ANZIAM J. 55(2013), 109–128
doi:10.1017/S1446181113000412

A CARTOPT METHOD FOR BOUND-CONSTRAINED GLOBAL OPTIMIZATION

B. L. ROBERTSON ✉[1], C. J. PRICE[2] and M. REALE[2]

(Received 5 March, 2012; revised 8 August, 2013)

Abstract

A stochastic algorithm for bound-constrained global optimization is described. The method can be applied to objective functions that are nonsmooth or even discontinuous. The algorithm forms a partition on the search region using classification and regression trees (CART), which defines a region where the objective function is relatively low. Further points are drawn directly from the low region before a new partition is formed. Alternating between partition and sampling phases provides an effective method for nonsmooth global optimization. The sequence of iterates generated by the algorithm is shown to converge to an essential global minimizer with probability one under mild conditions. Nonprobabilistic results are also given when random sampling is replaced with points taken from the Halton sequence. Numerical results are presented for both smooth and nonsmooth problems and show that the method is effective and competitive in practice.

2010 *Mathematics subject classification*: 90C26.

Keywords and phrases: CART, Halton sequence, numerical results, random search, stochastic global optimization.

1. Introduction

The bound-constrained global optimization problem is of the form

$$\min f(x) \quad \text{subject to } x \in \Omega, \tag{1.1}$$

where the search region Ω is defined by an n-dimensional box of the form

$$\Omega = \{x \in \mathbb{R}^n : l_i \leq x_i \leq u_i \text{ for all } i = 1, \ldots, n\}.$$

The objective function f maps Ω into $\mathbb{R} \cup \{+\infty\}$ and is assumed to be lower semicontinuous. The inclusion of $\{+\infty\}$ means that the objective can be assigned the

[1]Department of Statistics, University of Wyoming, Laramie, Wyoming, USA;
e-mail: brober25@uwyo.edu.
[2]Department of Mathematics and Statistics, University of Canterbury, Private Bag 4800, Christchurch, New Zealand; e-mail: chrisj.price@canterbury.ac.nz, marco.reale@canterbury.ac.nz.

value $+\infty$ at points or regions where it cannot be evaluated. Problems with general constraints can be handled using the extreme barrier approach [4], which defines $f(x) = +\infty$ at infeasible points. Under appropriate scaling, Ω can be simplified to a box of the form $\Omega = [-1, 1]^n$, and without loss of generality this is used hereafter.

Designing a global optimization algorithm to find a global minimum is usually very difficult because there is often no way to tell if a local minimum is indeed a global minimum. Nevertheless, an assortment of deterministic and random search methods have been proposed to solve a variety of global optimization problems. Random search methods have an element of randomness or probability in their design and deterministic methods do not. In the literature, random search methods may also be called Monte Carlo methods or stochastic algorithms.

Deterministic methods (such as branch and bound, interval analysis, and tunnelling methods [12, 13]) typically guarantee asymptotic convergence to the global minimum. Random search methods (such as simulated annealing, genetic algorithms, multi-start, and clustering algorithms [24, 26]) ensure convergence in probability. The trade-off between deterministic and random search methods is in terms of computational effort and the type of convergence [24, 25]. Random search methods are typically fast, but only offer convergence in terms of probability.

Random search algorithms are popular amongst practitioners for several reasons. First, they can provide a relatively good solution quickly and easily [25]. Second, they are relatively easy to implement (and program) on complex problems with "black box" function evaluations because they typically rely on function evaluations, rather than gradient or Hessian information [25]. This also makes random search methods useful for ill-conditioned global optimization problems where the objective function may be nonconvex, nonsmooth, and possibly discontinuous over the search region. Third, random search algorithms have been shown to be effective in solving some large-scale optimization problems for which deterministic methods struggle [24]. For example, Dyer and Frieze [7, 8] showed that estimating the volume of a convex body takes an exponential number of function evaluations for *any* deterministic algorithm, but if a random search algorithm is used then the volume can be estimated in polynomial time with a high probability of being correct [24].

Arguably the simplest method for solving (1.1) is the pure random search (PRS) algorithm. PRS evaluates f at a number of randomly generated points over Ω and uses the best function value as an estimate of the global minimum. However, PRS often performs poorly in practice and many authors have developed algorithms to improve its numerical performance. There is an extensive literature on increasing the performance of PRS [26]. The remainder of this section reviews several random search methods that are used for comparison in Section 8.

Controlled random search (CRS) [18] uses a set S of $35n$ randomly generated points over Ω. At each iteration, $n + 1$ points are randomly selected from S and one of these points is reflected through the centroid of the remaining n points. If the reflected point has f value lower than $\max f(x) : x \in S$, it is retained and the point with the greatest f value is removed from S; otherwise the reflected point is ignored. In any case, the method repeats until stopping conditions are satisfied. A new version of CRS uses a weighted centroid and weighted reflection, and forms a quadratic approximation using the $2n + 1$ points with the best f values under certain conditions [5]. Although effective in practice, both CRS algorithms have no formal convergence theory.

Accelerated random search (ARS) [3] randomly samples a finite sequence of contracting subregions (initially Ω) centred on the best iterate. If a point with a lower f value is found, or the sequence is exhausted, the search returns to Ω. This counterintuitive approach allows the algorithm to focus its search in the neighbourhood of its best point and can produce high-accuracy approximate solutions.

Another technique that is used to increase the efficiency of PRS is to partition Ω into a collection of subregions. The sample intensity can then be varied over Ω by selecting more points from subregions of the partition where f is presumed or known to be low (see, for example, stratified random search [9]).

An adaptive partitioning strategy can also be employed so that the partition is formed iteratively. The interested reader is referred to Zabinsky's book [24] for a full review on adaptive methods. Tilecutter [17] is in the spirit of the deterministic algorithm DIRECT [14] and forms a sequence of nested subregions called tiles. At each iteration, certain tiles are cut into smaller tiles and points are randomly drawn from the new tiles. The method continues cutting and sampling tiles until stopping conditions are satisfied.

In this paper, a random search method which produces a sequence of non-nested partitions is proposed. Each partition is used to iteratively update the sample intensity over Ω so that more points are selected from where f is presumed to be low. We begin by introducing the partitioning strategy that is used in our algorithm, called global CARTopt. The algorithm is described in Section 3, and particular steps are described in detail in Sections 4 and 5. A parallel implementation of the global CARTopt algorithm is given in Section 6. A global convergence proof for the algorithm is given in Section 7. Numerical results on a selection of bound-constrained global optimization problems are presented in Section 8, and Section 9 concludes the paper.

2. The partitions

To illustrate the advantages of forming a partition in a random search algorithm, consider minimizing an objective function f over Ω using the following two approaches:

(i) applying PRS on Ω using N points; and
(ii) partitioning Ω into N subregions A_i of equal positive Lebesgue measure and drawing one point randomly from each, where $1 \le i \le N$.

Let $F(y)$ and $F_i(y)$ be the cumulative distribution functions of objective function values induced from uniform sampling over Ω and each A_i, respectively, so that

$$F(y) = \frac{1}{N} \sum_{i=1}^{N} F_i(y).$$

Let Y be a random variable for the best function value obtained out of N draws over Ω. Then for approaches (i) and (ii), after N draws we have

$$\Pr(Y \le y \mid \text{approach (i)}) = 1 - \left(\frac{1}{N}\sum_{i=1}^{N}(1 - F_i(y))\right)^N, \tag{2.1}$$

$$\Pr(Y \le y \mid \text{approach (ii)}) = 1 - \prod_{i=1}^{N}(1 - F_i(y)). \tag{2.2}$$

Noting that $1 - F_i(y)$ is nonnegative for all i, the inequality of the arithmetic and geometric means,

$$\frac{1}{N}\sum_{i=1}^{N}(1 - F_i(y)) \ge \left[\prod_{i=1}^{N}(1 - F_i(y))\right]^{1/N},$$

can be applied to (2.1) and (2.2), giving

$$\begin{aligned}\Pr(Y \le y \mid \text{approach (i)}) &= 1 - \left(\frac{1}{N}\sum_{i=1}^{N}(1 - F_i(y))\right)^N \\ &\le 1 - \prod_{i=1}^{N}(1 - F_i(y)) = \Pr(Y \le y \mid \text{approach (ii)}).\end{aligned}$$

Therefore, the probability of generating a lower f value using approach (ii) is greater than or equal to that using PRS. Hence, simply by partitioning Ω into a set of equally sized subregions and drawing a sample from each, we would expect to obtain a lower f value. Rather than having a fixed partition as in this example, we propose a partitioning strategy that uses observed function values to form the partition.

2.1. The CART partition In this paper we use classification and regression trees (CART) [6] to partition Ω. Of particular interest is a partition that divides Ω into sets where the objective function is relatively low and high. To construct a CART partition of this form, a training data set T consisting of points with observed low and high function values is required. In this section we assume that there exists a set T of distinct points distributed over Ω with observed function values. The set of points is classified into two mutually exclusive and exhaustive groups as follows.

DEFINITION 2.1 (Low points). The $0 < \phi < |T|$ elements of T with the least function values are classified as low points and form the set ω_L.

DEFINITION 2.2 (High points). The set of points $T \setminus \omega_L$ are classified as high points and form the set ω_H.

Using the high and low points, a CART partition on Ω can be formed.

The CART partitioning algorithm begins by dividing Ω into two hyper-rectangular subregions A_1 and A_2 such that $A_1 \cup A_2 = \Omega$ and $A_1^{\circ} \cap A_2^{\circ} = \emptyset$, where A° denotes the interior of A. To find the boundary that separates A_1 and A_2, a series of hyperplane splits of the form

$$\frac{x_j + z_j}{2}, \quad x \in \omega_L \text{ and } z \in \omega_H, \tag{2.3}$$

are considered, where x_j denotes the jth coordinate of x and the splitting hyperplane is orthogonal to the jth coordinate axis. The method exhaustively searches over each coordinate to find the hyperplane split that maximizes the change in impurity [6], that is, the split that makes A_1 (or A_2) contain mostly points from ω_L or ω_H. For the next iteration, A_1 is divided further by considering splits of the form (2.3) using the points in $T \cap A_1$. The method continues splitting until each subregion contains points from ω_L or ω_H, but not from both.

The CART partition on Ω is a collection of nonempty hyper-rectangular subregions A_i aligned with the coordinate axes. The subsets where the objective function is presumed to be low and high can now be defined.

Definition 2.3 (Low region). The low region of the CART partition is the union of all the subregions A_i that contain an element from ω_L, namely

$$\bigcup_i \{A_i : A_i \cap \omega_L \neq \emptyset\}.$$

Definition 2.4 (High region). The high region of the CART partition is Ω itself.

Our primary goal is to define the low region so that the sample intensity can be increased where the objective is presumed to be low. Therefore, rather than setting the high region as the union of all the high subregions, Ω is set as a single high region. This simplification does not affect the convergence properties of our method, is computationally cheaper, and dramatically simplifies the algorithm.

The hyper-rectangular partition structure is aligned with the coordinate axes. To remove this restrictive alignment, T can be reflected using a Householder matrix so that the partition structure does not have to be aligned with the original coordinate system. The authors found that reflecting T with respect to the principal axis of the cloud of points in ω_L performed well [20, 21].

The training data set is reflected so that the x_1-axis is set parallel to the principal axis of the cloud of points in ω_L. Using $\omega_L^{(i)}$ to denote the ith point in ω_L, expressed as a row vector, the scatter matrix is defined by

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

where $\bar{\omega}_L$ is the sample mean of the points in ω_L. The dominant eigenvector d of the scatter matrix $\mathcal{M}$ is the direction vector for the principal axis of points in ω_L. Premultiplying each point in T with the Householder matrix

$$H = I - 2uu^{\mathsf{T}},$$

where

$$u = \frac{e_1 - d}{\|e_1 - d\|}, \tag{2.4}$$

gives the desired reflection.

To ensure that the reflected points are elements of $[-1, 1]^n$, each point is multiplied by the scalar $1/\varphi$, defined as follows. The Householder matrix H reflects points in the hyperplane given by (2.4) and preserves the length of vectors, $\|Hx\|_2 = \|x\|_2$. Therefore, $\|Hx\|$ is maximized at point(s) that maximize $\|x\|$. This occurs at the vertices of Ω, given by the set

$$V = \{z \in [-1, 1]^n : \|z\| = \sqrt{n}\}.$$

Therefore, the maximum coordinate value that Hx can take for all $x \in \Omega$ is given by

$$\varphi = \max_{z \in V} \|Hz\|_\infty = \max_{i=1,\dots,n} \sum_{j=1}^{n} |H_{ij}|.$$

The scalar φ takes values $1 \le \varphi \le \sqrt{n}$ and gives the minimal scaling such that $(1/\varphi)Hx \in [-1, 1]^n$ for all $x \in \Omega$.

The notation $\bar{\Omega} = \{(1/\varphi)Hx : x \in \Omega\}$ is used to denote the transformed search region, and $\bar{A}_i$ denotes a low subregion in $\bar{\Omega}$. A CART partition performed on $\bar{\Omega}$ defines a collection of hyper-rectangular subregions that partition $\bar{\Omega}$. It is possible that some $\bar{A}_i$ may have subsets that are not in Ω. The feasible subset of each low subregion in $\bar{\Omega}$ defines the low subregion in Ω,

$$A_i = \{\varphi Hx : x \in \bar{A}_i \cap \Omega\}.$$

To reflect a point $x \in \bar{\Omega}$ back to Ω, the reflection φHx is made.

3. The algorithm

The global CARTopt algorithm is a random search method that alternates between partition and sampling phases. The basic structure of the algorithm is given in Algorithm 3.1, and each step is described below. Here $\lceil a \rceil$ denotes the ceiling function, defined by $\lceil a \rceil = \min\{b \in \mathbb{Z} : b \ge a\}$.

Algorithm 3.1 (Global CARTopt algorithm).

(1) **Initialize.** *Choose* $N \ge 2$, $T_{\max} > 0$ *and* $1/N \le \sigma \le (N-1)/N$. *Generate a batch of* $2N$ *points* $X_1 \subset \Omega$, *and let* x_1 *be the best known point. Set* $k = 1$ *and* $T_1 = X_1$.

(2) **Classification.** *Set* ω_L *as the* $\lceil 0.8N \rceil$ *points in* T_k *with the least* f *values and* ω_H *as* $T_k \setminus \omega_L$.

(3) **Partition.** *Form a CART partition on the transformed search region* $\bar{\Omega}$ *using the reflected training data set to define the low region in* $\bar{\Omega}$.

(4) **Sample.** *Generate* $\lceil \sigma N \rceil$ *points from* Ω *and* $N - \lceil \sigma N \rceil$ *feasible points from the low region in* $\bar{\Omega}$. *Reflect the feasible points from* $\bar{\Omega}$ *back to* Ω *and call the new batch of points* X_k.

(5) **Update** T**.** *Let* x_{k+1} *be the best known point. If a restart is used, set* $T_{k+1} \subset T_k \cup X_k$ *such that* $x_{k+1} \in T_{k+1}$*; otherwise set* $T_{k+1} = T_k \cup X_k$*. If* $|T_{k+1}| > T_{\max}$*, the* $|T_{k+1}| - T_{\max}$ *points with largest* f *values are discarded. If stopping conditions are not met, increment* k *and go to Step* 2.

At Step 1, the user chooses a batch size $N \geq 2$, an upper bound $T_{\max}$ and a parameter σ. The upper bound $T_{\max}$ is the maximum number of points that are retained in T to form the CART partition. The parameter σ specifies the number of points to be selected from a uniform distribution over Ω (high region). A large σ value focuses the search effort globally and a small value increases the search intensity in the low region. To complete Step 1, an initial batch of $2N$ points are selected from Ω and f is evaluated at each point. The best point x_1 is set as the point with the best f value.

At Step 2, the batch of points are classified into the sets ω_L and ω_H using Definitions 2.1 and 2.2, respectively. The value $\phi = \lceil 0.8N \rceil$ is chosen here although other choices are possible. Choosing $\phi = \lceil 0.8N \rceil$ keeps the size of ω_L fixed for all iterations. As more points are added to T in Step 5, ω_L tends to cluster. This allows successive partitions to focus down in promising subregions.

A CART partition on the transformed search region $\bar{\Omega}$ is then formed at Step 3 using the reflected training data set. The CART partition is used to vary the sample intensity over Ω by selecting a batch of points from subregions of the partition on $\bar{\Omega}$. The newly generated points are then reflected back to points in Ω. The sampling phase at Step 4 is described in detail in Section 4.

The training data set is updated at Step 5 using the newly generated points. Because the algorithm is stochastic, it is possible to restart the algorithm from time to time. The restart feature is discussed in detail in Section 5. Discarding points with the largest f values ensures that $|T|$ remains bounded. It is necessary that $|T|$ remains bounded for all k to establish convergence to a global minimizer of f. If stopping conditions are not met, the training data is reclassified at Step 2 and the method repeats.

4. The sampling phase

At each pass of the main loop, a batch of N points is drawn from Ω. Rather than drawing all N points from the low region, a fraction

$$\frac{1}{N} \leq \sigma \leq \frac{N-1}{N} \tag{4.1}$$

of points are drawn from Ω (high region). Drawing points from each region provides a way to vary the sample intensity over Ω. Choosing $\sigma < 0.5$, for example, concentrates search effort where f is presumed to be low. The bounds given by (4.1) ensure that at least one point is drawn from the low region ($\lceil \sigma N \rceil \leq N - 1$) and at least one from the high region ($\lceil \sigma N \rceil \geq 1$) at each iteration.

Drawing $\lceil \sigma N \rceil$ points from the high region, Ω, is straightforward. The high region is sampled to ensure that the global minimum is not missed and is necessary to establish convergence.

To draw points uniformly from the low region in Ω, the following method is used. Consider drawing a point uniformly from the low region in $\bar{\Omega}$ with the following structure:

$$\bar{A}_1 \cup \bar{A}_2 \cup \cdots \cup \bar{A}_j,$$

where each $\bar{A}_i$ is a low subregion in $\bar{\Omega}$. Firstly, an $\bar{A}_i$ is selected using a simple discrete inverse transform method [15] and then a point is drawn uniformly from $\bar{A}_i$. A particular $\bar{A}_i$ is selected by choosing i using

$$i = \min\{\alpha \in \mathbb{N} : U \leq F(\bar{A}_\alpha) \text{ and } 0 < \alpha \leq j\},$$

where $U \in [0, 1]$ is a random variable and F is the cumulative distribution function given by

$$F(\bar{A}_i) = \frac{m(\{\bar{A}_1 \cup \bar{A}_2 \cup \cdots \cup \bar{A}_i\})}{m(\{\bar{A}_1 \cup \bar{A}_2 \cup \cdots \cup \bar{A}_j\})},$$

where $m(\cdot)$ denotes the Lebesgue measure. A point x is then drawn from a uniform distribution over the hyper-rectangular low subregion $\bar{A}_i$.

Because the CART partition is on $\bar{\Omega}$, it is possible that the newly generated x may not be an element of Ω. If

$$x \in \bar{A}_i \quad \text{such that } \|\varphi H_k x\|_\infty > 1,$$

the point is rejected ($x \notin \Omega$) and the method repeats. This acceptance/rejection sampling method is an effective technique to use here because $m(\{\bar{A}_i \setminus \Omega\})$ tends to be small, if it exists at all. Furthermore, the method generates the required number of points almost surely (see Proposition 4.1).

Once $N - \lceil \sigma N \rceil$ points have been accepted, the points are reflected back to Ω to give the sample over the low region in Ω. The $N - \lceil \sigma N \rceil$ reflected points and the $\lceil \sigma N \rceil$ points drawn uniformly from Ω complete the batch of N points.

Proposition 4.1. *The sampling method described above generates N points in the low region of Ω with probability* 1.

Proof. If the low region in $\bar{\Omega}$ is a subset of Ω, each point drawn is feasible and the result follows.

Otherwise, $m(\{\bar{A}_1 \cup \bar{A}_2 \cup \cdots \cup \bar{A}_j\} \setminus \Omega) > 0$ and acceptance/rejection sampling is required to reject infeasible points. Generating a feasible point is a Bernoulli trial—a feasible point is either generated or not—and the probability of success is given by

$$P = \frac{1}{m(\{\bar{A}_1 \cup \bar{A}_2 \cup \cdots \cup \bar{A}_j\})} \sum_{i=1}^{j} m(\bar{A}_i \cap \Omega),$$

where j is the number of low subregions. For any $x \in \omega_L$ and $\epsilon < 2$, $m(\mathcal{B}(x, \epsilon) \cap \Omega) \geq m(\mathcal{B}(x, \epsilon)) \cdot 2^{-n} > 0$, where $\mathcal{B}(x, \epsilon)$ is an open ball of radius $\epsilon > 0$. In most cases, $\mathcal{B}(x, \epsilon) \subset \Omega$ for small ϵ. Hence P is nonzero for all iterations.

Noting that repeated trials are independent with constant probability of success P, the number of points generated in the low region after k trials is a binomial random variable. The probability that at least N feasible points are drawn after $k > N$ trials is

$$\Pr(\text{number of feasible points} \geq N) = 1 - \sum_{q=0}^{N-1} \frac{k!}{q!(k-q)!} P^q (1-P)^{k-q}. \tag{4.2}$$

The proposed sampling method draws points indefinitely until N are obtained. Therefore, considering the limit of each term of the summation in (4.2) as k tends to infinity,

$$\frac{P^q}{q!} \lim_{k\to\infty} \frac{k!}{(k-q)!}(1-P)^{k-q},$$

allows the limit of (4.2) to be calculated. Noting that $0 < P \leq 1$ for all k, in the limit as k tends to infinity $(1-P)^{k-q}$ approaches zero. Hence, each term of the summation in (4.2) approaches zero in the limit as k tends to infinity and N points are drawn from the low region in Ω almost surely. □

In practice, the global CARTopt algorithm does generate N points in reasonable time because random number generators satisfy various dispersion conditions. To ensure that N points are drawn, one could, for example, divide the low region into N subregions of equal measure and draw one sample from each. If N feasible points are not obtained, continue dividing the low region into a set of nested subregions and draw one sample from each until N points are obtained. This would ensure that N points are drawn. The following assumption can be made to ensure that the required number of low points are drawn from the low region in finite time.

Assumption 4.2. *Let $N - \lceil \sigma N \rceil$ points be drawn from the low region using a finite number of rejections.*

4.1. The Halton sequence Setting Ω as the high region, rather than the collection of high subregions defined by CART, makes computational sense. Furthermore, it allows the algorithm to search Ω using efficient quasi-random sequences. The Halton sequence [10] is of particular interest here because it is known to quickly distribute points evenly over $[0, 1]^n$ in low dimensions. Exploring Ω using these evenly distributed points reduces the risk of CARTopt becoming trapped at a local minimizer. The Halton sequence also allows us to develop a nonprobabilistic convergence proof, which makes use of the following proposition.

Proposition 4.3. *The Halton sequence $\{u_t\}_{t=1}^{\infty}$ is dense in $[0, 1]^n$.*

Proof. A simple proof is given by Abramson et al. [1] and a detailed proof is given by Halton [10]. □

The Halton sequence gives the algorithm deterministic and stochastic layers. The stochastic layer directs the search where f is presumed to be low based on the CART partition, and the deterministic layer searches Ω in a highly uniform manner to reduce the risk of missing the global minimizer.

5. Restarts

The Global CARTopt algorithm is stochastic and so restarting the algorithm from time to time can produce different results. Restarting can increase the computational efficiency of the method and reduce the risk of becoming trapped at a local solution. Requiring $x_k \in T_{k+1}$ ensures that the sequence $\{f(x_k)\}$ is monotonically decreasing. The restart feature does not affect the convergence properties of the algorithm (see Theorem 7.6) and is not required to establish global convergence.

A simple restarting method is to set $T_{k+1} = x_k$ when $T_{\max}$ is reached or if user conditions are met. This essentially clears the training data set and allows a new sequence of partitions to be formed. However, given that the training data set contains useful information about f, it can be reused. If the sample points with the least f values are reused, the resulting partitions tend to direct the search back to areas previously explored. Reusing points that were drawn from the high region (Ω) gives an unbiased indication of where f is low and potentially directs the search to new areas.

There are many possible heuristics that could be used to restart the algorithm and two are considered in this paper. First, the algorithm restarts if there is a subregion A_i at iteration k such that $\|A_i\|_1 \leq \tau$. The notation $\|A_i\|_1$ is used for the 1-norm of subregion A_i's main diagonal (the diagonal that passes through the centre of A_i). Initially $\tau = 10^{-6}$ is used, and is then replaced by $\tau \leftarrow \tau^{3/2}$ after each restart. Decreasing τ after each restart allows the algorithm to refine its current iterate, producing high accuracy approximate solutions.

Second, the algorithm restarts if there is a subregion A_i at iteration k such that $\|A_i\|_{-\infty} \leq 10^{-16}$, where $\|A_i\|_{-\infty}$ is the smallest (absolute) vector component of subregion A_i's main diagonal. This restarts the algorithm if a particular subregion has become numerically degenerate in at least one dimension.

6. Parallel implementation

It is possible to implement various processes of the global CARTopt algorithm in parallel. Two processes that could benefit from parallelization are forming the CART partition and evaluating the objective function.

It is not necessary to evaluate the objective function sequentially in Step 1 or 5 of the global CARTopt algorithm. Rather, the function values at a batch of points need to be calculated. Therefore, if the objective function is computationally expensive to calculate, each evaluation can be performed in parallel.

The CART partition described in Section 2.1 can be parallelized in several ways. A detailed analysis of parallel CART formulations is beyond the scope of the current paper, but the interested reader is referred to the work of Srivastava et al. [22] and Yıldız and Dikmen [23] for further details. In this section we mention two simple forms of parallelization. Feature-based parallelization [23] searches for the best hyperplane split (see (2.3)) in subregion A_i on n slave processors. Each slave processor searches for the best split in one of the n dimensions and the host processor selects the best overall split.

Node-based parallelization [23] considers each subregion on a slave processor. Initially an optimal hyperplane split is found to divide Ω into two subregions A_1 and A_2. Rather than splitting A_1 and A_2 further on the host processor, $T_A = T \cap A_1$ and $T_B = T \cap A_2$ are sent to slave processors for splitting. Further subregions that are formed and require splitting (contain a mix of points from ω_L and ω_H) can be handled in this way to complete the partition.

7. Convergence

The convergence properties of the global CARTopt algorithm are analysed with the stopping conditions removed. This allows us to examine the asymptotic properties of the sequence of iterates generated by the algorithm. The convergence results show that every cluster point of the sequence $\{x_k\}$ is an essential global minimizer of f with probability one.

Definition 7.1 (Essential global minimizer). An essential global minimizer x_* is a point for which the set

$$L(x_*) = \{x \in \Omega : f(x) < f(x_*)\}$$

has Lebesgue measure zero.

If the objective function is continuous then $f(x_*)$ is also a global minimum in the classical sense.

Assumption 7.2. *Let the following conditions hold.*

(a) *The sequence of function values $\{f(x_k)\}$ is bounded below.*
(b) *The objective function f is lower semicontinuous.*

The first condition of Assumption 7.2 ensures that $f(x_k) \nrightarrow -\infty$ in the limit as $k \to \infty$. The second condition precludes the existence of a sequence $\{x_k\}$ converging to a point x_* for which $f(x_*) - \delta > f(x_k)$ for some $\delta > 0$ when $\|x_k - x_*\| < \epsilon$, where $\epsilon > 0$ and sufficiently small.

Theorem 7.3. *The sequence of iterates $\{x_k\}$ generated by the global CARTopt algorithm is an infinite sequence almost surely.*

Proof. For $\{x_k\}$ to be an infinite sequence, the main loop of the algorithm (Steps 2–5) must be a finite process. The cardinality of the training data set T_k is bounded above by $T_{\max}$ and so Steps 2 and 5 are finite processes. The partition phase uses a finite training data set and the classification method is a finite process, so Step 3 is a finite process. Step 4 draws N points from Ω almost surely. Thus, Step 4 is a finite process almost surely. □

Corollary 7.4. *Let Assumption 4.2 hold. The sequence of iterates $\{x_k\}$ generated by the global CARTopt algorithm is an infinite sequence.*

PROOF. The proof is similar to that of Theorem 7.3. Assumption 4.2 ensures that Step 4 is a finite process. □

THEOREM 7.5. *Let Assumption 7.2 hold. Each cluster point x_* of the sequence $\{x_k\}$ generated by the global CARTopt algorithm is an essential global minimizer of f with probability one.*

PROOF. Theorem 7.3 and the fact that Ω is bounded ensure the existence of cluster points in $\{x_k\}$ almost surely.

Let x_* be any cluster point of $\{x_k\}$ and assume, by contradiction, that x_* is not an essential global minimizer of f. Then there exists a subset $L(x_*) = \{z \in \Omega : f(z) < f(x_*)\}$ with positive Lebesgue measure. Because $N \geq 2$ for all k, at least one point is drawn from a uniform distribution over the high region Ω. The probability that L is sampled at each iteration is bounded below by the probability that L is sampled in a single draw, given by

$$\Pr(x \in L \mid \text{single draw over } \Omega) = \frac{m(L)}{m(\Omega)} > 0.$$

Thus, the probability that L is sampled after k iterations is bounded below by

$$\Pr(x \in L \mid k \text{ draws over } \Omega) = 1 - \left(1 - \frac{m(L)}{m(\Omega)}\right)^k. \tag{7.1}$$

Since there is an infinite number of iterations, in the limit as k tends to infinity expression (7.1) approaches 1 and L is sampled almost surely. Hence, in the limit as $k \to \infty$, $f(x_k) < f(x_*)$ almost surely, contradicting Assumption 7.2(b). Thus, x_* must be an essential global minimizer of f almost surely. □

The next theorem shows that restarting the algorithm does not affect its convergence properties.

THEOREM 7.6. *Let Assumption 7.2 hold. If restarts are used, each cluster point x_* of the sequence $\{x_k\}$ generated by the global CARTopt algorithm is an essential global minimizer of f with probability* 1.

PROOF. At each restart, $x_k \in T_k$ (see Step 5) and hence x_k remains an element of the training data set for all iterations. Thus, Theorem 7.3 and the fact that Ω is bounded ensure the existence of cluster points in $\{x_k\}$. The remainder of the proof follows directly from Theorem 7.5. □

The following theorem gives the proof for the instance when the Halton sequence is used to sample Ω, called the Halton global CARTopt algorithm. This instance is shown to converge to an open set essential global minimizer.

DEFINITION 7.7 (Open set essential global minimizer). An open set essential global minimizer $x_\#$ is a point whose function value $f_\#$ is not more than the supremum of the values $f_0 \in \mathbb{R}$ for which the level set

$$L(f_0) = \{x \in \Omega : f(x) < f_0\}$$

contains no open ball of positive radius.

The definition of $x_\#$ means that there is a dense set in Ω on which the objective function is not less than $f_\#$. We now show that the Halton global CARTopt algorithm's sequence of iterates is dense in Ω.

Theorem 7.8. *Let Assumptions 4.2 and 7.2 hold. Each cluster point x_* of the sequence $\{x_k\}$ generated by the Halton global CARTopt algorithm is an open set essential global minimizer of f.*

Proof. Assumption 4.2, Corollary 7.4, and the fact that Ω is bounded ensure the existence of cluster points in $\{x_k\}$.

Let x_* be any cluster point of $\{x_k\}$ and assume, by contradiction, that x_* is not an open set essential global minimizer of f. Then for some $z \in \Omega$ there exists a set

$$B(z, \epsilon) = \{\|z - x\| < \epsilon : x \in \Omega\} \tag{7.2}$$

such that $f(x) < f(x_*)$ for all $x \in B(z, \epsilon)$ for all sufficiently small $\epsilon > 0$. Because $N \geq 2$, at least one Halton point is drawn from Ω at each iteration. Noting that there is an infinite number of iterations and that the Halton sequence is dense in Ω (Proposition 4.3), it follows from the definition of a dense set that there exists a Halton point in (7.2) for all $\epsilon > 0$, contradicting Assumption 7.2(b). Thus, x_* must be an open set essential global minimizer of f. □

Theorem 7.6 is easily extended to accommodate restarts if the Halton sequence is not restarted from the same seed point at each restart.

8. Numerical results and discussion

The global CARTopt algorithm was tested on two sets of problems. The first are smooth global optimization problems from papers of Ali et al. [2] and Rinnooy Kan and Timmer [19]. The second set of problems are chosen from papers of Ali et al. [2] and Moré et al. [16], and are 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 10^{-5} on the nonsmooth function, for example, corresponds to a function value of approximately 10^{-10} on the original problem.

Numerical results were generated using a batch size $N = 20$ and $T_{\max} = 10\,000$. A post-partition modification [20, 21] was also applied to the CART partition. This modification replaces any low subregion that only contains one point with a hypercube whose volume occupies approximately $1/|\omega_L|$ of the volume of the low region [20, 21]. All the algorithms considered in this section are stochastic and so each reported value was averaged over 10 runs. A maximum number of 50 000 function evaluations was also used for all the algorithms considered. If a particular algorithm failed to obtain the desired accuracy within 50 000 function evaluations on a particular problem, the run was considered a failure. If a method failed at least once over the ten runs, the number of failures was reported.

Comparisons with the stochastic algorithms Tilecutter, ARS and CRS were also made. Tilecutter was implemented as in an earlier paper [17], with the tile cutting parameter $A = 3/2$ and restarts if the smallest 1-norm tile size (τ_{acc}) is 10^{-8}. ARS was implemented with a contraction factor of $\sqrt{2}$ and restarts if the smallest subregion size is 10^{-8}. CRS was implemented with its stopping rule removed and used an initial batch of $35n$ points as per Brachetti et al. [5]. We removed the stopping rule from CRS so we could make comparisons between the methods using the same stopping rule, details of which follow.

8.1. Smooth problems The high region sampling parameter σ changes how the global CARTopt algorithm operates. Selecting large values of σ means that the method performs more like PRS, directing its search globally. Choosing a small value turns the algorithm into a greedy method which directs its search locally. For small values of σ there is a greater risk of missing the global solution, but for large values the method becomes inefficient. In Tables 1–3 the rate of progress of the global CARTopt algorithm on the smooth problems is shown for the values $\sigma = 0.2, 0.5, 0.8$. The results from using the Halton sequence to sample the high region are not presented here because there was little difference between the two instances.

The rate of convergence was approximately sublinear for each choice of σ and the algorithm performed best with $\sigma = 0.2$. Although $\sigma = 0.2$ focuses the search effort locally, the global minimum was not missed on the problems considered. More function evaluations were required to solve each problem for the larger σ values.

Seven of the smooth test problems are reported in an earlier paper [17] for C-GRASP [11] and Tilecutter. These papers [11, 17] list the average number of function evaluations taken by each method to satisfy

$$|f(x_k) - f_*| < 10^{-6} + 10^{-4}|f_*|, \tag{8.1}$$

where f_* is the global minimum. Table 4 lists the averages (over ten runs) for CARTopt, CRS and ARS using the stopping criterion given by (8.1). The global CARTopt algorithm was implemented with $\sigma = 0.2$ and the high region was sampled using either random or Halton points.

With the exception of one problem (Goldstein), C-GRASP was considerably slower than all the other methods. All the methods solved the Goldstein problem, but C-GRASP was extremely fast. Tilecutter and CRS performed similarly on the smooth problems. ARS failed at least twice on four of the smooth problems. On the Shekel problems, ARS became trapped at a local minimizer for many iterations. ARS focused

Table 1. Rate of progress of the global CARTopt algorithm with $\sigma = 0.2$. This σ value concentrates most of the search effort in areas identified as low. f scores in bold show when the average function value over ten runs reached the global minimum to four decimal places.

Problem	n	f_*	Number of function evaluations					
			200	400	800	1600	3200	6400
Brannin	2	0.3979	0.4436	0.3980	**0.3979**	0.3979	0.3979	0.3979
Camel 6	2	−1.0316	−1.0074	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316
Cosine Mix.	8	−0.8000	0.1759	−0.0026	−0.5067	−0.7867	**−0.8000**	−0.8000
Exponential	15	−1	−0.5244	−0.6936	−0.8322	−0.9797	−0.9989	**−1.0000**
Goldstein	2	3	3.4448	3.0013	**3.0000**	3.0000	3.0000	3.0000
Hartmann 3	3	−3.8628	−3.8341	−3.8625	**−3.8628**	−3.8628	−3.8268	−3.8268
Hartmann 6	6	−3.3224	−2.8095	−3.1287	−3.3068	−3.3222	**−3.3224**	−3.3224
Neumaier 3	10	−210	4439	1918	298.26	−171.65	−209.53	**−210**
Paviani	10	−45.7785	−23.766	−32.905	−42.127	−45.344	**−45.7785**	−45.7785
Rastrigan	2	−2	−1.8237	−1.9321	−1.9395	−1.9758	**−2.0000**	−2.0000
Shekel 5	4	−10.1532	−2.0528	−5.9979	−10.0704	**−10.1532**	−10.1532	−10.1532
Shekel 7	4	−10.4029	−2.3158	−5.8392	−10.3378	**−10.4029**	−10.4029	−10.4029
Shekel 10	4	−10.5364	−1.9761	−5.4970	−10.4797	**−10.5364**	−10.5364	−10.5364

Table 2. Rate of progress of the global CARTopt algorithm with $\sigma = 0.5$. This σ value concentrates the search effort equally between areas identified as low and high.

Problem	n	f_*	Number of function evaluations					
			200	400	800	1600	3200	6400
Brannin	2	0.3979	0.3992	**0.3979**	0.3979	0.3979	0.3979	0.3979
Camel 6	2	−1.0316	−1.0298	**−1.0316**	−1.0316	−1.0316	−1.0316	−1.0316
Cosine Mix.	8	−0.8000	0.3101	−0.0333	−0.3024	−0.6678	−0.7998	**−0.8000**
Exponential	15	−1	−0.5355	−0.7193	−0.8675	−0.9652	−0.9985	**−1.0000**
Goldstein	2	3	3.0188	**3.0000**	3.0000	3.0000	3.0000	3.0000
Hartmann 3	3	−3.8628	−3.8606	**−3.8628**	−3.8628	−3.8628	−3.8628	−3.8628
Hartmann 6	6	−3.3224	−2.8504	−3.0713	−3.2492	−3.2974	**−3.3224**	−3.3224
Neumaier 3	10	−210	3113	1981	310.96	−154.74	−207.87	**−210**
Paviani	10	−45.7785	−24.623	−31.510	−42.046	−45.237	−45.768	**−45.7785**
Rastrigan	2	−2	−1.8216	−1.9349	−1.9757	−1.9876	**−2.0000**	−2.0000
Shekel 5	4	−10.1532	−2.0424	−4.1129	−9.4007	−10.1527	**−10.1532**	−10.1532
Shekel 7	4	−10.4029	−1.5601	−3.3229	−8.7804	−9.7344	−10.3997	**−10.4029**
Shekel 10	4	−10.5364	−1.9476	−2.8647	−7.4682	−8.2879	−10.5310	**−10.5364**

its search effort around the local minimizer, rather than searching globally, and in these cases missed the global minimum. On the Hartman 6 problem, ARS also converged to a local minimum on two runs. The CARTopt algorithm was fast on all of the problems considered, requiring the fewest function evaluations on five of the seven problems. There was little difference between searching Ω with the Halton sequence or using random sampling: both were effective.

8.2. Nonsmooth problems The global CARTopt algorithm with $\sigma = 0.2$ was also compared with Tilecutter, CRS and ARS on a set of nonsmooth problems,

TABLE 3. Rate of progress of the global CARTopt algorithm with $\sigma = 0.8$. This σ value concentrates most of the search effort in the high region.

Problem	n	f_*	Number of function evaluations					
			200	400	800	1600	3200	6400
Brannin	2	0.3979	0.5852	0.4300	0.4012	**0.3979**	0.3979	0.3979
Camel 6	2	−0.408	−0.8742	−0.9673	−1.0298	**−1.0316**	−1.0316	−1.0316
Cosine Mix.	8	−0.8000	0.6543	0.0045	−0.2367	−0.6879	−0.7965	**−0.8000**
Exponential	15	−1	−0.4538	−0.6969	−0.8460	−0.9706	−0.9982	**−1.0000**
Goldstein	2	3.0000	8.7839	4.0393	3.0320	3.0002	**3.0000**	3.0000
Hartmann 3	3	−3.8628	−3.7290	−3.8267	−3.8595	−3.8627	**−3.8628**	−3.8628
Hartmann 6	6	−3.3224	−2.5667	−3.0128	−3.2341	−3.3013	−3.3103	**−3.3224**
Neumaier 3	10	−210	5990	2091	587.63	−78.796	−200.53	−209.983
Paviani	10	−45.7785	−21.777	−29.275	−39.237	−31.463	−45.763	**−45.7785**
Rastrigan	2	−2	−1.7463	−1.8957	−1.9345	−1.9865	−1.9879	**−2.0000**
Shekel 5	4	−10.1532	−1.1685	−2.0408	−4.5301	−8.5904	−9.6388	**−10.1532**
Shekel 7	4	−10.4029	−1.5801	−2.3577	−3.6461	−8.6504	−9.8712	**−10.4029**
Shekel 10	4	−10.5364	−1.6977	−2.2051	−4.1761	−8.3667	−9.2225	**−10.5364**

TABLE 4. The average number of function evaluations required to satisfy the stopping criterion in (8.1). Figures in bold show the method which took the fewest function evaluations. CARTopt used $\sigma = 0.2$.

Problem	n	C-GRASP	Tilecutter	CRS	ARS	CARTopt (Random)	CARTopt (Halton)
Brannin	2	59 857	717	1287	**267**	425	436
Goldstein	2	**29**	771	1251	240	428	412
Hartmann 3	3	20 743	1205	1474	1095	381	**378**
Hartmann 6	6	79 685	12 504	12 673	2 fails	1787	**1774**
Shekel 5	4	5 545 982	5449	6952	2 fails	1285	**1275**
Shekel 7	4	4 052 800	4475	6038	2 fails	1096	**1075**
Shekel 10	4	4 701 358	6295	6345	3 fails	**1130**	1255

where $f(x_*) = 0$ for each problem. The average (over ten runs) number of function evaluations to obtain an approximate solution of low (10^{-2}), moderate (10^{-4}), and high (10^{-6}) accuracy are reported in Tables 5–7. The least number of function evaluations required to satisfy each condition is shown in bold.

Tilecutter performed well on most of the problems, but performed poorly on the variably dimensioned problem and failed to reduce the Schaffer 2 problem below 10^{-4}. Tilecutter was converging to the global minimum on the variably dimensioned problem, but required more than 50 000 function evaluations to satisfy the stopping criterion. The search region for the Schaffer 2 problem is given by $\Omega = [-100, 100]^2$. Tilecutter scales the search region to $[0, 1]^2$ and so $\tau_{\text{acc}} = 10^{-8}$ scales to $\tau_{\text{acc}} = 10^{-6}$. The first author decreased τ_{acc} from 10^{-8} to 10^{-10} and then Tilecutter was able to reduce the Schaffer 2 function below 10^{-4} using an average of 18 000 function evaluations. However, the first author could not find a value of τ_{acc} to reduce f below 10^{-6} within 50 000 function evaluations. Tilecutter performed the best on the Weka 3

TABLE 5. Average number of function evaluations required to reduce f below 10^{-2} on the nonsmooth problems.

Problem	n	Tilecutter	CRS	ARS	CARTopt (Random)	CARTopt (Halton)
Becker & Lago	2	609	1273	**184**	439	393
Bohachevsky 1	2	1461	1609	669	517	**507**
Bohachevsky 2	2	1496	1509	**310**	516	533
Levy & Montalvo 1	3	2345	2309	5782	606	**571**
Levy & Montalvo 2	3	2083	2711	3 fails	639	**624**
Mod. Rosenbrock	2	1976	12 441	4293	**1165**	2200
Periodic (Price)	2	719	1239	835	1118	**657**
Schaffer 2	2	4541	2479	**865**	893	882
Trigonometric	5	4320	11 982	2482	**1306**	1315
Variably dimensioned	4	23 577	3641	**1097**	1221	1202
Variably dimensioned	8	10 fails	22 610	8 fails	**4982**	5027
Weka 1	2	124	256	**52**	130	111
Weka 2	2	**383**	807	4695	1320	1458
Weka 3	2	**5983**	3 fails	10 fails	3 fails	10 fails

TABLE 6. Average number of function evaluations required to reduce f below 10^{-4} on the nonsmooth problems.

Problem	n	Tilecutter	CRS	ARS	CARTopt (Random)	CARTopt (Halton)
Becker & Lago	2	3014	2455	**400**	723	767
Bohachevsky 1	2	4908	2414	1015	831	**825**
Bohachevsky 2	2	4523	2351	945	812	**802**
Levy & Montalvo 1	3	4744	4097	6203	932	**930**
Levy & Montalvo 2	3	10 153	4523	8 fails	**1094**	1095
Mod. Rosenbrock	2	6119	13 437	10478	**1143**	1676
Periodic (Price)	2	1825	2722	1712	1341	**909**
Schaffer 2	2	10 fails	4132	10 fails	**2429**	2590
Trigonometric	5	17 655	17 617	4797	**2747**	3162
Variably dimensioned	4	6 fails	6751	2592	**1990**	2003
Variably dimensioned	8	10 fails	29 268	10 fails	8825	**8705**
Weka 1	2	922	582	**188**	289	275
Weka 2	2	1456	1663	6456	**1321**	2271
Weka 3	2	**6022**	3 fails	10 fails	4 fails	10 fails

problem (only failing once). This function has many local minima, and Tilecutter's success is largely due to its search being focused globally rather than locally.

CRS solved all but one of the nonsmooth problems, failing several times on the Weka 3 problem. For most of the other problems, the method was faster than Tilecutter to obtain approximate solutions of moderate and high accuracy, but slower at obtaining low-accuracy solutions.

TABLE 7. Average number of function evaluations required to reduce f below 10^{-6} on the nonsmooth problems.

Problem	n	Tilecutter	CRS	ARS	CARTopt	CARTopt
					(Random)	(Halton)
Becker & Lago	2	8915	3628	**968**	1365	1111
Bohachevsky 1	2	12 020	3415	**1645**	2050	2082
Bohachevsky 2	2	11 099	3298	1242	1123	**1116**
Levy & Montalvo 1	3	12 524	5417	6586	**1281**	1299
Levy & Montalvo 2	3	35 508	6286	10 fails	**1580**	1730
Mod. Rosenbrock	2	14 815	16 483	5 fails	**1844**	2354
Periodic (Price)	2	5936	3692	3450	2394	**1680**
Schaffer 2	2	10 fails	5790	10 fails	4370	**4301**
Trigonometric	5	45 917	22 319	14903	4505	**4266**
Variably dimensioned	4	9 fails	10 162	5368	**2958**	2960
Variably dimensioned	8	10 fails	37 049	10 fails	**16 820**	21 773
Weka 1	2	2804	963	**327**	436	422
Weka 2	2	3400	2546	10319	**2009**	2099
Weka 3	2	**1 fail**	5 fails	10 fails	6 fails	10 fails

The ARS algorithm focused its search locally and in doing so failed on a number of problems. The method failed to reduce f below 10^{-4} on four problems. ARS failed in a similar way to Tilecutter on the Schaffer 2 problem because the smallest subregion size was too large. Setting this parameter to 10^{-10} allowed the algorithm to reduce f below 10^{-4} using approximately 2500 function evaluations. Choosing a smallest subregion size of 10^{-14} allowed the algorithm to reduce f below 10^{-6} using an average of 4500 function evaluations. It should be noted that these choices performed poorly on the other nonsmooth problems considered and that ARS is extremely sensitive to this parameter. Choosing the smallest subregion size too large resulted in low-accuracy approximate solutions and choosing it too small promoted local convergence, missing the global solution altogether.

The global CARTopt algorithm performed well on all but one of the problems, failing many times on the Weka 3 problem. The first author also produced results using $\sigma = 0.5$. For this choice of σ, the random instance of the global CARTopt algorithm only once failed to reduce the Weka 3 problem below 10^{-6}. An average of 13 822 and 17 245 function evaluations were required to reduce f below 10^{-2} and 10^{-4}, respectively, and no failures were observed. However, the Halton instance still failed every time, converging to local minima. This suggests that reusing the Halton points at each restart produced similar partitions that led to local, rather than global, convergence. A better approach would be to continue with new Halton points only.

On all of the other problems, the global CARTopt algorithm performed well. There was little difference between the Halton and the random instances. The method was the fastest to obtain low-accuracy solutions on seven of the problems, moderate accuracy solutions on 11, and high accuracy solutions on ten of the problems.

9. Conclusion

A random search algorithm for bound-constrained global optimization of nonsmooth functions is presented, called global CARTopt. The method alternates between partition and sampling phases. At each partition phase, a CART partition is formed on the search region Ω to locate low and high regions with respect to the objective function. The sampling phase selects a batch of points from Ω using the partition to increase the sample intensity in the low region. Alternating between partition and sampling phases provides an effective method for global optimization. Convergence to an essential global minimizer with probability one is demonstrated under mild conditions. If the high region is sampled using the Halton sequence, each cluster point is an open set essential global minimizer. Numerical results show that theoretical convergence is achieved in practice.

Comparison with other random search methods shows that the global CARTopt algorithm is competitive in practice. Some global optimization methods focus their search locally (for example, ARS) and others focus their search globally (for example, Tilecutter). If the user anticipates many local minimizers in their problem, an algorithm with a global focus can be advantageous; if few minimizers are expected, an algorithm with local focus can be more efficient. Global CARTopt has the ability to vary its focus by changing a single parameter σ. Choosing $\sigma \approx 0.8$ gives a global focus and $\sigma \approx 0.2$ gives a local focus, rather than choosing two different algorithms.

Acknowledgement

We thank the anonymous referee for many helpful comments, which have improved this paper.

References

[1] M. A. Abramson, C. Audet, J. E. Dennis and S. Le Digabel, "OrthoMADS: a deterministic MADS instance with orthogonal directions", *SIAM J. Optim.* **20** (2008) 948–966; doi:10.1137/080716980.

[2] M. M. Ali, C. Khompatraporn and Z. B. Zabinsky, "A numerical evaluation of several stochastic algorithms on selected continuous global optimization problems", *J. Global Optim.* **31** (2005) 635–672; doi:10.1007/s10898-004-9972-2.

[3] M. J. Appel, R. LaBarre and D. Radulovic, "On accelerated random search", *SIAM J. Optim.* **14** (2003) 708–731; doi:10.1137/S105262340240063X.

[4] C. Audet and J. E. Dennis Jr, "Analysis of generalized pattern searches", *SIAM J. Optim.* **13** (2003) 889–903; doi:10.1137/S1052623400378742.

[5] P. Brachetti, M. De Felice Ciccoli, G. Di Pillo and S. Lucidi, "A new version of the Price's algorithm for global optimization", *J. Global Optim.* **10** (2010) 165–184; doi:10.1023/A:1008250020656.

[6] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, *Classification and regression trees* (Wadsworth International Group, Monterey, CA, 1984).

[7] M. E. Dyer and A. M. Frieze, "Computing the volume of convex bodies: a case where randomness provably helps", *Proc. Sympos. Appl. Math.* **44** (1991) 123–169; doi:10.1090/psapm/044/1141926.

[8] M. Dyer, A. Frieze and R. Kannan, “A random polynomial-time algorithm for approximating the volume of convex bodies”, *J. ACM* **38** (1991) 1–17; doi:10.1145/102782.102783.
[9] S. M. Ermakov, A. A. Zhigyavskii and M. V. Kondratovich, “Comparison of some random search procedures for a global extremum”, *USSR Comput. Math. Math. Phys.* **29** (1989) 112–117; doi:10.1016/0041-5553(89)90054-2.
[10] J. H. Halton, “On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals”, *Numer. Math.* **2** (1960) 84–90; doi:10.1007/BF01386213.
[11] M. J. Hirsch, C. N. Meneses, P. M. Pardalos and M. G. C. Resende, “Global optimization by continuous GRASP”, *Optim. Lett.* **1** (2007) 201–212; doi:10.1007/s11590-006-0021-6.
[12] R. Horst and T. Hoang, *Global optimization: deterministic approaches*, 3rd edn. (Springer, Berlin, 1996).
[13] R. Horst and P. M. Pardalos, *Handbook of global optimization* (Kluwer, Dordrecht, 1995).
[14] D. Jones, C. D. Perttunen and B. E. Stuckman, “Lipschitzian optimization without the Lipschitz constant”, *J. Optim. Theory Appl.* **79** (1993) 157–181; doi:10.1007/BF00941892.
[15] W. L. Martinez and A. R. Martinez, *Computational statistics handbook with Matlab* (Chapman & Hall/CRC, Boca Raton, FL, 2002).
[16] J. J. Moré, B. S. Garbow and K. E. Hillstrom, “Testing unconstrained optimization software”, *ACM Trans. Math. Softw.* **7** (1981) 17–41; doi:10.1145/355934.355936.
[17] C. J. Price, M. Reale and B. L. Robertson, “A cover partitioning method for bound constrained global optimization”, *Optim. Meth. Softw.* **27** (2012) 1059–1072; doi:10.1080/10556788.2011.557726.
[18] W. L. Price, “A controlled random search procedure for global optimisation”, *Comput. J.* **4** (1977) 367–370; doi:10.1093/comjnl/20.4.367.
[19] A. H. G. Rinnooy Kan and G. T. Timmer, “Stochastic global optimization methods part II: Multi level methods”, *Math. Program.* **39** (1987) 57–78; doi:10.1007/BF02592071.
[20] B. L. Robertson, “Direct search methods for nonsmooth problems using global optimization techniques”, Ph. D. Thesis, University of Canterbury, Christchurch, New Zealand, 2010.
[21] B. L. Robertson, C. J. Price and M. Reale, “CARTopt: a random search method for nonsmooth unconstrained optimization”, *Comput. Optim. Appl.* **56** (2013) 291–315; doi:10.1007/s10589-013-9560-9.
[22] A. Srivastava, E.-H. Han, V. Kumar and V. Singh, “Parallel formulations of decision-tree classification algorithms”, *High Performance Data Mining* **3** (2002) 237–261; doi:10.1007/0-306-47011-X_2.
[23] O. T. Yıldız and O. Dikmen, “Parallel univariate decision trees”, *Pattern Recog. Lett.* **28** (2007) 825–832; doi:10.1016/j.patrec.2006.11.009.
[24] Z. B. Zabinsky, *Stochastic adaptive search for global optimization* (Kluwer, Dordrecht, 2003).
[25] Z. B. Zabinsky et al., “Random search algorithms”, in: *Wiley encyclopedia of operations research and management science* (ed. J. J. Cochran), (Wiley, Hoboken, NJ, 2010); doi:10.1002/9780470400531.
[26] A. A. Zhigljavsky, *Theory of global random search* (Kluwer, Dordrecht, 1991).