Comput Optim Appl
DOI 10.1007/s10589-013-9560-9

# CARTopt: a random search method for nonsmooth unconstrained optimization

**B.L. Robertson · C.J. Price · M. Reale**

Received: 12 May 2011

**Abstract** A random search algorithm for unconstrained local nonsmooth optimization is described. The algorithm forms a partition on $\mathbb{R}^n$ using classification and regression trees (CART) from statistical pattern recognition. The CART partition defines desirable subsets where the objective function $f$ is relatively low, based on previous sampling, from which further samples are drawn directly. Alternating between partition and sampling phases provides an effective method for nonsmooth optimization. The sequence of iterates $\{z_k\}$ is shown to converge to an essential local minimizer of $f$ with probability one under mild conditions. Numerical results are presented to show that the method is effective and competitive in practice.

**Keywords** Nonsmooth optimization · CART · Partitioning random search · Numerical results

## 1 Introduction

The unconstrained optimization problem is of the form

$$\min_x f(x) \quad \text{subject to } x \in \mathbb{R}^n, \tag{1}$$

where a local minimizer is sought. The objective function $f$ maps $\mathbb{R}^n$ into $\mathbb{R} \cup \{+\infty\}$ and in this paper $f$ is assumed to be nonsmooth and may be discontinuous. However, we do require that $f$ is lower semi-continuous. The inclusion of $\{+\infty\}$ means that the method can be applied to barrier functions and discontinuous penalty functions. That

B.L. Robertson · C.J. Price (✉) · M. Reale
Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand
e-mail: C.Price@math.canterbury.ac.nz

Published online: 09 April 2013

is, constrained problems can be dealt with directly by assigning the value $f = +\infty$ to infeasible points.

Since $f$ is assumed to be nonsmooth or discontinuous the standard definition of a local minimizer is modified.

**Definition 1** (Essential local minimizer) A point $x_* \in \mathbb{R}^n$ for which the set

$$\mathfrak{E}(x_*, \epsilon) = \left\{x \in \mathbb{R}^n : f(x) < f(x_*) \text{ and } \|x - x_*\| < \epsilon\right\}$$

has Lebesgue measure zero for all sufficiently small positive $\epsilon$ is called an essential local minimizer of $f$.

If $f$ is continuous at $x_*$, then $x_*$ is a local minimizer in the classical sense. The objective value $f(x_*)$ is called an essential local minimum.

Direct search methods can be applied to (1) when $f$ is assumed to be nonsmooth or discontinuous because no gradient information is required. However, when such assumptions are placed on $f$ theoretical convergence properties of these methods can be compromised. Partial convergence results have been presented by others. Audet and Dennis [2] provide a method which looks asymptotically in all directions at each limit point at which the Clarke derivative [6] is non-negative. However, the non-negativity of the Clarke derivative is only a partial result and does not preclude the existence of descent directions [16, 17, 21]. Vicente and Custódio [25] generalize both the algorithm of Audet and Dennis, and its analysis, by requiring only lower semi-continuity of $f$, but still require that $f$ is directionally Lipschitz at cluster points. Price, Reale and Robertson [16] showed that finding a descent step for a nonsmooth problem is closely related to a global optimization problem. Exploiting this connection allows for a class of algorithms to be constructed with strong theoretical convergence results. In these algorithms the Clarke derivative approach is replaced with one that exhaustively searches for points of descent in the neighborhood of potential limit points of a local search algorithm. If descent is found in this localized global optimization phase the local search procedure restarts from the point of descent. Otherwise no descent is forthcoming and an essential local minimizer is located. Alternating between a local search algorithm and localized global phases provides an effective strategy for solving nonsmooth problems [16, 17, 21].

Of particular interest in this paper are partitioning random search algorithms. These methods attempt to increase the efficiency of Pure Random Search (PRS) [19], where $f$ is simply evaluated at a number of randomly generated points and an estimate of the global minimum is given by the lowest function value obtained. Partitioning algorithms modify PRS by forming a partition on an optimization region and drawing samples from the particular sub-regions of the partition. Using each batch of points, the partition and/or the sampling distribution is iteratively updated—usually drawing more samples from sub-regions of the partition where $f$ is observed to be low. The interested reader is referred to [1, 18, 21, 23] and references therein for a more detailed description and various algorithms. However, these algorithms are designed for constrained global optimization and often require successive partitions to be nested. Thus, applying these methods to local optimization can be computationally expensive and is heavily dependent on choosing an appropriate optimization region.

We seek a partitioning random search algorithm for unconstrained local optimization. Our approach is to use a set of points called the training data set, which is updated at the end of each iteration. The basic form of an iteration is as follows:

(i) Place an oriented box around the training data set.
(ii) Partition the box into low and high subregions.
(iii) Randomly sample in the low regions of the partitioned box.
(iv) Form the next training data set from the current one and the sample points generated in step (iii).

The intent here is to search randomly in neighborhoods of the lowest known points. As lower points are discovered, subsequent iterations place boxes around these new points, allowing the search to move as far as is necessary to reach a local minimizer. Once in the vicinity of a local minimizer, successive boxes will contain the minimizer, allowing step (iii) to explore in an open neighborhood of that minimizer. The partition is constructed using planar cuts parallel to the faces of the enclosing box. The box's orientation is chosen to try and make the partitioning and subsequent sampling as simple as possible. In practice our algorithm chooses the box's orientation in step (i), and implicitly constructs enough of the box and its partition in step (ii) to allow step (iii) to be done.

The paper is organized as follows. Firstly, the unconstrained CARTopt algorithm is introduced. Section 3 defines the training data set used to form the partition on $\mathbb{R}^n$. An invertible data transformation is proposed in Sect. 4 which potentially simplifies the partition. The partition itself is introduced in Sect. 5 and the sampling method in Sect. 6. Convergence results are given in Sect. 7, where $f$ is assumed to be nonsmooth or discontinuous. A new stopping rule based on CARTopt's local random sampling property is given in Sect. 8. Numerical results are presented in Sect. 9 and concluding remarks are given in Sect. 10.

## 2 Unconstrained CARTopt algorithm

CARTopt is a random search method that uses randomly generated sample points to form a partition on $\mathbb{R}^n$ using a statistical classification method. As the name suggests, Classification and Regression Trees (CART) [4, 8] is the method used to form the partition. Such a partition divides $\mathbb{R}^n$ into a set of non-empty hyperrectangular subregions aligned with the coordinate axes. To form a partition using CART a training data set of at least two different categories is required. Here we classify a fraction of randomly generated samples as being low with respect to $f$, and the rest of the samples as high. Applying the CART partitioning technique to such a training data set partitions $\mathbb{R}^n$ into a set of low and high sub-regions. Using the CART partition, we define an approximate level set $\mathcal{L}$.

**Definition 2** (Approximate level set) Let $A_i$ denote the $i$th low sub-region of the CART partition on $\mathbb{R}^n$, where $1 \leq i \leq |\text{low sub-regions}|$, then the approximate level set is

$$\mathcal{L} = \left\{ x \in \bigcup_i A_i \right\}.$$

```
1. Initialize: Set $k = 1$. Choose $N > 0$, $\delta > 0$, $h > 0$, $x_0 \in \mathbb{R}^n$
   and $T_1$. Store $x_0$ in $T_1$. Pick $\phi \in (0,1)$.
2. Generate $\max\{2N - |T_1|, 0\}$ random sample points $x \in x_0 +$
   $h[-1,1]^n$, store in $X_1$, and evaluate $f(x)$ at each $x \in X_1$.
   Let $z_1$ minimize $f$ over $\{X_1, T_1\}$.
3. while stopping conditions are not satisfied do
     (a) Update the training data set: $T_{k+1} \subseteq \{X_k, T_k\}$ such
         that $z_k \in T_{k+1}$. (see Sect. 3)
     (b) Classify $T_{k+1}$: Set $\omega_L$ as the $\min\{\lfloor \phi N \rfloor, |\{x \in T_{k+1} : f(x) \neq \infty\}|\}$
         elements of $T_{k+1}$ with the least function values, and set $\omega_H = T_{k+1} \setminus \omega_L$.
     (c) Transform the training data set: Calculate the
         Householder matrix $H_k$ and set

             $\hat{T} = H_k T_{k+1}^{\mathsf{T}}$.

     (d) Form CART partition on $\mathbb{R}^n$ using $\hat{T}$ to identify low
         sub-regions whose union is $\mathcal{L}_k$.
     (e) Generate next batch of $N$ random points $X_{k+1}$ in $\mathcal{L}_k$.
     (f) Apply inverse transform: Set $X_{k+1} = X_{k+1}^{\mathsf{T}} H_k$.
     (g) Evaluate $f(x)$ at each $x \in X_{k+1}$ and call the best
         point $\hat{x}$. If $f(\hat{x}) < f(z_k)$, set $z_{k+1} = \hat{x}$, otherwise set
         $z_{k+1} = z_k$. Check stopping conditions and increment $k$.
   end
```

**Fig. 1** CARTopt algorithm

The approximate level set defines a subset of $\mathbb{R}^n$ where $f$ is potentially low, based on where $f$ has been sampled, and is constructed in such a way that $m(\mathcal{L})$ is always finite. Here $m(.)$ denotes the Lebesgue measure. An extremely useful property of $\mathcal{L}$ is that it consists of a union of hyperrectangles (see Fig. 2). Thus, it is simple to draw further samples from $\mathcal{L}$. Therefore, alternating between partition and sampling phases in an algorithmic manner provides a method for sampling subsets of $\mathbb{R}^n$ which are relatively low, based on known function values. This is the basis for the new method.

The algorithm generates a sequence of iterates $\{z_k\}_{k=1}^{\infty} \subset \mathbb{R}^n$. Each iterate $z_{k+1}$ is the best point at the end of iteration $k$, and is generated from its predecessor $z_k$ by evaluating $f$ at a finite number of points in the approximate level set $\mathcal{L}_k$, defined from the CART partition. The partition is reflected using a Householder matrix so that the hyperrectangles do not have to be aligned with the original coordinate system. This is done by reflecting the training data set, partitioning $\mathbb{R}^n$, selecting the new sample points and mapping these points back to the original coordinate system. The loop is executed until the stopping conditions are satisfied, where the estimate of an essential local minimizer is $z_{k+1}$.

A precise statement of CARTopt is given in Fig. 1. The algorithm consists of an initialization phase (steps 1 and 2) and a single loop (step 3). Step 1 sets the iteration counter $k = 1$ and the user chooses the batch size $N > 0$, a minimum sub-region radius $\delta > 0$, an initial search box radius $h > 0$ and an initial point $x_0 \in \mathbb{R}^n$ such that $f(x_0)$ is finite. Choosing $\delta > 0$ ensures that $m(\mathcal{L})$ is bounded away from zero for all iterations, where $m(.)$ denotes the Lebesgue measure. This property is needed for convergence on nonsmooth problems. Numerical experience suggests choosing a large $h$ is advantageous. This allows the algorithm to take larger steps early on rather than lots of little ones. In addition, an input training data set $T_1$ can be used if the user has a priori knowledge of $f$. If no input training data exists, $T_1$ contains $x_0$ only. Step 2 generates the first set of points $X_1$, drawn from a uniform distribution over $x_0 + h[-1, 1]^n$ and corresponding objective function values. Each point is stored as a row vector of the matrix $X_1$.

The steps of the loop are discussed in detail in the sections that follow.

## 3 Training data and classification

The training data set $T$ is used as a model from which CART forms a partition on $\mathbb{R}^n$. Here a dynamic training data set which reflects information about $f$ obtained during the previous sampling phase is proposed. Of particular interest is locating subsets of $\mathbb{R}^n$ where $f$ is relatively low. Sampling such sets further and updating the training data set allows a new partition to be formed, defining a new subset to be explored.

An initial classification must be placed on $T$ before any partition can be formed. There is a great deal of freedom when imposing a classification on $T$. Here two categories are chosen, points with relatively low and relatively high function values respectively, defined formally below. The notation $\lfloor \lambda \rfloor$ denotes the greatest integer less than or equal to $\lambda$.

**Definition 3** (Low points) Given $0 < \phi < 1$, the $\min\{\lfloor \phi N \rfloor, |\{x \in T : f(x) \neq \infty\}|\}$ elements of $T$ with the least function values are classified as low and form the set $\omega_L$.

**Definition 4** (High points) The set of points $T \setminus \omega_L$ is classified as high and form the set $\omega_H$.

If $f$ is finite at all training data points, $\omega_L$ is simply the $\lfloor \phi N \rfloor$ elements of $T$ with the least function values. The best iterate, $z_k$, is always included in $\omega_L$. Furthermore, each element of $\omega_L$ is finite and with $f(x_0)$ finite, $\omega_L \neq \emptyset$. Hereafter, $\phi = 0.8$ is used although other choices are possible. Therefore $|\omega_L| \leq 0.8N$ for all $k$, which has two important consequences. Firstly, the number of distinct low sub-regions is bounded by $0.8N$, even if $|T|$ becomes large. Secondly, $|\omega_L| < |\omega_H|$ for all $k$ since $2N$ points are initially generated, which can cause $\omega_L$ to cluster as $k$ increases.

Let us now consider how $T$ is updated. The primary interest here is local (not global) optimization and so it is sufficient for $T$ to reflect local information about $f$. Forming the $k$th partition using all $(k+1)N$ points generated from previous iterations (global information), can complicate the partition and is computationally expensive.

A sufficient number of points surrounding $\omega_L$ are required to define the approximate level set $\mathcal{L}$ effectively. Loosely speaking, at least $2n$ high points are required to define each low hyperrectangular sub-region, one for each face. Therefore, as dimension increases, CART requires a larger training data set to effectively partition $\mathbb{R}^n$. Fortunately the number of faces grows linearly with $n$, as does the maximum training data size used here. Numerical simulations performed by the first author found a maximum size $|T| \leq \max\{2N, 2(n-1)N\}$, performed well in practice on a variety of problems.

**Definition 5** (Full size training data) A training data set $T$ such that,

$$|T| = \max\bigl\{2N, 2(n-1)N\bigr\},$$

is called a full size training data set.

The training data is updated iteratively using the batch of newly generated points $X_k$ via $T_{k+1} \subseteq \{X_k, T_k\}$. The most recent sample points appear first in the update. This gives an indication of the *age* of points as $T$ grows. When $T_{k+1} = \{X_k, T_k\}$ exceeds full size, the oldest points with relatively high function values are discarded. Specifically, $T_{k+1} = \{X_\gamma, X_R\}$, where $X_\gamma$ is the set of $\gamma > 0$ sample points with the least function values and $X_R$ is the set of most recent points (not included in $X_\gamma$) to ensure $T_{k+1}$ is full size. Hereafter $\gamma = 2N$ is chosen, which is the number of function values required for the stopping condition [21].

Ideally, updating and classifying $T$ causes $\omega_L$ to cluster in the neighborhood of an essential local minimizer $x_*$. Fixing $|\omega_L|$, keeping points with the least $f$ values and the most recent sample points tries to achieve this. Numerical simulations performed by the first author found that although initially $\omega_L$ may be disjoint, $\omega_L$ tends to cluster into a hyper-elliptical cloud, eventually in the neighborhood of $x_*$.

## 4 Transforming training data sets

When forming a partition on $\mathbb{R}^n$ the fundamental principle is that of simplicity. The CART partition performs best when the components of $T$ have an alignment with the coordinate axes. Since all potential splits are orthogonal to the coordinate axes [20], such an alignment can simplify the partition. Multivariate splits, as in Binary Space Partition Trees (BSP) [8], can also simplify the partition because potential splits are expressed as linear combinations of the coordinate axes. However, such splits can be computationally expensive and it is difficult to draw samples directly from the convex polyhedral sub-regions of the partition. Another approach is preprocessing $T$ before the partition is formed. However, choosing a transform to simplify the partition for a general problem is often difficult. The interested reader is referred to [8] for various preprocessing techniques.

Ideally, a transform $\mathfrak{D} : T \to \hat{T}$ is desired such that, the partition induced from $\hat{T}$ is *simpler* than the one using $T$. Here simpler means fewer splits in the training data, and hence less sub-regions in the partition. The CARTopt approach alternates

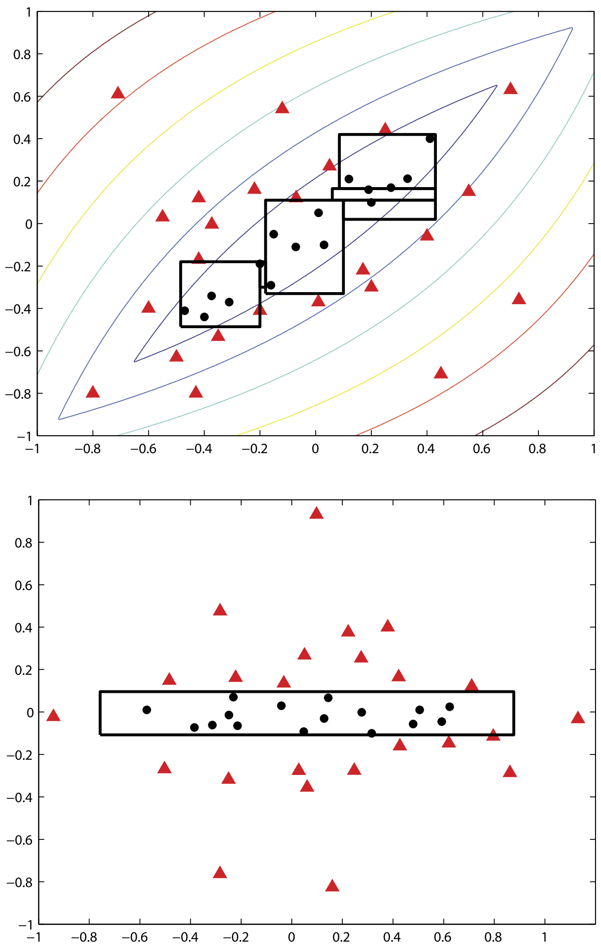

**Fig. 2** *The first figure* shows the level curves of $f = 2|x_1 - x_2| + x_1 x_2$, along with the training data set $T = \{\omega_L, \omega_H\}$, with $x \in \omega_L$ and $\xi \in \omega_H$ denoted $\bullet$, $\blacktriangle$ respectively. Six low sub-regions are formed in the CART partition which define $\mathcal{L}$ (the 6th sub-region is between the two large sub-regions on the left hand side). *The second figure* shows the transformed training data using the Householder transformation. The transformation simplifies the partition to one sub-region, producing a good approximate level set

between partition and sampling phases, and thus $\mathfrak{D}$ must be invertible so $f$ can be evaluated at points generated in the transformed space.

Numerical simulations performed by the first author found that although initially $\omega_L$ may be disconnected, $\omega_L$ often forms a hyper-elliptical cloud of points. Thus, transforming $T$ so that principal axis of the hyper-elliptical cloud aligns with a coordinate axis can be advantageous. Such a transform can dramatically simplify the partition, particularly in the neighborhood of some essential local minimizers where hyper-elliptical contours are present. This is illustrated in Fig. 2.

The principal axes of the hyper-elliptical cloud $\omega_L$ are obtained using Principal Component Analysis (PCA) [8]. Using $\omega_L^{(i)}$ to denote the $i$th element of $\omega_L$, expressed as a row vector, the scatter matrix is defined by

$$\mathcal{M} = \sum_{i=1}^{|\omega_L|} \left[\omega_L^{(i)} - \bar{\omega}_L\right]^{\mathrm{T}} \left[\omega_L^{(i)} - \bar{\omega}_L\right],$$

where $\bar{\omega}_L$ is the sample mean,

$$\bar{\omega}_L = \frac{1}{|\omega_L|} \sum_{i=1}^{|\omega_L|} \omega_L^{(i)}.$$

The dominant eigenvector $d$, of the scatter matrix $\mathcal{M}$, is the direction vector for the principal axis of the hyper-elliptical cloud. Using the Householder transformation,

$$\begin{aligned} H &= I - 2uu^{\mathsf{T}}, \\ u &= (e_1 - d)/\|e_1 - d\|, \end{aligned} \tag{2}$$

the $x_1$ axis is set parallel to $d$, i.e. $He_1 = d$. Pre-multiplying each element of $T$ with $H$ gives the desired transformed training data $\hat{T}$. This transformation is not only appealing because it keeps the coordinate directions orthogonal, but also the inverse transformation is trivial ($H^{\mathsf{T}} \equiv H^{-1}$).

In summary, the transformed training data $\hat{T}$ is defined as

$$\mathfrak{D}(T) = HT^{\mathsf{T}} = \hat{T},$$

which is implemented in Step 3(c) of CARTopt. The inverse transform is simply

$$\mathfrak{D}^{-1}(\hat{T}) = \hat{T}^{\mathsf{T}} H = T,$$

which is implemented in Step 3(f) of CARTopt. Numerical simulations found this transform to be extremely effective in practice, increasing the numerical performance of the algorithm significantly.

## 5 CART partition

CART partitions $\mathbb{R}^n$ into a set of non-empty hyperrectangular sub-regions $A_i$, such that $\cup_i A_i = \mathbb{R}^n$ and $A_i^{\circ} \cap A_j^{\circ} = \emptyset$, $i \neq j$, where $A_i^{\circ}$ denotes the interior of $A_i$. The partition is formed by considering splits in $T$ of the form

$$s = (x_j + \xi_j)/2 \quad \text{where } x \in \omega_L \text{ and } \xi \in \omega_H.$$

The partition is formed iteratively using a greedy strategy which chooses splits that maximize an impurity relation [20]. That is, each split tries to make the resulting sub-regions as pure as possible—mainly low, or mainly high points in each sub-region. When each sub-region contains either low or high points only, the partition is complete. The interested reader is referred to [20, 21] for full details on the partition. However, in this paper post-partition modifications are made. The modifications are simple to implement, keep the hyperrectangular structure of the partition and are not necessarily implemented at each iteration, only when required. Each is discussed in the subsections which follow.

A matrix $B$ contains the bounds of each low sub-region $A_i$ of the partition. Each row of $B$ defines the lower and upper dimension bounds of $A_i$ and has the following structure

$$B_i = [b_1, \dots, b_n, b_{n+1}, \dots, b_{2n}],$$

where $b_j \le x_j \le b_{j+n}$ for all $x \in A_i$ and $j = 1, \dots, n$. If $A_i$ is unbounded below (above) in a particular dimension $j$, then $b_j = -\infty$ ($b_{j+n} = +\infty$).

### 5.1 Minimum sub-region radius

To exploit the connection between nonsmooth local optimization and global optimization [16], it is necessary for the $m(\mathcal{L}_k)$ to be bounded away from zero for all $k$. This ensures CARTopt performs localized global optimization in the neighborhood of potential cluster points by applying random search in a subset of $\mathbb{R}^n$ with positive measure at each iteration. To enforce this condition a minimum splitting distance $2\delta > 0$ between elements of $\omega_L$ and $\omega_H$ is imposed in CARTopt, post-partition. Formally, for each low sub-region $A_i$, each $x \in \omega_L \cap A_i$ must be at least a distance $\delta$ away from all points on the boundary of $A_i$. For sufficiently close splits, the splitting hyperplane is *pushed* away from closest $x \in \{\omega_L \cap A_i\}$. That is, each lower bound $b_j \in \{B_i(j) : 1 \le j \le n\}$ and each upper bound $b_{j+n} \in \{B_i(j+n) : 1 \le j \le n\}$ is set to

$$B_i(j) = \min\{b_j, x_j^- - \delta\}, \tag{3}$$

$$B_i(j+n) = \max\{b_{j+n}, x_j^+ + \delta\}, \tag{4}$$

for each $A_i$, where $j = 1, \dots, n$. Here $x_j^+ = \max\{x_j \in \{\omega_L \cap A_i\}\}$, and $x_j^- = \min\{x_j \in \{\omega_L \cap A_i\}\}$.

The update given by (3) and (4) has two important consequences. Firstly, it removes the possibility of samples converging to an impassable hyperplane boundary. Secondly, a minimum sub-region radius $\delta > 0$ on each $A_i$ is forced. Therefore, $m(\mathcal{L}_k) \ge \delta^n$ for all $k$ and so is bounded away from zero as required. The latter giving convergence on nonsmooth problems, see Sect. 7. Herein $\delta = 1\text{e}{-10}$ is chosen.

Applying the minimum sub-region radius update can destroy the desirable property, $A_i^o \cap A_j^o = \emptyset$ for all $i \neq j$, inherent to the CART partition. To remove this problem the boundary of $\mathcal{L}$ could be extended, rather than individual sub-regions. However, this update is usually only applied when $m(\mathcal{L})$ is approaching the limiting size and has no adverse effects on the algorithm.

### 5.2 Updating problematic sub-region bounds

The partition is conducted in $\mathbb{R}^n$ and hence, it is possible that particular low sub-regions are unbounded. That is, there may exist a $b \in \{B_i(q) : 1 \le q \le 2n\}$ such that $|b| = \infty$ for a particular $A_i$. Such a bound is considered problematic, defined formally below.

**Definition 6** (Problematic bound) A sub-region bound $b \in \{B_i(q) : 1 \le q \le 2n\}$ for a low sub-region $A_i$ is considered a problematic bound if $|b| = \infty$.

All problematic bounds warrant further investigation. Here two distinct low sub-regions that can be formed with the CART partition are considered.

**Definition 7** (Singleton sub-region) A low sub-region $A_i$ is considered a singleton sub-region if $|\{x : x \in \omega_L \cap A_i\}| = 1$.

**Definition 8** (Non-singleton sub-region) A low sub-region $A_i$ is considered a non-singleton sub-region if $|\{x : x \in \omega_L \cap A_i\}| > 1$.

Firstly, non-singleton low sub-regions with problematic bounds are considered and singleton low sub-regions are left until the next subsection.

To remove problematic bounds from non-singleton low sub-regions the following method is used. The method is computationally cheap to evaluate and maintains the hyperrectangular structure of $\mathcal{L}$. Each problematic bound is replaced by a new bound which fits the training data best.

Consider a low sub-region $A_i$ with at least one problematic bound. Each problematic bound corresponds to a face on the hyperrectangle which defines $A_i$. Initially, each problematic lower bound $b_j \in \{B_i(j) : 1 \le j \le n\}$ is updated to

$$B_i(j) = x_j^- - \alpha \max(r_j, \delta) \tag{5}$$

and each problematic upper bound $b_{j+n} \in \{B_i(j+n) : 1 \le j \le n\}$ is updated to

$$B_i(j+n) = x_j^+ + \alpha \max(r_j, \delta), \tag{6}$$

where $\alpha = \frac{1}{3}$, $\delta$ is the minimum sub-region radius and $r_j = x_j^+ - x_j^-$ is the range in dimension $j$ of the set $\{x : x \in \omega_L \cap A_i\}$. Other choices of $\alpha > 0$ are possible provided they are finite. This initial step brings all problematic bounds closer to the convex hull of points in $A_i$. The updated bounds are tested by drawing a point $x$ from an uniform distribution over each updated face of $A_i$ and evaluating $f(x)$.

**Definition 9** (Upper-face) For a low sub region $A_i$ with bounds $B_i$, the face defined by setting $b_j \leftarrow b_{j+n}$ is called the $j$th upper-face of $A_i$, where $1 \le j \le n$.

**Definition 10** (Lower-face) For a low sub region $A_i$ with bounds $B_i$, the face defined by setting $b_{j+n} \leftarrow b_j$, is called the $j$th lower-face of $A_i$, where $1 \le j \le n$.

The $j$th upper and lower-faces define subsets of the hyperplanes $x_j = b_{j+n}$ and $x_j = b_j$ respectively. Therefore, generating an $x$ uniformly on $j$th face is equivalent to uniformly sampling $A_i$ with the $j$th coordinate fixed (see Sect. 6.2). If $f(x) > f(x_j^+)$ on the $j$th upper-face of $A_i$, a higher function value has been generated and $x$ is included in $\omega_H$. In this case the upper bound $b_{j+n} \in \{B_i(j+n) : 1 \le j \le n\}$ is fixed (by (6)). Similarly, if $f(x) > f(x_j^-)$ on the $j$th lower-face, the lower bound $b_j \in \{B_i(j) : 1 \le j \le n\}$ is fixed (by (5)). Otherwise, descent was made, $x$ is added to $\omega_L$ and the bound remains problematic.

For all remaining problematic bounds, $\alpha$ is increased iteratively in (5) and (6), in a standard forward-tracking manner. Here the sequence $\alpha = \frac{1}{3}, 1, 3, \ldots, 3^{10}$ is used although other choices are possible. The method terminates when either every $b \in \{B_i(q) : 1 \leq q \leq 2n\}$ is not considered problematic, or $3^{10}$ is reached. For the latter the bound is fixed with $\alpha = 3^{10}$. Each non-singleton $A_i \subset \mathcal{L}_k$ is considered in turn and the update is applied ensuring $\mathcal{L}_k$ is finitely bounded. The additional samples are included in the next training data set as recent points. Updating $T$ reduces the risk of having problematic bounds during the next iteration.

### 5.3 Singleton low sub-regions

The CART method can form a partition on $\mathbb{R}^n$ such that, singleton low sub-regions exist. These sub-regions can be problematic and are often partially defined by two close, parallel hyperplanes. Rather than applying the update described above, each singleton sub-region is replaced by a hypercube.

In the extremely unlikely case when every low sub-region is a singleton sub-region, the partition is updated as follows. For each $x \in \omega_L$ at iteration $k$, a hypercube with center $x$ and radius $r$ defines each sub-region. The hypercube radius is based on $m(\mathcal{L}_{k-1})$ to ensure the algorithm searches at a similar level to the previous iteration, keeping the search focused. Specifically,

$$r = \frac{1}{2}\max\left\{\left(\frac{\sum_{i=1}^{|A|}\prod_{j=1}^{n}(b_{j+n}^{(i)} - b_j^{(i)})}{|\omega_L|}\right)^{1/n}, \delta\right\},$$

where, $b_{j+n}^{(i)}$ and $b_j^{(i)}$ are the bounds on sub-region $A_i$ on the previous approximate level set $\mathcal{L}_{k-1}$, with $\mathcal{L}_0 = h[-1, 1]^n$. Here $|A|$ denotes the number of low sub-regions. The approximate level set $\mathcal{L}_k$ is defined by the union of hypercubes. The inclusion of $\delta$ ensures $m(\mathcal{L}_k) > 0$ for all $k$. Each singleton sub-region has equal measure and thus, has equal probability of being sampled during the next sampling phase, see next section.

When there exists at least one non-singleton low sub-region, the partition is updated as follows. The notation $\bar{A}_i$ is used to denote singleton sub-regions of $\mathcal{L}_k$. For each $x \in \{\omega_L \cap \bar{A}_i\}$, a hypercube replaces each $\bar{A}_i$ with center $x$ and radius $r$, defined by

$$r = \frac{1}{2}\max\left\{\left(\frac{\sum_{i=1}^{m}\prod_{j=1}^{n}(b_{j+n}^{(i)} - b_j^{(i)})}{|\omega_L| - |\bar{A}|}\right)^{1/n}, \delta\right\}, \tag{7}$$

where $|\bar{A}|$ is the number of singleton sub-regions and the $m$ non-singleton sub-regions $A_i \subset \mathcal{L}_k$, $i = 1, \ldots, m$, have upper and lower bounds $b_{j+n}^{(i)}$ and $b_j^{(i)}$ on the $j^{\text{th}}$ coordinate for $j = 1, \ldots, n$. Replacing each $\bar{A}_i$ with a hypercube, $\mathcal{L}_k$ is updated as the union of low sub-regions. After applying (7), each singleton sub-region occupies $1/|\omega_L|$ of the volume of $\mathcal{L}_k$. This is chosen because each $x \in A_i$ occupies $1/|\omega_L|$ of the point space of $\omega_L$.

## 6 Generating the next batch

At each iteration the batch of $N$ points $X_{k+1}$ is distributed into the approximate level set $\mathcal{L}_k$. This increases the chance of reducing $f$, and also increases the chance of reducing the measure of $\mathcal{L}$ at each iteration, down to the limiting size. Ideally each $\mathcal{L}$ would be nested, however, as each is an approximate level set this is usually not true. A fraction of samples can be drawn outside of $\mathcal{L}$, particularly if (1) is a bound constrained problem. The interested reader is referred to [21] for details on sampling outside of $\mathcal{L}$ and such sampling is considered no further here.

### 6.1 Batch size

Before describing how each batch of points is generated, the batch size $N > 0$ is chosen. To choose a suitable $N$ the efficiency of uniform sampling over $\mathcal{L}$ is considered. Firstly, let us define the $\eta$-percentage set of $\mathcal{L}$.

**Definition 11** ($\eta$-Percentage set) Let $f_-$ and $f_+$ denote the minimum and maximum of $f$ on $\mathcal{L}$, respectively. For any $\eta \in [0, 1]$, assume there exists an $f_\eta \in [f_-, f_+]$ for which the set

$$\mathcal{S}(\eta) = \{x \in \mathcal{L} : f(x) < f_\eta\},$$

has $m(\mathcal{S})/m(\mathcal{L}) = \eta$. Such a set is called the $\eta$-percentage set of $\mathcal{L}$.

Therefore, $\mathcal{S}(1) = \mathcal{L}$ and in the limit as $\epsilon \to 0$, $\mathcal{S}(\epsilon)$ converges to the set of minimizers $\{x \in \mathcal{L} : f(x) = f_-\}$.

Let us now consider how many sample points are required to generate at least one point in each $\mathcal{S}(\eta)$ as $\eta$ is decreased. The probability that there exists at least one point $x \in \mathcal{S}(\eta)$ out of $N$ uniform draws over $\mathcal{L}$, is given by,

$$\Pr = 1 - (1 - \eta)^N. \tag{8}$$

Choosing a high probability, $\Pr = 0.99$ and rearranging (8) with respect to $N$,

$$N = \frac{\ln(0.01)}{\ln(1 - \eta)},$$

the expected number of points required to generate an $x \in \mathcal{S}(\eta)$ is obtained.

From Fig. 3 it is clear that uniform sampling is very effective at reducing $f$ in the early stages of sampling, only requiring 44 sample points to generate an $x \in \mathcal{S}(0.1)$ with probability 0.99. However, to reduce $f$ significantly further, generating an $x \in \mathcal{S}(0.05)$ say, an extremely large sample size is required.

At each iteration $\mathcal{L}_k$ is sampled using a near-uniform distribution. Therefore, to maintain algorithm efficiency, it would be advantageous to alternate between partition and sampling phases with relatively small batch sizes. Sampling $\mathcal{L}_k$ for too long would cause CARTopt to become inefficient, with a large number of samples generated in $\mathcal{L}_k$ failing to reduce $f$ below the current lowest function value. Furthermore, each partition phase defines a new, possibly smaller, promising subset of $\mathbb{R}^n$ to search. Here the value $N = 20$ is chosen. Hence, $\mathcal{L}_k$ is sampled until an $x \in \mathcal{S}(0.2)$ has been obtained with probability 0.99, before another partition phase is implemented. Numerical simulations performed by the first author support this batch size.

For the initial batch $2N = 40$ samples are used. Therefore, a generated point $x \in \mathcal{L}_0 = x_0 + h[-1, 1]^n$ will be an element of $\mathcal{S}(0.1)$ with high probability. This gives

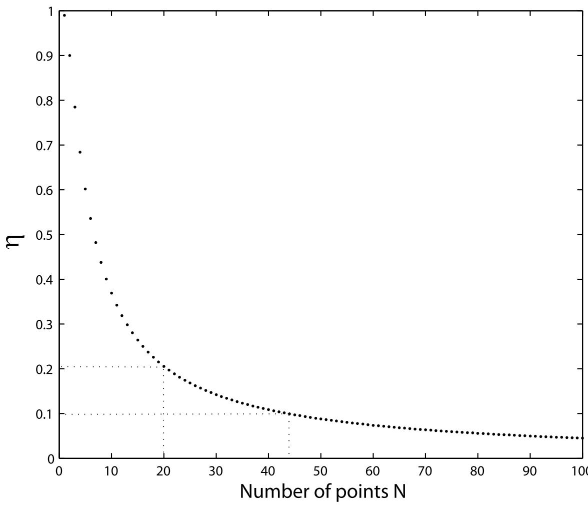

**Fig. 3** The expected number of sample points required to generate an $x \in \mathcal{S}(\eta)$ from uniform sampling over $\mathcal{L}$, with probability 0.99

the algorithm at least one sufficiently low point in the initial partition and hence, a promising sub-region to begin the search for an essential local minimizer.

## 6.2 Delivery method for new sample points

To sample the approximate level set a near-uniform distribution over $\mathcal{L}_k$ is used. The delivery method requires a two stage process. Firstly a low sub-region $A_i \subset \mathcal{L}_k$ is selected, then a point $x$ is drawn from a uniform distribution over $A_i$. To select each $A_i$ a simple discrete inverse transform method is used [13]. The cumulative distribution function $F$ for the sub-region measure of $\mathcal{L}_k$ is given by

$$F(A_p) = \frac{\sum_{q=1}^{p} m(A_q)}{\sum_i m(A_i)} = \frac{\sum_{q=1}^{p} \prod_{j=1}^{n} (b_{j+n}^{(q)} - b_j^{(q)})}{\sum_i \prod_{j=1}^{n} (b_{j+n}^{(i)} - b_j^{(i)})},$$

where $m(.)$ denotes Lebesgue measure and $b_{j+n}^{(i)}$ and $b_j^{(i)}$ are the bounds on sub-region $A_i$. A particular $A_i$ is selected using

$$A_i = \min\bigl(i : U \le F(A_i)\bigr),$$

where $U \in [0, 1]$ is a random variable. Hence, more samples are expected to be drawn from larger sub-regions of $\mathcal{L}_k$.

Upon selecting a low sub-region $A_i$ a point $x \in A_i$ is delivered using

$$x = \bigl[b_{n+1}^{(i)} - b_1^{(i)}, \ldots, b_{2n}^{(i)} - b_n^{(i)}\bigr] U_n + \bigl[b_1^{(i)}, \ldots, b_n^{(i)}\bigr],$$

where $U_n$ is a diagonal matrix of rank $n$ with each non-zero element $u_{jj} \in [0, 1]$ a random uniform variable. The method repeats until the batch of $N$ points is obtained.

If no post-partition modifications were required at iteration $k$, $A_i^{\circ} \cap A_j^{\circ} = \emptyset$ for all $i \neq j$. Therefore, the proposed delivery method samples $\mathcal{L}_k$ uniformly. Whereas, a modified partition can have an overlap such that $A_i^{\circ} \cap A_j^{\circ} \neq \emptyset$. All overlaps have a greater probability of being sampled as they can be sampled from multiple sub-regions. However, such an overlap tends to be small, giving near-uniform sampling on $\mathcal{L}_k$ and does not have any adverse effects on the algorithm.

## 7 Convergence analysis

The convergence properties of CARTopt are analyzed with the stopping conditions removed. This allows us to examine the asymptotic properties of the sequence of iterates generated by the algorithm. Stopping conditions are included from a practical point of view.

Before the result is presented, we show that the CARTopt algorithm generates an infinite sequence of iterates.

**Theorem 12** *The sequence $\{z_k\}$ generated by the CARTopt algorithm is an infinite sequence.*

*Proof* For $\{z_k\}$ to be an infinite sequence, Steps 3(a–g) of CARTopt must be finite processes. Noting that $|T_k|$ is finite for all $k$, Steps 3(a, b, c, f, g) are finite processes. Step 3(e) is a finite process with $N$ (finite) points drawn from the finitely bounded $\mathcal{L}_k$.

The CART partition in Step 3(d) uses a finite data set $T_k$ and only considers a finite number of splits from the set $\{s = (x_j + \xi_j)/2 : x, \xi \in T_k\}$. The resulting partition on $\mathbb{R}^n$ is a finite set of non-empty hyperrectangles, where the number of distinct low sub-regions is bounded by $|\omega_L| = \lfloor 0.8N \rfloor$. Each low sub-region update requires only a finite number of steps. Therefore, the partition phase is a finite process and thus, Step 3(d) is finite. $\square$

The convergence result for the unconstrained CARTopt algorithm can now be given. To establish convergence to an essential local minimizer of $f$ the following assumptions are required.

**Assumption 13** The following conditions hold:

(a) the points at which $f$ is evaluated at lie in a compact subset of $\mathbb{R}^n$;
(b) the sequence of function values $\{f(z_k)\}$ is bounded below; and
(c) the objective function $f$ is lower semi-continuous.

The first two assumptions ensure $\{z_k\}$ is bounded and exclude the case where $f(z_k) \to -\infty$ as $k \to \infty$. Assumption 13(c) ensures that

$$\liminf_{\zeta \to z_*} f(\zeta) \geq f(z_*),$$

where $z_*$ is a cluster point of $\{z_k\}$. Without Assumption 13(c), the CARTopt algorithm does not always converge to an essential local minimizer. Consider, for example, the function

$$f(x) = \begin{cases} \|x\| & x \neq 0 \\ 1 & x = 0. \end{cases} \tag{9}$$

The origin is not an essential local minimizer of (9), but the algorithm will give $\{z_k\} \to 0$ as $k \to \infty$ almost surely. Modifying (9) so that $f = -1$ at $x = 0$ makes $f$ lower semi-continuous and the origin is now an essential local minimizer.

The convergence result for the unconstrained CARTopt algorithm shows that every cluster point of the sequence $\{z_k\}$ is an essential local minimizer of $f$ with probability one.

**Theorem 14** *Let Assumption 13 hold. Each cluster point $z_*$ of the sequence $\{z_k\}$ is an essential local minimizer of $f$ with probability one.*

*Proof* Theorem 12 and Assumption 13 ensure the existence of cluster points in $\{z_k\}$.

Let $z_*$ be any cluster point of $\{z_k\}$ and assume, by contradiction, that $z_*$ is not an essential local minimizer of $f$. Then, there exists a set

$$\mathfrak{E} \subset \left\{z \in \mathbb{R}^n : f(z) < f(z_*) \text{ and } \|z - z_*\| < \delta/2\right\}, \tag{10}$$

with $m(\mathfrak{E}) > 0$. Also there exists an infinite subsequence $\mathcal{K} \subset \mathbb{Z}_+$ such that

$$\|z_k - z_*\| < \delta/2 \quad \text{for all } k \in \mathcal{K}, \tag{11}$$

where all members of $\mathcal{K}$ are sufficiently large to ensure (11) holds. Then,

$$\|z_k - z\| \leq \|z_k - z_*\| + \|z_* - z\| < \delta$$

for all $z \in \mathfrak{E}$. For all $k$ sufficiently large, CARTopt samples near-uniformly in the set

$$\left\{x \in \mathbb{R}^n : \|z_k - x\| < \delta\right\} \tag{12}$$

with probability density greater than or equal to $(|\omega_L| \cdot m(\mathcal{L}_k))^{-1}$. Therefore, CARTopt samples in $\mathfrak{E}$ with probability

$$\Pr(x \in \mathfrak{E}) \geq \frac{m(\mathfrak{E})}{|\omega_L| \cdot m(\mathcal{L}_k)} > 0$$

for all sufficiently large $k$. Hence, in the limit as $k \to \infty$, $f(z_k) < f(z_*)$ almost surely, contradicting Assumption 13(c). Thus, $z_*$ must be an essential local minimizer of $f$, almost surely. $\square$

## 8 The stopping rule

Numerical results were obtained using the stopping rule from [21]. This stopping rule is an adaptation of the rules of Dorea [7] and Hart [9] for global optimization by random sampling. These rules can not be applied directly as they require sampling from a fixed probability distribution. These rules rely on Dorea's [7] observation that the measure of the level set

$$\mathcal{E}\left(x^*, \delta, \epsilon\right) = \left\{x \in \mathbb{R}^n : f(x) < f\left(x^*\right) + \epsilon \text{ and } \|x - x^*\| < \delta\right\}$$

is almost always a power law function of $\epsilon$ (for small $\epsilon$) when $x^*$ is a local minimizer and $\delta$ is fixed. This observation can still be exploited, and we do so by estimating the power law fit around a prospective local minimizer $x$ using known function values at sample points near $x$. If the power law accurately fits these known points, and it indicates it is unlikely that points near $x$ which are significantly better than $x$ exist, then the algorithm accepts $x$ as an acceptable approximation to a local minimizer, and the method halts.

The relevance of Dorea's observation rests on the fact that CARTopt samples near uniformly in a neighborhood of its best known point. Without this property the logic behind our new stopping rule is invalid. Hence, for example, it can not be applied to methods such as MADS, which sample on a sequence of grids. The situation is symmetric: MADS' stopping rule is based on a minimum grid size, which can not be applied to CARTopt as there is no equivalent of a grid. In any case a stopping rule is an integral part of an algorithm, and may (as ours does) crucially rely on properties particular to that algorithm. This is especially true of global optimization in the absence of global information such as a Lipschitz constant, or knowledge of the global minimum [24]. For example estimating the global minimum of an arbitrary continuous function on the interval [0, 1] using sample points will always eventually succeed, but deciding when enough sample points have been used is impossible because arbitrarily narrow, deep notches might exist in parts of [0, 1] not yet sampled. Hence the stopping rule is not only the hardest part of such a global optimization problem, it can not be made totally reliable. It has been shown [16] that finding a descent direction in non-smooth local optimization is tightly linked to global optimization, with attendant difficulties.

Our stopping rules uses the set $Y$ containing the $\Gamma$ least function values in ascending order

$$Y = \{f_1, \dots, f_\Gamma\}.$$

Following Dorea [7] we assume that the Cumulative Distribution Function (CDF) of the function value $f_0$ at a random point in the level set $\{x \in \mathbb{R}^n : f(x) < f_\Gamma\}$ is

$$F(f,\kappa) = \Pr(f_0 \le f) = \left(\frac{f_0 - f_*}{f_\Gamma - f_*}\right)^\kappa \tag{13}$$

where $f_*$ is the essential local minimum. In fact this assumption is somewhat more general than Dorea's [21], and usually applies even when $f$ is discontinuous at the local minimizer $x^*$. The values for $\kappa$ and $f_*$ must be estimated. The estimate $\hat{f}_*$ of $f_*$ was chosen from three possible values. These are $f_1 - \mathcal{R}$, $f_1 - \mathcal{R}/2$, and $f_1 - \mathcal{R}/4$, where $\mathcal{R}$ is the range of $Y$ given by $\mathcal{R} = \max(f_\Gamma - f_1, \epsilon_o/2)$. The positive constant $\epsilon_o$ ensures $\mathcal{R}$ is bounded away from zero. This choice of values for $\hat{f}_*$ appears restrictive, but worked well in practice [21].

For each possible value of $\hat{f}_*$, the value of $\kappa \in [n/2, 2n]$ is chosen which minimizes the supremum norm distance between the CDF in (13) and the Empricial Distribution Function (EDF) of the actual data $Y$, where the EDF is

$$\hat{F}(f) = \frac{1}{\Gamma}\sum_{i=1}^{\Gamma} \mathcal{U}(f - f_i) \tag{14}$$

and the unit step function $\mathcal{U}(t)$ is one if $t \geq 0$, and zero otherwise. The pair of values $\hat{f}_*$, $\hat{\kappa}$ for which the supremum norm

$$\sup_{f \in [0, f_\Gamma]} \left| \hat{F}(f) - F(f) \right|$$

is least are used in (13) to give the fitted CDF.

The range of $\kappa$ values considered is fairly tight. The lower bound corresponds to the case when $f$ grows quadratically as one moves away from a local minimizer, whereas the upper bound corresponds to the case when $f$ grows to the $1/2$ power. A linear growth rate gives $\kappa = n$. The range of $\kappa$ values $[n/2, 2n]$ gave good results, but if a wider range is needed, it is simple to alter it.

The goodness of fit is measured using the Kolmogorov-Smirnov test for continuous data with the null hypothesis that the two distributions are the same. The null hypothesis is rejected at the 0.05 level of significance, in which case the stopping rule does not halt the algorithm. If the null hypothesis is not rejected, the probability that a random sample point in the level set $\{x \in \mathbb{R}^n : f(x) < f_\Gamma\}$ will have a function value less than $\hat{f}_* - \epsilon_o$ is calculated. If this is less than $\beta$ the stopping rule halts the algorithm. Numerical results were generated using $\epsilon_o = 10^{-8}$ and $\beta = 10^{-6}$.

## 9 Numerical results and discussion

CARTopt was tested on two sets of problems. The first set was chosen from Hock et al. [10], Moré et al. [14] and Schittkowski [22]. The problems selected were expressed as a sum of squares $\sum_i f_i^2$ with a global minimum of zero. These problems are easily made nonsmooth by replacing the sum of squares with absolute values $\sum_i |f_i|$. The nonsmooth versions share the same global minimizer(s) as their smooth counterparts because $f_i = 0$ for all $i$ at the solution. However, these modified functions can make the results seem deceptively poor. A final function value of 1e$-$5 on the nonsmooth function, for example, corresponds to a function value of approximately 1e$-$10 on the original problem. Here we consider anything less than 1e$-$4 an acceptable approximate solution to a particular problem.

The second set of problems was taken from Lukšan and Vlček [12], Brachetti et al. [3], and Vicente and Custódio [25]. The problems from [12] and [25] are nonsmooth and the two problems from [3] were made nonsmooth by the first author. The reader is referred to the Appendix for their exact formulation.

Numerical results comparing CARTopt to Pure Random Search (PRS), Controlled Random Search (CRS) [3], and two new versions of CRS [3] are presented in Table 1. The CRS algorithms use a set $S$ of $35n$ randomly generated points over the search region. At each iteration $n + 1$ points are randomly selected from $S$ and one of these points is randomly selected and reflected through the centroid of the remaining $n$ points. If the reflected point has a lower $f$ value than $\max\{f(x) : x \in S\}$, it is retained and the point with the greatest $f$ value is removed from $S$ and the method repeats. The method terminates when $|\max f(x) - \min f(x)| < 1\text{e}{-6}$, where $x \in S$. One new version of the algorithm uses a weighted centroid, weighted reflection, and forms a quadratic approximation using the $2n + 1$ points with the least $f$ values under

**Table 1** Comparison of CARTopt with Controlled Random Search (CRS) and PRS using a maximum of 50000 $f$ evaluations. The first two columns list the problem name and dimension. Columns headed with $|f - f_*|$ list the absolute error in the final $f$ values and the 'nf' columns list the number of function evaluations, averaged over ten runs. The $f$ value in *bold* shows the lowest value obtained

| Problem | $n$ | PRS (50k) | CRS | | New CRS | | CRS no Quad | | CARTopt | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|f - f_*\|$ | $\|f - f_*\|$ | nf | $\|f - f_*\|$ | nf | $\|f - f_*\|$ | nf | $\|f - f_*\|$ | nf |
| Beale | 2 | 0.01 | 2e−7 | 2829 | 2e−7 | 1593 | 1e−7 | 1589 | **4e−9** | 986 |
| CB2 | 2 | 3e−3 | **8e−11** | 2456 | 8e−10 | 1527 | 4e−8 | 1414 | 5e−9 | 782 |
| CB3 | 2 | 6e−3 | 2e−8 | 3568 | 1e−8 | 1763 | 9e−8 | 1600 | **3e−9** | 1067 |
| Cosine Mix. | 4 | 0.56 | 0.25 | 50000 | 1e−6 | 13749 | 2e−7 | 13752 | **2e−8** | 3763 |
| | 6 | 1.45 | 0.72 | 50000 | 1e−3 | 25259 | 3e−7 | 28759 | **2e−8** | 7120 |
| Crescent | 2 | 1e−3 | 7e−9 | 2737 | 4e−9 | 1514 | 6e−8 | 1513 | **2e−9** | 809 |
| Exponential | 6 | 0.17 | 7e−8 | 24580 | 3e−8 | 29571 | 3e−7 | 9572 | **2e−8** | 4495 |
| | 8 | 0.38 | 1e−7 | 41471 | 6e−4 | 50000 | 5e−7 | 11716 | **3e−8** | 6678 |
| Ext. Rosenbrock | 4 | 1.94 | 3e−8 | 14949 | 2e−8 | 10493 | 2e−7 | 9979 | **1e−8** | 4197 |
| Gulf | 3 | 2.69 | 3e−7 | 7013 | **1e−7** | 4508 | 2e−7 | 4465 | 1e−6 | 17252 |
| Helical Valley | 3 | 0.28 | 2e−7 | 6908 | 1e−7 | 4360 | 2e−7 | 4441 | **7e−9** | 1722 |
| LQ | 2 | 3e−4 | 5e−8 | 2464 | 4e−8 | 1453 | 6e−8 | 1321 | **4e−8** | 777 |
| Mifflin 1 | 2 | 4e−4 | 1e−8 | 2914 | 3e−9 | 1641 | 3e−8 | 1404 | **2e−9** | 1233 |
| Mifflin 2 | 2 | 9e−4 | 1e−8 | 2641 | 4e−9 | 1524 | 6e−8 | 1024 | **2e−9** | 900 |
| Powell | 4 | 1.78 | 4e−7 | 11723 | 3e−7 | 8938 | 1e−7 | 6561 | **1e−8** | 2329 |
| QL | 2 | 9e−3 | 8e−9 | 2677 | 4e−9 | 1489 | 5e−8 | 1306 | **2e−9** | 875 |
| Rosenbrock | 2 | 0.03 | 2e−7 | 3207 | 1e−7 | 1902 | 1e−7 | 1962 | **3e−9** | 1102 |
| Trigonometric | 5 | 0.47 | 5e−7 | 21840 | 3e−7 | 18924 | 3e−7 | 19009 | **2e−8** | 3945 |
| Variably Dim. | 4 | 0.21 | 4e−8 | 11848 | 2e−8 | 6248 | 3e−7 | 5739 | **1e−8** | 3053 |
| | 8 | 1.33 | 8e−7 | 37016 | 4e−7 | 12950 | 5e−7 | 13056 | **4e−8** | 11508 |
| Wolfe | 2 | 0.03 | 7e−9 | 2920 | 6e−9 | 1584 | 4e−8 | 1644 | **1e−9** | 965 |
| 240 [22] | 3 | 2.32 | 2e−7 | 7777 | 1e−7 | 4577 | 1e−7 | 4616 | **1e−8** | 1800 |
| 261 [22] | 4 | 0.31 | 4e−7 | 10643 | 2e−7 | 6885 | 3e−7 | 7223 | **9e−9** | 3483 |
| 291 [22] | 10 | 1.55 | 3e−7 | 27233 | **1e−27** | 12157 | 9e−8 | 24722 | 9e−9 | 5520 |

**Table 2** Comparison of CARTopt with MADS and Algorithm 2.1 from [25] on three lower semicontinuous functions from [25]. All results are 10 run averages

| Problem | $n$ | MADS | | Suff. Decrease | | CARTopt | |
|---|---|---|---|---|---|---|---|
| | | failures | nf | failures | nf | failures | nf |
| $f_1$ | 2 | 0 | 233 | 0 | 175.6 | 0 | 764 |
| $f_2$ | 2 | 0 | 193.9 | 0 | 494.6 | 0 | 476 |
| $f_4$ | 2 | 2 | 220.6 | 1 | 177.7 | 0 | 403 |

certain conditions [3]. The second new version is obtained from the first by omitting the quadratic approximation. Although the CRS algorithms are global optimization algorithms, they are direct search methods which can be applied to the problems considered here if a bounded search region is specified. However, the CRS algorithms have no formal convergence theory for the types of problems considered.

An initial hypercube search region of the form

$$(x_0 + x_*)/2 + h[-1, 1]^n, \tag{15}$$

was used, where $x_0$ and $x_*$ are the standard starting point and a known solution for each problem respectively. The radius $h$ is chosen sufficiently large so that both $x_0$ and $x_*$ are interior points of the initial search region. This allows us to compare CARTopt with the other methods applied in an unbiased search region. The interested reader is referred to [21, Appendix A.1] for exact values.

PRS marginally solved three of the test problems using 50000 function evaluations. Unsurprisingly both CARTopt and the CRS algorithms were superior to PRS, requiring fewer function evaluations to produce far more accurate approximations to the solutions of each problem. The new versions of CRS required fewer function evaluations than the original CRS on most of the problems to obtain similar or more accurate approximate solutions. The first author found that the weighted centroid and weighted reflection reduced the total function evaluations, whereas the quadratic model had little effect on these problems. These observations are similar to those reported in [3].

CARTopt solved all the problems considered to the desired standard and required fewer function evaluations to obtain similar or better approximate solutions on all but one of the problems. The CRS algorithms required fewer function evaluations to obtain a more accurate approximate solution to the Gulf problem. For all other problems CARTopt took approximately half (or fewer) function evaluations to solve each problem to the desired standard.

The CARTopt algorithm was also tested on the four lower semicontinuous functions from [25], presented in Table 2. Functions $f_1$, $f_3$, and $f_4$ are discontinuous at the unique solution $x_* = (0, 0)$ and $f_2$ is continuous at $x_* = (0, 0)$ but not Lipschitz continuous near this point [25]. The authors in [25] used the Mesh Adaptive Direct Search (MADS) algorithm [2] to solve the problems and reported the number of failures and the average number of function evaluations. CARTopt solves problems $f_1$, $f_2$, and $f_4$ with no failures but required more function evaluations to do so. Problem $f_3$ is not reported here because CARTopt finds an essential local minimizer and hence solves the problem by our definition. In [25] the classical definition of a minimizer is used and they report ten failures from ten runs on this problem. Reporting

**Table 3** Comparison of CARTopt with two other direct search methods for nonsmooth unconstrained optimization created by the authors. The results for CARTopt are averages over 10 runs. The $f$ value in *bold* shows the lowest value obtained

| Problem | $n$ | Results from [16] | | Results from [17] | | CARTopt | |
|---|---|---|---|---|---|---|---|
| | | $\|f - f_*\|$ | nf | $\|f - f_*\|$ | nf | $\|f - f_*\|$ | nf |
| Beale | 2 | 4e−8 | 3638 | 2e−8 | 1119 | **1e−9** | 1083 |
| Brown 2D | 2 | 2e−3 | 10598 | **4e−4** | 950 | 2e−3 | 50000 |
| Gulf | 3 | 1e−5 | 15583 | 6e−6 | 31306 | **5e−6** | 16405 |
| Helical Valley | 3 | 7e−8 | 8406 | **1e−9** | 2773 | 5e−9 | 1891 |
| Powell | 4 | 4e−7 | 11074 | 3e−3 | 3659 | **7e−9** | 2756 |
| Rosenbrock | 2 | 5e−8 | 4438 | 2e−8 | 1154 | **3e−9** | 1184 |
| Trigonometric | 5 | 5e−8 | 14209 | 4e−8 | 6678 | **2e−8** | 4105 |
| Variably Dim. | 8 | 2e−7 | 34679 | 5e−7 | 55647 | **4e−8** | 16182 |
| Woods | 4 | **3e−7** | 15610 | 5e−4 | 4682 | 0.02 | 3852 |

that CARTopt solved all ten problems would be misleading. The authors in [25] also include results using different starting positions. However in the CARTopt approach, an initial batch of $2N$ random points is used and changing a single start point does not change our results.

CARTopt was also compared with two other direct search algorithms created by the authors, presented in Table 3. An initial hypercube search region of the form

$$x_0 + 2[-1, 1]^n, \tag{16}$$

was used, where $x_0$ is the standard starting point for each problem. This gives CARTopt the same starting point as the algorithms in [16, 17]. The CARTopt algorithm was superior to the algorithm from [16], using fewer function evaluations to produce more accurate approximate solutions on all problems except on the Brown, Gulf, and Woods problems. On the Gulf problem CARTopt obtained a more accurate approximate solution, but required more function evaluations to do so. On the Woods function one of CARTopt's 10 runs halted early, leading to a poor average function value. For the other six problems approximately one third of the function evaluations were required. CARTopt matched or outperformed the algorithm in [17] on all test functions except the Brown and Woods functions.

CARTopt (using (16) as the initial search region) was compared with the MADS algorithm for unconstrained nonsmooth minimization. Each MADS implementation was obtained using the `optimtool` function of MATLAB's optimization toolbox, with the patternsearch and MADS options selected. An initial mesh size of $\exp(1)/2$ was chosen to ensure the solution of a particular problem was not on the initial grid, preventing misleading results. A terminating mesh size of 1e−10 was used. Smaller values were also tested but this tended to increase the number of function evaluations and had little effect on accuracy. The MADS algorithm has a 'search step', where the user can choose to implement another optimization method. This step is not needed for theoretical convergence and is included to potentially increase the rate of convergence. Here we considered no search method, the Nelder–Mead method [5, 15], and

**Table 4** Comparison of CARTopt with MADS using three different search methods. All CARTopt results are 10 run averages

| Problem | $n$ | No search | | Nelder–Mead | | latin hypercube | | CARTopt | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\|f-f_*\|$ | nf | $\|f-f_*\|$ | nf | $\|f-f_*\|$ | nf | $\|f-f_*\|$ | nf |
| Beale | 2 | 0.07 | 640 | 1e−5 | 4150 | 1e−7 | 2984 | **1e−9** | 1083 |
| CB2 | 2 | 4e−3 | 358 | 5e−9 | 3718 | 2e−3 | 1768 | **4e−9** | 833 |
| CB3 | 2 | 6e−3 | 541 | 1e−6 | 4342 | 1e−8 | 1869 | **3e−9** | 1086 |
| Cosine Mix. | 4 | 2e−10 | 1550 | **2e−15** | 25278 | 8e−3 | 50000 | 2e−8 | 3496 |
| | 6 | 6e−10 | 3397 | **1e−13** | 45157 | 6e−3 | 50000 | 2e−8 | 6731 |
| Crescent | 2 | 0.02 | 620 | 1e−8 | 566 | 6e−4 | 1800 | **1e−9** | 828 |
| Exponential | 6 | 9e−11 | 2732 | 1e−10 | 3228 | **1e−16** | 6170 | 2e−8 | 4595 |
| | 8 | 9e−11 | 4437 | 2e−10 | 6907 | **3e−16** | 9271 | 2e−8 | 6998 |
| ext. Rosenb'k | 4 | 2.55 | 1260 | 3e−5 | 14307 | 1.79 | 3953 | **1e−8** | 3679 |
| Gulf | 3 | 4e−3 | 50000 | **2e−7** | 16900 | 0.77 | 50000 | 5e−6 | 16405 |
| Helical Valley | 3 | 0.36 | 657 | 7e−6 | 4303 | 0.58 | 2684 | **5e−9** | 1891 |
| LQ | 2 | 1e−3 | 400 | 4e−8 | 3384 | 1e−4 | 1872 | **4e−8** | 788 |
| Mifflin 1 | 2 | 0.19 | 299 | 3e−7 | 2040 | 6e−3 | 50000 | **4e−9** | 1268 |
| Mifflin 2 | 2 | **2e−11** | 625 | 2e−11 | 2409 | 1e−10 | 14164 | 2e−9 | 924 |
| Powell | 4 | 0.95 | 1373 | 2e−2 | 5265 | 0.38 | 4154 | **7e−9** | 2756 |
| QL | 2 | 0.85 | 484 | **7e−10** | 3653 | 0.09 | 1674 | 2e−9 | 897 |
| Rosenbrock | 2 | 1.49 | 395 | 1e−5 | 3508 | 2e−1 | 1932 | **3e−9** | 1184 |
| Trigonometric | 5 | 3e−10 | 2069 | 3e−10 | 17620 | **1e−15** | 5110 | 2e−8 | 4105 |
| Variably Dim | 4 | 0.98 | 1782 | 1e−5 | 21590 | 1.04 | 50000 | **1e−8** | 3067 |
| | 8 | 3.01 | 7400 | 4e−5 | 50000 | 1.22 | 12356 | **4e−8** | 16182 |
| Wolfe | 2 | 4e−10 | 536 | 1e−9 | 1818 | **1e−11** | 2609 | 1e−9 | 963 |
| 240 | 3 | 22.2 | 6205 | 6e−6 | 2516 | 0.02 | 50000 | **6e−9** | 1943 |
| 261 | 4 | 0.58 | 4441 | 2e−6 | 18022 | 2e−5 | 50000 | **5e−8** | 3960 |
| 291 | 10 | 8e−20 | 7245 | 6e−20 | 8856 | **5e−39** | 12790 | 1e−8 | 5368 |

Latin Hypercube design [11]. These methods were chosen because they require no gradient information and can be applied to nonsmooth problems. Results are present in Table 4.

The MADS instance with no search step solved one third of the problems to the desired standard. The Nelder–Mead and Latin Hypercube search step methods dramatically increased the number of function evaluations on most of the problems. This additional computational effort did not significantly increase the performance of MADS using the Latin Hypercube search method, still failing to solve half the problems to the desired standard within 50000 function evaluations. However, the Nelder–Mead search step greatly increased the performance of MADS, only failing to reach the desired accuracy on one problem (Powell $n = 4$).

Comparing CARTopt to the best of the MADS algorithms (the one using the Nelder Mead search step) we see that CARTopt solved all the problems to the desired standard, requiring fewer function evaluations than MADS on 21 of the 24 problems considered. Similar function evaluations were required on the remaining three

**Table 5** Comparison of CARTopt with three versions of MADS. All values are averaged over ten runs. The $f$ value in *bold* shows the lowest value obtained. The nine discontinuous test functions are listed in the Appendix

| Problem | $n$ | No search | | Nelder–Mead | | latin hypercube | | CARTopt | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\|f-f_*\|$ | nf | $\|f-f_*\|$ | nf | $\|f-f_*\|$ | nf | $\|f-f_*\|$ | nf |
| $B1$ | 2 | 0.85 | 553 | 1e−6 | 5994 | 0.04 | 3553 | **3e−9** | 1291 |
| $B2$ | 2 | 1.65 | 906 | 1e−6 | 6606 | 0.89 | 2932 | **2e−9** | 1396 |
| $B3$ | 2 | 1.48 | 624 | 6e−5 | 4488 | 1.17 | 2928 | **4e−9** | 1641 |
| Cosine Mix. | 4 | 2e−10 | 1550 | **2e−15** | 25278 | 8e−3 | 50000 | 2e−8 | 3496 |
| Cosine Mix. | 6 | 6e−10 | 3397 | **1e−13** | 45157 | 6e−3 | 50000 | 2e−8 | 6731 |
| $R1$ | 2 | 5.20 | 392 | 3e−5 | 2929 | 0.78 | 1705 | **4e−9** | 1489 |
| $R2$ | 2 | 1.12 | 386 | 1e−5 | 3609 | 0.98 | 2287 | **4e−9** | 1473 |
| $R3$ | 2 | 4.94 | 375 | **3e−9** | 8013 | 2.17 | 2332 | 5e−9 | 2045 |
| $R4$ | 2 | 3.42 | 366 | 3e−6 | 4522 | 0.92 | 2975 | **2e−9** | 1398 |

problems. CARTopt obtained the most accurate approximate solutions on most of the problems considered. The first author tried adjusting the termination parameters of MADS to increase its final estimates. However, no significant change was observed and MADS became increasingly computationally expensive.

Results on discontinuous variants of three problems are presented in Table 5 for the three versions of MADS and also for CARTopt. Two of the three MADS versions (no search and Latin Hypercube) failed on seven and all nine problems respectively. The Nelder Mead version of MADS and CARTopt both solved all nine problems, but CARTopt was by far the faster of the two. Interestingly MADS without a search step outperformed CARTopt on the two cosine mixture problems. These results show that CARTopt was the most effective of the four methods on discontinuous problems. Overall CARTopt outperformed the other methods on a range of smooth, non-smooth, and discontinuous problems, showing it is clearly an effective and robust method for non-smooth optimization.

## 10 Conclusion

A random search algorithm for unconstrained local nonsmooth optimization has been presented. The method forms a partition on $\mathbb{R}^n$ using CART to identify subsets where $f$ is relatively low. Due to the hyperrectangular structure of the partition, the low subsets can be sampled directly. Alternating between partition and sampling phases provides an effective method for nonsmooth optimization. Convergence to an essential local minimizer of $f$ with probability one has been demonstrated under mild conditions. It is possible to apply this convergence theory to a general class of algorithms that ultimately exhaustively search for points of descent in a neighborhood $\Omega$ of cluster points, provided $m(\Omega)$ is bounded above zero and $z_* \in \Omega$. Numerical results have been presented which verify that theoretical convergence is achieved in practice. Comparison with other direct search methods shows that the CARTopt algorithm is competitive in practice.

**Acknowledgements** We acknowledge the helpful comments of two anonymous referees which led to an improved version of the paper.

## Appendix

A nonsmooth version of the Cosine Mixture problem [3] is

$$f(x)=\begin{cases}0.1\sum_{i=1}^{n}\cos(5\pi x_i)-\sum_{i=1}^{n}|x_i|, & \text{if } \|x\|_\infty \le 1\\ \infty & \text{otherwise,}\end{cases}$$

where $x_0=(0,0,\dots,0)$. The cases $n=4$ and $n=6$ were studied, where $f_*=-4.4$ and $f_*=-6.6$ respectively.

A nonsmooth version of the Exponential problem [3] is

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

where $x_0=(1,1,\dots,1)$ and $f_*=-1$. The cases $n=6$ and $n=8$ were studied.

The discontinuous versions of the Rosenbrock function are defined as follows.

$$f_{R1}(x)=\begin{cases}10|x_2-x_1^2|+|x_1-1| & \text{if } x_1\ge 1,\\ 10|x_2-x_1^2|+|x_1-1|+4 & \text{otherwise.}\end{cases}$$

$$f_{R2}(x)=\begin{cases}10|x_2-x_1^2|+|x_1-1|+4 & \text{if } x_1> 1,\\ 10|x_2-x_1^2|+|x_1-1| & \text{otherwise.}\end{cases}$$

$$f_{R3}(x)=\begin{cases}10|x_2-x_1^2|+|x_1-1|+4 & \text{if } x_1< 1,\\ 10|x_2-x_1^2|+|x_1-1|+2 & \text{if } x_1\ge 1 \text{ and } x_2>1,\\ 10|x_2-x_1^2|+|x_1-1| & \text{otherwise.}\end{cases}$$

$$f_{R4}(x)=\begin{cases}10|x_2-x_1^2|+|x_1-1| & \text{if } x_1\le 1 \text{ and } x_2\le x_1,\\ 10|x_2-x_1^2|+|x_1-1|+4 & \text{otherwise.}\end{cases}$$

Each problem uses $x_0=(-1.2,1)$ and has an essential local minimizer at $x_*=(1,1)$ with $f_*=0$.

The discontinuous versions of the Beale function are defined as follows. Let

$$f_1(x)=1.5-x_1(1-x_2);$$

$$f_2(x)=2.25-x_1\bigl(1-x_2^2\bigr);$$

$$f_3(x)=2.625-x_1\bigl(1-x_2^3\bigr).$$

$$f_{B1}(x) = \begin{cases} \sum_i |f_i(x)| & \text{if } x_1 \geq 3 \text{ and } x_2 \geq 0.5, \\ \sum_i |f_i(x)| + 2 & \text{otherwise.} \end{cases}$$

$$f_{B2}(x) = \begin{cases} \sum_i |f_i(x)| & \text{if } x_2 \geq 0.5 \text{ and } x_2 - 0.5x_1 \leq -1, \\ \sum_i |f_i(x)| + 2 & \text{otherwise.} \end{cases}$$

$$f_{B3}(x) = \begin{cases} \sum_i |f_i(x)| & \text{if } x_2 - 0.25x_1 \geq -0.25 \text{ and } x_2 - 0.5x_1 \leq -1, \\ \sum_i |f_i(x)| + 2 & \text{otherwise.} \end{cases}$$

Each problem uses $x_0 = (1, 1)$ and has an essential local minimizer at $x_* = (3, 0.5)$ with $f_* = 0$.

## References

1. Appel, M.J., Labarre, R., Radulovic, D.: On accelerated random search. SIAM J. Optim. **14**, 708–731 (2003)
2. Audet, C., Dennis, J.E.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**, 188–217 (2006)
3. Brachetti, P., Ciccoli, M.D.F., Di Pillo, G., Lucidi, S.: A new version of the Price's algorithm for global optimization. J. Glob. Optim. **10**, 165–184 (1997)
4. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth, Belmont (1984)
5. Burmen, Á., Puhan, J., Tuma, T.: Grid restrained Nelder Mead algorithm. Comput. Optim. Appl. **34**, 359–375 (2006)
6. Clarke, F.H.: Optimization and Nonsmooth Analysis. Classics in Applied Mathematics. SIAM, Philadelpha (1990)
7. Dorea, C.C.Y.: Stopping rules for a random optimization method. SIAM J. Control Optim. **28**, 841–850 (1990)
8. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley-Interscience, New York (2001)
9. Hart, W.E.: Sequential stopping rules for random optimization methods with applications to multistart local search. SIAM J. Optim. **9**, 270–290 (1998)
10. Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems, vol. 187. Springer, Berlin (1981)
11. Loh, W.: On latin hypercube sampling. Ann. Stat. **24**, 2058–2080 (1996)
12. Lukšan, L., Vlček, J.: Test problems for unconstrained optimization. Technical Report no. 8. Academy of Sciences of the Czech Republic, Institute of Computer Science (2003)
13. Martinez, W.L., Martinez, A.R.: Computational Statistics Handbook with MATLAB. Chapman & Hall/CRC Press, London/Boca Raton (2002)
14. Moré, J.J., Garbow, B.S., Hillstrom, K.E.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**, 17–41 (1981)
15. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
16. Price, C.J., Reale, M., Robertson, B.L.: A direct search method for smooth and nonsmooth unconstrained optimization. ANZIAM J. **48**, C927–C948 (2006)
17. Price, C.J., Robertson, B.L., Reale, M.: A hybrid Hooke and Jeeves—Direct method for nonsmooth optimization. Adv. Model. Optim. **11**, 43–61 (2009)
18. Price, C.J., Reale, M., Robertson, B.L.: A cover partitioning method for bound constrained global optimization. Optim. Methods Softw. **27**, 1059–1072 (2012). doi:10.1080/10556788.2011.557726
19. Rinnooy Kan, A.H.G., Timmer, G.T.: Stochastic global optimization methods. Part I. Clustering methods. Math. Program. **39**, 27–56 (1987)
20. Robertson, B.L., Price, C.J., Reale, M.: Nonsmooth optimization using classification and regression trees. In: Proceedings of the 18th IMACS World Congress and MODSIM09 International Congress on Modelling and Simulation, Cairns, Australia, July 13–17, 2009, pp. 1195–1201 (2009)

21. Robertson, B.L.: Direct search methods for nonsmooth problems using global optimization techniques. Ph.D. Thesis, University of Canterbury, Christchurch, New Zealand (2010)
22. Schittkowski, K.: More Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems, vol. 282. Springer, Berlin (1987)
23. Tang, Z.B.: Adaptive partitioned random search to global optimization. IEEE Trans. Autom. Control **39**, 2235–2244 (1994)
24. Torn, A., Žilinskas, A.: Global Optimization. Lecture Notes in Computer Science, vol. 350. Springer, Berlin (1989)
25. Vicente, L.N., Custódio, A.L.: Analysis of direct searches for discontinuous functions. Math. Program. **133**, 299–325 (2012)