Stochastic global optimization using random forests

B. L. Robertson [a], C. J. Price[a], Marco Reale[a]

[a]*School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand*
Email: blair.robertson@canterbury.ac.nz

Abstract: Global optimization problems occur in many fields including mathematics, statistics, computer science, engineering, and economics. The purpose of a global optimization algorithm is to efficiently find an objective function's global minimum. In this article we consider bound constrained global optimization, where the search is performed in a box, denoted Ω. The global optimization problem is deceptively simple and it is usually difficult to find the global minimum. One of the difficulties is that there is often no way to verify that a local minimum is indeed the global minimum. If the objective function is convex, the local minimum is also the global minimum. However, many optimization problems are not convex. Of particular interest in this article are objective functions that lack any special properties such as continuity, smoothness, or a Lipschitz constant.

The CARTopt algorithm is a stochastic algorithm for bound constrained global optimization. This algorithm alternates between partition and sampling phases. At each iteration, points sampled from Ω are classified low or high based on their objective function values. These classified points define training data that is used to partition Ω in boxes using classification and regression trees (CART). The objective function is then evaluated at a number of points drawn from the boxes classified low and some are drawn from Ω itself. Drawing points from the low boxes focuses the search in regions where the objective function is known to be low. Sampling Ω reduces the risk of missing the global minimum and is necessary to establish convergence. The new points are then added to the existing training data and the method repeats.

In this article we provide an alternative partitioning strategy for the CARTopt algorithm. Rather than using a CART partition at each iteration, we propose using a random forest partition. Pure CART partitions are known to over-fit training data and have low bias and high variance. A random forest partition has less variance than a CART partition, and hence provides a more stable approximation to where the objective function is expected to be low. We also provide a method for sampling low regions defined by random forest partitions. A preliminary simulation study showed that using random forests in the CARTopt algorithm was an effective strategy for solving a variety of global optimization test problems. The authors are currently refining the method and extending the set of test problems.

Keywords: *Bound constrained global optimization, CART, CARTopt, random search*

1 INTRODUCTION

The bound constrained global optimization problem is of the form

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

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

$$\Omega = \{\mathbf{x} \in \mathbb{R}^n : l_j \leq x_j \leq u_j \text{ for all } j = 1, \ldots, n\}.$$

The objective function f maps Ω into $\mathbb{R} \cup \{+\infty\}$ and is assumed to be lower semi-continuous. The inclusion of $\{+\infty\}$ means certain constrained optimization problems can be handled using the extreme barrier approach of Audet and Dennis (2003), which assigns the value $+\infty$ to infeasible points or where it is otherwise not defined.

Despite the deceptively simple form of (1), finding a global minimum is usually difficult because there is often no way to verify that a local minimum is indeed the global minimum. A simple method for solving (1) is called pure random search (PRS). PRS evaluates f at a number of points randomly drawn from Ω and returns the lowest f value as an approximate global minimum. However, PRS often returns low accuracy solutions because the probability of generating a point near a global minimizer is usually very low. A simple modification of PRS was proposed by Appel et al. (2003), called accelerated random search (ARS). ARS evaluates f at points drawn from a finite sequence of contracting boxes (initially Ω itself) centred on the point with the best f value. If a point with a lower f value is found, or if the sequence is exhausted, the search returns to Ω. This approach may appear counter intuitive, but can be very effective at refining the current estimate of the global minimum.

Numerous deterministic and stochastic optimization methods have been proposed for bound constrained optimization. Stochastic methods have an element of randomness or probability in their design and deterministic methods do not. Examples of deterministic methods include branch-and-bound methods (c.f. Horst and Hoang (1996)) and the DIvide a hyper-RECTangle (DIRECT) algorithm of Jones et al. (1993). Stochastic methods that have enjoyed recent interest include simulated annealing (proposed for continuous problems by Bélisle et al. (1993)), genetic algorithms (also called evolutionary algorithms) introduced by Holland (1975) and particle swam optimization introduced by Kennedy and Eberhart (1995). Zabinsky (2009) describes the trade-off between deterministic and stochastic methods in terms of computational effort and the type of convergence. Stochastic methods are typically fast, but only offer convergence to the global minimum with probability one.

This article considers the stochastic method of Robertson et al. (2013b), called CARTopt. This algorithm alternates between partition and sampling phases. At each iteration, points sampled from Ω are classified as low or high based on their evaluated f values. These classified points are then used as training data for a binary classification problem (low or high) using the classification and regression trees (CART) method of Breiman et al. (1984). CART is chosen because it partitions Ω into boxes (see Figure 1(b)), which means particular regions of the partition can be sampled directly. Once the partition is built, f is evaluated at a number of points drawn from the boxes classified low and some from those classified high. More points are drawn from the low boxes to focus the search in regions where f is known to be low to increase the chance of reducing f further. Sampling the high boxes reduces the risk of missing the global minimum and is necessary to establish convergence. The new points are then added to the existing training data and the method repeats.

An acceptable to solution to (1) is called an essential global minimizer (EGM) (c.f. Price et al. (2012)). An EGM, $\mathbf{x}_*$, is a point for which the set

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

has Lebesgue measure zero. If f is continuous, then $\mathbf{x}_*$ is also a global minimizer in the classical sense. Under mild conditions, Robertson et al. (2013b) show that the sequence of best points generated by CARTopt converges to an EGM of f with probability one.

In this article, we propose using a random forest partition in the CARTopt algorithm instead of using a CART partition. In Section 2, CARTopt's training data and CART partition are explained. Partitions built using random forests are discussed in Section 3 and a way to sample from these partitions is proposed in Section 4. Concluding remarks are given in Section 5.

2 CARTOPT PARTITION

CARTopt's partition phase divides Ω into low and high boxes using CART and binary classified (low or high) training data

$$D = \{\mathbf{x}_i, y_i : \mathbf{x}_i \in \Omega \text{ and } i = 1, 2, \ldots, N\},$$

where N is the number of training observations and

$$y_i = \begin{cases} low & \text{if } f(\mathbf{x}_i) < f_c \\ high & \text{otherwise.} \end{cases}$$

Robertson et al. (2013b) suggest choosing f_c so that a relatively small number of points are classified low at each iteration. This choice promotes clustering in D, particularly in areas where f is relatively low. In this article we classify the 20 points in D with the least f values low.

CART forms its partition on Ω using a greedy recursive binary splitting strategy. Initially, $\Omega_1 = \Omega$ is split into two boxes, Ω_2 and Ω_3, by considering a series of hyperplane splits of the form

$$x_j = \frac{1}{2}\left(\mathbf{x}_{ij} + \mathbf{x}_{kj})\right) \text{ such that } y_i = low \text{ and } y_k = high,$$

where $\mathbf{x}_{ij}$ denotes the jth coordinate of $\mathbf{x}_i \in \Omega_1$. The method exhaustively searches over each coordinate to find the hyperplane split that maximises

$$\Delta = i(\Omega_1) - \frac{|\Omega_2|}{|\Omega_1|} i(\Omega_2) - \frac{|\Omega_3|}{|\Omega_1|} i(\Omega_3), \tag{2}$$

where $|\Omega_j|$ is number of points in Ω_j and $i(\Omega_j)$ is the impurity of box Ω_j. In this article we measure impurity using the Gini index

$$i(\Omega_j) = 2\frac{low_j}{|\Omega_j|}\left(1 - \frac{low_j}{|\Omega_j|}\right),$$

where low_j is the number of low points in Ω_j. The method then repeats on each box until all boxes are pure — containing either low or high points, but not both. An example of a CART partition is given in Figure 1(b).

To potentially simplify the CART partition, Robertson et al. (2013a) reflect the training data using a Householder matrix

$$H = I - 2\frac{(\mathbf{e}_1 - \mathbf{v})(\mathbf{e}_1 - \mathbf{v})^{\mathrm{T}}}{\|\mathbf{e}_1 - \mathbf{v}\|^2},$$

where I is the identity matrix, $\mathbf{e}_1$ is the first column of I and $\mathbf{v}$ is the dominant eigenvector of the covariance matrix for the low points. Pre-multiplying each training observation with H aligns CART splits with respect to $\mathbf{v}$ rather than the coordinate axes. A CART partition using reflected training data is illustrated in Figure 1(c). This is an effective strategy for simplifying CART partitions and its effectiveness has been demonstrated by Wickramarachchi et al. (2016) for a number of classification problems.

To increase the performance of CARTopt, Robertson et al. (2013a) perform three post-partition modifications on the CART partition. The first modification stops low boxes from becoming numerically degenerate by forcing a minimum box diameter. The second potentially reduces the size of low boxes that share at least one face with Ω. This modification requires at least one additional f evaluation for each low box that is considered. The third modification replaces singleton low boxes (boxes containing one low point) with hypercubes whose size is set relative to the size of the other low boxes. This modification can be seen in Figure 1(b) where five singleton low boxes have been replaced with square boxes.

3 RANDOM FOREST PARTITIONS

In this section, we propose using a random forest partition in the CARTopt algorithm. The purpose of CARTopt's partition phase is to define where f is expected to be low using a partition that can be efficiently sampled. A CART partition is efficient to sample, but forcing each box to be pure over-fits the training data.

These pure CART partitions are known to have low bias and high variance (c.f. Hastie et al. (2013)), meaning small changes in the training observations can produce substantial changes in the partition.

To reduce over-fitting and the variance of CART partitions, bootstrap aggregation (bagging) can be used. In bagging, B CART partitions are built using bootstrapped training data and classifications are made using a majority vote. The bias of the bagged partition is the same as an individual CART partition, but the variance of the bagged partition is typically less than an individual CART partition. Breiman (2001) introduced random forests (Algorithm 1) in an attempt to improve the variance reduction further by decorrelating the partitions. This is achieved by randomly choosing m dimensions for splitting each impure box while building the partition (line 7 of Algorithm 1). Setting $m = n$ is bagging and choosing $1 \leq m < n$ attempts to decorrelate the partitions. Here we choose $m = \lceil \sqrt{n} \rceil$ following from Breiman (2001), where $\lceil \cdot \rceil$ is the ceiling function. CART partitions can be simplified by using the Householder reflected training data, particularity in the neighbourhood of a global minimizer (see Figure 1(c)). Therefore, half of the partitions in the forest are built using bootstrapped samples from Householder reflected D. The remaining partitions use bootstrapped samples from D.

Algorithm 1 *Random Forest Partition*

```
1:  Input: $N$ training observations $D = \{\mathbf{x}_i, y_i\}_{i=1}^N$.
2:  for $k = b$ to $B$ do
3:      Draw a bootstrap sample of size $N$ from $D$ if $k$ is even. Otherwise, bootstrap from reflected $D$.
4:      Form a random forest partition $T_b$ to the bootstrapped data, by recursively repeating
5:      the following steps for each impure box in the partition.
6:      repeat
7:          Select $m = \lceil \sqrt{n} \rceil$ dimensions at random from the $n$ dimensions.
8:          Divide the box using the split that maximises (2) from among the $m$ dimensions.
9:      until all boxes are pure
10: end for
11: Output: The ensemble of partitions $\{T_b\}_{b=1}^B$.
```

Random forest classifiers typically classify observations using a majority vote. That is, if at least $B/2$ of the partitions classify an $\mathbf{x} \in \Omega$ low, then $\mathbf{x}$ is classified low. However, we have imbalanced training data (many high points and few low points) and we are primarily interested in defining where f is expected to be low. Therefore, rather than using a majority vote, we classify $\mathbf{x}$ low if at least $1 \leq \alpha \leq B/2$ partitions classify $\mathbf{x}$ low. In this article we choose $B = 20$ (the number of low points drawn) and $\alpha = 3$. We recommend relatively low α values to reduce the risk of misclassifying promising low regions. An example of a random forest partition with $B = 20$ and $\alpha = 3$ is shown in Figure 1(e).

Two post-partitions modifications from Robertson et al. (2013a) are also applied to the random forest partition. We force a minimum low box diameter to stop these boxes becoming numerically degenerate and replace singleton low boxes with hypercubes. These modifications are simple to implement, do not require additional f evaluations, and are not necessarily implemented at each iteration, only when required.

4 SAMPLING FROM RANDOM FORESTS

CARTopt's sampling phase draws points from the low and high boxes in the partition on Ω. Hence, to implement CARTopt with the random forest partition, we need to draw observations from the random forest. Rather than sampling the high boxes directly, Robertson et al. (2013b) simplify the algorithm by sampling Ω instead. Sampling the high boxes (or Ω) is necessary to establish convergence and sampling the low boxes focuses the search where f is expected to be low. This simplified sampling approach is used in this article.

The low boxes in the random forest partition are sampled using a three-phase approach. First, randomly select one partition from the random forest. A low box is then randomly selected from the partition using selection probabilities proportional to the relative size of each low box. A point $\mathbf{x}$ is then randomly drawn from the selected box. If $\mathbf{x} \in \Omega$ and at least α partitions in the random forest classify $\mathbf{x}$ low, it is accepted. Otherwise the point is rejected. In either case, the method repeats until the required batch size is achieved. A sample drawn from the random forest partition in Figure 1(e) is shown Figure 1(f), where the search has been focused

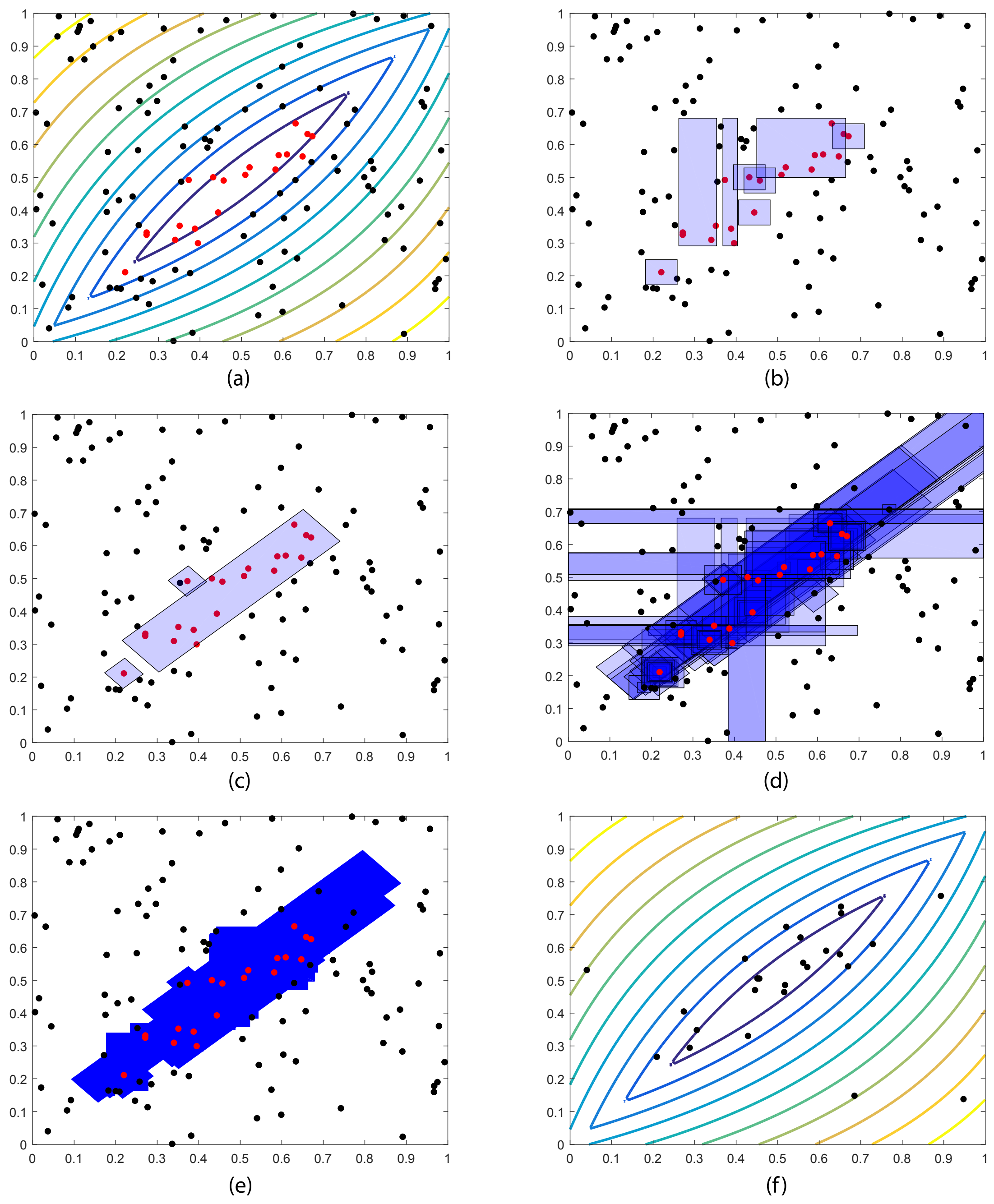

Figure 1. (a) Contours of $f = 2|2x_1 - 2x_2| + (2x_1 - 1)(2x_2 - 1)$ on $\Omega = [0, 1]^2$ and 125 training data points, where 20 points were classified low (red) and 105 points high (black). (b) Low boxes in the CARTopt partition on Ω using the training data. (c) Low boxes in the CARTopt partition on Ω using Householder reflected training data. (d) Low boxes in a random forest partition on Ω using $B = 20$ trees. (e) The intersection of low boxes from at least $\alpha = 3$ trees in the random forest. (f) A new CARTopt sample of 25 points, with 20 points drawn from the random forest partition in (e) and 5 points drawn from Ω.

where f was expected to be low.

The distribution of points drawn using the method above depends on the overlap of the low regions from each partition. If the low regions in each partition are similar, the points will be approximately uniformly distributed over the intersection of all low boxes (see Figure 1(f)). Otherwise, the point density will tend to be greater where the low regions have greatest overlap. The regions of greatest overlap are where the random forest is most confident that f is relatively low, so increasing the sample density here is advantageous.

5 DISCUSSION

In this article we provide an alternative partitioning strategy for the CARTopt algorithm. Rather than using a CART partition at each iteration, we suggest using a random forest partition. Pure CART partitions are known to over-fit training data and have low bias and high variance. A random forest partition has less variance than a CART partition and hence, provides a more stable approximation to where the objective function is expected to be low. Sample points can then be drawn from these low regions using the random forest. The random forest partition also simplifies the original CARTopt algorithm by removing a post-partition modification that required additional f evaluations. Convergence to an essential global minimizer of f can be demonstrated with small changes to the results of Robertson et al. (2013b). A preliminary simulation study showed that using random forests in the CARTopt algorithm was an effective strategy for solving a variety of global optimization test problems. The authors are currently refining the method and extending the set of test problems.

REFERENCES

Appel, M. J., R. Labarre, and D. Radulović (2003). On accelerated random search. *SIAM J. Optimization* (14), 708–731.

Audet, C. and J. E. Dennis (2003). Analysis of generalized pattern searches. *SIAM J. Optimization* (13), 889–903.

Bélisle, C. J., H. E. Romeijn, and R. L. Smith (1993). Hit-and-run algorithms for generating multivariate distributions. *Math. Oper. Res.* (18), 255–266.

Breiman, L. (2001). Random forests. *Machine Learning* (45), 5–32.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). *Classification and regression trees*. Monterey (CA): Wadsworth and Brooks.

Hastie, T., R. Tibshirani, and J. Friedman (2013). *The Elements of Statistical Learning, Data Mining, Inference, and Prediction*. New York: Springer.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press.

Horst, R. and T. Hoang (1996). *Global Optimization: Deterministic Approaches*. Berlin: Springer-Verlag.

Jones, D., C. D. Perttunen, and B. E. Stuckman (1993). Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Applications* (79), 157–181.

Kennedy, J. and R. Eberhart (1995). Particle swarm optimization. *In: Proceedings of the IEEE International Conference on Neural Networks. Piscataway*, 1942–1948.

Price, C. J., M. Reale, and B. L. Robertson (2012). A cover partitioning method for bound constrained global optimization. *Optimization Methods and Software* (27), 1059–1072.

Robertson, B. L., C. J. Price, and M. Reale (2013a). CARTopt: a random search method for nonsmooth unconstrained optimization. *Computational Optimization and Applications* (56), 291–315.

Robertson, B. L., C. J. Price, and M. Reale (2013b). A CARTopt method for bound-constrained global optimization. *ANZIAM J.* (55), 109–128.

Wickramarachchi, D. C., B. L. Robertson, M. Reale, C. J. Price, and J. A. Brown (2016). HHCART: An oblique decision tree. *Computational Statistics and Data Analysis* (96), 12–23.

Zabinsky, Z. B. (2009). Random search algorithms. *Department of Industrial and Systems Engineering, University of Washington, Seattle, WA*.