JOURNAL OF COMPUTATIONAL PHYSICS 99, 28–32 (1992)

An Evaluation of the Sniffer Global Optimization Algorithm Using Standard Test Functions*

ROGER A. R. BUTLER AND EDWARD E. SLAMINKA

Mathematics Department, Auburn University, Auburn, Alabama 36849-5310

Received June 29, 1990; revised April 5, 1991

The performance of Sniffer—a new global optimization algorithm—is compared with that of Simulated Annealing. Using the number of function evaluations as a measure of efficiency, the new algorithm is shown to be significantly better at finding the global minimum of seven standard test functions. Several of the test functions used have many local minima and very steep walls surrounding the global minimum. Such functions are intended to thwart global minimization algorithms.
© 1992 Academic Press, Inc.

1. INTRODUCTION

In 1984, Vanderbilt and Louie [1] described a version of Simulated Annealing for unconstrained optimization over continuous variables. To demonstrate the efficacy of their algorithm, and to compare it with a variety of (then) current methods, they applied it to the seven test functions proposed in Dixon and Szego [2]. Since that time, Simulated Annealing has become the method of choice for a variety of optimization settings.

In this paper we compare a new global optimizer, Sniffer, to Simulated Annealing using these same test functions. The Sniffer algorithm (named by Donnelly whose original FORTRAN program used a subroutine named SNIFR) was introduced by Donnelly and Rogers in [6] and has since been applied, with much success, to a wide class of optimization problems in engineering and the natural sciences [4, 5, 3, 7, 8]. Some of these problems have over 400 degrees of freedom and contain thousands of local minima. In Section 2 we present a description of the original Sniffer algorithm, and in Section 3 we describe an extension of the algorithm that is capable of altering some of its own parameters in an attempt to solve individual problems more efficiently. This algorithm should be considered as one of a family of possible extensions to or variations on the original Sniffer algorithm. The purpose of this paper is to demonstrate that this family of algorithms can provide an efficient global optimization algorithm that compares favorably with Simulated Annealing. We demonstrate that this is, in fact, the case in Section 4, where we compare the results of the Sniffer algorithm with those of Simulated Annealing when applied to the test functions of Dixon and Szego.

* The authors thank the reviewers for their helpful suggestions.

2. DESCRIPTION OF THE DONNELLY AND ROGERS SNIFFER ALGORITHM

In 1981, Griewank [9] developed a global optimizer that attempted to find the global minimum of a differentiable objective function by following carefully constructed search trajectories. If we assume that c is a *target level* which is higher than the desired global minimum value of the objective function f, then Griewank's search trajectories are solutions of the second-order differential equation

$$x''(t) = \frac{-e(I - x'(t)\, x'^{T}(t))\, \nabla f(x(t))}{f(x(t)) - c}, \qquad e > 0,$$

with any initial point (x_0, x_0') satisfying $f(x_0) > c$, and $\|x_0'\| = 1$.

By construction, Griewank's search trajectories are solutions of the above differential equation and, as such, possess several desirable properties, including:

1. Trajectories cannot converge to minima with values greater than c.

2. If a particle follows a search trajectory, then its speed at position $x(t)$ is proportional to $f(x(t)) - c$. Thus, as a trajectory gets closer to the target level c, the speed at which it is followed is reduced, and the trajectory does a more thorough job of minimizing the function f at lower levels. Conversely, as a trajectory moves further from the target level c, the speed at which it is followed is increased and so little time is wasted minimizing the function at higher levels. When $f \leqslant c$, the technique of following search trajectories amounts to a local minimization technique.

0021-9991/92 $3.00
Copyright © 1992 by Academic Press, Inc.
All rights of reproduction in any form reserved.

3. Trajectories are invariant with respect to translations of the variables and multiplication of the term $f(x(t)) - c$ by a positive scalar.

When following one of Griewank's search trajectories, a particle will tend to turn towards the negative gradient direction. The extent to which the gradient influences a particle's direction of motion is determined by the *gradient sensitivity parameter* e in Griewank's differential equation. Larger values of e give rise to trajectories that more closely follow the gradient of the objective function.

In his paper, Griewank proves that for a certain class of objective functions, and for appropriate settings of the parameters e and c, search trajectories satisfying his differential equation are able to escape local minima and converge to the global minimum of the objective function. However, if implemented exactly as described in [9], Griewank's algorithm is extremely expensive computationally since the solution of an ordinary differential equation must be computed at each step. To overcome this problem, Donnelly and Rogers [3, 6] introduced a discrete analog of the algorithm. Their algorithm, which they later called Sniffer, attempts to find the global minimum of an objective function by stepping along discrete analogs of Griewank's search trajectories.

In its barest form the Sniffer algorithm can be described as follows: let x_0 be an initial position and let d_0 be an initial unit direction vector. Let ε be a *gradient sensitivity parameter* similar in effect to Griewank's e parameter, and let μ be a *step size parameter* which will be used to determine the size of the algorithm's steps. Usually d_0 is initialised as $-\nabla f(x_0)/|\nabla f(x_0)|$, and typical values for ε and μ are $\varepsilon = 1.0$ and $\mu = 0.2$. Given the pair (x_n, d_n) we compute (x_{n+1}, d_{n+1}) in the following manner. Define

$$\alpha = \max\{0, 1 + (1+\varepsilon)\, d_n^T \nabla f(x_n)\}$$
$$\delta = -\varepsilon \nabla f(x_n) + \alpha d_n$$

and use these to compute d_{n+1} and x_{n+1} from the equations

$$d_{n+1} = \delta/|\delta|$$
$$x_{n+1} = x_n + \mu(f(x_n) - c)\, d_{n+1}.$$

The parameter α is, by definition, guaranteed to be non-negative, and it is defined so that if the angle between d_n and $-\nabla f(x_n)$ is less than 90° and $|\nabla f(x_n)|$ is large, or if the angle is very close to 0°, then the algorithm should follow the negative gradient instead of crossing back and forth over it in a zigzag fashion reminiscent of steepest-descent methods in valleys.

To find a global minimum using the Sniffer algorithm we successively compute pairs (x_1, d_1), (x_2, d_2), ... By computing the value of f at the points $x_1, x_2, \ldots$ we are then able to investigate the function f's minima. The discrete "walks" produced by the Sniffer algorithm possess properties similar to those of Griewank's search trajectories. They slow down at lower levels and speed up at higher levels thus wasting little time minimizing the function.

3. A VARIATION OF THE DONNELLY AND ROGERS SNIFFER ALGORITHM

In an attempt to improve the performance of the original Sniffer algorithm, we have enabled the algorithm to repeatedly alter its own parameters based on the preliminary results of its continuing search. These modifications, the result of many numerical experiments on several different "test functions," in no way constitute a new algorithm, but rather the incorporation of some limited "intelligence" into the existing one. In this section we provide complete descriptions of, and at least partial justifications for, all of these modifications. The changes we have made generally increase the speed at which the Sniffer algorithm finds the global minimum of the test functions.

As stated in Section 2, the original Sniffer algorithm requires the user to provide initial values for several (program) parameters. These include an initial value for c (an estimate of the minimum value of the function), ε (a parameter measuring the algorithm's sensitivity to the function's gradient), and μ (a parameter that is used to determine the algorithm's step size). The original algorithm's performance is intimately connected with the choice of all of these parameters. If ε is chosen close to zero then the gradient information will be largely ignored and the particle will move in a direction, d_{n+1}, close to the direction d_n. If ε is relatively large, the particle will move in the direction of $-\nabla f(x_n)$, the negative gradient. The current function value and the choices of c and μ all influence the step size. If $f(x_n)$ is close to the value c, or if μ is small, then the step size will be small. However, if $f(x_n)$ is far from c, or if μ is large, then the step size will be large. In the original implementation of the algorithm, c was chosen to be slightly smaller than a known value of f, the parameter ε was generally chosen to be 1.0, and μ was set at 0.2. Clearly, these initial values cannot be adequate for every optimization problem, and even when they are initially adequate, they are unlikely to remain so for the entire optimization process.

The first attempts at modifying the algorithm were of an interactive nature. As the function f approached the value c, c was lowered. Also, while observing the motion of the particle, ε and μ were altered to either avoid known local minima or to investigate wells containing possible global minima. It was also observed that some trajectories, after having reached a fairly low value of f, proceeded to "higher ground" for too many iterations. After a number of iterations, the algorithm was restarted at the point x_n with the lowest function value encountered and given a random

direction vector or the direction of the negative gradient of f at x_n. We have now incorporated some of these heuristics into the algorithm itself.

By enabling the algorithm to alter its own parameters, we were able to gradually reduce the area within which the algorithm searches for a global minimum and, hence, to increase the speed with which it finds a global minimum. (In Simulated Annealing a similar reduction of area is affected by reducing the "temperature.") The present variation of the Sniffer algorithm uses the "original" algorithm (as defined in [6, 3]) as a subroutine called MINIM. The subroutine MINIM is given an initial position at which to start searching for the global minimum (either a "random" position at the very beginning of the algorithm's use, or the position of the best minimum found so far), and is then allowed to run for a fixed (maximum) number of steps called MAXSTEPS. Once the subroutine MINIM has been run, the values of ε, μ, c, and MAXSTEPS are all adjusted as follows:

$$\varepsilon \leftarrow \varepsilon \times M_\varepsilon$$
$$\mu \leftarrow \mu / M_\mu$$
$$\text{MAXSTEPS} \leftarrow M_{\text{steps}} \times \text{MAXSTEPS}$$
$$c \leftarrow (c + \min)/2 - c_{\text{bias}}$$
$$c_{\text{bias}} \leftarrow c_{\text{bias}} / M_{\text{bias}}.$$

In order to make the algorithm increasingly sensitive to the local gradient as more steps are taken (and hence as we hopefully come closer to finding a global minimum), the value of ε is increased—in the present algorithm we have simply multiplied by a constant factor M_ε. To reduce the area in which MINIM searches for the global minimum we also reduce the value of μ—this corresponds to reducing the step size and is achieved in our present algorithm by dividing the value of μ by the factor M_μ. To guarantee that the region searched by MINIM is bounded, we also introduce a maximum allowable step size $\mu_{\max}$. This parameter is the maximum value allowed for the term $\mu(f(x_n) - c)$ in the equation $x_{n+1} = x_n + \mu(f(x_n) - c)\, d_{n+1}$. In order to give the MINIM subroutine more time to search the reduced area, we increase MAXSTEPS (the maximum number of steps that MINIM will be allowed to take) by multiplying by the factor M_{steps}. Finally, the value of c is adjusted using the lowest value of the function that has been found so far. The initial value of c is set at a somewhat arbitrary value (-100.0 for the present numerical experiments), and this value is then periodically adjusted by the program according to the best information that the algorithm has as to the exact value of the minimum. If we denote the best minimum found so far by *min*, then the value of c is computed using $c = (c + \min)/2 - c_{\text{bias}}$. Here c_{bias} is a positive number (set initially to $|c|/10$) that is used to ensure that the value of c is always below the best minimum found so far. As with all the other parameters, we also adjust the value of c_{bias} after each call of the subroutine MINIM. In the present program we repeatedly decrease the size of c_{bias} by dividing it by M_{bias}.

The values of the constants used in the present variation of the Sniffer algorithm were chosen after performing many numerical experiments and are as follows: $M_\varepsilon = 1.1$, $M_\mu = 1.7$, $M_{\text{steps}} = 1.1$, and $M_{\text{bias}} = 1.1$. (We note that in [9] Griewank suggests altering his gradient sensitivity parameter, e, by multiplying by 1.1 in order to reduce the probability that his search trajectories will leave the search area.)

In order to compare the results of [1] with those obtained using our variation of the Sniffer algorithm, we attempted to use the very same stopping criterion. The algorithm was considered to have found the global minimum if the minimum value it found, $\min_{\text{found}}$, when compared with the true minimum, $\min_{\text{true}}$, satisfied the following condition:

$$\left| \frac{\min_{\text{found}} - \min_{\text{true}}}{\min_{\text{true}}} \right| \leqslant 10^{-3}.$$

In the present paper we have used the above variation of the Sniffer algorithm to find the global minimum of each of several standard test functions. In the process of producing the present variation several other similar schemes were tried and tested. The results of these preliminary tests motivated the choice of the present constants used to adapt the parameters that appear in the algorithm. Even though the present constants give excellent results, they are very unlikely to be optimum—further numerical studies need to be performed. The method by which we change the algorithm's parameters is also an area in which many other schemes could be employed. However, even using our simplistic approach, the performance of our variation (as we see in the next section) compares very favorable with that of Simulated Annealing.

4. OPTIMIZATION FUNCTIONS AND RESULTS

The seven test functions described by Dixon and Szego were intended both to thwart global minimizers and to model minimizations which occur in practice. These functions have either very steep walls surrounding the local minima or an enormous number of local minima. For completeness we include a detailed description of these functions.

SHEKEL'S FAMILY (F1, F2, and F3).

$$f(x) = -\sum_{i=1}^{m} \frac{1}{(x - a_i)^{\mathrm{T}} (x - a_i) + c_i},$$

where $x=(x_1, x_2, x_3, x_4)^T$, $a_i=(a_{i1}, a_{i2}, a_{i3}, a_{i4})^T$, and the region of interest is $0 \leqslant x_j \leqslant 10$, for $j=1, 2, 3, 4$. The function F1 has $m=5$, F2 has $m=7$, and F3 has $m=10$. The global minimum value for F1 is -10.1532, for F2 is -10.40294, and for F3 is -10.53641:

i	a_i				c_i
1	4.0	4.0	4.0	4.0	0.1
2	1.0	1.0	1.0	1.0	0.2
3	8.0	8.0	8.0	8.0	0.2
4	6.0	6.0	6.0	6.0	0.4
5	3.0	7.0	3.0	7.0	0.4
6	2.0	9.0	2.0	9.0	0.6
7	5.0	5.0	3.0	3.0	0.6
8	8.0	1.0	8.0	1.0	0.7
9	6.0	2.0	6.0	2.0	0.5
10	7.0	3.6	7.0	3.6	0.5

HARTMAN'S FAMILY (F4, F5).

$$f(x) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{4} a_{ij}(x_j - p_{ij})^2},$$

where $x=(x_1, ..., x_n)$, $p_i=(p_{i1}, ..., p_{in})$, $a_i=(a_{i1}, ..., a_{in})$, and the region of interest is $0 \leqslant x_i \leqslant 1$ for $i=1, ..., n$. The global minimum value for F4 is -3.86278, and for F5 is -3.32237. F4 has $n=3$:

i	a_{ij}			c_i	p_{ij}		
1	3.0	10.0	30.0	1.0	0.36890	0.1170	0.2673
2	0.1	10.0	35.0	1.2	0.46990	0.4387	0.7470
3	3.0	10.0	30.0	3.0	0.10910	0.8732	0.5547
4	0.1	10.0	35.0	3.2	0.03815	0.5743	0.8828

F5 has $n=6$:

i	a_{ij}						c_i
1	10.00	3.00	17.00	3.50	1.70	8.00	1.0
2	0.05	10.00	17.00	0.10	8.00	14.00	1.2
3	3.00	3.50	1.70	10.00	17.00	8.00	3.0
4	17.00	8.00	0.05	10.00	0.01	14.00	3.2

i	p_{ij}					
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

BRANIN (F6).

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

where $a=1$, $b=5.1/(4\pi^2)$, $c=5/\pi$, $d=6$, $e=10$, $f=1/(8\pi)$, and the region of interest is $-5 \leqslant x_1 \leqslant 10$ and $0 \leqslant x_2 \leqslant 15$. The global minimum value for F6 is 0.39789.

GOLDSTEIN AND PRICE (F7).

$$f(x_1, x_2) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)],$$

where the region of interest is $-2 \leqslant x_1, x_2 \leqslant 2$. The global minimum value for F7 is 3.0.

Table I shows the initial parameter values used for the Sniffer algorithm. Sniffer 1 denotes the algorithm in which the parameter values were chosen to achieve the greatest success percentage using the lowest number of function evaluations. Sniffer 2 denotes the algorithm in which the success percentage was intended to equal that of Simulated Annealing with the least number of function evaluations.

In [1, p. 265] one of the principal free parameters, T_0, was "essentially chosen on the basis of the variance of a random sample" and the other, χ_T, was chosen "by trial and error." To provide a methodology as similar as possible to that used by Vanderbilt and Louie, we chose the required program parameters on the basis of a *small* number of numerical experiments. The program parameters ε, μ, μ_{max}, and MAXSTEPS were chosen in two stages. As an initial approximation we set $\varepsilon=1.0$, $\mu=0.2$, $\mu_{max}=\mu$, and MAXSTEPS = 100. For Sniffer 1, we perturbed these values one at a time to determine the effect of increasing or decreasing that parameter on the success rate and then used the information obtained to produce a single set of parameter values for each function. For functions F1, F2, and F3, this method very quickly produced the values shown in Table I. For the functions F5 and F6 a large change was made to MAXSTEPS, and for F4 and F7, increasing the *gradient sensitivity parameter* also improved performance.

TABLE I

Parameter Values Used in the Minimization of the Seven Standard Test Functions

	Sniffer 1				Sniffer 2			
Function	ε	μ	μ_{max}	MAXSTEPS	ε	μ	μ_{max}	MAXSTEPS
F1	1.1	0.15	0.17	95	1.4	0.19	0.19	22
F2	1.7	0.24	0.24	104	1.9	0.25	0.25	24
F3	1.4	0.25	0.25	95	1.6	0.24	0.24	44
F4	20.0	0.01	0.50	20	20.0	0.01	0.50	20
F5	1.0	0.10	0.20	22	1.0	0.05	0.10	5
F6	1.0	0.10	1.00	5	1.0	0.10	1.00	5
F7	4.0	0.80	0.80	10	4.0	0.80	0.80	10

TABLE II

Comparison of the Efficiency of Simulated Annealing to That of Sniffer

	Simulated annealing[a]		Sniffer 1		Sniffer 2	
Function	Function Evaluations	Percent Success	Function Evaluations	Percent Success	Function Evaluations	Percent Success
F1	3910	54	3695	90	1040	54
F2	3421	64	2655	96	1092	64
F3	3078	81	3070	95	1589	81
F4	1224	100	534	99	534	99
F5	1914	62	1760	99	364	62
F6	557	100	205	100	205	100
F7	1186	99	664	100	664	100

[a] The values given for Simulated Annealing are from Vanderbilt and Louie [1].

Starting from the parameter values used by Sniffer 1, we attempted to make minimal changes in order to match Simulated Annealing's success rates. No changes were necessary for the parameters for functions F4, F6, and F7 because the success rates were very close to 100% for both methods. Our major strategy for the remaining functions was to decrease the total number of steps used while at the same time approximating the success rates of Simulated Annealing as accurately as possible. We achieved this by decreasing MAXSTEPS and making minor perturbations to ε, μ, and $\mu_{\max}$. Only a small number of numerical experiments were necessary to obtain the very same success rates as those of Simulated Annealing.

Table II gives the number of function evaluations (including those needed for the gradient calculations) based upon 100 runs for each function and the percentage of success.

Two things are apparent from Table II. First, the Sniffer 1 success rate is at least 90% for these seven test functions and, except for F1, the success rate is at least 95%. In all cases this compares favorably with Simulated Annealing's success rate. Perhaps the most demanding function is F3 on which Sniffer 1 uses essentially the same number of function evaluations as Simulated Annealing, but achieves a 14% higher success rate. Second, Sniffer 2 achieves success rates comparable to those of Simulated Annealing, but with fewer function evaluations, at worst 56% of the number of evaluations Simulated Annealing performed for function F7, and at best 19% for function F5.

These results indicate that the Sniffer algorithm is a viable optimization algorithm for continuous variables. Other work [7] shows that for optimization problems involving hundreds of variables, the Sniffer's advantage over other optimization methods is even more striking.

REFERENCES

1. D. Vanderbilt and S. G. Louie, *J. Comput. Phys.* **56**, 259 (1984).
2. L. C. W. Dixon and G. P. Szego, in *Towards Global Optimization 2* (North-Holland, New York/Amsterdam, 1978), p. 1.
3. J. W. Rogers and R. A. Donnelly, *J. Optim. Theory Appl.* **61**, 111 (1989).
4. R. A. Donnelly, *A discrete dynamical system for conformational energy optimization* (unpublished).
5. M. L. Papay, GLIDE Program Optimization Results, TRW Defense Systems Group, San Bernardino, CA, 1989 (unpublished).
6. R. A. Donnelly and J. W. Rogers, *Int. J. Quantum Chem.* **22**, 507 (1988).
7. E. E. Slaminka and K. D. Woerner, *Celestial Mech. Dyn. Astron.* **48**, 347 (1990).
8. J. Rogers, R. A. Donnelly, E. E. Slaminka, and R. A. R. Butler, Bounded orbits and chaotic behavior for a dynamical system used in global optimization, 1990(unpublished).
9. A. O. Griewank, *J. Optim. Theory Appl.* **34**, 11 (1981).