Anal. Chem*. **1991**, *63*, 2383−2386.
(18) Lipkowitz, K. B.; Boyd, D. B. *Reviews in Computational Chemistry*; VCH: New York, 1990; p 299−308.
(19) Collins, N. E.; Eglese, R. W.; Golden, B. L. Simulated Annealing−An Annotated Bibliography. *Am. J. Math. Manage. Sci*. **1988**, *8*, 209−307.
(20) Ramachandran, G. N.; Sasisekharan, V. Conformation of Polypeptides and Proteins *Adv. Protein Chem*. **1968**, *23*, 438−463.
(21) Ponder, J. W.; Richards, W. Tertiary Templates for Proteins−Use of Packing Criteria in the Enumeration of Allowed Sequences for Different Structural Classes. *J. Mol. Biol*. **1987**, *193* (4), 775−791.
(22) Bohachevsky, I. O.; Johnson, M. E.; Stein, M. L. Generalized Simulated Annealing for Function Optimization *Technometrics* **1986**, *28* (3), 209−217.
(23) Powell, M. J. D. Convergence Properties of Algorithms for Nonlinear Optimization. *SIAM Rev.*. **1986**, *28* (4), 487−500.
(24) Fletcher, F.; Powell, M. J. D. A Rapidly Convergent Descent Method for Minimization. *Comput. J.* **1963**, *6*, 163−168.
(25) Nelder, J. A.; Mead, R. A Simplex Method for Functional Minimization. *Comput. J.* **1965**, *7*, 308−313.
(26) Jurs, P. C. *Computer Software Applications in Chemistry*; Wiley: New York, 1986; p 125.
(27) Pulfer, J. D. Numerical Modeling of the Action of Acetolactate Synthase Isozyme. II. Using Simplex Optimization. *Comput. Chem*. **1991**, *15*, 287−292.
(28) Breiman, L.; Cutler, A. A Deterministic Algorithm for Global Optimization. *Math. Program*. **1993**, *58* (2), 179−199.
(29) Fagiuoli, E.; Pianca, P.; Zecchin, M. In *Toward Global Optimization 2*; Dixon, L. C. W. Szegö, G. P., Eds.; North-Holland: Amsterdam, 1978; p 103.

CI970103+