Article

An improved Controlled Random Search method

Vasileios Charilogis [1], Ioannis Tsoulos [2*], Alexandros Tzallas [3], Nikolaos Anastasopoulos [4]

[1] Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece; v.charilog@uoi.gr
[2] Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece; itsoulos@uoi.gr
[3] Department of Informatics and Telecommunications, University of Ioannina, 47100 Arta, Greece; tzallas@uoi.gr
[4] Computer Engineering and Information Department, University of Patras, Greece; ece8268@upnet.gr

Abstract: A modified version of common global optimization method named Controlled Random Search is presented here. This method is designed to estimate the global minimum of multidimensional symmetric and asymmetric functional problems. The new method modifies the original algorithm by incorporating a new sampling method, a new termination rule and the periodical application of local search optimization algorithm to the points sampled. The new version is compared against the original using some benchmark functions from the relevant litearature.

Citation: Charilogis, V.; Tsoulos, I.G.; Tzalas, A.; Anastasopoulos, N. An improved Controlled Random Search method. *Symmetry* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

1. Introduction

Global optimization[1] is considered a problem of high complexity with many applications. The problem is defined as the location of the global minimum of a multi - dimensional function $f(x)$:

$$x^* = \arg\min_{x \in S} f(x) \tag{1}$$

Where $S \subset R^n$ is formulated as:

$$S = [a_1, b_1] \otimes [a_2, b_2] \otimes \ldots [a_n, b_n] \tag{2}$$

In global optimization many functional problems that need to be solved can have symmetric solutions - minimus without this being the rule. The location of the global optimum finds application in many areas such as physics [2,3], chemistry [6,7], medicine [4,5], economics[8] etc. In modern theory there are two different categories of global optimization methods: the stochastic methods and the deterministic methods. The first category contains the vast majority of methods such as Simulated Annealing methods [9–11], Genetic Algorithms [12–14], Tabu Search methods [15], Particle Swarm Optimization [16–18] etc. A common method that also belongs to stochastic methods is the Controlled Random Search (CRS) method [19], which is a procedure that uses a population of trial solutions. This method initially creates a set with randomly selected points and repeatedly replaces the worst point in that set with a randomly generated point and this process can continue until some termination criterion is satisfied. The CRS method has been used intensively in many problems such as geophysics problems [20,21], optimal shape design problems [22], the animal diet problem [23], the heat transfer problem [24] etc.

This CRS method has been thoroughly analyzed by many researchers in the field such as the work A of Ali and Storey where two new variants of the CRS method are proposed [25]. These variants have proposed alternative techniques for the selection of the initial sample set and usage of local search methods. Also, Pillo *et al* [26] suggested a hybrid CRS method where the base algorithm is combined with a Newton - type unconstrained minimization algorithm [27] to enhance the efficiency of the method in various test problems. Another work is of Kaelo and Ali , in which they suggested

[28] some modifications on the method and especially in the new point generation step. Also, Filho and Albuquerque have suggested [29] the usage of a distribution strategy to accelerate the Controlled Random Search method. Tsoulos and Lagaris [30] suggested the usage of a new line search method based on Genetic Algorithms to improve the original CRS method. The current work proposed three major modifications in the CRS method: a new point replacement strategy, a stochastic termination rule and a periodical application of some local search method. The first modification is used to better explore the domain range of the function. The second modification is made in order to achieve a better termination of the method without wasting valuable computational time. The third modification is used in order to speed up the method by applying a small amount of steps of a local search method. The new method introduces a new method to create trial points that was not present in the previous work [30] and also replaces the expensive call to line search method with a few calls to a local search optimization method.

The rest of this article is organized as follows: in Section 2 the major steps of the CRS method as well as the proposed modifications are presented, in section 3 the results from the application of the proposed method on a series of benchmark functions are listed and finally in section 4 some conclusions and guidelines for future research are presented.

2. Method Description

The Controlled Random Search has a series of steps that are described in Algorithm 1. The changes proposed by the new method focus on three points:

1. The creation of a test point (**New_Point** step) is performed using a new procedure described in subsection 2.1.
2. In the **Min_Max** step the stochastic termination rule described in subsection 2.2 is used. The aim of this rule is to terminate the method when, with some certainty, no lower minimums are to be found.
3. Apply a few steps of a local search procedure after **New_Point** step in the $\tilde{z}$ point. This procedure is used to bring the test points closer to the corresponding minimums. This speeds up the process of searching for new minima, although it obviously leads to an increase in function calls

2.1. A new method for trial points

The proposed technique to compute the trial point $\tilde{z}$ is shown in Algorithm 2. According to this, the calculation of the test point $\tilde{z}$ does not contain product with high values as in the basic algorithm, so that the test point is not too far from the centroid. This technique avoids vector jumps from the centroid, where it has great gravity in the calculation for starting the local optimization. This method considers in the calculation also the current minimum point and not only a random point as in the original technique. With this modification, knowledge that has already been found in the past is used to create a new point and in such a way that it is close to the area of attraction of a local minimum.

2.2. A new stopping rule

It is quite common in the optimization techniques to use a predefined number of maximum iterations as the stopping rule of the method. Even though this termination rule is easy to implement sometime could require an excessive number of functions calls before termination and a more sophisticated termination rule is needed. The termination rule proposed here is inspired from [31]. At every iteration k the variance $\sigma^{(k)}$ of the quantity $f_{\min}$ is calculated. If the optimization technique did not manage to find a new estimation of the global minimum for some iterations, then probably the global minimum has been discovered and the algorithm should terminate. The termination rule is defined as follows, terminate when

Initialization Step:

1. **Set** the value for the parameter N. Typically this value could be set to $N = 25n$.
2. **Set** ϵ as a small positive value, used in comparisons.
3. **Create** randomly the set $T = \{z_1, z_2, ..., z_N\}$ from S.

Min_Max Step:

1. **Calculate** the points $z_{\text{min}} = \text{argmin} f(z)$ and $z_{\text{max}} = \text{argmax} f(z)$ and their function values
$$f_{\text{max}} = \max_{z \in T} f(z)$$
and
$$f_{\text{min}} = \min_{z \in T} f(z)$$
2. **If** $|f_{\text{max}} - f_{\text{min}}| < \epsilon$, **then goto Local_Search** Step.

New_Point Step:

1. **Select** randomly the reduced set $\tilde{T} = \{z_{T_1}, z_{T_2}, .., z_{T_{n+1}}\}$ from T.
2. **Compute** the centroid G:
$$G = \frac{1}{n} \sum_{i=1}^{n} z_{T_i}$$
3. **Compute** a trial point $\tilde{z} = 2G - z_{T_{n+1}}$.
4. **If** $\tilde{z} \notin S$ **or** $f(\tilde{z}) \geq f_{\text{max}}$ **then** goto New_Point step.

Update Step:

1. $T = T \cup \{\tilde{z}\} - \{z_{\text{max}}\}$.
2. **Goto Min_Max** Step.

Local_Search Step:

1. $z^* = \text{localSearch}(z)$.
2. The final outcome of the algorithm is discovered global minimum z^*.

Algorithm 1: The original Controlled Random search method. The basic steps of the method.

1. **Calculate** the centroid G:
$$G = \frac{1}{n} \sum_{i=1}^{n} z_{T_i}$$
2. Set $G = G + \frac{1}{n} z_{\text{min}}$
3. **Compute** a trial point $\tilde{z} = G - \frac{1}{n} z_{T_{n+1}}$.

Algorithm 2: The steps of the new proposed method to create more efficient trial points for the Controlled Random Search method.

$$\sigma^{(k)} \leq \frac{\sigma^{\left(k_{\textbf{last}}\right)}}{2} \tag{3}$$

The term $k_{\textbf{last}}$ represents the last iteration where a new global minimum was located.

The amount $\sigma^{(k)}$ decreases continuously over time as either the method will find a lower estimate for the global minimum or the global minimum will have already been found. In addition, this quantity is de facto permanently positive and therefore is a good candidate for use in termination criteria. If the global minimum has already been found or the method is no longer able to find a new estimate for it, then this quantity will tend to zero and therefore we can interrupt the execution of the algorithm when this quantity falls below a value. This value may be a fraction of the value of $\sigma^{(k)}$ the last time a new estimate for the global minimum was found. If we want to allow the algorithm to continue for several generations this fraction can be small eg 0.25. If we want it to stop more immediately, a good estimate for the fraction can be 0.75. A good compromise between these prices is the 0.5 price chosen here.

3. Experiments

3.1. Test functions

The modified version of the CRS was tested against the traditional CRS on series of benchmark functions from the relevant literature [32,33]. The following functions were used:

- **Bf1** function defined as:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1) - \frac{4}{10}\cos(4\pi x_2) + \frac{7}{10}$$

with $x \in [-100, 100]^2$.

- **Bf2** function:

$$f(x) = x_1^2 + 2x_2^2 - \frac{3}{10}\cos(3\pi x_1)\cos(4\pi x_2) + \frac{3}{10}$$

  where $x \in [-50, 50]^2$.

- **Branin** function: $f(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$ with $-5 \leq x_1 \leq 10,\ 0 \leq x_2 \leq 15$.
- **CM - Cosine Mixture** function.

$$f(x) = \sum_{i=1}^{n} x_i^2 - \frac{1}{10}\sum_{i=1}^{n}\cos(5\pi x_i)$$

  with $x \in [-1, 1]^n$. In our experiments we have used $n = 4$.
- **Camel** function.

$$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, \quad x \in [-5, 5]^2$$

- **Easom** function.

$$f(x) = -\cos(x_1)\cos(x_2)\exp\left((x_2 - \pi)^2 - (x_1 - \pi)^2\right)$$

  with $x \in [-100, 100]^2$.
- **Exponential** function.

$$f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right), \quad -1 \leq x_i \leq 1$$

In the conducted experiments the values with $n = 2, 4, 8, 16, 32, 64, 100$ were used and the corresponding functions was denoted as EXP2, EXP4, EXP8, EXP16, EXP32, EXP64, EXP100.

- **GoldStein & Price**

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

- **Griewank2** function.

$$f(x) = 1 + \frac{1}{200}\sum_{i=1}^{2} x_i^2 - \prod_{i=1}^{2}\frac{\cos(x_i)}{\sqrt{(i)}}, \quad x \in [-100, 100]^2$$

- **Gkls** function. $f(x) = \mathrm{Gkls}(x, n, w)$, is a function with w local minima, described in [34] with $x \in [-1, 1]^n$ In the conducted experiments we have used $n = 2, 3$ and $w = 50$ and the functions are denoted by the labels GKLS250 and GKLS350.
- **Guilin Hills** function. $f(x) = 3 + \sum_{i=1}^{n}\left(c_i \frac{x_i+9}{x_i+10}\sin\left(\frac{\pi}{1-x_i+\frac{1}{2k_i}}\right)\right)$, $x \in [0, 1]^n$, $c_i > 0$ and k_i being positive integers. In our experiments we have used $n = 5, 10$ with 50 local minima in each function. The produced functions are entitled GUILIN550 and GUILIN1050.
- **Hansen** function. $f(x) = \sum_{i=1}^{5} i\cos[(i-1)x_1 + i]\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]$, $x \in [-10, 10]^2$.
- **Hartman 3** function.

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$$

with $x \in [0, 1]^3$ and $a = \begin{pmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{pmatrix}$$

- **Hartman 6** function.

$$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$$

with $x \in [0,1]^6$ and $a = \begin{pmatrix} 10 & 3 & 17 & 3.5 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{pmatrix}$, $c = \begin{pmatrix} 1 \\ 1.2 \\ 3 \\ 3.2 \end{pmatrix}$ and

$$p = \begin{pmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{pmatrix}$$

- **Rastrigin** function.

$$f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \quad x \in [-1,1]^2$$

- ***Rosenbrock** function*

$$f(x) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \quad -30 \le x_i \le 30.$$

  In our experiments we used this function with $n = 20$.
- **Shekel 7** function.

$$f(x) = -\sum_{i=1}^{7} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

  with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 3 & 5 & 3 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \end{pmatrix}$.
- **Shekel 5** function.

$$f(x) = -\sum_{i=1}^{5} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

  with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \end{pmatrix}$.
- **Shekel 10** function.

$$f(x) = -\sum_{i=1}^{10} \frac{1}{(x-a_i)(x-a_i)^T + c_i}$$

with $x \in [0,10]^4$ and $a = \begin{pmatrix} 4 & 4 & 4 & 4 \\ 1 & 1 & 1 & 1 \\ 8 & 8 & 8 & 8 \\ 6 & 6 & 6 & 6 \\ 3 & 7 & 3 & 7 \\ 2 & 9 & 2 & 9 \\ 5 & 5 & 3 & 3 \\ 8 & 1 & 8 & 1 \\ 6 & 2 & 6 & 2 \\ 7 & 3.6 & 7 & 3.6 \end{pmatrix}$, $c = \begin{pmatrix} 0.1 \\ 0.2 \\ 0.2 \\ 0.4 \\ 0.4 \\ 0.6 \\ 0.3 \\ 0.7 \\ 0.5 \\ 0.6 \end{pmatrix}$.

- **Sinusoidal** function.

$$f(x) = -\left(2.5\prod_{i=1}^{n}\sin(x_i - z) + \prod_{i=1}^{n}\sin(5(x_i - z))\right), \quad 0 \le x_i \le \pi.$$

  In our experiments we used $n = 4, 8, 16, 32$ and $z = \frac{\pi}{6}$ and the corresponding functions are denoted by the labels SINU4, SINU8, SINU16, SINU32.
- **Test2N** function. This function is given by the equation

$$f(x) = \frac{1}{2}\sum_{i=1}^{n} x_i^4 - 16x_i^2 + 5x_i, \quad x_i \in [-5,5].$$

  In the conducted experiments the n has the values 4,5,6,7.
- **Test30N** function. This function is given by

$$f(x) = \frac{1}{10}\sin^2(3\pi x_1)\sum_{i=2}^{n-1}\left((x_i - 1)^2\left(1 + \sin^2(3\pi x_{i+1})\right)\right) + (x_n - 1)^2\left(1 + \sin^2(2\pi x_n)\right)$$

  with $x \in [-10,10]$. The function has 30^n local minima in the specified range and we used $n = 3, 4$ in our experiments.

3.2. Results

In the experiments two different values were measured: the rejection rate in the **New_Point** step and the average number of function calls required. In the first case we measure the percentage of points rejected during the **New_Point** step, i.e. points created that are outside the domain range of the function. All the experiments were conducted 30 times and different seed for the random number generator was used each time. The local search method that used in the experiments and denoted as localsearch(x), was a BFGS variant due to Powell[35]. The experiments were conducted on a i7-10700T CPU at 2.00GHz equipped with 16GB of RAM. The operating system used was Debian Linux and the all the code were compiled using ANSI C++ compiler.

The experimental results are listed in Table 1. The column FUNCTION stands for the name of the objective function, the column CRS-R stands for the rejection rate for the CRS method while the column NEWCRS-R displays the same measure for the current method. Similar, the column CRS-C represents the average function calls for the CRS method and the column NEWCRS-C stands for the average function calls of the proposed method. Also a statistical comparison between the CRS and the proposed method is shown in Figure 1.

The proposed method almost annihilates the rejection rate in every test function. This is an evidence that the new mechanism proposed here to create new point is more accurate than the traditional one. Also, the proposed method requires lower number of function calls than the CRS method as one can deduce from the relevant columns and the statistical comparison. The same information is presented graphically in Figure **??**, where the percentage comparison of times of functional problems is outlined. Also, in

Figure 1. Statistical comparison for the function calls using box plots.

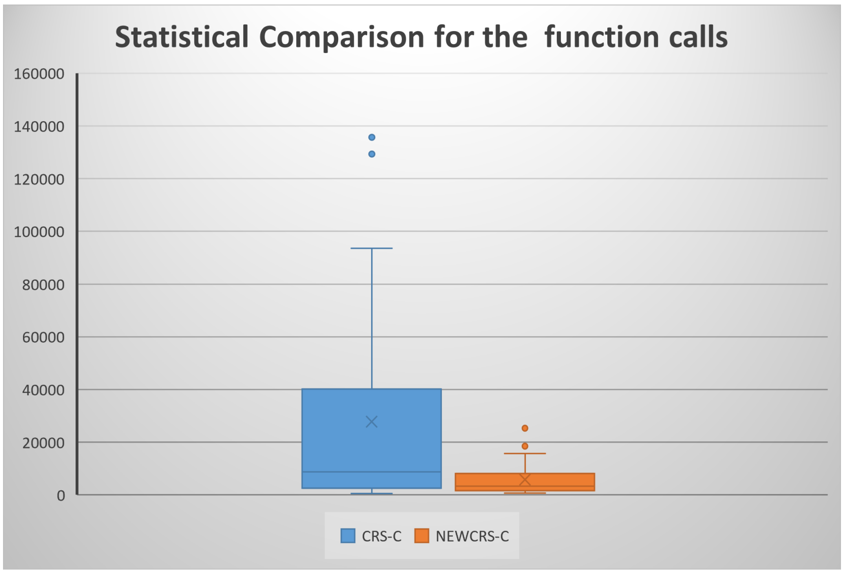
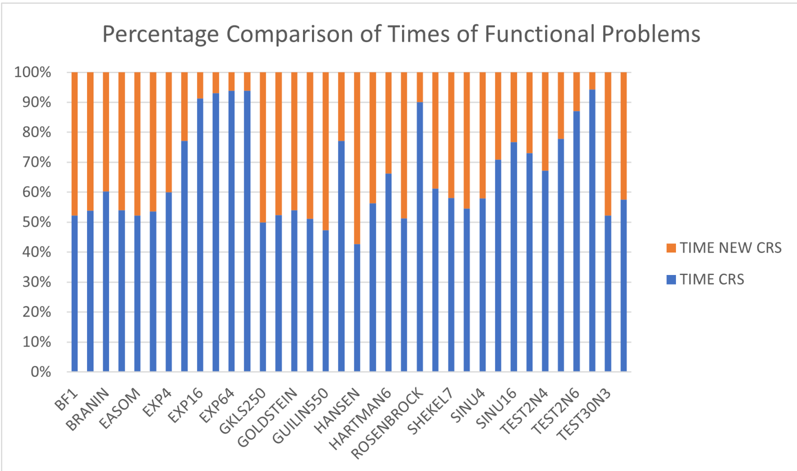


Figure 2. Percentage comparison for time execution between the two methods.



the most difficult problems, the proposed method seems to be even more superior to the original one in number of calls, as the combination of the termination rule together with the improved new point generation technique terminate the method much faster and more correctly than the original method.

Additionally, the execution time for every test function was measured and this information is outlined in Table 2. The column CRS-TIME stands for the average execution time of the original CRS method, the column NEWCRS-TIME represents the average execution time for the proposed method and the column DIFF is the calculated percentage difference between the previously mentioned columns. It is evident that the proposed method requires shorter execution times than the original one and in addition the difference between the two methods is more obvious in large problems. This phenomenon is also reflected in Figure 3, where a graphical representation of the average execution times of the two methods for the EXP problem for a different number of dimensions is made.

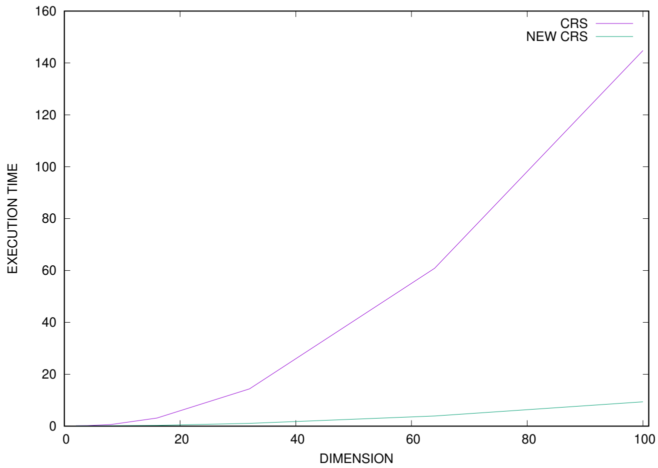
Table 1: Experimenting with rejection rates.

FUNCTION	CRS-R	CRS-C	NEWCRS-R	NEWCRS-C
BF1	1.37%	2523	0.00%	1689
BF2	1.33%	2506	0.17%	1569
BRANIN	16.00%	2014	9.13%	851
CAMEL	1.67%	2235	0.20%	1487
EASOM	51.03%	591	11.43%	635
EXP2	3.03%	1290	0.70%	644
EXP4	2.67%	4688	0.00%	1302
EXP8	2.77%	16453	0.00%	2601
EXP16	4.00%	47400	0.00%	5207
EXP32	7.70%	93520	0.00%	10414
EXP64	18.80%	135638	0.00%	13602
EXP100	38.53%	129327	0.00%	14506
GKLS250	3.87%	1784	0.27%	1684
GKLS350	6.43%	3881	0.03%	2088
GOLDSTEIN	3.60%	2154	0.70%	1829
GRIEWANK2	1.20%	2503	0.03%	2742
GUILIN550	8.33%	9129	0.00%	25333
GUILIN1050	9.63%	30806	0.00	10561
HANSEN	47.60%	2643	4.03%	1736
HARTMAN3	9.97%	3009	6.13%	1331
HARTMAN6	13.37%	13615	0.00%	6091
RASTRIGIN	9.17%	2130	1.33%	2986
ROSENBROCK	0.00%	59024	0.00%	15719
SHEKEL5	4.73%	8974	0.00%	2967
SHEKEL7	3.70%	8606	0.00%	3236
SHEKEL10	2.73%	9264	0.00%	3479
SINU4	3.90%	6525	0.00%	2889
SINU8	5.10%	21561	0.00%	4946
SINU16	8.43%	62194	0.00%	9539
SINU32	14.40%	135986	0.00%	18456
TEST2N4	24.57%	10198	0.00%	3756
TEST2N5	34.17%	20850	0.00%	4806
TEST2N6	42.50%	43290	0.00%	6075
TEST2N7	50.37%	92658	0.00%	7005
TEST30N3	24.10%	4011	0.00%	5691
TEST30N4	27.30%	7432	0.00%	8579
TOTAL	**13.67%**	1000412	**0.86%**	208031

Table 2: Time comparisons.

FUNCTION	CRS-TIME	NEWCRS-TIME	DIFF
BF1	0.168	0.154	8.33%
BF2	0.180	0.154	14.44%
BRANIN	0.209	0.138	33.97%
CAMEL	0.165	0.141	14.55%
EASOM	0.165	0.151	8.48%
EXP2	0.165	0.143	13.33%
EXP4	0.228	0.152	33.33%
EXP8	0.629	0.187	70.27%
EXP16	3.142	0.299	90.48%
EXP32	14.364	1.082	92.47%
EXP64	60.861	3.932	93.54%
EXP100	144.794	9.386	93.52%
GKLS250	0.592	0.593	-0.17%
GKLS350	0.658	0.599	8.97%
GOLDSTEIN	0.191	0.163	14.66%
GRIEWANK2	0.174	0.166	4.60%
GUILIN550	0.475	0.529	-11.37%
GUILIN1050	1.524	0.453	70.28%
HANSEN	0.217	0.292	-34.56%
HARTMAN3	0.21	0.163	22.38%
HARTMAN6	0.514	0.262	49.03%
RASTRIGIN	0.168	0.16	4.76%
ROSENBROCK	5.31	0.584	89.00%
SHEKEL5	0.321	0.203	36.76%
SHEKEL7	0.302	0.218	27.81%
SHEKEL10	0.325	0.271	16.62%
SINU4	0.283	0.206	27.21%
SINU8	0.897	0.369	58.86%
SINU16	4.775	1.448	69.68%
SINU32	24.413	8.999	63.14%
TEST2N4	0.389	0.19	51.16%
TEST2N5	0.733	0.209	71.49%
TEST2N6	1.714	0.256	85.06%
TEST2N7	4.326	0.264	93.90%
TEST30N3	0.222	0.203	8.56%
TEST30N4	0.324	0.239	26.23%
TOTAL	274.127	32.958	**87.98%**

Figure 3. Time comparison between the two methods for the EXP function for a variety of problem dimensions.



4. Conclusions

Three important modifications were proposed in the current work for the CRS method. The first modification has to do with the new test point generation process, which seems to be more accurate than the original one. The new method almost every time creates points that are within the domain range of the function. The second change adds a new termination rule based on stochastic observations. The third proposed modification applies a few steps of a local search procedure to every trial point created by the algorithm. Judging by the results, it seems that the proposed changes have two important effects. The first is that the success of the algorithm in creating valid test points is significantly improved. The second is the large reduction in the number of function calls required to locate the global minimum.

Future research may include the exploration of usage additional stopping rules and the parallelization of different aspects of the method in order to speed up the optimization procedure as well as to take advantage of multicore programming environments.

1. Törn A. and Žilinskas A., Global Optimization Volume 350 of Lecture Notes in Computer Science, Springer, Heidelberg, 1987.
2. Patrice Ogou Yapo, Hoshin Vijai Gupta, Soroosh Sorooshian, Multi-objective global optimization for hydrologic models, Journal of Hydrology 204, pp. 83-97, 1998.
3. Q. Duan, S. Sorooshian, V. Gupta, Effective and efficient global optimization for conceptual rainfall-runoff models, Water Resources Research **28**, pp. 1015-1031 , 1992.
4. Balsa-Canto E., Banga J.R., Egea J.A., Fernandez-Villaverde A., de Hijas-Liste G.M. (2012) Global Optimization in Systems Biology: Stochastic Methods and Their Applications. In: Goryanin I., Goryachev A. (eds) Advances in Systems Biology. Advances in Experimental Medicine and Biology, vol 736. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7210-1_24
5. P. Boutros, A. Ewing, K. Ellrott et al., Global optimization of somatic variant identification in cancer genomes with a global community challenge, Nat Genet **46**, pp. 318–319, 2014.
6. David J. Wales, Harold A. Scheraga, Global Optimization of Clusters, Crystals, and Biomolecules, Science **27**, pp. 1368-1372, 1999.
7. P.M. Pardalos, D. Shalloway, G. Xue, Optimization methods for computing global minima of nonconvex potential energy functions, Journal of Global Optimization **4**, pp. 117-133, 1994.
8. Zwe-Lee Gaing, Particle swarm optimization to solving the economic dispatch considering the generator constraints, IEEE Transactions on **18** Power Systems, pp. 1187-1195, 2003.
9. S. Kirkpatrick, CD Gelatt, , MP Vecchi, Optimization by simulated annealing, Science **220**, pp. 671-680, 1983.
10. L. Ingber, Very fast simulated re-annealing, Mathematical and Computer Modelling **12**, pp. 967-973, 1989.
11. R.W. Eglese, Simulated annealing: A tool for operational research, Simulated annealing: A tool for operational research **46**, pp. 271-281, 1990.
12. D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, Massachussets, 1989.

13. Z. Michaelewicz, Genetic Algorithms + Data Structures = Evolution Programs. Springer - Verlag, Berlin, 1996.
14. S.A. Grady, M.Y. Hussaini, M.M. Abdullah, Placement of wind turbines using genetic algorithms, Renewable Energy **30**, pp. 259-270, 2005.
15. A. Duarte, R. Marti, F. Glover et al., Hybrid scatter tabu search for unconstrained global optimization, Ann Oper Res **183**, pp. 95–123m, 2011.
16. J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.
17. Riccardo Poli, James Kennedy kennedy, Tim Blackwell, Particle swarm optimization An Overview, Swarm Intelligence **1**, pp 33-57, 2007.
18. Ioan Cristian Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters **85**, pp. 317-325, 2003.
19. W.L. Price, Global Optimization by Controlled Random Search, Computer Journal **20**, pp. 367-370, 1977.
20. David N. Smith and John F. Ferguson, Constrained inversion of seismic refraction data using the controlled random search, Geophysics **65**,pp. 1622-1630, 200.
21. Cassiano Antonio Bortolozo, Jorge Luis Porsani, Fernando Acacio Monteiro dos Santos, Emerson Rodrigo Almeida, VES/TEM 1D joint inversion by using Controlled Random Search (CRS) algorithm, Journal of Applied Geophysics **112**, pp. 157-174, 2015.
22. J. Haslinger, D. Jedelsky, T. Kozubek et al, Genetic and Random Search Methods in Optimal Shape Design Problems, Journal of Global Optimization **16**, pp. 109–131, 2000.
23. Radha Gupta, Manasa Chandan, Use of "Controlled Random Search Technique for Global Optimization" in Animal Diet Problem, International Journal of Emerging Technology and Advanced Engineering **3**, pp. 284-287, 2013.
24. R.C. Mehta, S.B. Tiwari, Controlled random search technique for estimation of convective heat transfer coefficient, Heat Mass Transfer **43**, pp.1171–1177, 2007.
25. M.M. Ali, C. Storey, Modified Controlled Random Search Algorithms, International Journal of Computer Mathematics **53**, pp. 229-235, 1994.
26. Di Pillo G., Lucidi S., Palagi L., Roma M. (1998) A Controlled Random Search Algorithm with Local Newton-type Search for Global Optimization. In: De Leone R., Murli A., Pardalos P.M., Toraldo G. (eds) High Performance Algorithms and Software in Nonlinear Optimization. Applied Optimization, vol 24. Springer, Boston, MA. https://doi.org/10.1007/978-1-4613-3279-4_10
27. S. Lucidi, F. Rochetich, and M. Roma, Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization, SIAM Journal on Optimization **8**, pp. 916–939, 1998.
28. P. Kaelo, M.M. Ali, Some Variants of the Controlled Random Search Algorithm for Global Optimization, Journal of Optimization Theory and Applications **130**, pp. 253–264, 2006.
29. Nelson Manzanares-filho , Rodrigo B. F. Albuquerque, Accelerating Controlled Random Search Algorithms Using a Distribution Strategy, In: EngOpt 2008 - International Conference on Engineering Optimization, Rio de Janeiro, Brazil, 01 - 05 June 2008.
30. Ioannis G. Tsoulos,I.E. Lagaris, Genetically controlled random search: a global optimization method for continuous multidimensional functions, Computer Physics Communications **174**, pp. 152-159, 2006.
31. Ioannis G. Tsoulos, Modifications of real code genetic algorithm for global optimization, Applied Mathematics and Computation **203**, pp. 598-607, 2008.
32. M. Montaz Ali, Charoenchai Khompatraporn, Zelda B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization **31**, pp 635-672, 2005.
33. C.A. Floudas, P.M. Pardalos, C. Adjiman, W. Esposoto, Z. Gümüs, S. Harding, J. Klepeis, C. Meyer, C. Schweiger, Handbook of Test Problems in Local and Global Optimization, Kluwer Academic Publishers, Dordrecht, 1999.
34. M. Gaviano, D.E. Ksasov, D. Lera, Y.D. Sergeyev, Software for generation of classes of test functions with known local and global minima for global optimization, ACM Trans. Math. Softw. **29**, pp. 469-480, 2003.
35. M.J.D Powell, A Tolerant Algorithm for Linearly Constrained Optimization Calculations, Mathematical Programming **45**, pp. 547-566, 1989.