1997 年 12 月 Dec.,1997 运 筹 学 学 报 OR Transactions 第 1 卷 第 2 期 Vol.1 No.2

A Simple Multistart Algorithm for Global Optimization [*]

Fred J. Hickernell

(Department of Mathematics, Hong Kong Baptist University, Hong Kong)

Ya-xiang Yuan[†]

(State Key Laboratory of Scientific and Engineering Computing
Institute of Computational Mathematics and Scientific/Engineering Computing
Chinese Academy of Sciences, Beijing 10080, China)

Abtstract

A generalization of the multistart algorithm is proposed for finding the global minimizer of a nonlinear function of n variables. Our method concentrates a quasirandom sample by performing a few inexpensive local searches. The sample is then reduced by replacing worse points by new quasirandom points. A complete local search is performed only on those points with small function values. This method performs favorably in comparison to other global optimization methods.

Key word: nonlinear optimization, global minimizer, random points, quasirandom points.

1. Introduction

Consider the unconstrained optimization problem: find x^* such that

$$f(x^*) = \min_{x \in X} f(x), \tag{1}$$

where $f(x)$ is a nonlinear function defined on $\Re^n$ and $X \subset \Re^n$. Our objective is to find the global minimizer of $f(x)$ in the feasible set. Without assuming any conditions on $f(x)$ global optimization problems are unsolvable in the following sense: no algorithm can be guaranteed to find a global minimizer of a general nonlinear function within finitely many iterations. Suppose that an algorithm applied to a nonlinear function $f(x)$ produces iterates x_i and terminates after K iterations. Unless $X = \{x_i \quad | \quad i = 1, \ldots, K\}$, one can construct a new function $\bar{f}$ with the same function and gradient values as f at all the x_i and with

$$f(\bar{x}) < \min_{1 \le i \le K} f(x_i) - 1$$

Received: Feb. 18, 1997.

*This work was partially supported by Chinese NNSF and the project "large scale scientific computing" of State Commission of Science and Technology, China

†Research carried out while visiting Hong Kong Baptist University, supported by Croucher Foundation

for some $\bar{x} \in X$. This implies that this algorithm cannot find the global minimizer of $\bar{f}(x)$. Even assuming continuity one must evaluate a function on a dense subset of the feasible set to be certain of finding the global minimizer. This unsolvability of global optimization problems is well known, namely, unless special assumptions are made about the objective function any method designed to solve the global optimization requires infinitely many function evaluations (see Dixon [1], Rinnooy Kan and Timmer [9]).

There have been two approaches in the development of global optimization algorithms: deterministic and stochastic (see Rinnooy Kan and Timmer [9], Dixon and Szegö [2]). Deterministic methods can guarantee absolute success, but only by making certain restrictive assumptions on the objective function. Stochastic methods evaluate the objective function at randomly generated points. The convergence results for these methods are not absolute. However, the probability of success approaches one as the sample size tends to infinity under very mild assumptions about the objective function. Reviews of various global optimization approaches are given by Rinnooy Kan and Timmer [9] and Törn and Žilinskas [15].

The simplest stochastic method is the Monte Carlo method, which evaluates the objective function on a set of random points and takes the point with the least function value as the approximate minimizer. The Monte Carlo method is straightforward and very easy to implement. However, it has a serious disadvantage of requiring many function evaluations to find an acceptable approximate solution.

Many algorithms have been proposed that combine the Monte Carlo method with an efficient quasi-Newton local search procedure. For each local minimizer, x_i^*, let X_i^* denote its basin of attraction. By this we mean that if a local search starts from a point in X_i^*, it will converge to x_i^*. The multistart method (see [2][9]) applies local searches to each point in a random sample drawn from the feasible region. If one of these points happens to be in X^*, the basin of attraction of the global minimizer x^*, then the multistart method succeeds. However, the multistart method is inefficient in that it performs complete local searches starting from all sample points, including those not in X^*.

Extensions of the multistart method have been proposed by Törn [14], Rinnooy Kan and Timmer [7][8], Mayne and Meewella [10] and others. These extensions seek to reduce the number of complete local searches that are performed and increase the probability that they are started from points in X^*. At each major iteration a sample is drawn from the feasible region. This sample is then transformed by removing points with larger function values and/or performing one or a few local search steps. A complete local search is performed only on those points that are unlikely to belong to X_i^* for some known x_i^*. Several algorithms use a form of cluster analysis, in particular the single-linkage method, to identify points belonging to X_i^*. Another approach proposed by [10] is to check whether the point satisfies a quadratic approximation to the objective function centered at x_i^*.

Several authors have proposed global optimization methods based on quasirandom, as opposed to random, samples. Quasirandom samples are sets of deterministic points that are evenly distributed over a set (see Hua and Wang [6] and Niederreiter [11]). Niederreiter and Peart [12] and Wang and Fang [16] independently developed methods that perform a sequence of quasirandom searches on domains of decreasing size. An algorithm for solving nonlinear equations that combines quasirandom and quasi-Newton searches has recently been proposed by Hickernell and Fang [5].

The advantage of quasirandom samples over random samples may be illustrated by considering a particular case: the infinite sequences developed by Faure [3]. For every prime number t Faure constructed an infinite sequence $\{z_i | i = 0, 1, \ldots\}$ on the t-dimensional unit cube with the following property: any cube of the form $\left[i_1 t^{-j_1}, (i_1+1)t^{-j_1}\right] \times \cdots \times \left[i_t t^{-j_t}, (i_t+1)t^{-j_t}\right]$ with volume t^{-m} contains exactly one of the t^m points $z_{lt^m+1}, \ldots, z_{(l+1)t^m}$. However, for a random sample of t^m points there is a probability of $(1-t^{-m})^{t^m}$ ($\approx e^{-1}$ for large m) that this same cube contains no sample points. Thus, the Faure points are more evenly distributed than random points.

The algorithm proposed in this article is an extension of the multistart method. Having drawn a quasirandom sample of N points from the feasible set, p iterations of an inexpensive local search are applied to concentrate the sample. The q points with the smallest function values are retained, while the other $N-q$ points are replaced by new quasirandom points. Then the concentration step is repeated. Any point that is retained for s iterations is used to start an efficient complete local search, provided that its function value is not significantly larger than the smallest function value obtained so far. The algorithm terminates when no new local minimum is found after several iterations.

The next section describes our algorithm in detail. It is also recast in a form suitable for solving systems of nonlinear equations. Numerical results are reported in Section 3.

2. The Algorithm

Our algorithm consists of "major iterations". At the beginning of the k^{th} major iteration, a set of N starting points $\{x_i^{(k)} | i = 1, \ldots, N\}$ is available. These points are concentrated by applying p iterations (usually $p = 1$ or 2) of an inexpensive local search (such as steepest descent) to each $x_i^{(k)}$ to obtain a new point $y_i^{(k)}$. To reduce the sample, the q points with smallest function values are identified. That is, the set $I(k) = \{i_j |\ j = 1, ..., q\}$ is chosen such that

$$f(y_{i_1}^{(k)}) \leq f(y_{i_2}^{(k)}) \leq ... \leq f(y_{i_q}^{(k)}) \leq f(y_l^{(k)}) \quad \text{for all } l \notin I(k).$$

The points $y_{i_1}^{(k)}, \ldots, y_{i_q}^{(k)}$ become the starting points $x_{i_1}^{(k+1)}, \ldots, x_{i_q}^{(k+1)}$ for the next major iteration. $N-q$ new quasirandom points comprise the remainder of the $x_i^{(k+1)}$.

At the end of a major iteration, we check whether any index has remained in the set $I(k)$ for s consecutive iterations. If so, an efficient local search algorithm (such as BFGS) is applied to find a local minimizer, z. If $f(z)$ is smaller than $FBEST$, the smallest function value found so far, then NSP, the number of stationary points, is increased by one and $NWSP$, the number of worse stationary points, is set to zero. Otherwise, $NWSP$ is increased by one.

Stopping conditions for global optimization are difficult to choose. If we know, f_{min}, the global minimum value of $f(x)$ in X, the algorithm can be terminated when a point with a function value very close to f_{min} is found. However, in general f_{min} is unknown. When $NWSP$ is large, it seems likely that the current best local minimizer is also a global minimizer, since the previous $NWSP$ local minimizers found have higher function values. Therefore, we terminate the algorithm when $NWSP$ is greater than or equal to $r \times NSP$ for some integer $r > 1$.

The algorithm can be stated as follows:

Algorithm 2.1

```
Step 0 INITIALIZE. Given integers $N \geq q \geq 1$, $p \geq 1$, $r > 1$, $s \geq 1$ and a positive number $\epsilon$,
       Choose a quasirandom sample of $N$ initial points $x_i^0$ $(i = 1, ..., N)$.
       Set $k = NSP = NWSP = 0$ and $NTIX(j) = 0$ $(j = 1, ..., N)$.
Step 1 CONCENTRATE. Apply $p$ iterations of an inexpensive local search algorithm (Algorithm A) to each of the points $x_i^{(k)}$ $(i = 1, ..., N)$, obtaining $y_i^{(k)}$ $(i = 1, ..., N)$.
Step 2 REDUCE. Find $I(k) \subset \{1, \ldots, N\}$ such that $I(k)$ has $q$ elements and that $f(y_i^{(k)}) \leq f(y_j^{(k)})$ holds for all $i \in I(k)$ and $j \notin I(k)$.
       Set $NTIX(j) = \begin{cases} 0 & \text{if } j \notin I(k) \\ NTIX(j) + 1 & \text{if } j \in I(k) \end{cases}$ .
Step 3 FIND LOCAL MINIMUM. For $j = 1, ..., N$ such that $NTIX(j) \geq s$:
       Set $NTIX(j) = 0$.
       If $NSP = 0$ or $f(y_j^{(k)}) \leq FBEST + \epsilon$ then
            Apply an efficient local optimization algorithm (Algorithm B) starting from $y_j^{(k)}$ to obtain a local minimizer $z$,
            If $f(z) < FBEST$ then
                 Set $NSP = NSP + 1$, $NWSP = 0$, $FBEST = f(z)$.
            Else
                 Set $NWSP = NWSP + 1$.
            End
       Else
            Set $NWSP = NWSP + 1$.
       End
       If $NWSP \geq r \times NSP$ then stop (success).
Step 4 SAMPLE ADDITIONAL POINTS. For $j = 1, 2, ....N$
       If $NTIX(j) = 0$ then
            Generate $x_j^{(k+1)}$ by quasi-random techniques.
       Else
            Set $x_j^{(k+1)} = y_j^{(k)}$.
       End
       Set $k = k + 1$.
       If the total number of function calls $> MFCALL$ then stop (failure).
       Go to Step 1.
```

The algorithm proposed here concentrates the sample (applies one or two local search steps to every point) before reducing the sample (replacing points with high function values). The reason is that while X^* may contain many points with high function values, but applying a few iterations of a local search procedure to these points should substantially decrease the function values. Törn and Žilinskas [15] report that in Törn's experiments with clustering methods concentration was found to be more effective than simple reduction. Furthermore, they remark that the reason several authors prefer reduction is because it allows a theoretical basis for stopping conditions which concentration does not.

The extensions to the multistart algorithm reviewed in Section 1 record all local minimizers found and their approximate domains of attraction. Our algorithm does not take this approach in order to avoid the inherent calculation involved. The overhead required to perform clustering increases with the number of iterates. Even the approach of Mayne and Meewella [10] may be computationally intensive if the number of local minimizers is large.

Our algorithm is a general one, allowing different choices of N, p, q, r, s and ϵ. Some of the methods described in the Section 1 are special cases of the above algorithm. $N = q$ and $\epsilon = \infty$ corresponds to the original multistart algorithm. $N = q = 1$ and $\epsilon = 0$ is comparable to the S2 algorithm of Hartman [4]. The quasirandom search methods of Niederreiter and Peart [12] and Wang and Fang [16] are similar to Algorithm 2.1 when $p = 0$ and there is no local search in Step 3.

However, our interest is not these special cases. The value of N may be moderate to large depending on the assumed complexity of the function. The optimal values of p and q depend on the shape of the objective function. If the function has small values on X^*, the basin of attraction of the global minimizer, then p and q can be small, because the more crucial requirement is generating enough sample points for one to lie in X^*. If, on the other hand, the objective function has large values on X^* then concentration is important and reduction should be done conservatively. This would suggest larger values for p and q. In the absence of any special considerations q be should set small in relation to N (e.g. $q \sim \sqrt{N}$). In order to avoid the cost of excessive local searches the value for p is recommended to be 1 or 2. The value of $s = 2, 3$ works well in practice. The value of ϵ should be small enough to prevent unnecessary local searches, but large enough to allow useful local searches. A value of approximately 10^{-4} times the typical function value is reasonable.

Algorithm A in Step 2 for computing y_i^k can be chosen by the user. If the steepest descent method is used, then

$$\bar{x}_{i,0}^{(k)} = x_i^{(k)}, \quad \bar{x}_{i,j}^{(k)} = x_{i,j-1}^{(k)} - \alpha_{i,j}^{(k)} \nabla f(x_{i,j-1}^{(k)}) \quad (j = 1, 2, ..., p), \quad y_i^{(k)} = \bar{x}_{i,p}^{(k)},$$

for all $i = 1, 2, \ldots, N$, where $\alpha_{i,j}^{(k)} > 0$ are stepsizes. If D is a typical length scale of the feasible region, X, then setting $\alpha_{i,j}^{(k)} = D/(N^{1/n} ||\nabla f(x_{i,j-1}^{(k)})||_2)$ as a first approximation attempts to obtain a significant decrease in the function value without taking too large a step.

The new sample points in Steps 1 and 4 are generated by quasirandom methods rather than random methods for the reasons given in the Introduction. In particular the Faure points [3] are employed in the numerical experiments in the next section. The number used to generate them is chosen to be the smallest prime number greater than or equal to n. Any infinite sequence of quasirandom points could be used just as easily. See Niederreiter [11] for a description of other sequences such as those of Halton and Hammersley.

Most quasirandom number generators give points on an n-dimensional unit cube. To obtain a sample on the feasible set, a suitable transformation must be used. For example, if the feasible set is a finite cube, the desired sample can be obtained by a linear transformation in each coordinate direction. If $X = \Re^n$, then a transformation from $[0,1]$ to $(-\infty, \infty)$ such as the inverse Gaussian distribution function may be used for each coordinate.

Although our algorithm does not have a strong theoretical basis, it is possible to understand under what conditions it will not obtain the global minimum. Suppose that our algorithm stops after K major iterations, with $f_{stop} = f_{min} + \Delta$ as the approximation to the global minimum, where f_{min} is the true global minimum and Δ is some positive number. Let $X^*(\Delta, p)$ be the subset of X^* with the following property: p iterations of a local search starting from any point in $X^*(\Delta, p)$ leads to a point y such that $f(y) < f_{min} + \Delta$. Therefore, in the course of K iterations, none of the $M \geq N + K(N - q)$ quasirandom sample points were in $X^*(\Delta, p)$. If the sample were random, the probability of this happening is would be $(1 - \mu(X^*(\Delta, p)))^M$, where $\mu(Y)$ denotes the probability of a single random point lying in the set Y. Because quasirandom points are more evenly distributed the probability should be even less. Therefore, or our algorithm to fail $X^*(\Delta, p)$ must be very small, especially if M is large.

The global optimization algorithm may be modified to solve nonlinear equations. Specifically the problem is to find x^* satisfying

$$F(x^*) = 0, \tag{2}$$

where F is a nonlinear function from $\Re^n$ into $\Re^m$. Solving (1) is equivalent to solving (2) if $f(x)$ is taken to be $||F(x)||$ for some norm $||.||$ Therefore we can apply Algorithm 2.1 to solve (2). However, since the global minimum of the objective function is known the algorithm should be modified. There is no need for the variables NSP and $NWSP$ nor the parameter r. The stopping criterion in Step 3. should be changed to the following: *If $FBEST < \delta$, then stop (success).*

If $f(x) = ||F(x)||_2$, we suggest that the steepest steps, namely, Cauchy directions for Algorithm A in Step 1 and the Gauss-Newton method or the Levenberg-Marquardt method can be used for Algorithm B in Step 3.

Törn and Žilinskas [15], Smith, Eskow and Schnabel [13] and others have discussed parallel implementations of global optimization. Although we have not had the opportunity to perform numerical experiments on a parallel computer, we note that the bulk of our algorithm, that is the concentration step, is inherently parallelizable.

3. Numerical Results

A double precision FORTRAN program has been written to test our algorithm. Steepest descent is used for Algorithm A and BFGS with inexact line searches is used for Algorithm B. The termination criterion for Algorithm B is $||\nabla f(x)|| \leq 10^{-6}$. The quasirandom sample points are generated by Faure's algorithm.

The test problems for the numerical experiments come from Dixon and Szegö [2]. They are as follows:

Example 3.1 *Goldstein and Price:*

$$\begin{aligned} f(x) = \;& [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2] \times \\ & [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \end{aligned} \tag{3}$$

$X = \{x \mid ||x||_\infty \leq 2, x = (x_1, x_2)^T \in \Re^2\}$.

Example 3.2 *Branin:*

$$f(x_1, x_2) = c_1(x_2 - c_2x_1^2 + c_3x_1 - c_4)^2 + c_5(1 - c_6)\cos(x_1) + c_5, \tag{4}$$

where $c_1 = 1, c_2 = 5.1/(4\pi^2), c_3 = 5/\pi, c_4 = 6, c_5 = 10, c_6 = 1/(8\pi)$. $X = \{x \mid -5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15\}$.

Example 3.3 *Hartman's Family:*

$$f(x) = -\sum_{i=1}^{m} c_i \exp\Big(-\sum_{j=1}^{n} a_{ij}(x_j - p_{ij})^2\Big) \tag{5}$$

where $p_i = (p_{i_1}, p_{i2}, \ldots, p_{in})^T \in \Re^n$, $a_i = (a_{i1}, a_{i2}, \ldots, a_{in})^T \in \Re^n$, $c_i > 0$ $(i = 1, 2, \ldots\ldots, m)$ *and* $x = (x_1, x_2, \ldots, x_n)^T \in \Re^n$. $X = \{x \mid 0 \leq x_i \leq 1\}$. *The data are as follows:*

1) Hartman3: $n = 3$, $m = 4$,

i	a_{ij}			c_i	p_{ij}		
1	3.0	10	30	1.0	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3.0	10	30	3.0	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.03815	0.5743	0.8828

2) Hartman6: $n = 6$, $m = 4$,

i	a_{ij}						c_i
1	10.0	3.0	17	3.5	1.7	8.0	1.0
2	0.05	10.	17	0.1	8.0	14	1.2
3	3.00	3.5	1.7	10	17	8	3.0
4	17.0	8.0	0.05	10	0.1	14	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

Example 3.4 *Shekel's family:*

$$f(x) = -\sum_{i=1}^{m} \frac{1}{(x-a_i)^T(x-a_i)+c_i} \tag{6}$$

where $a_i = (a_{i1}, a_{i2}, \ldots, a_{in})^T \in \Re^n$, $c_i > 0$ $(i = 1, 2, \ldots\ldots, m)$ *and* $x = (x_1, x_2, \ldots, x_n)^T \in \Re^n$. $X = \{x \mid 0 \le x_i \le 10\}$.

Data: $n = 4$, $m = 5, 7, 10$

i	1	2	3	4	5	6	7	8	9	10
a_{i1}	4	1	8	6	3	2	5	8	6	7
a_{i2}	4	1	8	6	7	9	5	1	2	3.6
a_{i3}	4	1	8	6	3	2	3	8	6	7
a_{i4}	4	1	8	6	7	9	3	1	2	3.6
c_i	0.1	0.2	0.2	0.4	0.4	0.6	0.3	0.7	0.5	0.5

The parameter values used for the numerical experiments are $N = 15$, $p = 1$, $q = 3$, $r = 3$, $s = 2$ and $\epsilon = 10^{-4}$. The numerical results are reported in Tables 3.1 and 3.2. As suggested by Dixon and Szegö [2], we give the number of function evaluations and the running time measured in *units of standard time*, where one unit corresponds to the time required for 1000 evaluations of the S5 test function at the point $(4, 4, 4, 4)^T$. The global minimum was found in every case.

Table 3.1 Results of the Algorithm $(N = 15, p = 1, q = 3, r = 3, s = 2)$

Problem	GP	BR	H3	H6	S5	S7	S10
NF	159	172	143	145	121	127	157
NG	69	79	66	71	60	62	77
T	0.5	0.5	0.75	1.25	0.5	0.75	1.0

Table 3.2 Solutions found by the Algorithm

Problem	Solution Found	Function Value
GP	-1.234733348E-10 -1.0000000036761	2.99999999999994
BR	3.14159265091356 2.27500000358534	.3978873577411121
H3	.1146143435546542 .5556488500545595 .8525469541408391	-3.86278214782076
H6	.2016895034585899 .1500106658026912 ..4768739746403644 .2753324316807096 .311651622367135 .6573005449766441	-3.3223680114155

S5	4.00003715289352 4.00013327657369 4.00003715289352 4.00013327657369	-10.1531996790582
S7	4.00057291797521 4.0006893683435 3.99948970726924 3.99960615763753	-10.4029405668187
S10	4.0007465348935 4.00059293675117 3.99966339657596 3.99950979843363	-10.536409816692

The numerical results for several global optimization algorithms are given by Rinnooy Kan and Timmer [9] and Törn and Žilinskas [15]. Mayne and Meewella [10] give the numerical results for their algorithm. Our algorithm requires fewer function evaluations than most of the other methods. However, it is noted that our algorithm requires gradient calculations, while most other methods do not compute any gradients. Our results also revealed that our algorithm requires less running time than nearly all other methods. In comparison to the the multi-level single linkage method running times for our algorithm were greater for GP, BR and H3 but less for H6, S5, S7 and S10.

Since none of the above 7 test problems is highly oscillating, our algorithm succeeded in finding a global minimizer for each problem with relatively small parameters N, p, q and r ($N = 15$, $p = 1$, $q = 3$, $r = 3$). When the objective function has many locate minima, it is likely that our algorithm will fail to find a global minimum if small parameters N, p, q and r are used. To explore this, we constructed the following test function:

Example 3.5 *(Guilin Hills)*

$$f(x) = 3 + \sum_{i=1}^{n} c_i \frac{x_i + 9}{x_i + 10} \sin\left(\frac{\pi}{1 - x_i + 1/(2k_i)}\right), \tag{7}$$

where $c_i > 0 (i = 1, ..., n)$ are parameters and $k_i (i = 1, ..., n)$ are positive integers. $X = \{x| \quad 0 \leq x_i \leq 1\}$. We consider two cases:

1) $n = 2$, $c_1 = 1$, $c_2 = 1.5$, $k_1 = 5$, $k_2 = 3$

2) $n = 3$, $c_1 = 1$, $c_2 = 1.5$, $c_3 = 2$, $k_1 = 5$, $k_2 = 3$, $k_3 = 10$.

The function (7) has $\prod_{i=1}^{n} k_i$ local minima in the region X. We call this function by *Guilin Hills* because of its similarity to the mountains in Guilin, China. Only one of the local minima of (7) is the global minimum, which is very close to the point

$$\left(1 - \frac{1}{8k_1^2 - 4k_1}, 1 - \frac{1}{8k_2^2 - 4k_2}, \cdots, 1 - \frac{1}{8k_n^2 - 4k_n}\right)^T. \tag{8}$$

The global minimum value of (7) in X is 0.72750432 when $n = 2$, and -1.09065629 when $n = 3$.

For Example 3.5, we found that our algorithm with $N = 15$, $p = 1$, $q = 3$, $r = 3$ and $s = 2$ can not find the global minimum. We tested our algorithm for Example 3.5 with $s = 2$ and different choices of parameters N, p, q and r. The results are given in Tables 3.3 and 3.4. All the runs failed to find the global minimizer. We also tested our algorithm with $s = 3$. Only one run with $N = 15$, $p = 5$, $q = 3$ and $r = 6$ located the global minimizer. Results are reported in Tables 3.5 and 3.6.

Table 3.3 Results of Example 3.5($n = 2$) by the Algorithm($s = 2$)

N-p-q-r	NF	NG	T	NSP	FBEST
15-1-3-3	141	66	0.5	1	.738701490
40-1-3-3	649	87	2.0	2	.733701252
30-2-10-4	226	123	1.2	1	.738701490
30-3-3-5	4094	2266	4.75	4	.729021968
15-1-3-9	1372	615	3.75	3	.729021968
30-3-2-6	3999	2115	4.5	4	.729021968
60-3-3-5	4558	2539	5.1	2	.729021968

Table 3.4 Results of Example 3.5($n = 3$) by the Algorithm($s = 2$)

N-p-q-r	NF	NG	T	NSP	FBEST
15-1-3-3	171	83	0.5	1	-1.06868404
40-1-3-3	487	208	1.6	1	-1.06868404
30-2-10-4	228	125	1.2	1	-1.06868404
30-3-3-5	961	555	2.0	1	-1.06868404
15-1-3-9	299	143	1.25	1	-1.06868404
30-3-2-6	1262	738	3.5	1	-1.06868404
60-3-3-5	1976	1102	5.5	1	-1.06868404
60-4-3-10	7556	4341	19.5	2	-1.08387466

Table 3.5 Results of Example 3.5($n = 2$) by the Algorithm($s = 3$)

N-p-q-r	NF	NG	T	NSP	FBEST
15-1-3-3	185	109	0.5	1	0.73870149
40-1-3-3	425	243	1.0	1	0.73870149
30-2-10-4	352	184	0.5	1	0.73239742
30-3-3-5	6758	3701	8.0	5	0.72773712
15-1-3-20	3721	2317	8.3	4	0.72814205
15-4-3-6	5034	2586	5.5	4	0.72773712
15-5-3-6	7077	3609	8.0	6	0.72750432

Table 3.6 Results of Example 3.5($n = 3$) by the Algorithm($s = 3$)

N-p-q-r	NF	NG	T	NSP	FBEST
15-1-3-3	154	96	0.5	1	-1.06868404
40-1-3-3	862	572	2.5	2	-1.08436242
30-2-10-4	340	184	1.0	1	-1.06868404
30-3-3-5	1369	823	3.0	1	-1.06868404
15-1-3-20	1498	991	3.75	2	-1.08387466
15-4-3-6	4993	2838	11.0	4	-1.08475457
15-5-3-6	1084	600	2.0	1	-1.06868404
60-4-3-10	16848	10093	28.0	3	-1.08693563
100-3-10-10	8682	5434	17.0	3	-1.08952663

The above tables tell us that our algorithm is not efficient for the test function (7). For this kind of highly oscillating functions, special techniques must be found to construct faster algorithms.

4. Conclusion

The algorithm proposed is a simple extension of the original multistart algorithm. By using quasirandom, rather than random, sample points better coverage of the feasible region can be achieved. As found by previous authors concentration and reduction of the sample eliminates unnecessary local searches. Although our algorithm is simpler in form than some clustering algorithms it performs as well or better on several test functions.

References

[1] L.C.W. Dixon. *Global optima without convexity.* Technical Report, Numerical Optimization Centre, Hatfield Polytechnic, Hatfield, England, 1978.

[2] L.C.W. Dixon and G.P. Szegö. *The global optimization: An introduction,* in: Dixon and Szegö, eds., *Towards Global Optimisation 2*, North-Holland, Amsterdam, 1978, pp. 1-15.

[3] H. Faure. *Discrépance de suites associés ê un syst/'eme de num/'eration (en dimension s),* *Acta Arithmetica*, 61 (1982) 337-351.

[4] J.K. Hartman. *Some experiments in global optimization.* Naval Research Logistics Quarterly 20(1973)569-576.

[5] F. J. Hickernell and K. T. Fang. *Combining quasi-random search and Newton-like methods for nonlinear equations.* Technical Report MATH-037, Hong Kong Baptist College, 1993.

[6] L. K. Hua and Y. Wang. *Application of Number Theory to Numerical Analysis.* Springer-Verlag and Science Press, Berlin and Beijing, 1981.

[7] A.H.G. Rinnooy Kan and G.T. Timmer. *Stochastic global optimization methods: Part I: Clustering methods. Mathematical Programming* 39(1987) 27-56.

[8] A.H.G. Rinnooy Kan and G.T. Timmer. *Stochastic global optimization methods: Part II: Multi-level methods. Mathematical Programming* 39(1987) 57-78.

[9] A.H.G. Rinnooy Kan and G.T. Timmer. *Global Optimization.* in: G.L. Nemhauser, A.H.G. Rinnooy Kan and M.J. Todd, eds., *Handbooks in Operations Research and Management Science: Vol. I: Optimization*, (North-Holland, Amsterdam, 1989) pp. 631-662.

[10] D. Q. Mayne and C. C. Meewella. *A non-clustering multistart algorithm for global optimization. Analysis and Optimization of Systems*, A. Bensoussan and J. L. Lions, ed. Lecture Notes in Control and Information Sciences, Vol. 111, Springer-Verlag, Berlin, 1988.

[11] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* SIAM, Philadelphia, 1992.

[12] H. Niederreiter and P. Peart. *Localization of search in quasi-Monte Carlo methods for global optimization. SIAM J. Sci. Stat. Comput.*, 7 (1986) 660-664.

[13] S. L. Smith, E. Eskow, and R.B. Schnabel. *Large adaptive, asynchronous stochastic global optimization algorithms for sequential and parallel computation.* in: T. Coleman and Y. Li, eds., *Large-Scale Numerical Optimization* (SIAM, Philadelphia, 1990) pp. 207-227.

[14] A. Törn. *A search-clustering approach to global optimization.* in: Dixon and Szegö, eds., *Towards Global Optimisation 2* (North-Holland, Amsterdam, 1978), pp. 49-62.

[15] A. Törn and A. Žilinskas. *Global Optimization.* Lecture Notes in Computer Science, Vol. 350, G. Goos and J. Hartmanis, eds., Springer-Verlag, Berlin, 1987.

[16] Y. Wang and K.T. Fang. *A sequential numer-theoretic method for optimization and its application in statistics.* in: L. Wang and Y. Wang, eds., *Lecture Notes in Contemporary Mathematics* (Science Press, Beijing, 1990).