

https://dspace.org/

Some Population Set-Based Methods for Unconstrained Global Optimization

Kaelo, Professor

2006-11-16T08:40:11Z
http://hdl.handle.net/10539/1771

Downloaded from a DSpace repository. https://dspace.org/

Some variants of the controlled random search algorithm for global optimization

P. Kaelo† and M. M. Ali‡

Abstract

Some modifications are suggested to the controlled random search (crs) algorithm for global optimization. We introduce new trial point generation schemes in crs, in particular, point generation schemes using linear interpolation and mutation. However, central to our modifications is the probabilistic adaptation of point generation schemes within the crs algorithm. A numerical study is carried out using a set of 50 test problems many of which are inspired by practical applications. Numerical experiments indicate that the resulting algorithms are considerably better than the previous versions. The new crs algorithms are also compared with the DIRECT algorithm developed by Jones et. al. [4]. The comparison shows that the crs algorithms are better than the DIRECT algorithm in high dimensional problems. Thus, they offer a reasonable alternative to many currently available stochastic algorithms, especially for problems requiring 'direct search type' methods.

Keywords: Global optimization, direct search methods, linear interpolation, probabilistic adaptation.

---

† Postgraduate Student, School of Computational and Applied Mathematics, Witwatersrand University, 1 Jan Smuts Avenue, Johannesburg, South Africa.

‡ Associate Professor, School of Computational and Applied Mathematics, Witwatersrand University, 1 Jan Smuts Avenue, Johannesburg, South Africa.

1 Introduction

We consider the problem of finding the global minimum of the optimization problem

$$\text{minimize } f(x) \text{ subject to } \ x \in \Omega, \tag{1}$$

where $f(x) : \Omega \subset R^n \rightarrow R$ is a continuous real-valued function and x is a n-dimensional continuous variable vector. The search region Ω is assumed to be either a box or some other region easy to sample. A point x_{opt} is said to be a global minimizer of f if $f_{opt} = f(x_{opt}) \leq f(x), \forall x \in \Omega$. In many applications, for example, in applied sciences and engineering, the function of interest may be non-linear, non-smooth or simulation based. It is with this view in mind that some search methods that do not require much information about the function were developed. They include simulated annealing [1, 2, 3], DIRECT algorithm [4], differential evolution [5, 6], genetic algorithms [7, 8] and controlled random search [9, 10]. Unlike gradient based methods, these search methods use no properties of the function being optimized. The only requirement on the problem is that $f(x)$ can be computed for any $x \in \Omega$. They are also easy to implement. Among other recent global optimization methods, the interval methods [11] are well-known. However, these methods require explicit expression of the function being optimized. This paper is concerned with the crs algorithm of Price [10].

The main drawbacks of crs are that it is not very robust in locating the global minimum and is less efficient with respect to convergence, especially after reaching the region of global minimum [12]. We have suggested some modifications to make crs more robust in obtaining the global minimum and efficient in terms of the number of function evaluations. We propose modifications to some previous versions of crs namely crs2 [10] and crsi [13]. In particular, we suggest modifications to the trial point generation schemes of these versions. We then suggest a scheme which probabilistically adapts point generation schemes for a given problem. While the probabilistic adaptation guides the algorithms to be robust in locating the global minimum, a local technique is introduced for faster convergence.

The organisation of the paper is as follows. In section 2 we briefly describe crs. Section 3 contains the full description of our proposed modifications. In section 4 numerical results and comparisons are made and section 5 contains the concluding remarks based on the results obtained.

2 Brief introduction to the crs algorithms

The controlled random search algorithm is a direct search technique and is purely heuristic. It starts by initially filling a set S with a sample of N ($N \gg n$) points uniformly distributed over the search space Ω. The sample S is then gradually contracted by replacing the current worst point in it with a better point, called a trial point. In the original crs1 [9], a trial point is obtained by forming a simplex, using $n+1$ distinct points chosen at random, with replacement, from the sample S, and reflecting one of the points in the centroid of the remaining n points of the simplex, as in the Nelder and Mead algorithm [14]. This process of finding a trial point and replacing the current worst point in S, if the trial point is better than the current worst point in S, is repeated until a certain stopping condition is met. Below we present the crs1 algorithm.

Algorithm 1 : the crs1 algorithm

Step 1 Initialize. Generate N ($N \gg n$) uniformly distributed random points from the search region Ω and store the points and their corresponding function values in an array S. Set iteration counter $k = 0$.

Step 2 Stopping rule based on best and worst points. Find the best and worst points in S, x_l and x_h, where the best point x_l has the lowest function value f_l and the worst point x_h has the highest function value f_h. If the stopping condition (e.g. $f_h - f_l \leq \epsilon$) is achieved, then stop.

Step 3 Generate a trial point $\tilde{x}$. Choose randomly $n+1$ points $x_1, x_2, \cdots, x_{n+1}$, with replacement, from S. Compute

$$\tilde{x} = 2G - x_{n+1}, \tag{2}$$

where the centroid G is given by

$$G = \frac{1}{n}\sum_{j=1}^{n} x_j. \tag{3}$$

If $\tilde{x} \notin \Omega$ go to Step 3; otherwise compute $f(\tilde{x})$. If $f(\tilde{x}) \geq f_h$ then go to Step 3.

Step 4 Update S. Replace, in S, the point x_h and function value f_h by those of $\tilde{x}$ and $f(\tilde{x})$ respectively. Set $k = k + 1$ and go to Step 2.

There are various modifications that have been suggested to the original crs1. The first two, crs2 and crs3, were suggested by Price [10, 15]. In crs2, the point with the least function value in S is always used in forming the simplex while in crs3, a Nelder-Mead-type [14] local search from the best $n+1$ points in S is incorporated. Ali and Storey [16] modified crs2 by exploring the region around the best point using a β-distribution. This version is known as crs4. In crs4, every time a new best point x_l is found, $f(x)$ is evaluated for M points from a β-distribution using the current best point x_l as the mean and the distance between x_l and x_h as the standard deviation. Ali and Storey [16] also proposed crs5 which uses a few steps of a gradient based local search from the best point x_l instead of a β-distribution. Another modification, crsi,

was introduced by Mohan and Shanker [13], who used coordinate-wise quadratic interpolation to find trial points instead of making use of simplexes. Ali et. al. [17] then proposed crs6 which uses the coordinate-wise quadratic interpolation to find trial points and a β-distribution for local exploration around the best point x_l just as in crs4. The coordinate-wise quadratic interpolation scheme uses $x_1 = x_l$ and two other randomly selected points $\{x_2, x_3\}$ with replacement from S to determine the coordinates of the trial point $\tilde{x} = (\tilde{x}^1, ..., \tilde{x}^n)$, where

$$\tilde{x}^i = \frac{1}{2}\left[\frac{(x_2^{i^2} - x_3^{i^2})f(x_1) + (x_3^{i^2} - x_1^{i^2})f(x_2) + (x_1^{i^2} - x_2^{i^2})f(x_3)}{(x_2^i - x_3^i)f(x_1) + (x_3^i - x_1^i)f(x_2) + (x_1^i - x_2^i)f(x_3)}\right], \tag{4}$$

for $i = 1, 2, \cdots, n$. If $\tilde{x} \notin \Omega$, the denominator is zero or $f(\tilde{x}) \geq f_h$, then another two random points $\{x_2, x_3\}$ are chosen from S and a new trial point is found by the above quadratic interpolation using $\{x_1, x_2, x_3\}$. Brachetti et. al. [18] also presented a modification to crs1. The main features of their version, ncrs, are the use of weighted centroid and weighted reflection in generating trial points. ncrs is similar to crs1 except for these features and the use of a quadratic model for local exploration using $2n + 1$ best points in S.

It is clear from the crs1 algorithm that the core step, that is, the operation through which the trial points are obtained in the crs algorithms, is Step 3. Therefore, most crs algorithms were derived by modifying this step. In particular, modifications were done by either replacing the point generation scheme in Step 3 and/or introducing a local search technique to obtain trial points locally every time a new best point x_l is obtained by a point generation scheme.

We denote a global point generation scheme by $P_{g\bullet}$ where the g represents global and the dot ($\bullet$) represents a scheme, e.g. s represents simplex in P_{gs} and q represents coordinate-wise quadratic interpolation in P_{gq}. We also denote a local point generation scheme by $P_{l\bullet}$ where l represents local and the dot ($\bullet$) represents the scheme. We can therefore write a local search using a β-distribution as $P_{l\beta}$, a gradient based local search as P_{lg}, and a Nelder-Mead type local search as P_{ln}. In order to facilitate the understanding and to make the difference between the methods more explicit we append to each individual algorithm name the appropriate parameter containing '(global scheme, local scheme)'. Using these notation we can write crs2, crs3, crs4, crs5, crsi and crs6 as crs(P_{gs},·), crs(P_{gs},P_{ln}), crs(P_{gs},$P_{l\beta}$), crs(P_{gs},P_{lg}), crs(P_{gq},·) and crs(P_{gq},$P_{l\beta}$) respectively.

All previous modified versions were suggested to make the convergence faster and to improve the robustness in locating the global minimizer. However, the modifications were justified by numerical experiments using a small set consisting of a maximum of 15 low dimen-

sional test problems. We carried out an extensive numerical study using some crs algorithms that proved superior to other versions in previous studies [19, 20], e.g. crs2, crsi and crs4. We used a set of 50 test problems from the literature of dimensions ranging from 2 to 20 for testing these algorithms. We compare first crs(P_{gs},$P_{l\beta}$) and crs(P_{gs},$\cdot$) to see the effect of $P_{l\beta}$. We observed that although crs(P_{gs}, $P_{l\beta}$) greatly improves the efficiency of crs(P_{gs}, $\cdot$), it failed to solve the same problems that crs(P_{gs}, $\cdot$) failed to solve. Thus crs(P_{gs}, $P_{l\beta}$) mostly improves the efficiency of crs(P_{gs}, $\cdot$) in terms of the number of function evaluations. A comparison of crs(P_{gs}, $\cdot$) and crs(P_{gq}, $\cdot$) showed that crs(P_{gq}, $\cdot$) was very good in terms of the number of function evaluations but it was inferior to crs(P_{gs}, $\cdot$) in finding the global minima. This shows that the global point generation scheme P_{gs} is superior to P_{gq} in terms of global exploration. Intuitively speaking, P_{gq} will be more robust in replacing the worst point in S if the points used in the quadratic interpolation are in a convex area around a minimizer, resulting in faster convergence. It is therefore desirable that a crs algorithm uses a global point generation scheme that is more exploratory at the earlier stages and then switches to a different scheme that is robust in the latter stages of the algorithm.

During our numerical study with crs, we observed that the robustness and efficiency of the crs algorithms are largely problem dependent. Some versions are superior to others on some problems while on other problems the converse statement is true. This means some particular point generation schemes favour some problems more than others. We also observed that a crs algorithm failing consistently on a problem can be made to succeed considerably by simply generating, say 10% trial points using a different scheme. We also observed that for some problems, a crs algorithm could spend a significant amount of time and function evaluations without finding a trial point that could replace the worst point in S. This led us to believe that a crs algorithm that gets trapped in a local minimizer could be re-directed to a global minimum point by generating some trial points using a different scheme. It is with this view in mind that we propose a scheme that combines two or more point generation schemes probabilistically. We felt that a probabilistic combination of point generation schemes would improve the performance of the crs algorithm. Motivated by the efficiency of coordinate-wise quadratic interpolation, we propose a new point generation scheme for crs, namely a global point generation scheme using linear interpolation (P_{gl}). We also introduce a local scheme using mutation (P_{lm}). These modifications, as will be shown later, indeed improve the efficiency and robustness of the crs algorithm considerably.

3 Proposed modifications to crs

In this section, we propose four new versions of crs. These are based on either introducing a global point generation scheme and/or adding a local generation scheme.

The first version modifies the point generation scheme P_{gs} in Step 3 of the crs($P_{gs}, \cdot$) algorithm by introducing a local mutation technique P_{lm}. Whenever a trial point $\tilde{x}$ generated by P_{gs} in crs($P_{gs}, \cdot$) fails to replace the current worst point x_h in S, local mutation P_{lm} generates a second trial point $\tilde{y}$ exploring the region around the current best point x_l in S by reflecting $\tilde{x}$ through x_l. In particular, this is done by coordinate-wise reflection of $\tilde{x}$ through x_l as follows:

$$\tilde{y}^i = (1 + \omega_i){x_l}^i - \omega_i \tilde{x}^i, \tag{5}$$

where $\tilde{y}^i$, $\tilde{x}^i$ and x_l^i are the i-th coordinate of $\tilde{y}$, $\tilde{x}$ and x_l respectively and ω_i is a random number in $[0, 1]$ for each i. This version of crs is referred to as controlled random search with local mutation. Using the notation introduced earlier we denote this version by crs(P_{gs},P_{lm}).

The second modified version generates global trial points using some probability distribution over the global generation scheme set $\{P_{gs}, P_{gl}\}$. That is, a point is generated using P_{gs} with some probability and using P_{gl} with the remaining probability. Therefore, a trial point is either generated using a simplex scheme P_{gs} or by using a linear interpolation scheme P_{gl}. Initially equal probabilities (0.5) are assigned to both generation schemes and these probabilities are updated according to some rule based on reward (for being successful) and penalty (for being unsuccessful). A probabilistic adaptation in the algorithm guides the algorithm in deciding on which scheme to use most in generating trial points for any given problem. This adaptive procedure allows the algorithm to bias the trial points generation to the scheme that solves a given problem most efficiently. We refer to this version as probabilistic crs using simplex and linear interpolation and denote it by crs(P_{gs+gl},$\cdot$) where P_{gs+gl} represents the combined scheme that uses the global schemes P_{gs} and P_{gl}.

The third modification introduces a local technique, in particular, the local mutation P_{lm}, in the second version crs(P_{gs+gl},$\cdot$) to give a more robust and efficient algorithm. We refer to this algorithm as probabilistic crs using simplex and linear interpolation with local mutation or crs(P_{gs+gl},P_{lm}).

The fourth version does not have a local technique and it generates global trial points with some probability distribution over the global scheme set $\{P_{gl}, P_{gq}\}$. This version, therefore, generates trial points using either linear interpolation or quadratic interpolation. We refer to

this algorithm as probabilistic crs using quadratic and linear interpolation or crs(P_{gq+gl},$\cdot$).

3.1 Controlled random search with local mutation : crs(P_{gs},P_{lm})

In the original crs($P_{gs}, \cdot$) algorithm, if a trial point $\tilde{x}$ obtained in Step 3 of the crs($P_{gs}, \cdot$) algorithm gives a function value $f(\tilde{x})$ that is not better than the current worst point in the sample S then it is discarded and a new simplex is formed using a new set of $n+1$ points from S. In crs(P_{gs},P_{lm}), we do not discard the unsuccessful trial point. The unsuccessful trial point $\tilde{x}$ is used to obtain a second trial point $\tilde{y}$ as defined by (5). This has an effect of improving the robustness and efficiency of the crs algorithm. This kind of exploration around x_l was not looked into before. Therefore, this motivated us to introduce this modification whenever a trial point fails to give a function value that can replace the current worst point in S. Below, we present the crs(P_{gs},P_{lm}) algorithm.

Algorithm 2 : the crs(P_{gs},P_{lm}) algorithm

Step 1 Initialize. Same as in Algorithm 1.

Step 2 Stopping rule based on best and worst point. Same as in Algorithm 1.

Step 3 Generate a trial point $\tilde{x}$ using P_{gs}. Choose n random points $x_2, x_3, \cdots, x_{n+1}$ from S and let $x_1 = x_l$. Generate a trial point $\tilde{x}$ as in Algorithm 1. If $\tilde{x} \notin \Omega$ repeat Step 3; otherwise compute $f(\tilde{x})$. If $f(\tilde{x}) \geq f_h$ then go to Step 4; otherwise go to Step 5.

Step 4 Mutate $\tilde{x}$ using P_{lm}. Generate another trial point $\tilde{y}$ using the trial point $\tilde{x}$ and x_l using (5). If $f(\tilde{y}) \geq f_h$ then go to Step 3.

Step 5 Update S. If this step is reached from Step 3 then replace x_h and f_h by $\tilde{x}$ and $f(\tilde{x})$ respectively. Else replace x_h and f_h by $\tilde{y}$ and $f(\tilde{y})$ respectively. Set $k = k+1$ and go to Step 2.

Remark

1. The local mutation scheme P_{lm} has a global effect (more exploration) at the earlier stages of the algorithm when points in S are scattered and a local effect at the later stages.

3.2 Probabilistic controlled random search using simplex and linear interpolation : crs(P_{gs+gl},$\cdot$)

In this version we introduce a linear interpolation concept, as used in model-based trust region methods [21], to generate global trial points. In other words we replace the trial point generation scheme P_{gs} in crs($P_{gs}, \cdot$) with P_{gs+gl}. To generate a trial point using linear interpolation,

$n+1$ points, say $Z = \{y_1, y_2, \cdots, y_{n+1}\}$, are drawn at random, with replacement, from S. We then let y_1 be the best point in Z and let y_{n+1} be the furthest point from y_1. A model $s \rightarrow m_c(y_1 + s)$ is then created to approximate $f(x)$ around y_1. We require that this model interpolate f at the points in Z, i.e., $m_c(y_i) = f(y_i)$ for all $y_i \in Z$. We then define this model at y_1 as

$$m_c(y_1 + s) = f(y_1) + g_c^T s, \tag{6}$$

where the vector $g_c \in R^n$ must be determined. Since we impose the interpolation conditions $m_c(y_i) = f(y_i)$, $i = 2, \cdots, n+1$, we have that

$$g_c^T s^i = f(y_i) - f(y_1) \text{ , } i = 2, \cdots, n+1,$$

where s^i is the displacement from y_i to y_1, i.e.,

$$s^i = y_i - y_1 \text{ , } i = 2, \cdots, n+1.$$

It then follows that the linear model (6) is uniquely determined if and only if the set $Z = \{y_1, \cdots, y_{n+1}\}$ is such that the set $\{s^i : i = 2, \cdots, n+1\}$ is linearly independent. This model (6) is then minimized with respect to s subject to $||s||_2 \leq \rho_c$, where the radius ρ_c is given, to generate a step $s_c \in R^n$ that gives a trial point $\tilde{x} = y_1 + s_c$. Thus, we have

$$\min_s \quad m_c(y_1 + s) = f(y_1) + g_c^T s, \tag{7}$$

$$\text{subject to} \qquad ||s||_2 \leq \rho_c. \tag{8}$$

For full explanations on trust region methods and linear models see [21, 22].

Generating trial points using linear interpolation only showed that for some problems it is very good in locating the global minimum where simplex alone failed while for some it proved inefficient in terms of the number of function evaluations and time taken to obtain a solution (see [12] for detailed results). This motivated us to introduce a scheme that combines two point generation schemes probabilistically. This probabilistic scheme penalizes (rewards) a point generation scheme for not making (making) good progress (see [23] for more on probabilistic adaptation in the context of discrete optimization using learning automata). Thus, we combined the linear interpolation scheme (P_{gl}) with the simplex scheme (P_{gs}), so that a trial point can either be generated with some probability α_k using simplex or with probability $\gamma_k = 1 - \alpha_k$ using linear interpolation. Initially equal probabilities (say α_0=γ_0=0.5) are assigned to both schemes P_{gs} and P_{gl}. If a trial point is generated say using P_{gs} and is found

to be successful in replacing the worst point in S then the probability for P_{gs} is increased (reward) using

$$\alpha_k = \alpha_{k-1} + \beta_0 \alpha_{k-1}(1 - \alpha_{k-1}), \tag{9}$$

and γ_k is obtained using $\gamma_k = 1 - \alpha_k$. If the trial point generated falls outside Ω or is not successful in replacing the worst point then the probability for P_{gs} is decreased (penalty) using

$$\alpha_k = \alpha_{k-1} - \beta_1 \alpha_{k-1}(1 - \alpha_{k-1}). \tag{10}$$

Similarly we can reward or penalize the probability γ_k. However, rewarding γ_k means penalizing α_k and penalizing γ_k means rewarding α_k. We can therefore work with only one probability, say α_k. This adaptive process tends to let the algorithm decides which scheme to use most in generating trial points for any given problem so that it solves the problem in a much more robust and efficient way. This is done by increasing the value of α_k whenever the simplex scheme gives more favourable points, thus reducing the probability of using linear interpolation. On the other hand, if linear interpolation gives better points than simplex, then α_k is reduced. In this way, the algorithm adapts itself to the combination of schemes that solves a given problem in a more robust and efficient way. The algorithm for crs(P_{gs+gl},$\cdot$) is described below.

Algorithm 3 : the crs(P_{gs+gl},$\cdot$) algorithm

Step 1. **Initialize**. Same as in Algorithm 1.

Step 2. **Stopping rule based on best and worst point**. Same as in Algorithm 1.

Step 3. **Select point generation procedure**. Go to Step 3a with probability α_k else go to Step 3b with probability γ_k.

Step 3a. **Generate a trial point $\tilde{x}$ using P_{gs}**. Same as in Algorithm 2.
If $\tilde{x} \notin \Omega$ then go to Step 5. Otherwise find $f(\tilde{x})$. If $f(\tilde{x}) \geq f_h$ then go to Step 5 else go to Step 4.

Step 3b. **Generate a trial point $\tilde{x}$ using P_{gl}**. Choose randomly $n+1$ distinct points $Z = \{y_1, y_2, \cdots, y_{n+1}\}$ from S. Let y_1 be the best point in Z and let

$$y_{n+1} = \arg\max_{2 \leq i \leq n+1} ||y_i - y_1||_2. \tag{11}$$

Compute s_c by solving problem (7-8). If $\tilde{x} \notin \Omega$ then go to Step 5. Otherwise compute $f(\tilde{x})$. If $f(\tilde{x}) \geq f_h$ then go to Step 5 else go to Step 4.

Step 4. **Update** S. If this step is reached from Step 3a then go to Step 4a. Else go to Step 4b.

Step 4a **Replace and reward**. Set $k = k+1$. Replace x_h and f_h by $\tilde{x}$ and $f(\tilde{x})$ respectively. $\alpha_k = \alpha_{k-1} + \beta_0 \alpha_{k-1}(1 - \alpha_{k-1})$. Go to Step 2.

Step 4b **Replace and penalize**. Set $k = k+1$. Replace x_h and f_h by $\tilde{x}$ and $f(\tilde{x})$ respectively. $\alpha_k = \alpha_{k-1} - \beta_1 \alpha_{k-1}(1 - \alpha_{k-1})$. Go to Step 2.

Step 5. **Update** α_k. Set $k = k+1$. If this step is reached from Step 3a then go to Step 5a else go to Step 5b.

Step 5a. **Penalize** $\alpha_k = \alpha_{k-1} - \beta_1 \alpha_{k-1}(1 - \alpha_{k-1})$. Go to Step 3.

Step 5b. **Reward** $\alpha_k = \alpha_{k-1} + \beta_0 \alpha_{k-1}(1 - \alpha_{k-1})$. Go to Step 3.

Remarks

2. All probabilistic adaptations use the same reward and penalty rules given by (9) and (10). The values β_0, β_1 and α_0 (or γ_0) given by (9) and (10) have to be provided by the user, where $0 < \beta_0, \beta_1 \leq 1$. The value β_0 controls the increment on α_k and β_1 controls the reduction in α_k.

3. It can be seen from Algorithm 3 above that when $\alpha_k = 0$ then we have an algorithm crs(P_{gl},$\cdot$) that uses only linear interpolation in generating trial points and when $\alpha_k = 1$ then we have an algorithm crs(P_{gs},$\cdot$) that uses only simplexes.

4. In (6)-(7) ρ_c is user provided.

3.3 Probabilistic crs using simplex and linear interpolation with local mutation : crs(P_{gs+gl},P_{lm})

Motivated by the results of crs(P_{gs},P_{lm}), i.e., the effect of local mutation in crs(P_{gs},$\cdot$), we introduce the local mutation technique in global probabilistic adaptation of the simplex and linear

interpolation. In other words, we incorporate a local mutation phase in crs($P_{gs+gl}, \cdot$). However, unlike in crs($P_{gs+gl}, \cdot$) where the probability distribution is defined over the set $\{P_{gs}, P_{gl}\}$, we define the probability over the combined simplex and mutation, and linear interpolation, i.e., over the set $\{P_{gs+lm}, P_{gl}\}$. This is because assigning a probability to any local technique would force the algorithm to go to a local minimum quickly. Therefore, if the trial point $\tilde{x}$ generated by simplex proves favourable we reward the combined simplex and mutation scheme by increasing α_k. However, if $\tilde{x}$ generated by the simplex scheme fails to give a favourable function value, we do not reduce α_k immediately, but we try mutation first. If mutation fails then we reduce α_k else we increase it. The mutation makes the algorithm even more exploratory. Next we present the algorithm for crs(P_{gs+gl},P_{lm}).

Algorithm 4 : the crs(P_{gs+gl}, P_{lm}) algorithm

Step 1. **Initialize**. Same as in Algorithm 1.

Step 2. **Stopping rule based on best and worst point**. Same as in Algorithm 1.

Step 3. **Select point generation procedure**. Go to Step 3a with probability α_k else go to Step 3c with probability γ_k.

Step 3a. **Generate a trial point $\tilde{x}$ using P_{gs}**. Same as in Algorithm 2.
If $\tilde{x} \notin \Omega$ then go to Step 5a. Otherwise find $f(\tilde{x})$. If $f(\tilde{x}) \geq f_h$ then go to Step 3b else go to Step 4.

Step 3b. **Mutate $\tilde{x}$ to find $\tilde{y}$ using P_{lm}**. Same as in Algorithm 2.
If $f(\tilde{y}) \geq f_h$ then go to Step 5a else go to Step 4.

Step 3c. **Generate a trial point $\tilde{x}$ using linear interpolation P_{gl}**. Same as in Algorithm 3. If $f(\tilde{x}) \geq f_h$ then go to Step 5b else go to 4.

Step 4. **Update** S. If this step is reached from Step 3a or 3b then go to Step 4a. Else go to Step 4b.

Step 4a **Replace and reward**. Set $k = k + 1$. Replace x_h and f_h by $\tilde{x}$ and $f(\tilde{x})$ respectively if reached from Step 3a. Else replace x_h and f_h by $\tilde{y}$ and $f(\tilde{y})$ respectively. $\alpha_k = \alpha_{k-1} + \beta_0\alpha_{k-1}(1 - \alpha_{k-1})$. Go to Step 2.

Step 4b **Replace and penalize**. Set $k = k + 1$. Replace x_h and f_h by $\tilde{x}$ and $f(\tilde{x})$ respectively. $\alpha_k = \alpha_{k-1} - \beta_1\alpha_{k-1}(1 - \alpha_{k-1})$. Go to Step 2.

Step 5. **Update** α_k. Set $k = k + 1$. If this step is reached from Step 3a or Step 3b then go to Step 5a else go to Step 5b.

Step 5a. **Penalty**. Same as in Step 5 in Algorithm 3.

Step 5b. **Reward**. Same as in Step 5 in Algorithm 3.

3.4 Probabilistic crs using quadratic and linear interpolation: crs(P_{gq+gl},·)

In this version of the controlled random search algorithm we combine linear and quadratic interpolations probabilistically. We define the probability distribution over the set $\{P_{gq}, P_{gl}\}$. Updating of the probabilities α_k and γ_k are done as before, i.e. using reward and penalty. For example, α_k will be reduced if the point generated by P_{gq} is unsuccessful which in effect means γ_k being increased. Below we give the algorithm for crs(P_{gq+gl},·).

Algorithm 5 : the crs(P_{gq+gl}, ·) algorithm

Step 1. **Initialize**. Same as in Algorithm 1.

Step 2. **Stopping rule based on best and worst point**. Same as in Algorithm 1.

Step 3. **Select point generation procedure**. Go to Step 3a with probability α_k else go to Step 3b with probability γ_k.

Step 3a. **Generate a trial point $\tilde{x}$ using quadratic interpolation** P_{gq}. Choose 2 random points x_2, x_3 from S, different from x_l, and let $x_1 = x_l$. Find a trial point $\tilde{x}$ using (4). If $\tilde{x} \notin \Omega$ or the denominator is zero then go to Step 5. Otherwise find $f(\tilde{x})$. If $f(\tilde{x}) \geq f_h$ then go to Step 5 else go to Step 4.

Step 3b. **Generate a trial point $\tilde{x}$ using linear interpolation** P_{gl}. Same as Step 3b in Algorithm 3. If $\tilde{x} \notin \Omega$ then go to Step 5. Otherwise compute $f(\tilde{x})$. If $f(\tilde{x}) \geq f_h$ then go to Step 5 else go to Step 4.

Step 4. **Update** S. Same as Step 4 in Algorithm 3.

Step 5. **Update** α_k. Same as Step 5 in Algorithm 3.

4 Numerical results and discussion

In this section numerical results of crs(P_{gs}, ·), crs(P_{gs}, $P_{l\beta}$), crs(P_{gq}, ·) and the new algorithms using 50 test problems are presented. These problems have a variety of inherent difficulty [24]. All the problems have continuous variables and a detailed description of each problem can be found in [12, 20]. We compare the new algorithms with the crs(P_{gs}, ·), crs(P_{gs}, $P_{l\beta}$) and crs(P_{gq}, ·) algorithms to assess their robustness in terms of success in finding the global minima and their efficiency in terms of the number of function evaluations. We answer the following research questions:

- does local mutation improve the crs(P_{gs}, ·) algorithm and how does it compare with crs(P_{gs}, $P_{l\beta}$), i.e. how do crs(P_{gs}, ·), crs(P_{gs}, $P_{l\beta}$) and crs(P_{gs}, P_{lm}) compare?
- does linear interpolation improve the robustness of the crs(P_{gs}, ·) algorithm?

- does linear interpolation and local mutation together improve crs($P_{gs}, \cdot$)?
- does linear interpolation improve the robustness of the crs($P_{gq}, \cdot$)?

The first three research questions focus on the improvement of crs($P_{gs}, \cdot$) while the fourth question focuses on crs($P_{gq}, \cdot$).

The algorithms were run 100 times on each of the 50 test problems to determine the success rate (or percentage success), sr, of each algorithm. There were 5000 runs in total. In every case, a run was terminated when the function values of all points in S were identical to an accuracy of four decimal places, i.e,

$$f_h - f_l \leq \epsilon = 10^{-4} \tag{12}$$

or when the maximum number of iterations was reached, in this case, $T = 1000n^2$, where n is the dimension of the problem being solved. A success was counted when the value f_l of a run was such that $f_l - f_{opt} \leq 0.01$ where f_{opt} is the known global minimum of the problem being solved. We calculated the average number of function evaluations (fe) and cpu time (cpu) for those problems for which the global minima were found by at least one of the algorithms. We note that none of the algorithms succeeded in finding the global minimum for four 10 dimensional functions, namely Epistatic Michalewicz (EM), Odd Square (OSP), Schwefel (SWF) and Shekel's Foxholes (SF). These problems are highly multi-modal and their global minima lie at a location which is either close to the boundary of Ω or further away from the centre of Ω. Except for these four functions, all other functions were solved by at least one of the algorithms. Therefore, results for these four functions are not reflected in our presentation. We used the sr, fe and cpu as the criteria for comparison.

Each of the algorithms has some parameter values to be provided by the user that are common to all algorithms. These common parameters are the number of elements N of S and the parameter ϵ in the stopping rule. We took the value of N to be $10(n+1)$ where n is the dimension of the problem. It is a heuristic choice and can therefore always be increased for obtaining the global minimum with higher probability. The parameter ϵ was as given in (12).

A parameter associated with crs($P_{gs}, P_{l\beta}$) is M where M is the number of trial points to be generated from the β-distribution. We used $M = 2$ for problems with dimensions $n \leq 5$ and $M = 3$ for problems with $n > 5$, see [16] for the choice of M. Three parameters associated with crs($P_{gs+gl}, \cdot$), crs(P_{gs+gl}, P_{lm}) and crs($P_{gq+gl}, \cdot$) are the trust region radius ρ_c in (8), β_0 in (9) and β_1 in (10). In trust region methods, a specific value is given for ρ_c at the beginning

of the algorithm and this value is either increased or reduced as the iterations progress. For our implementations we calculated ρ_c every time the linear interpolation scheme was called. In particular, ρ_c was found by

$$\rho_c = \min_{1 \leq i \leq n} |y_{n+1}^i - y_1^i|, \tag{13}$$

where y_{n+1}^i and y_1^i are the i-th component of y_{n+1} and y_1 respectively. y_{n+1} and y_1 are the furthest and the best points respectively in the set Z as defined in section 3.2. We also restricted ρ_c to $\rho_c \geq 10^{-5}$. Different combinations of β_0 and β_1 values can be used. We carried out numerical experiments using various combinations of values for these parameters. Although the results for other combinations are also good, the results presented here are the best results obtained for $\beta_0 = 0.35$ and $\beta_1 = 0.65$. For results using other combinations of β_0 and β_1 see [12]. We forced $\alpha_k, k \geq 1$, to lie in $[0.050, 0.95]$. For example, if α_k goes below 0.05, we set $\alpha_k = 0.05$ by clipping. This was done in order to avoid the algorithm switching entirely to one scheme. Thus α_k and γ_k lie in $(0, 1)$ to allow the algorithm to be able to switch from one scheme to the other.

To address our first research question, we compare the results of crs($P_{gs}, \cdot$) and crs($P_{gs}, P_{l\beta}$) with those of crs(P_{gs}, P_{lm}). These results of crs($P_{gs}, \cdot$) and crs($P_{gs}, P_{l\beta}$) along with the results of the new algorithms are presented in Table 1, where tr represents the total result. To see the effect of introducing local mutation in crs($P_{gs}, \cdot$), we begin by looking at the robustness and efficiency of crs(P_{gs}, P_{lm}) as compared to the crs($P_{gs}, \cdot$) algorithm. We then compare the results of crs($P_{gs}, \cdot$) and crs(P_{gs}, P_{lm}) with those of crs($P_{gs}, P_{l\beta}$). The total results show that crs(P_{gs}, P_{lm}) is much more superior to both crs($P_{gs}, \cdot$) and crs($P_{gs}, P_{l\beta}$) in terms of fe and sr. For instance, in terms of sr crs(P_{gs}, P_{lm}) is superior to crs($P_{gs}, \cdot$) and crs($P_{gs}, P_{l\beta}$) by 479 and 200 successes respectively. crs(P_{gs}, P_{lm}) was able to secure this higher number of total sr for a total fe considerably less than those of the other two. Also, crs(P_{gs}, P_{lm}) was able to locate the global minima for the Extended Rosenbrock (RB) and Price's Transistor Modelling (PTM) functions. Though the RB function is unimodal and the global minimum is at the centre of Ω, it has a saddle point. The PTM function is highly non-linear with the global minimum situated near the boundary of Ω. For a fair comparison, we exclude the results of these functions from the total result of crs(P_{gs}, P_{lm}). If we now compare all three algorithms based on functions for which they all succeeded, we see that crs(P_{gs}, P_{lm}) makes about 63% improvement on total fe compared to crs($P_{gs}, \cdot$) and about 49% improvement compared to crs($P_{gs}, P_{l\beta}$). This shows the overall effectiveness of the introduction of local mutation in crs($P_{gs}, \cdot$).

Table 1: Results of crs($P_{gs}, \cdot$), crs(P_{gs}, P_{lm}), crs($P_{gs+gl}, \cdot$),crs(P_{gs+gl}, P_{lm}) and crs($P_{gs}, P_{l\beta}$)

		crs($P_{gs}, \cdot$)		crs(P_{gs}, P_{lm})		crs($P_{gs+gl}, \cdot$)		crs(P_{gs+gl}, P_{lm})		crs($P_{gs}, P_{l\beta}$)	
P	n	fe	sr	fe	sr	fe	sr	fe	sr	fe	sr
ACK	10	17586	100	8010	100	9059	100	7999	100	9389	100
AP	2	406	83	487	100	624	100	499	100	458	93
BL	2	578	98	477	100	753	100	563	100	427	100
B1	2	819	74	726	100	437	100	452	100	664	92
B2	2	825	76	730	100	437	100	450	100	670	98
BR	2	465	85	489	100	608	97	512	100	438	100
CB3	2	554	97	474	100	299	100	299	100	435	96
CB6	2	615	97	522	100	806	100	584	100	486	100
CM	4	2266	100	1185	100	1364	100	1315	100	1701	100
DA	2	789	63	836	100	988	100	901	100	734	100
EP	2	487	98	464	100	562	100	438	100	385	100
EXP	10	6425	100	3118	100	3695	100	3133	100	3526	100
GP	2	671	88	587	99	860	100	634	100	546	99
GW	10	10037	100	4768	100	5438	100	4755	100	5375	100
GRP	3	977	100	951	100	1049	100	977	100	872	100
H3	3	908	100	671	100	942	100	685	100	720	100
H6	6	3993	90	1980	83	3908	97	1932	77	2580	87
HV	3	1876	100	1551	100	2495	100	1584	100	1544	100
HSK	2	467	100	403	100	500	100	413	100	372	100
KL	4	519	100	480	100	570	100	475	100	461	100
LM1	3	1377	100	888	100	1569	100	927	100	1073	100
LM2	10	8945	100	3900	100	14144	100	4033	100	5018	100
MC	2	462	100	415	100	567	100	454	100	379	100
MR	3	1266	41	879	35	2932	43	1179	39	1151	37
MCP	4	577	100	440	100	934	100	662	100	501	100
ML	10	75780	17	13049	36	86866	25	8979	37	44737	29
MRP	2	694	39	640	46	787	67	641	62	569	45
MGP	2	623	25	571	36	707	100	571	88	522	26
NF2	4	75214	93	5753	100	73201	92	5640	100	71717	92
NF3	10	12686	100	7215	100	14581	85	7280	100	7524	100
PP	10	10407	69	4791	100	10410	69	4799	100	5960	98
PRD	2	497	100	424	100	268	100	263	100	388	100
PQ	4	2189	100	1641	100	1537	100	1666	100	1747	100
PTM	9	22734	0	53360	4	26453	0	43343	5	17626	0
RG	10	100000	0	12891	0	7002	100	6974	100	100000	0
RB	10	24067	0	14744	97	16031	0	14661	100	69018	0
SAL	10	45060	0	29624	0	18894	100	20888	99	34231	0
SF1	2	474	100	424	100	274	100	290	100	393	100
SF2	2	1503	41	1514	100	1129	100	1095	100	1347	100
SBT	2	418	7	842	100	1183	83	1300	100	678	90
S5	4	3239	95	1721	67	3669	98	1871	74	2429	82
S7	4	2973	98	1708	82	3689	98	1771	82	2261	97
S10	4	3074	100	1707	71	3773	96	1754	87	2322	94
SIN	20	20788	3	11580	100	20102	3	11632	100	13388	1
ST	9	10128	100	13870	100	15036	100	18929	100	9744	100
WP	4	3919	100	3021	100	5608	100	3225	100	3323	100
tr		480357	3477	216521	3956	366740	4048	193427	4250	429829	3756

To address our second research question, we compare crs($P_{gs}, \cdot$) with crs($P_{gs+gl}, \cdot$). We note that crs($P_{gs}, \cdot$) is a special case of crs($P_{gs+gl}, \cdot$) where the probability of generating trial points using P_{gl} is always zero. Table 1 shows that in terms of total sr crs($P_{gs+gl}, \cdot$) is much more superior to crs($P_{gs}, \cdot$), with 571 more successes than crs($P_{gs}, \cdot$). We also see that in terms of total fe, crs($P_{gs+gl}, \cdot$) is again more efficient than crs($P_{gs}, \cdot$). crs($P_{gs+gl}, \cdot$), like crs($P_{gs}, \cdot$), also failed to find the global minima of the PTM and RB functions. However, crs($P_{gs+gl}, \cdot$) was able to locate the global minimum value for two difficult functions, Rastrigin (RG) and Salomon (SAL) where crs($P_{gs}, \cdot$) failed. These two problems are highly multi-modal and their global minima lie at the centre of Ω. crs($P_{gs+gl}, \cdot$), therefore, not only increased sr, it also solved two difficult problems where crs($P_{gs}, \cdot$) failed.

We now address our third research question. We have shown the positive effect of P_{gl} in crs($P_{gs+gl}, \cdot$). We look at the effect of introducing local mutation in crs($P_{gs+gl}, \cdot$). The total results in Table 1 show that crs(P_{gs+gl}, P_{lm}) is much more superior to crs($P_{gs+gl}, \cdot$) both in terms of sr and fe. crs(P_{gs+gl}, P_{lm}) secured 202 more sr than crs($P_{gs+gl}, \cdot$). In addition crs(P_{gs+gl}, P_{lm}) reduced fe of crs($P_{gs+gl}, \cdot$) by 47%. We now compare these two algorithms by excluding the results of two problems (PTM and RB) that crs(P_{gs+gl}, P_{lm}) solved but crs($P_{gs+gl}, \cdot$) failed. We see that crs(P_{gs+gl}, P_{lm}) reduces fe of crs($P_{gs+gl}, \cdot$) by 58%. It is therefore quite clear that incorporation of local mutation in crs($P_{gs+gl}, \cdot$) has a significant effect in reducing fe and increasing sr. Introduction of local mutation in crs($P_{gs+gl}, \cdot$) means introducing linear interpolation and local mutation together in crs($P_{gs}, \cdot$). Results from Table 1 show that incorporation of both linear interpolation and local mutation in the crs($P_{gs}, \cdot$) algorithm reduces its fe by 60% and increases sr by 22%. Thus, from the above analysis we see that both linear interpolation and local mutation have a great effect in improving both the robustness and efficiency of the crs($P_{gs}, \cdot$) algorithm.

Table 2: Results of problems solved by all algorithms in Table 1

	crs($P_{gs}, \cdot$)	crs(P_{gs}, P_{lm})	crs($P_{gs+gl}, \cdot$)	crs(P_{gs+gl}, P_{lm})	crs($P_{gs}, P_{l\beta}$)
fe	288496	105902	298360	107561	208954
sr	3477	3855	3848	3946	3756
cpu	6.45	3.37	38.23	22.47	4.86

Next we compare crs($P_{gs}, \cdot$), crs(P_{gs}, P_{lm}), crs($P_{gs+gl}, \cdot$) and crs(P_{gs+gl}, P_{lm}) using the results of those functions for which all of these algorithms succeeded in finding their global

minimum. These results are summarised in Table 2 where we have also presented the total cpu (total averages). These results are extracted from Table 1, where we excluded the results of problems that were not solved by at least one algorithm. We see that although a probabilistic combination of linear interpolation and simplex scheme improves fe and sr, it is at a cost of high cpu. On the other hand, introduction of local mutation technique in both crs($P_{gs}, \cdot$) and crs($P_{gs+gl}, \cdot$) shows significant improvements on fe, sr and cpu. It follows also from Table 2 that crs(P_{gs+gl}, P_{lm}) is the best in terms of sr while crs(P_{gs}, P_{lm}) is the best in terms of fe and cpu.

Table 3: Rank order of algorithms

Rank	1	2	3	4	5
fe	crs(P_{gs}, P_{lm})	crs(P_{gs+gl}, P_{lm})	crs($P_{gs}, P_{l\beta}$)	crs($P_{gs}, \cdot$)	crs($P_{gs+gl}, \cdot$)
sr	crs(P_{gs+gl}, P_{lm})	crs(P_{gs}, P_{lm})	crs($P_{gs+gl}, \cdot$)	crs($P_{gs}, P_{l\beta}$)	crs($P_{gs}, \cdot$)
cpu	crs(P_{gs}, P_{lm})	crs($P_{gs}, P_{l\beta}$)	crs($P_{gs}, \cdot$)	crs(P_{gs+gl}, P_{lm})	crs($P_{gs+gl}, \cdot$)

In Table 2 we presented the results of problems for which all algorithms succeeded at least once. It is therefore easy to rank order these algorithms by comparing their effort (fe and cpu) needed to find the global minimum as well their percentage of success, i.e. sr. Table 3 presents the rank ordering of the algorithms. It follows from Table 3 that the algorithms that use P_{gl} and P_{lm} are the best in all respect and are runners-up with respect to fe and sr. crs($P_{gs}, P_{l\beta}$) is the runner-up with respect to cpu. The rank ordering in Table 3 therefore justifies the introduction of P_{lm} and probabilistic adaptation of P_{gs} and P_{gl}.

In order to address the fourth research question, we look at the effect of combining, probabilistically, the quadratic P_{gq} and linear interpolation P_{gl}. The total results of the resulting algorithm crs($P_{gq+gl}, \cdot$) alongside those of crs($P_{gq}, \cdot$) are presented in Table 4. We note that in addition to the problems that crs($P_{gq}, \cdot$) solved, crs($P_{gq+gl}, \cdot$) also solved the Salomon (SAL) function. However, the results under crs($P_{gq+gl}, \cdot$) do not include the results (fe=11588, sr=100 and cpu = 1.53) for the Salomon function. Table 4 shows that crs($P_{gq+gl}, \cdot$) needed about the same fe as crs($P_{gq}, \cdot$) but it achieved 227 more successes than crs($P_{gq}, \cdot$). It is also clear from Table 4 that the incorporation of linear interpolation, like in the other algorithms, increases the cpu. Although crs($P_{gq+gl}, \cdot$) is superior to crs($P_{gq}, \cdot$) with respect to sr, it is inferior to other modified algorithms presented in Table 2.

Table 4: Results of crs($P_{gq}, \cdot$), crs($P_{gq+gl}, \cdot$) and crs($P_{gq+gs}, \cdot$)

	crs($P_{gq}, \cdot$)			crs($P_{gq+gl}, \cdot$)			crs($P_{gq+gs}, \cdot$)		
	fe	sr	cpu	fe	sr	cpu	fe	sr	cpu
tr	112138	3301	1.67	112065	3528	3.69	143512	3285	5.33

In Table 4, we also presented the results of crs($P_{gq+gs}, \cdot$) to see the combined effect of P_{gq} and P_{gs}. The results show that crs($P_{gq+gs}, \cdot$) is inferior to both crs($P_{gq}, \cdot$) and crs($P_{gq+gl}, \cdot$). Results of crs($P_{gl}, \cdot$), i.e. crs using linear interpolation scheme alone, are not very competitive, especially in terms of fe and cpu (see [12]). Therefore, these results are not presented here.

We now show how the new crs algorithms compare with the DIRECT algorithm. The DIRECT algorithm is a heuristic algorithm developed by Jones et. al. [4]. This algorithm was found to compete very well with some existing global optimization algorithms [4]. We compare the best performing crs algorithm, the crs(P_{gs+gl}, P_{lm}) algorithm, with the DIRECT algorithm. We used 30 problems from Table 1. These problems do not have their global minimum at the centre of the search space. We did not use the problems with their global minima at the centre since the DIRECT algorithm uses the centre point as the initial starting point. Hence, for these problems, the global minimizer would be used as the starting point. The DIRECT algorithm was terminated using percent error from the known globally optimal value f_{opt}. That is, if f_l is the best function value at some point in the search, then the percent error is given by

$$E = 100\frac{f_l - f_{opt}}{|f_{opt}|}. \tag{14}$$

For a fair comparison of the algorithms, we ran the algorithms using the same stopping condition (14) with $E = 0.01\%$. We present the number of function evaluations (fe) of the algorithms in Table 5.

A comparison of the results shows that in 17 out of 23 low dimensional problems ($n \leq 6$), the DIRECT algorithm is superior in terms of fe. In terms of success rate, sr, the DIRECT algorithm is superior to the crs algorithm in 6 out of 23 low dimensional problems, i.e. the Shekel family (S5, S7, S10), 6 dimensional Hartman (H6), Modified Rosenbrock (MRP) and Multi Gaussian (MGP). For the other low dimensional problems, both algorithms were very competitive except for the Neumaier 2 (NF2) problem where the DIRECT algorithm failed. For high dimensional problems ($n > 6$), the DIRECT algorithm is inferior to crs(P_{gs+gl}, P_{lm})

Table 5: Results of crs(P_{gs+gl}, P_{lm}) and DIRECT

		crs(P_{gs+gl}, P_{lm})	DIRECT
Problem	n	fe	fe
AP	2	491	157
BL	2	646	777
BR	2	510	195
CB6	2	637	285
DA	2	575	129
EP	2	421	497
GP	2	588	191
GRP	3	978	2849
H3	3	555	199
H6	6	1553	571
HV	3	1606	1481
HSK	2	370	109
LM1	3	921	159
LM2	10	4136	19895
MC	2	327	149
MCP	4	600	277
ML	10	7595	†
MRP	2	633	357
MGP	2	587	141
NF2	4	5681	†
NF3	10	5410	31693
PP	10	3593	33343
RB	10	15279	†
SBT	2	994	2967
S5	4	1487	155
S7	4	1514	145
S10	4	1500	145
SIN	20	11301	†
ST	9	18155	†
WP	4	2903	6579

† No convergence after 60 000 function evaluations.

with respect to both fe and sr. The DIRECT algorithm failed to converge after 60 000 function evaluations on the 9 dimensional Storn's Tchebychev (ST) problem, the 10 dimensional modified Langerman (ML) and Extended Rosenbrock (RB) problems, and the 20 dimensional Sinusoidal (SIN) problem. From Table 1, it is also clear that for high dimensional problems, the new crs algorithms are superior to the DIRECT algorithm in terms of success.

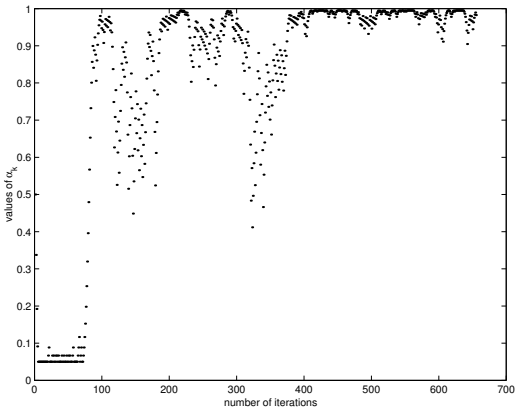

Figure 1: Probabilistic adaptation of crs($P_{gs+gl}, \cdot$) on GP

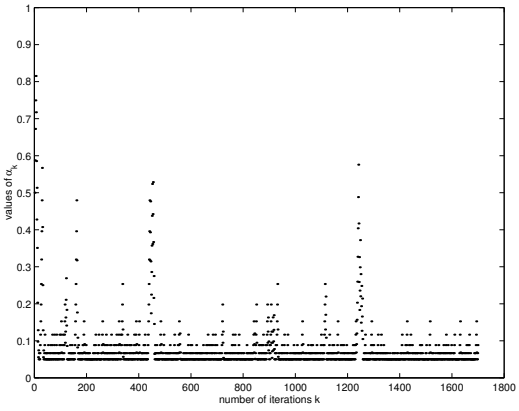

Figure 2: Probabilistic adaptation of crs($P_{gs+gl}, \cdot$) on PQ

Finally, we show how crs($P_{gs+gl}, \cdot$), crs($P_{gq+gl}, \cdot$) and crs(P_{gs+gl}, P_{lm}) probabilistically adapts to a point generation scheme or schemes. We present a number of figures to illustrate these adaptations. The figures have been plotted using the number of iterations k as the horizontal axis and α_k as the vertical axis. For a particular function, we observed slight variations in plots from run to run. However, the general trend is the same for all successful runs. Therefore, we presented each figure from a single run. We first use four figures, Figures 1-4, to

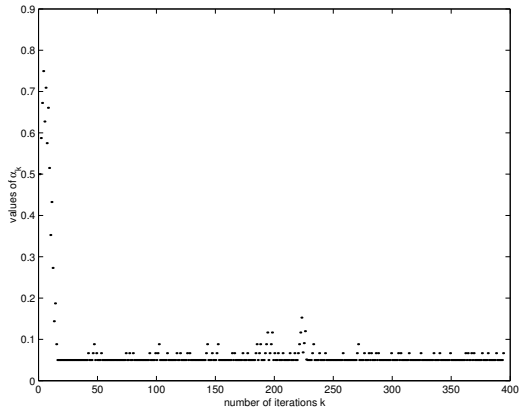

Figure 3: Probabilistic adaptation of crs($P_{gs+gl}, \cdot$) on B1

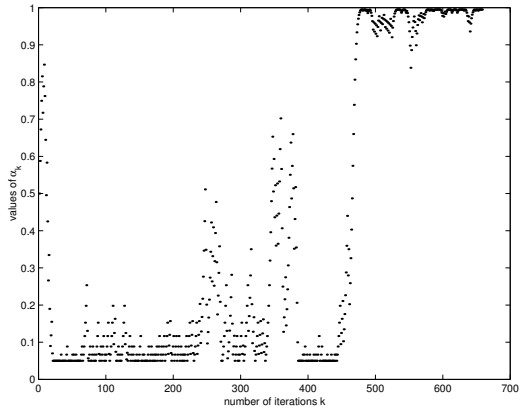

Figure 4: Probabilistic adaptation of crs($P_{gs+gl}, \cdot$) on MGP

illustrate the probabilistic adaptation of crs($P_{gs+gl}, \cdot$). Figures 1-4 are drawn respectively for Goldstein and Price (GP), Powell's Quadratic (PQ), Bohachevsky 1 (B1) and Multi-Gaussian (MGP) problems. Other problems can be used to illustrate the probabilistic adaptation in the algorithms. The problems used were taken as representatives. crs($P_{gs+gl}, \cdot$) uses the simplex scheme P_{gs} for $\alpha_k = 1$, linear interpolation scheme P_{gl} for $\alpha_k = 0$ and a mixture of the two for any $\alpha_k \in (0, 1)$. For Goldstein and Price function (Figure 1), this algorithm mostly uses linear interpolation up to about 100 iterations, then a mixture of (less) linear interpolation and (more) simplex up to 400 iterations, and finally switching almost entirely to simplex. For the Powell function (Figure 2) and Bohachevsky 1 function (Figure 3) it consistently uses the linear interpolation to solve these problems. For Multi-Gaussian function (Figure 4) the algorithm uses linear interpolation mostly for most of the times up to 450 iterations before

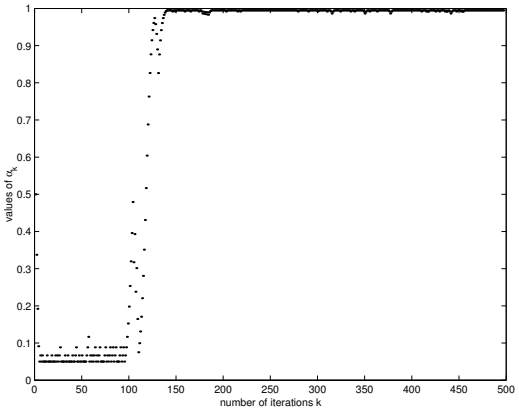

Figure 5: Probabilistic adaptation of crs(P_{gs+gl}, P_{lm}) on GP

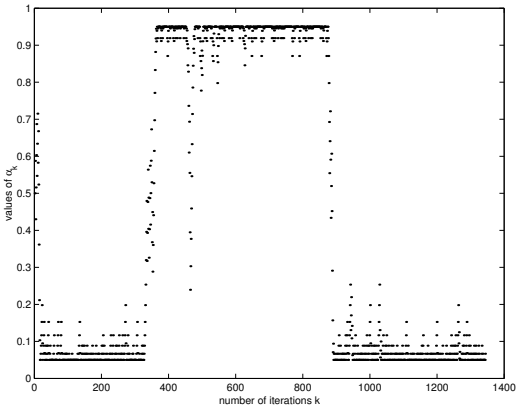

Figure 6: Probabilistic adaptation of crs(P_{gs+gl}, P_{lm}) on PQ

gradually switching to simplex.

Next we analyse the adaptation of crs(P_{gs+gl}, P_{lm}) on the same functions using Figures 5-8. One can see the effect of mutation by comparing Figure 1 and 5 for the Goldstein and Price function. In Figure 1 crs($P_{gs+gl}, \cdot$) starts to use a mixture of simplex and linear interpolation after about 100 iterations. For iterations greater than 100 the trend of using more simplexes and less linear interpolations continues until the algorithm stops. On the other hand, Figure 5 shows that crs(P_{gs+gl}, P_{lm}) uses linear interpolation at the beginning for a short while and just after 100 iterations it switches completely to combined simplex and mutation. This trend of quickly switching to the combined simplex and mutation scheme is also noticeable for the Multi-Gaussian function (see Figure 8). In figure 6, simplex and local mutation dominate between the 400th and 800th iterations and linear interpolation dominates in the earlier and final stages.

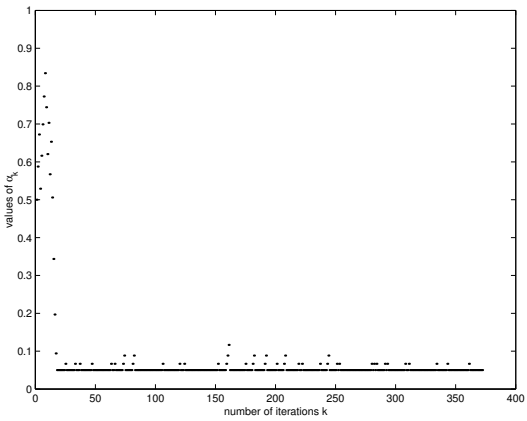

Figure 7: Probabilistic adaptation of crs(P_{gs+gl}, P_{lm}) on B1

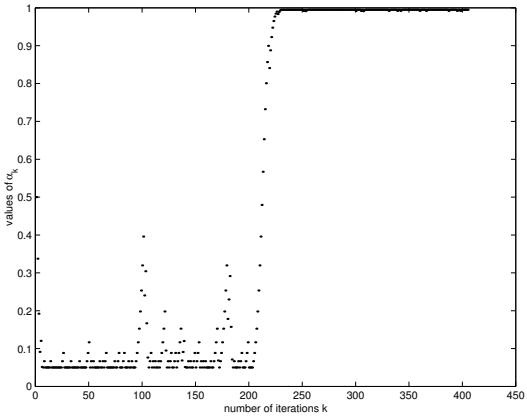

Figure 8: Probabilistic adaptation of crs(P_{gs+gl}, P_{lm}) on MGP

We now show figures of the same functions using crs($P_{gq+gl}, \cdot$). Figure 9 shows that quadratic interpolation is very efficient in solving the Goldstein and Price function. Therefore, the algorithm quickly adapts itself to using quadratic approximation. Figure 10 shows that quadratic interpolation dominates up to about 200 iterations before the algorithm switches completely to linear interpolation. Figures 10 - 12 show that linear interpolation is dominant compared to quadratic interpolation.

We also present Figure 13 for the Salomon function (SAL). We found that all algorithms that use a probabilistic combination of linear interpolation with another point generation scheme were able to solve this problem. Here, we only present the adaptation of the crs($P_{gs+gl}, \cdot$) algorithm since the other algorithms, crs(P_{gs+gl}, P_{lm}) and crs($P_{gq+gl}, \cdot$), showed the same trend. Figure 13 shows that trial point generation here is more dominated by the linear interpolation scheme. Since the figures obtained using crs(P_{gs+gl}, P_{lm}) and crs($P_{gq+gl}, \cdot$) showed similar

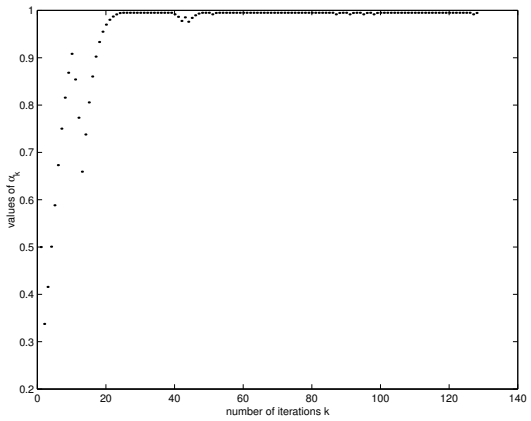

Figure 9: Probabilistic adaptation of crs(P_{gq+gl}, $\cdot$) on GP

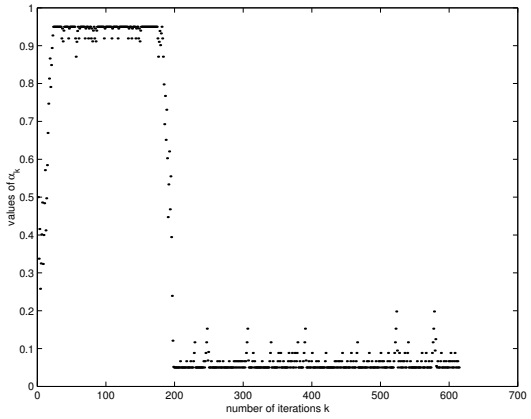

Figure 10: Probabilistic adaptation of crs(P_{gq+gl}, $\cdot$) on PQ

trends, it follows therefore that the linear interpolation scheme is best suited for this function.

From the above discussions using the results and figures it is clear that the probabilistic adaptation has a significant role to play in solving some difficult problems. Analysis of the results also shows that the low dimensional problems with a small number of local minima were frequently solved by all algorithms, except for some problems whose global minimizers lie close to the boundary of the search region Ω. The low dimensional problems with saddle points were also difficult to solve, e.g. the multi-Gaussian problem (MGP). The high dimensional problems whose global minimizers lie close to the boundary of Ω were particularly difficult to solve, e.g. modified Langerman (ML) and PTM. However, the probabilistic adaptation of the point generation schemes was able to overcome some of the above mentioned difficulties.

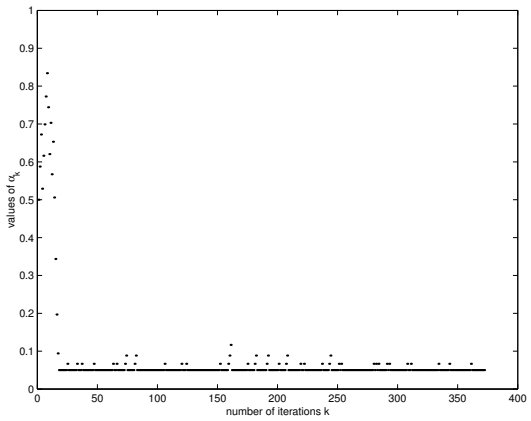

Figure 11: Probabilistic adaptation of crs($P_{gq+gl}, \cdot$) on B1

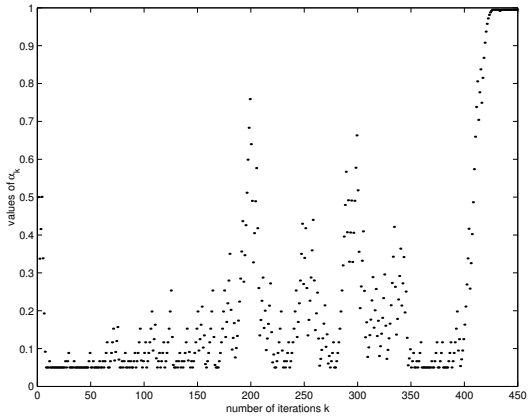

Figure 12: Probabilistic adaptation of crs($P_{gq+gl}, \cdot$) on MGP

5 Conclusion

We have developed and tested four new versions of the crs algorithms on a large set of problems. Numerical results have shown that the new versions are considerably better than the currently known crs algorithms.

We have shown the effect of local mutation on algorithms that use either simplex or a combination of simplex and linear interpolation. Introduction of the local mutation has made these algorithms more robust in finding the global minimum and efficient in reducing the number of function evaluations and cpu time. The local mutation technique also expedites the convergence as soon as the region of the global minimizer is reached.

Although the probabilistic use of linear interpolation has improved the crs algorithm in terms of success and the number of function evaluations, its use made the algorithm inferior

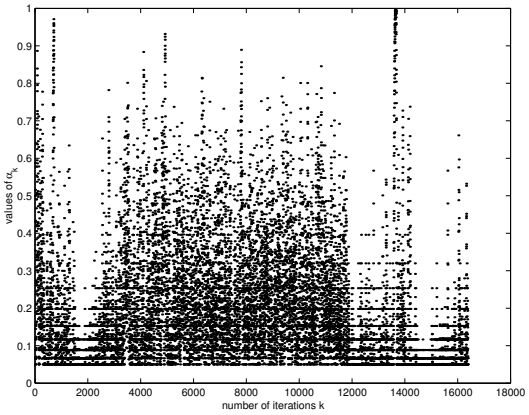

Figure 13: Probabilistic adaptation of crs($P_{gs+gl}, \cdot$) on SAL

in terms of cpu time. This is because of the time needed for fitting the linear model. However, there are problems for which function evaluations are extremely expensive in terms of cpu time. Under these circumstances, the use of linear interpolation can be justified as the cpu time needed by this algorithm will be negligible.

The new versions have more flexibilities than their original counterparts. In effect, we have generalised the old versions in that they are special cases of the new versions. We have also shown how the probabilistic adaptation guides an algorithm to improve its robustness and efficiency. Research is underway to develop even more efficient crs algorithms.

References

[1] A. Dekkers and E. Aarts, Global Optimization and Simulated Annealing, Mathematical Programming, Vol. 50, pp.367-393, 1991.

[2] M. M. Ali and C. Storey, Aspiration based Simulated Annealing Algorithms, Journal of Global Optimization, Vol. 11, pp.181-191, 1997.

[3] M. M. Ali, A. Törn and Viitanen, S., A Direct Search Variant of the Simulated Annealing Algorithm for Global Optimization involving Continuous Variables, Computers and Operations Research, Vol.29 (1), pp.87-102, 2002.

[4] D. R. Jones, C. D. Perttunen and B. E. Stuckman, Lipschitzian Optimization without the Lipschitz constant, Journal of Optimization Theory and Applications, Vol.79 (1), pp.157-181, 1993.

[5] R. Storn and K. Price, Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, Vol.11, pp.341-359, 1997.

[6] P. Kaelo and M. M. Ali, A Numerical Comparison of some Modified Differential Evolution Algorithms, European Journal of Operations Research, to appear in 2005.

[7] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, 1996.

[8] M. M. Ali and A. Törn, Population set based Global Optimization Algorithms : some Modifications and Numerical Studies, Computers and Operations Research, Vol.31 (10), pp.1703-1725, 2004.

[9] W. L. Price, A controlled Random Search Procedure for Global Optimization, in Towards Global Optimization 2, Dixon, L.C.W. and Szegö, G.P. (eds.), North-Holland, Amsterdam, Holland, pp.71-84, 1978.

[10] W. L. Price, Global Optimization by Controlled Random Search, Journal of Optimization Theory and Applications, Vol. 40, pp.333-348, 1983.

[11] R.B. Kearfott, Rigorous Global Search: Continuous Problems, Kluwer Academic, Dordrecht, 1996.

[12] P. Kaelo, Some Population set based Methods for Unconstrained Global Optimization, *PhD thesis*, to be submitted, 2005.

[13] Mohan, C. and Shanker, K., A Controlled Random Search technique for Global Optimization using quadratic approximation, Asia-Pacific Journal of Operational Research Vol.11, pp.93-101, 1994.

[14] J. A. Nelder and R. Mead, A Simplex Method for Function Minimization, Computer Journal, Vol. 7, pp.308-313, 1965.

[15] W. L. Price, Global Optimization Algorithms for a CAD Workstation, Journal of Optimization Theory and Applications, Vol. 55, pp.133-146, 1987.

[16] M. M. Ali and C. Storey, Modified Controlled Random Search Algorithms, International Journal of Computer Mathematics, Vol. 54, pp. 229-235, 1994.

[17] M. M. Ali, A. Törn and S. Viitanen, A Numerical Comparison of Some Modified Controlled Random Search Algorithms, Journal of Global Optimization, Vol. 11, pp. 377-385, 1997.

[18] P. Brachetti, M. De Felice Ciccoli, G. Di Pillo and S. Lucidi, A new version of the Price's Algorithm for Global Optimization, Journal of Global Optimization, Vol.10, pp.165-184, 1997.

[19] M. M. Ali, C. Storey and A. Törn, Application of some Stochastic Global Optimization Algorithms to Practical Problems, Journal of Optimization Theory and Applications, Vol. 95, No. 3, pp. 545-563, 1997.

[20] M. M. Ali, C. Khompatraporn and Z. B. Zabinsky, A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems, Journal of Global Optimization, In press, 2005.

[21] Marazzi, M. and Nocedal, J., Wedge Trust Region Methods for Derivative free Optimization, Mathematical Programming, Series A vol.91, 289-305, 2002.

[22] Nocedal, J. and Wright, S.J., Numerical Optimization. Springer Series in Operations Research, Springer, 1999.

[23] K. Najim, L. Pibouleau and M. V. Le Lann, Optimization Technique Based on Learning Automata, Journal of Optimization Theory and Applications, Vol. 64, pp.331-347, 1990.

[24] A. Törn, M. M. Ali and S. Viitanen, Stochastic Global Optimization: Problem Classes and Solution Techniques, Journal of Global Optimization Vol. .14, pp.437-447, 1999.