A Numerical Comparison of Some Modified Controlled Random Search Algorithms

Montaz Ali

Turku Centre for Computer Science (TUCS), Turku, Finland

Aimo Törn

Department of Computer Science, Åbo Akademi University, Finland

Sami Viitanen

Department of Computer Science, Åbo Akademi University, Finland

Turku Centre for Computer Science
TUCS Technical Report No 98
March 1997

ISBN 951-650-971-1
ISSN 1239-1891

Abstract

In this paper we propose a new version of the Controlled Random Search (CRS) algorithm of Price [13, 14, 15]. The new algorithm has been tested on thirteen global optimization test problems. Numerical experiments indicate that the resulting algorithm performs considerably better than the earlier versions of the CRS algorithms. The algorithm, therefore, could offer a reasonable alternative to many currently available stochastic algorithms, especially for problems requiring 'direct search' type methods. Also a classification of the CRS algorithms is made based on 'global technique' – 'local technique' and the relative performance of classes is numerically explored.

Keywords: Global optimization, β-distribution, controlled random search

1 Introduction

A global optimization algorithm aims at finding a global minimizer or its close approximation of a function $f : S \subset R^n \rightarrow R$. A point x^* is said to be a global minimizer of f if $f^* = f(x^*) \leq f(x)$, $\forall x \in S$. We assume that the function f is essentially unconstrained, i.e., all of its minimizers are in the interior of S. For practical interests, especially in applied sciences and in engineering, it is required that an approximation $\hat{x}^*$ of a global minimizer of f be found with $|f^* - f(\hat{x}^*)| < \varepsilon$. However, in many applications the function of interest is not differentiable and it is with this view in mind the CRS algorithm was initially developed. CRS is a 'direct search' technique and purely heuristic. It is a kind of contraction process where an initial sample of N points is iteratively contracted by replacing the worst point with a better point. The replacement point is either determined by a global or a local technique. The global technique is an iterative process in which a trial point, the 'new' point, is defined in terms of $n + 1$ points selected from the whole current sample of N points until a replacement point is found. Some CRS algorithms also apply a local technique where the replacement point is searched for near a subset of best points in the current sample.

In the original version of CRS, CRS1 [13], a simplex is formed from the subset of $n + 1$ points. One of the points of the simplex is reflected in the centroid of the remaining points (as in [11]) to obtain a new trial point and this process is repeated until a replacement point is found. The contraction phase ends when some stopping condition is met. In the second version, CRS2 [14], a more sophisticated use is made of the simplexes in obtaining new trial points. In the third version of the algorithm, CRS3 [15], a Nelder-Mead-type local technique is incorporated.

Performances of the CRS were slightly enhanced in [10] by introducing a larger simplex and by taking the weighted centroid rather than the geometric centroid of n points. In order to make the CRS algorithm more efficient, modifications to the CRS2 algorithms have been suggested in [2]. As a result two new versions, namely the CRS4 and CRS5 algorithms, were proposed which improved the CRS algorithms significantly both in terms of the number of function evaluations and run time. Of the two new versions the CRS4 algorithm proved robust when applied to some difficult practical problems [1, 3].

In CRS4 a modification to CRS2 was sought by exploring the region around the best point using a β-distribution [5] instead of carrying out local searches as in CRS3. The CRS5 algorithm uses a gradient based local search with a pre-set probability instead of a Nelder-Mead-type local technique as in CRS3 or the use of β-distribution as in CRS4. Both CRS4 and CRS5 use Hammersley sequences [8] rather than a uniform distribution to select the initial N sample points. The aim in using the Hammersley distribution in generating the initial points was to explore the search region more evenly. However, recently it was shown in [17] that although the distribution of Hammersley points is fairly even for regions of lower dimensions it is not necessarily so for higher dimensions. The use of β-distribution in CRS4 was a salient feature whereby it explores the region S when the N points are sparse and expedite its convergence as soon as they form a dense cluster. In our new approach we modify CRS4 by replacing its simplex approach as global technique with a quadratic approximation [12].

2 CRS6, a New CRS Algorithm

The algorithm like the other CRS algorithms uses a set A of N points generated at random in S. This set is then successively transformed until the function values of all points in A are close enough. The transformation has a global part and a local part. The global part consists of choosing three points from A instead of $n+1$, i.e., the best point, call it $r_1 = (r_{11}, r_{12}, \ldots, r_{1n})$, and two other points at random, call them r_2 and r_3. For each coordinate i the algorithm determines the minimum point $p_{.i}$ of the quadratic through the points r_{1i}, r_{2i}, r_{3i} giving the new trial point $p = (p_{.1}, p_{.2}, \ldots, p_{.n})$. If this new point is better than the worst point in A it replaces the worst point. If it is a new best point then M local steps (the local part) using an appropriate β distribution are made otherwise a new p is determined. This is then repeated until the stopping condition is fulfilled.

1. Generate N uniformly distributed random points and store the points and corresponding function values in an array A. Find the best and worst points in A and their functions values: l, h, f_l, f_h. (The worst and best point in the array are the points with highest and lowest function values respectively).

2. Choose randomly two distinct points r_2, r_3 from A; $r_2, r_3 \neq l$ and set $r_1 = l$. Compute $p = (p_{.1}, p_{.2}, \ldots, p_{.n})$ as

$$p_{.i} = \frac{1}{2}\left[\frac{(r_{2i}^2 - r_{3i}^2)f(r_1) + (r_{3i}^2 - r_{1i}^2)f(r_2) + (r_{1i}^2 - r_{2i}^2)f(r_3)}{(r_{2i} - r_{3i})f(r_1) + (r_{3i} - r_{1i})f(r_2) + (r_{1i} - r_{2i})f(r_3)}\right],$$

   $i = 1, 2, \ldots, n$ where $r_j = (r_{j1}, r_{j2}, \ldots, r_{jn})$, $j = 1, 2, 3$.

3. If $p \notin S$ go to step 2. Evaluate f_p. Ir $f_p \geq f_h$ go to step 2.

4. Replace h by p in A and find h, f_h in new A. If $f_p < f_l$ then set p, f_p as new l, f_l in A; else go to step 6.

5. Repeat M times or until the stopping condition is satisfied: Choose a new trial point $p = (x_1, x_2, \ldots, x_n)$ from an appropriately scaled β-distribution as follows. Each $x_i, i = 1, 2 \ldots, n$, is found from the β-distribution with mean the i^{th} coordinate of the current best point $l = (l_1, l_2, \ldots, l_n)$ and standard deviation

$$s = \delta d \ , \tag{1}$$

   where

$$d = |l_i - h_i|,$$

$$h = (h_1, h_2, \ldots, h_n)$$

   and δ is a user-supplied parameter. Evaluate f_p. If $f_p < f_h$ replace h by p in A, find h, f_h (and l, f_l if $f_p < f_l$) in new A.

6. If the stopping condition is satisfied stop; else go to step 2.

Remarks

1. A suggested value for N, the number of points in the array A, is $N = 10(n+1)$. It is however a heuristic choice and therefore could always be increased for obtaining the global minimum with higher probability. This might be important when solving problems with large n.

2. M is the number of points generated from the β-distribution in step 5 of the CRS6 algorithm. The effect of M was examined by taking it as a fixed integer and also as a variable M_ν. For variable M, initially we

set $M_\nu = 0$ and if a new best point is found, M_ν is increased by one, $\lfloor \gamma M_\nu \rfloor$ trial points are then generated from the β-distribution and so on until the algorithm stops. By using variable M the algorithm could switch from global explorations in early stages to progressively more local searches in later stages. We carried out an extensive series of tests to see the effects of varying M and the results are reported in [1].

3. δ in (1) is a preset real number supplied by the user. It was tested on test problems as well as on practical problems. The most suitable values of δ were empirically found to lie roughly in the interval $[0.05, 0.2]$ with the overall best value equal to 0.1 (see [1]).

4. β-distribution: The β-distribution on $(0,1)$ has the probability density function given by

$$d(x) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1}, \quad 0 \le x \le 1, \quad \alpha, \beta > 0 \ , \qquad (2)$$

   with mean $\frac{\alpha}{\alpha+\beta}$ and variance $\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$. We use the algorithm of Cheng [5] for generating the β-variates. The values of α and β are determined from the given mean and given standard deviation as follows: Let $K_i = \overline{x}_i - \underline{x}_i$ be the difference between the upper and lower limits on the i^{th} coordinate. Let $\theta = (l_i - \underline{x}_i)/K_i$ and $A = [K_i^2\theta(1-\theta)/s^2] - 1$, then $\alpha = A\theta$, and $\beta = A(1-\theta)$.

   Clearly, the calculated parameters α and β for the β-distribution can take both positive and negative values but to get reasonable distributions we restrict them to values greater than or equal to one by 'clipping'. Notice that in the limiting case $\alpha = \beta = 1$, the β-distribution is a uniform distribution. Hence, if our required standard deviation is high we will merely be generating a realisation from a uniform distribution.

5. The algorithm stops when all points in the array A are within agreement to a fixed number of decimal places, i.e. $|f_l - f_h| < \epsilon_o$, where ϵ_o is a preset small number. For all numerical calculation we took $\epsilon_o = 10^{-4}$.

3 Numerical Investigation and Comparisons

In this section we compare the CRS6 algorithm with the original CRS1-CRS4 algorithms. CRS5, being a gradient based algorithm, is excluded from this comparison as we are giving emphasis only on non-gradient based algorithms. Moreover, within the previously modified versions the CRS4 algorithms proved to be the best one in terms of the number of function evaluations, run time and reliability in finding the global solution. Such an investigation has already been carried out for some standard test problems as well as for some complex practical problems [1, 3]. We have coded all algorithms and run them on a SUN Sparc ELC workstation. Random numbers were generated using the well tested procedure given in [16].

3.1 Test Problems

The 13 test problems have been taken from different global optimization text sources, see Table I. For the convenience of the reader some characteristics of the problems are given in the table, i.e., dimensionality n, region of interest S, number of known local minima #min, probability to find the global minimum by a gradient method starting from a random point P^*, global minimum f^*.

Our numerical experiments showed that the problems KL and PW have only one minimum. In fact for PW the result was confirmed both analytically and numerically. We therefore argue that these problems should not be used as test problems. We have, however, used them here to see what accuracies could be achieved by our CRS algorithm. Notice that P^* for P16 is rather small compared to the P^* of the other problems.

3.2 Test Results

To begin with, we compare CRS6 with CRS4 on these problems for three independent test runs. The results are summarized in Table II. In order to facilitate run time comparisons for runs made on different computers we give the time of 1000 evaluations of the test function S5 which is 0.07. For an initial comparison of the algorithms we have used γM_ν, $\gamma = 1$ to generate points from the β-distribution. For all calculations we took $\delta = 0.1$ (see remark 3 in Section 2). In Table II FE denotes the average number of function evaluations, cpu the average run times, f_w^* the worst of three runs and LM

Table 1: The Test Problems

Function	n	Region S	#min	P^*	f^*	Ref
Branin (BR)	2	$-5 \leq x_1 \leq 10$	3	1.00	0.3978	[1]
		$0 \leq x_2 \leq 15$				
Goldprice(GP)	2	$-5 \leq x_i \leq 10$	4	0.40	3.0000	[1]
Shekel5 (S5)	4	$0 \leq x_i \leq 10$	4	0.35	-10.1532	[1]
Shekel7 (S7)	4	$0 \leq x_i \leq 10$	7	0.35	-10.4029	[1]
Shekel10 (S10)	4	$0 \leq x_i \leq 10$	10	0.35	-10.5364	[1]
Hartman3 (H3)	3	$0 \leq x_i \leq 1$	4	0.70	-3.8627	[1]
Hartman6 (H6)	6	$0 \leq x_i \leq 1$	4	0.70	-3.3223	[1]
Schubert3 (P8)	3	$-10 \leq x_i \leq 10$	5^3	0.35	0.0000	[2]
Schubert5 (P16)	5	$-5 \leq x_i \leq 5$	15^5	0.05	0.0000	[2]
Levy10 (L10)	10	$-10 \leq x_i \leq 10$	10^{10}	0.85	0.0000	[3]
Kowalik (KL)	4	$0 \leq x_i \leq 0.42$	1	1.00	0.0003	[3]
Hosaki (HK)	2	$0 \leq x_1 \leq 5$	2	0.65	-2.3460	[4]
		$0 \leq x_2 \leq 6$				
Powell(PW)	4	$-10 \leq x_i \leq 10$	1	1.00	0.0000	[3]

P^* is the probability to find the global minimum based on steepest descent
[1]:(Dixon,Szegö), [2]:(Dekkers,Aarts), [3]:(Jansson,Knüppel), [4]:(Bekey,Ung)

the number of failures to locate the global minimum in three independent runs. Because the results in the individual runs varied only little three runs were considered enough for obtaining reliable average behaviour. The average results were calculated for data for which the global minima were achieved.

From the total figures in Table II it is clear that CRS6 reduces the number of function evaluations and cpu time by about 60% and 50% respectively. For CRS6 the values obtained for f_w^* are the same as for CRS4 except for problem PW.

Next we compare both algorithms for variable and for fixed values of M. We totalled the average results for three independent runs, see Table III. PL is the problems for which only a local minimum was found and LM is the count of these.

From Table III it is clear that CRS6 reduced the computational efforts significantly while obtaining all the global minima almost as successfully as CRS4. Notice from Tables 2 and 3 that for CRS6 the total results differs only little for variables $\lfloor \gamma M_\nu \rfloor$ with $\gamma = 1$ and 0.5, except that CRS6 with

Table 2: Comparison of CRS4 and CRS6

	CRS4				CRS6			
	FE	cpu	f_w^*	LM	FE	cpu	f_w^*	LM
BR	354	0.06	0.3978	0	205	0.05		0
GP	433	0.06	3.0000	0	152	0.05		0
S5	1364	0.17	-10.1487	1	526	0.10		1
S7	1551	0.23	-10.3984	0	538	0.11		0
S10	1670	0.28	-10.5319	0	522	0.12		0
H3	565	0.11	-3.8627	0	218	0.08		0
H6	1802	0.52	-3.3223	0	526	0.17		1
P8	824	0.12	0.0000	0	238	0.06		0
P16	1392	0.25	0.0000	0	415	0.11		0
L10	3102	1.23	0.0000	0	1160	0.41		0
KL	462	0.10	0.0003	0	169	0.05		0
HK	267	0.05	-2.3458	0	141	0.03		0
PW	1336	0.17	0.0000	0	963	0.37	0.0981	0
Total	15122	3.35		1/39	5773	1.71		2/39

FE: number of function evaluations; LM: number of non-global minima

$\gamma = 1$ failed more often. This indicates that the number of failures increases with M which is to be expected since with larger M more local exploration is done giving less global cover.

3.3 Effects of Global and Local Techniques

The essential difference between CRS4 and CRS6 is the use of quadratic approximations in CRS6 as opposed to the use of simplexes in CRS4, while both are using the β-distribution. To see how the effects of the proposed modification to the CRS4 algorithm prevails its superiority over the others, we carried out a comprehensive numerical comparison. In order to facilitate the understanding and to make the differences between the methods more explicit we append to each individual algorithm name the appropriate parameters containing '(global technique, local technique)'. Using this otation we thus write CRS2, CRS4 and CRS6 as CRS(Δ,$\cdot$), CRS(Δ, β) and CRS(q, β).

Motivated by the results of CRS(q, β) and since CRS(Δ, β) itself replaces the local search in CRS(Δ,N-M) (N-M = Nelder-Mead) with β-distribution

Table 3: Comparison of CRS4 and CRS6 over total figures

	FE	cpu	LM	PL	M
CRS4	19162	4.13	2/39	S5,S5	3
CRS6	6605	1.69	3/39	S5,S10,H6	3
CRS4	19032	4.08	1/39	S5	$\lfloor 0.5M_\nu \rfloor$
CRS6	5713	1.49	1/39	H6	$\lfloor 0.5M_\nu \rfloor$

PL: Functions for which non-global minima were obtained

sampling we performed more experiments using quadratic approximation. We thus also compare with CRS(q,·), CRS(Δ, q) and with CRS(q, q). In both CRS(Δ, q) and CRS(q, q) we use quadratic approximation using the three best points when a point found by the simplex for the former and by the quadratic for the latter has function value less than or equal to the third best within A. (Note that a quadratic approximation through the three best points does not guarantee an improved solution.) We do not order the entire array but keep the three best points on the bottom of the array A and the worst point at the top. For the others we keep the worst and the best at the top and bottom of A. The results are shown in Table IV where the parameters for CRS(Δ, β) and CRS(q, β) are $0.5M_\nu$ and $\delta = 0.1$. To make the comparison of the methods easier we have also totalled the (average) number of function evaluations. Except for CRS(Δ,·) the others fail occasionally to produce global minima for some functions. Although CRS(q, q) proved to be the runner-up in terms of FE it fails three times in 39 runs. However, the overall winner is CRS(q, β).

In Table V we compare the five best CRS algorithms from Table IV based on total figures over the problems of Table I. As can be seen from the table the dominant factor is the introduction of both the β-distribution and the quadratic approximation in combination. It is also clear that only introducing the quadratic approximation has already improved the results considerably. However, as Table V indicates, the CRS(q, β) algorithm outperforms the rest in terms of both FE and cpu.

Table 4: Comparison of CRS versions based on FE for functions of Table I

CRS	(Δ,N-M)	($\Delta,\cdot$)	(Δ,q)	(Δ,β)	($q,\cdot$)	(q,q)	(q,β)
BR	415	694	544	478	211	222	198
GP	543	738	582	490	165	150	182
S5	2754	2986	2848	2027	830	741	680
S7	2652	2929	2308	1614	863	743	505
S10	2857	2970	2467	2033	585	662	509
H3	939	929	694	733	246	232	230
H6	6423	3855	3242	2378	995	704	645
P8	1371	1429	1078	872	267	287	234
P16	2973	3318	2612	1604	514	511	450
L10	27054	8988	6474	4395	1646	1630	1014
KL	895	508	472	491	243	235	137
HK	425	512	389	334	154	126	175
PW	3245	2184	1616	1583	808	1180	754
PL	S5	-	S5	S5	S5,S5*	S5,S7,S10	H6
Total	52546	32040	25326	19032	7527	7423	5713

$*$: One of the runs did not converge

4 Discussion and Conclusion

We have developed a new CRS method for global optimization. In this the simplex search originally proposed by Price is replaced by quadratic search and β-distribution sampling. The performance of the new method is quite effective and efficient.

The comparisons in the previous chapter are fair in the sense that we have used the same stopping condition. Although one could argue that the extrapolation from numerical experiments has to be treated with caution since the test functions used are well defined mathematical functions some of which having a small number of local minima, our previous experiments with complex practical problems suggest that these methods are very effective and superior to many currently available stochastic methods ([3]). Consequently we feel that the method could be used as a general purpose global optimization technique. Research is continuing to develop even more efficient CRS methods.

A disturbing fact concerning CRS algorithms is their totally heuristic

Table 5: Comparison on total figures

	FE	cpu	LM	PL
CRS(Δ, q)	25326	9.87	1/39	S5
CRS(Δ, β)	19032	4.08	1/39	S5
CRS($q, \cdot$)	7527	2.56	2/39	S5,S5*
CRS(q, q)	7423	3.02	3/39	S5,S7,S10
CRS(q, β)	5713	1.49	1/39	H6

$*$: One of the runs did not converge

nature with no theoretical convergence properties. It cannot be asserted that the algorithm will converge in probability to the global minimum for $\epsilon_o = 0$. However, convergence in probability is easily achieved for the CRS algorithms by adding an alternative random sampling step in the global technique (Step 2 of CRS6) and by letting this step be applied with a probability decreasing with each application.

References

[1] Ali, M. M.:1994, 'Some Modified Stochastic Global Optimization Algorithms with Applications', Loughborough University of Technology, *Ph.D Thesis*.

[2] Ali, M. and Storey, C.:1994, 'Modified Controlled Random Search Algorithms', *International Journal of Computer Mathematics*, 53, pp. 229-235.

[3] Ali, M. M, Storey, C. and Törn, A.:1995, 'Application of some stochastic global optimization algorithms to practical problems', submitted to the *Journal of Optimization Theory and Applications*.

[4] Bekey, G. A. and Ung, M. T.:1974, 'A Comparative Evaluation of Two Global Search Algorithms', *IEEE Trans. Syst. Man, Cybern.*, SMC-4, No. 1, pp. 112-116.

[5] Cheng, R. C. H.:1978, 'Generating Beta Variates with Nonintegral Shape Parameters', *Communications of the ACM*, 21, pp. 317-322, 1978.

[6] Dekkers, A. and Aarts, E.:1991, 'Global Optimization and Simulated Annealing', *Mathematical Programming*, 50, pp. 367-393.

[7] Dixon, L. C. W., and Szegö, G. P. (eds.):1978, *Towards Global Optimization 2*, North-Holland, Amsterdam.

[8] Halton, J. H.:1960, 'On the Efficiency of certain Quasi-random Sequences of Points in Evaluating Multi-dimensional Integrals', *Numerische Mathematik*, 2, pp.84-90.

[9] Jansson, C. and Knüppel, O.:1995, 'A Branch and Bound Algorithm for Bound Constrained Optimization Problems without Derivatives', *Journal of Global Optimization*, 7, pp.297-331.

[10] Mohan, C. and Shanker, K.:1988, 'A Numerical Study of Some Modified Versions of Controlled Random Search Method for Global Optimization', *International Journal of Computer Mathematics*, 23, pp.325-341.

[11] Nelder, J. A. and Mead, R.:1965, 'A Simplex Method for Function Minimization', *The Computer Journal*, 7, pp.308-313.

[12] Palosaari, S. M., Parviainen, S., Hiironen, J., Reunanen, J. and Neittaanmaki, P.:1986, 'A random Search Algorithm for Constrained Global Optimization', *Acta Polytechnica Scandinavica*, Chemical Technology at Metallurgy Series No. 172, Helsinki, pp.1-45.

[13] Price, W. L.:1978, 'A Controlled Random Search Procedure for Global Optimization', *Towards Global Optimization 2*, Edited by L.C.W. Dixon and G.P. Szegö, North-Holland, Amsterdam, Holland, pp.71-84.

[14] Price, W. L.:1978, 'Global Optimization by Controlled Random Search', *Journal of Optimization Theory and Applications*, 40, pp.333-348.

[15] Price, W. L.:1978, 'Global Optimization Algorithms for a CAD Workstation', *Journal of Optimization Theory and Applications*, 55, pp.133-146.

[16] Tezuka, S. and L'Ecuyer, P.:1991, 'Efficient and Portable Combined Tausworthe Random Number Generators', *ACM Transactions on Modelling and Computer Simulation*, 1, pp.99-112.

[17] Törn, A. and Viitanen, S.:1996, 'Iterative Topographical Global Optimization', in: C.A. Floudas and M. Pardalos (Eds.) *State of the Art in Global Optimization, Computational Methods and Applications*, Kluwer Academic Publishers, Dordrecht, pp.353-363.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**