THE CARTOPTIMIZER PACKAGE VERSION 1.0 USER'S GUIDE

B. L. ROBERTSON, C. J. PRICE AND M. REALE

Department of Mathematics and Statistics
University of Canterbury
Private Bag 4800
Christchurch, New Zealand

Report Number: UCDMS2012 / 1 FEBRUARY 2012

The CARToptimizer Package version 1.0 User's Guide

B. L. Robertson, C. J. Price and M. Reale,
Department of Mathematics and Statistics,
University of Canterbury,
Private Bag 4800,
Christchurch, New Zealand.

Abstract

The CARToptimizer package is a suite of MATLAB functions for solving numerical optimization problems. The suite has algorithms for local, global and constrained optimization problems, where the objective function can be nonsmooth or even discontinuous. The underlying optimizer used in these functions is the random search CARTopt algorithm. This User's Guide describes the target class of problems, the CARTopt algorithm, the CARToptimizer package programme files, the files needed to solve a problem, and the graphical user interface (GUI), along with its parameter choices and menu options.

keywords: CART, random search, stochastic global optimization, nonsmooth optimization, nonlinear programming, optimization software.

1 Introduction

The CARToptimizer package is a MATLAB implementation of four different direct search CARTopt algorithms to solve a variety of nonsmooth optimization problems. These are:

(a) Local optimization with no constraints;

(b) Local optimization with bound constraints;

(c) Local optimization with general constraints; and

(d) Global optimization with bound constraints.

No derivatives need to be programmed (or even need to exist) to implement the CARTopt algorithms. These algorithms are a direct search methods that do not require any gradient information. The objective function to be minimized can be nonsmooth and may even be discontinuous. Nonsmooth constraint functions are also permitted. If the objective function is smooth, derivative based algorithms may be more efficient at solving local optimization problems than the CARTopt algorithms. However, although the package is designed for minimizing nonsmooth problems, it is still effective at minimizing smooth problems as well.

1.1 Problem (a) — Local optimization with no constraints

The unconstrained optimization problem is of the form

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

where a local minimizer is sought. The CARToptimizer package has two algorithms, the CARTopt algorithm [9] and the Hooke and Jeeves — CARTopt hybrid algorithm [8], to find approximate solutions to such problems. These algorithms can be shown to converge to an essential local minimizer of f under mild conditions [8, 9].

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

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

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

Loosely speaking $\mathfrak{E}(x_*, \epsilon)$ has Lebesgue measure zero means a randomly selected point satisfying $\|x - x_*\| < \epsilon$ has zero chance of also satisfying $f(x) < f(x_*)$. This does not mean $\mathfrak{E}(x_*, \epsilon)$ has to be empty, but its size is vanishly small compared to the set of all points within ϵ of x_*. 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.

In practice, the algorithm terminates at its final iterate, x_k, and the size of $\mathfrak{E}(x_*, \epsilon)$ is small compared the size of the set $\{x \in \mathbb{R}^n : \|x - x_k\| < \epsilon\}$. That is, there are more points with larger f values in satisfying $\|x - x_k\| < \epsilon$, than ones with values less than $f(x_k)$.

1.2 Problem (b): Local optimization with bound constraints

The CARTopt algorithm can also solve bound constrained local optimization problems. These problems restrict the search domain to a region, Ω, defined by an n-dimensional box of the form

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

where $l_i < u_i$ for all i. In this case, convergence to a feasible essential local minimizer can be demonstrated under mild conditions [7].

Definition 2. (Feasible essential local minimizer). *A point $x_* \in \Omega$ for which the set*

$$\mathfrak{E}(x_*, \epsilon) = \{x \in \Omega : f(x) < f(x_*) \text{ and } \|x - x_*\| < \epsilon\}$$

has Lebesgue measure zero for all sufficiently small positive ϵ is called a feasible essential local minimizer of f.

This is just an essential local minimizer on a restricted domain, where $\mathbb{R}^n$ is replaced with $\Omega \subset \mathbb{R}^n$.

1.3 Problem (c) — Local optimization with general constraints

The constrained optimization problem can be written as

$$\min_{x \in \mathbb{R}^n} f(x) \text{ such that } C(x) \leq 0,$$

where $C : \mathbb{R}^n \to (\mathbb{R} \cup \{+\infty\})^m$ are functions with $C = (c_1, c_2, \ldots, c_m)^\mathsf{T}$. The CARToptimizer package has the CARTopt filter algorithm [10] to solve these problems. This algorithm can be shown to converge to an essential local minimizer of the objective function, a constraint violation function, $\mathfrak{h}$, or a penalty function of the form

$$f(x) + \sigma_k \mathfrak{h}(x),$$

where σ_k is a penalty parameter [10].

These problems can also be solved using the CARTopt algorithm and the Hooke and Jeeves — CARTopt hybrid algorithm provided the closure of the interior of the feasible region is equal to itself. In this case the objective function, f, can be expressed as a barrier function of the form

$$\mathfrak{B}(x) = \begin{cases} f(x) & \text{if } x \text{ is feasible} \\ +\infty & \text{otherwise.} \end{cases}$$

It is not necessary to evaluate f at infeasible points, rather the barrier function assigns the value $+\infty$.

1.4 Problem (d) — Global optimization with bound constraints

The bound constrained global optimization problem is of the form

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

where the search region Ω is defined by an n-dimensional box given by (1). The global CARTopt algorithm [11] is included in the CARToptimizer package to find an approximate solution to bound constrained global optimization problems. Under mild conditions, convergence to an essential global minimizer of f can be demonstrated [11].

Definition 3. (Essential global minimizer). *An essential global minimizer, x_*, is a point for which the set*

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

has Lebesgue measure zero.

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

In practice, the algorithm terminates at its final iterate, x_k, and the size of $\mathfrak{E}(x_*)$ is small compared the size of Ω. That is, there are many more points with larger f values in Ω, than ones with values less than $f(x_k)$.

The remainder of the article is as follows. Section 2 briefly describes the CARTopt algorithms used in this package and gives references to academic articles for detailed descriptions. The installation and setup procedure is given in Section 3. The objective function M-file and constraint violation function M-file are defined in Sections 4 and 5, respectively. The graphical user interface (GUI), menu options, and algorithm parameters are explained in Section 6. The article concludes with a glossary of all the parameters used in the CARToptimizer package. The `typewriter` font is used hereafter to differentiate file names and GUI parameter names from the standard text.

2 CARTopt Algorithms

In this section the reader is given a basic overview of the CARTopt algorithms in the CARToptimizer package. It is not possible to go into great detail because it has taken several academic papers to fully describe the algorithms. The reader is referred to the academic papers cited throughout this article for full details.

2.1 CARTopt algorithm

The CARTopt algorithm [9] is a partitioning random search algorithm that alternates between partition and sampling phases. The algorithm generates a sequence of iterates $\{x_k\}_{k=1}^{\infty} \subset \mathbb{R}^n$, where each iterate x_{k+1} is the point with the lowest objective function value at the end of iteration k.

Initially a batch of points are drawn randomly from an optimization region, Ω. The batch of points are classified into two sets, points with relatively high f values, ω_H, and points with relatively low f values, ω_L. It is necessary that ω_H and $\omega_L \neq \emptyset$. The set of classified points, $T = \omega_L \cup \omega_H$, is called the training data set. The training data set is used to form a partition on Ω using classification and regression trees (CART) [1, 2].

The CART partition defines desirable subsets of Ω where f is presumed to be relatively low based on T. The partition itself consists of a set of non-empty hyper-rectangular subregions, A_i, such that $\cup_i A_i = \Omega$ and $A_i^o \cap A_j^o = \emptyset$ for all $i \neq j$, where A_i^o denotes the interior of A_i. Each A_i contains an element(s) from ω_L or ω_H, but not elements from both. The hyper-rectangular partition structure is aligned with the coordinate axes which makes it simple to draw points directly from the union of subregions containing low points. The union of these low subregions is called an approximate level set, $\mathcal{L}$ [7, 9].

For the next iteration a batch of points, X_k, is drawn from $\mathcal{L}$. The best points from $T_k \cup X_k$ are retained in T_{k+1} and the training data is reclassified into new sets, ω_L and ω_H. A new CART partition on Ω is formed with T_{k+1} and the method repeats. The method continues alternating between these sampling and partition phases, defining new subsets of Ω to explore, until stopping conditions are satisfied [7]. The best point, x_{k+1}, is taken as the approximation to an essential local minimizer of f.

2.2 Hooke and Jeeves — CARTopt hybrid algorithm

As the name suggests, the Hooke and Jeeves — CARTopt hybrid algorithm [8] is a hybrid algorithm that uses the Hooke and Jeeves algorithm [5] and the unconstrained CARTopt algorithm [9]. The algorithm generates a sequence of iterates $\{x_k\}_{k=1}^{\infty} \subset \mathbb{R}^n$, where each iterate x_{k+1} is the point with the lowest objective function value at the end of iteration k.

The hybrid algorithm alternates between the two methods. The first is an altered Hooke and Jeeves algorithm that uses mesh rotations, a scaled pattern move, and permits uphill steps to be taken under certain conditions [8]. Iterations of the altered Hooke and Jeeves algorithm are applied until no descent is forthcoming from the current iterate x_k. The unconstrained CARTopt algorithm is then applied in a neighborhood of x_k until a point, x_{k+1}, is found such that $f(x_{k+1}) < f(x_k)$. If such a point is found, the method reverts back to the Hooke and Jeeves phase. Otherwise, the CARTopt algorithm fails to reduce f below $f(x_k)$, confirming x_k is an essential local minimizer of f with probability one [8]. The method continues alternating between the two algorithms until stopping conditions are satisfied. The best point, x_{k+1}, is taken as the approximation to an essential local minimizer of f.

2.3 CARTopt filter algorithm

The CARTopt filter algorithm [10] generates two sequences of iterates; $\{x_k\}_{k=1}^{\infty} \subset \mathbb{R}^n$ and $\{z_k\}_{k=1}^{\infty} \subset \mathbb{R}^n$. Each iterate x_{k+1} is the point with the least constraint violation at the end of iteration k. Each iterate z_{k+1} is the point with the least

$$f(x) + \sigma_k \mathfrak{h}(x) \tag{2}$$

value at the end of iteration k, where σ_k is a penalty parameter and $\mathfrak{h}(x)$ is a constraint violation function [10] (see Section 5).

The CARTopt filter algorithm differs from the CARTopt algorithm in the way it defines its training data set. The training data is classified using a filter [10] so that some infeasible points are considered desirable. The resulting CART partition on $\mathbb{R}^n$ not only defines subsets where f is relatively low, but also promising subsets to explore to drive feasibility.

The method alternates between partition and sampling phases — updating the training data set at each iteration — until stopping conditions are satisfied. The algorithm then outputs the best point with the least constraint violation, x_{k+1}, and the best point that minimizes (2), z_{k+1}, along with their respective f and $\mathfrak{h}$ values.

2.4 Global CARTopt algorithm

The global CARTopt algorithm [11] is similar to the CARTopt algorithm described in Subsection 2.1. The algorithm generates a sequence of iterates $\{x_k\}_{k=1}^{\infty} \subset \Omega$, where the search region $\Omega = [0,1]^n$ is used. Each iterate x_{k+1} is the point with the lowest objective function value at the end of iteration k.

General hyper-rectangular search regions can be considered by scaling the independent variables in the objective function. Consider, for example, a search region in $\mathbb{R}^2$ defined by

$$-2 \leq x_1 \leq 5 \quad \text{and} \quad 1 \leq x_2 \leq 3. \tag{3}$$

Then the scaling, $x_1 \leftarrow 7x_1 - 2$ and $x_2 \leftarrow 2x_2 + 1$ computes the objective function in the search region defined by (3).

The global CARTopt algorithm forms a CART partition on Ω to define low subregions in the same way that the CARTopt algorithm does. However, the global algorithm sets Ω as a high region. Samples are then drawn from the high and low regions of the partition. Then the training data is updated and a new partition is formed and the method repeats. The method stops when the maximum number of function evaluations is reached and x_{k+1} is taken as an approximation to an essential global minimizer over the search region.

3 Installation and Setup

The CARToptimizer package files are stored in a zipped file, `CARToptimizer package.zip`, which is publicly available for download at the first author's website: `http://www.math.canterbury.ac.nz/php/people/~b.robertson/`. A brief description of each file is given below.

`Help`: This folder contains the files that are used for the help menu in each algorithm's graphical user interface (GUI).

`camel6.m`: This is an example of a smooth, bound constrained global optimization problem taken from Rinnooy Kan and Timmer [6] called the Six Hump Camel-back function of Brannin.

`CARTopt.m`: The CARTopt optimization algorithm.

`CARToptfilter.m`: The CARTopt filter optimization algorithm

`CARToptfiltertool.m`: The CARTopt filter algorithm's GUI.

`CARToptimizer.m`: The CARToptimizer package GUI.

`CARTopttool.m`: The CARTopt algorithm's GUI.

`constraint32.m`: This is an example of a squared 2-norm constraint violation function to handle the constraints in `testproblem32.m`.

`GlobalCARTopt.m`: The global CARTopt optimization algorithm

`GlobalCARTopttool.m`: The global CARTopt algorithm's GUI.

`HJ_CARTopt.m`: The Hooke and Jeeves — CARTopt hybrid optimization algorithm.

`HJ_CARTopttool.m`: The Hooke and Jeeves — CARTopt hybrid algorithm's GUI.

`rosenbrock.m`: This is a nonsmooth version of an unconstrained optimization problem taken from Morè et al. [12] called the Rosenbrock function. This problem is expressed as a sum of squares $\sum_i f_i^2$ with a global minimum of zero. Replacing the sum of squares with a sum of absolute values $\sum_i |f_i|$ gives the nonsmooth version, `rosenbrock.m`. The nonsmooth version shares the same global minimizer because $f_i = 0$ for all i at the solution (see Subsection 4.1).

`testproblem32.m`: This is a nonsmooth version of a constrained optimization problem taken from W. Hock and K. Schittkowski [4] called Test Problem 32. This problem is made nonsmooth in the same way that the Rosenbrock function was giving the nonsmooth version, `testproblem32.m` (see Subsection 5.1).

`The CARToptimizer User Guide.pdf`: This document, The CARToptimizer Package version 1 User's Guide.

All of these files need to be stored in the same directory, which must be added to the MATLAB path.

4 Objective Function File

To use the optimization algorithms in the CARToptimizer package an M-file, `myproblem.m`, that computes the objective function to be minimized needs to be created by the user. The M-file should accept a vector, whose length is equal to the number of independent variables for the objective function, and return a scalar. The M-file needs to be stored in the same directory as the CARToptimizer package. The first line of the function `myproblem.m` is of the form:

```
function [fcn] = myproblem(x),
function [fcn, x_scaled] = myproblem(x),
```

where

`fcn` = the objective function value;
`x` = $n \times 1$ vector of $\mathbb{R}^n$; and
`x_scaled` = the point where the objective was computed.

4.1 Example — Writing an objective function for local optimization

The following example shows how to write an M-file that computes the objective function to be minimized. Suppose we wish to minimize a nonsmooth version of Rosenbrock's function given by

$$f(x_1, x_2) = 10|x_2 - x_1^2| + |1 - x_2|. \tag{4}$$

The M-file, `rosenbrock.m`, that computes (4) must accept a vector $x \in \mathbb{R}^2$ and return the scalar value $f(x)$. The following lines of code are entered into `rosenbrock.m`:

```
function [fcn] = rosenbrock(x)
x1 = x(1);      x2 = x(2);
fcn = 10*abs(x2 - x1^2) + abs(1 - x2);
return
```

The M-file is then saved in the directory of the MATLAB path.

4.2 Example — Writing an objective function for global optimization

The following example shows how to write an M-file that computes the objective function to be minimized. Suppose we wish to find the global minimum of the Six Hump Camel-back function of Brannin given by

$$f(x_1, x_2) = (4 - 2x_1^2 + x_1^{4/3})x_1^2 + x_1x_2 - 4(1 - x_2^2)x_2^2, \tag{5}$$

over the search region

$$-3 \le x_1 \le 3 \quad \text{and} \quad -2 \le x_2 \le 2.$$

The bound constrained global optimization algorithm in the CARToptimizer package searches over $[0, 1]^n$. Thus, the independent variables for the objective function must be scaled when general bounded regions are considered. The M-file, `camel6.m`, that computes (5) must accept a vector $x \in [0, 1]^2$, scale the vector to match the search region, `x_scaled`, and return the scalar value $f(x)$ and `x_scaled`. The following lines of code are entered into `camel6.m`:

```
function [fcn, x_scaled] = camel6(x)
x1 = x(1);          x2 = x(2);
x1 = 6*x1 - 3;      x2 = 4*x2 - 2;
x_scaled = [x1, x2];
fcn = (4 - 2*x1^2 + x1^4/3)*x1^2 + x1*x2 - 4*(1 - x2^2)*x2^2
return
```

The M-file is then saved in the directory of the MATLAB path.

5 Constraint Violation Function File

A constraint violation function $\mathfrak{h} : \mathbb{R}^n \to \mathbb{R}^+ \cup \{\infty\}$ is used to handle constraints in the CARTopt filter algorithm. The other CARTopt algorithms do not require $\mathfrak{h}$. The constraint violation function is measure of how infeasible an iterate is and satisfies two simple conditions. Firstly, $\mathfrak{h}(x) \ge 0$ for all x. Secondly, $\mathfrak{h}(x) = 0$ if and only if x is a feasible point.

Consider, for example, an optimization problem with m constraints, $\{c_i(x)\}$, defining the feasible region such that $C(x) \leq 0$, where $C : \mathbb{R}^n \to (\mathbb{R} \cup \{\infty\})^m$ with $C = (c_1, c_2, \ldots, c_m)^\top$. Then, a constraint violation function of the form,

$$\mathfrak{h}(x) = \|[C(x)]_+\|,$$

where $\|.\|$ is a vector norm and $[y]_+ = \max\{y, 0\}$, satisfies the conditions stated above.

The constraint violation file, `myconstraint.m`, is the M-file that computes the constraint violation. The M-file should accept a vector, whose length is equal to the number of independent variables for the objective function, and return a non-negative scalar. This file needs to be stored in the same directory as `myproblem.m`. The first line of the function `myconstraint.m` is of the form:

```
function [hvalue] = myconstraint(x),
```

where

`hvalue` = the constraint violation value, and
`x` = $n \times 1$ vector of $\mathbb{R}^n$.

5.1 Example — Constraint violation function

The following example shows how to write an M-file that computes the constraint violation function at a point $x \in \mathbb{R}^n$. Suppose we wish to minimize a nonsmooth version of Test Problem 32 from W. Hock, and K. Schittkowski [4] given by

$$f(x_1, x_2, x_3) = |x_1 + 3x_2 + x_3| + 2|x_1 - x_2| - 1,$$

such that

$$\begin{aligned} x_1 + x_2 + x_3 - 1 &= 0; \\ x_1^3 - 6x_2 - 4x_3 + 3 &\leq 0; \\ -x_1 &\leq 0; \\ -x_2 &\leq 0; \\ -x_3 &\leq 0. \end{aligned}$$

The M-file, `constraint32.m`, that computes the constraint violation for Test Problem 32 must accept a vector $x \in \mathbb{R}^3$ and return a non-negative scalar value for the violation. Here we use the squared 2-norm constraint violation function [10]. The following lines of code are entered into `constraint32.m`:

```
function [hvalue] = constraint32(x)
x1 = x(1);      x2 = x(2);      x3 = x(3)
h1 = (x1 + x2 + x3 - 1)^2;
h2 = max(0, x1^3 - 6*x2 - 4*x3 + 3)^2;
h3 = max(0, -x1)^2;
h4 = max(0, -x2)^2;
h5 = max(0, -x3)^2;
hvalue = h1 + h2 + h3 + h4 + h5;
return
```

The M-file is then saved in the directory of the MATLAB path.

6 The CARToptimizer Package Display Screen and Menu Options

To launch the CARToptimizer package simply enter `CARToptimizer` into the MATLAB command window. This launches the algorithm selection GUI. From this GUI the user selects the algorithm they want to use to solve their particular problem. The package includes four different optimization algorithms:

1. `CARTopt algorithm`: Bound constrained and unconstrained local optimization;
2. `Hooke and Jeeves --- CARTopt hybrid algorithm`: Unconstrained local optimization;
3. `Global CARTopt algorithm`: Bound constrained global optimization;
4. `CARTopt filter algorithm`: Constrained local optimization.

To select a particular algorithm, the user clicks on the algorithm's red `Run` button. This launches the particular algorithm's GUI. To aid the user's decision, a brief description of each algorithm is given in the drop down `About` menu.

The remainder of this section describes the GUI for each algorithm. Section 7 is a glossary of all the parameters used in the CARToptimizer package.

6.1 GUI drop down menu

Each algorithm's GUI has a drop down menu with four options to help the user:

`Help`: Clicking `Help` displays a list of all the parameters in the algorithm's GUI. Clicking on one of the parameters in the list displays a brief description of that parameter.

`About`: Clicking `About` and clicking on a particular algorithm, displays a brief description of the selected algorithm and gives references to academic articles for further details.

`Example`: Gives an example of an optimization problem suitable for that particular algorithm. It also includes the MATLAB M-file that computes the objective function to be minimized and states the parameter values to be entered into the algorithm's GUI.

`User's Guide`: This gives the user a link to this document, The CARToptimizer Package version 1.0 User's Guide.

6.2 CARTopt algorithm GUI

The CARTopt algorithm is designed to find an essential local minimizer of an objective function. The algorithm can be implemented as a stochastic or deterministic method. If `Stochastic` is selected all the points are generated using MATLAB's random number generator. Otherwise the `Deterministic` is selected and all the points are generated using the Halton sequence [3].

The algorithm can also be implemented as an unconstrained or bound constrained algorithm. If `Unconstrained` is selected the search is conducted over $\mathbb{R}^n$. Otherwise `Bound Constrained` is selected and the search is conducted over

$$\texttt{Box centre} + \texttt{Box radius}[-1,1]^n. \tag{6}$$

General hyper-rectangular regions can be considered by choosing the unit hypercube and scaling the independent variables in the objective function M-file, `myproblem.m` (see Subsection 2.4 for details on scaling the independent variables).

The algorithm has three parameters to control the CART partition and sampling phase. To ensure the low subregions do not become numerically degenerate a lower bound, `Minimum subregion size`, is used on the each subregion's radius. This value needs to be finite and positive and ensures the Lebesgue measure of the low region is bounded above zero for all iterations. `Batch size` is the number of points that are used at each iteration. The effectiveness of random sampling to reduce the objective function becomes increasingly inefficient after approximately 40 points [9], so values less than 40 are recommended by the first author. Finally, `# low points` is the number of points that are classified as being low in the training data set. The default values of `Batch size` = 20 and `# low points` = 16 are chosen to promote clustering in the low points [9]. This allows the algorithm to focus down in promising regions.

The algorithm can terminate in two ways. Firstly, the algorithm halts if `Max # of f evaluations` is reached and secondly, if the probabilistic stopping rule is satisfied. The stopping rule terminates the algorithm if probability of reducing f below the current lowest value, f_k, is sufficiently small. That is, the algorithm halts if

$$Pr\Big(f_{k+1} < (f_k - \texttt{Accuracy in f})\Big) < \texttt{Probability of reducing f},$$

where f_{k+1} is a potential iterate such that $f_{k+1} < f_k$. The default values are `Accuracy in f` = 1e-8 and `Probability of reducing f` = 1e-6. If the algorithm halts because `Max # of f evaluations` is reached, the user should consider increasing this parameter so that the probabilistic stopping rule is used.

The algorithm outputs an approximation to an essential local minimum, `Minimum value`, the total number of objective function evaluations, `# f evaluations`, and an approximation to an essential local minimizer, `x value`. The user can also choose to output the objective function values after 50, 100, 200,... function evaluations in the command window by checking `Display f after 50, 100, 200,... evals`. The points generated by the algorithm can also be plotted (two dimensions only) by checking `Plot points generated`.

6.2.1 User defined parameters

The user is required to enter four parameter values; `Problem`, `Dimension`, `Box centre` and `Box radius`.

`Problem`: This is where the user enters the name of the M-file that computes the objective function to be minimized (see Section 4). The name is not entered as a string, just simply typed into the cell.

`Dimension`: The user enters the number of independent variables in the objective function to be minimized.

`Box centre`: The user must enter the centre of the initial hypercube search region. If `Bound Constrained` is selected, this will be the centre of the optimization region.

`Box radius`: The user must enter the radius of the initial hypercube search region. If `Bound Constrained` is selected, this will be the radius of the optimization region.

All the other parameters have default values. If the user changes a default value and enters an invalid value, the algorithm runs using the default value and informs the user why their choice was invalid in the MATLAB command window. Once all the parameters are entered, the user is required to click the red `Optimize` button to start the optimization.

6.3 Hooke and Jeeves — CARTopt algorithm GUI

The Hooke and Jeeves — CARTopt hybrid algorithm is designed to find an essential local minimizer of an objective function. This hybrid algorithm uses the unconstrained CARTopt algorithm and an altered Hooke and Jeeves algorithm. The reader is referred to Section 6.2 for details on the CARTopt algorithm's parameters. This subsection provides details on the altered Hooke and Jeeves algorithm's parameters.

The altered Hooke and Jeeves algorithm can be implemented in a way that generates a sequence of decreasing function values, or in a way that permits increases from time to time. A strictly decreasing sequence of f values is generated if `Downhill steps` is selected. Otherwise `Uphill steps` is selected and the sequence of f values is not necessarily decreasing. Allowing for uphill steps can be advantageous in nonsmooth optimization [8], potentially preventing the algorithm becoming trapped.

The altered Hooke and Jeeves algorithm has three parameters to control the mesh. The `Minimum mesh reduction` parameter puts a bound on how much the mesh can be reduced at each iteration. This value must be finite and greater than one. Choosing a value of two, for example, means the mesh can be reduced by a maximum factor of two at each reduction. The mesh size is also bounded below by `Minimum mesh size`. The `Pattern move scalar` allows the user to scale the pattern move to take larger steps across the mesh. This value must be an integer that is greater than or equal to one.

If the objective function is known to be smooth, the `Known to be smooth` box can be checked. If checked, a Hooke and Jeeves mesh reduction is made after each CARTopt phase. This potentially increases the rate of convergence on smooth optimization problems.

The hybrid algorithm can terminate in three different ways. Firstly, the algorithm halts if `Minimum mesh size` is reached. Secondly, if the probabilistic stopping rule in the CARTopt algorithm is satisfied. Finally, if `Maximum # of function evaluations` is reached. If the algorithm halts because `Max # of f evaluations` is reached, the user should consider increasing this parameter so that one of the other rules is used.

The algorithm outputs an approximation to an essential local minimum, `Minimum value`, the total number of objective function evaluations, `# f evaluations`, and an approximation to an essential local minimizer, `x value`.

6.3.1 User defined parameters

The user is required to enter four parameter values; `Problem`, `Dimension`, `Initial point` and `Initial mesh`.

`Problem`: This is where the user enters the name of the M-file that computes the objective function to be minimized (see Section 4). The name is not entered as a string, just simply typed into the cell.

`Dimension`: The user enters the number of independent variables in the objective function to be minimized.

`Initial point`: The user chooses an initial point $x \in \mathbb{R}^n$, which is a vector whose length is equal to the number of independent variables in the objective function. This point must also have a finite objective function value.

`Initial mesh`: The user chooses an initial Hooke and Jeeves mesh size. This value must be finite and greater than `Minimum mesh size`. If the user knows little about the objective function, choosing a large initial mesh size can be advantageous. This allows the algorithm to make larger steps early on, rather than a series of small ones.

All the other parameters have default values. If the user changes a default value and enters an invalid value, the algorithm runs using the default value and informs the user why their choice was invalid in the MATLAB command window. Once all the parameters are entered, the user is required to click the red `Optimize` button to start the optimization.

6.4 CARTopt filter algorithm GUI

The CARTopt filter algorithm is designed to solve constrained optimization problems. The CARTopt filter algorithm is similar to the CARTopt algorithm and shares many of its parameters. `Stochastic`, `Deterministic`, `Batch size`, `# low points`, `Maximum # of f evaluations`, `Minimum subregion size`, `Accuracy in f` and `Probability of reducing f` are all the same (see Section 6.2 for further details).

The CARTopt filter algorithm generates two sequences of iterates; points with least constraint violation ($\{x_k\}$ in Section 2.3) and points with the least

$$\texttt{Problem} + \sigma_k * \texttt{Constrain function} \tag{7}$$

value ($\{z_k\}$ in Section 2.3). The initial penalty parameter value, σ_k, is entered in the `Penalty parameter` cell, which must be finite and non-zero.

The user is required to select `Equality constraints` if the optimization problem has equality constraints present. This means the `Feasibility parameter` is used, which is finite and positive. This parameter defines any point with a constraint violation less than `Feasibility parameter` as feasible. This ensures there exists a feasible region with positive Lebesgue measure when equality constraints are present. Otherwise `No equality constraints` is selected and `Feasibility parameter` is zero.

The algorithm can terminate in two ways. Firstly, the algorithm halts if `Maximum # of f evaluations` is reached and secondly, if the probabilistic stopping rule is satisfied. If the algorithm halts because `Maximum # of f evaluations` is reached, the user should consider increasing this parameter so that probabilistic rule is used.

The algorithm outputs the total number of objective function evaluations, `Number of f evaluations`, and two sets of results. The first set displays the approximate solution with the least constraint violation. The second set outputs the approximate solution with the least (7) value. In both cases, `Minimum value` and `Constraint violation` are the objective

function value and the constraint violation, respectively, at their final iterates, `x value`. The user can also choose to plot the points generated by the algorithm (two dimensions only) by checking `Plot points generated`.

6.4.1 User defined parameters

The user is required to enter five parameter values; `Problem`, `Constraint function`, `Dimension`, `Box centre` and `Box radius`.

`Problem`: This is where the user enters the name of the M-file that computes the objective function to be minimized (see Section 4). The name is not entered as a string, just simply typed into the cell.

`Constraint function`: This is where the user enters the name of the M-file that computes the constraint violation function (see Section 5). The name is not entered as a string, just simply typed into the cell.

`Dimension`: The user enters the number of independent variables in the objective function to be minimized.

`Box centre`: The user must enter the centre of the initial hypercube search region.

`Box radius`: The user must enter the radius of the initial hypercube search region. If not much is known about the objective function, it can be advantageous to choose a large radius. This allows the algorithm to make large steps early on, rather than a series of small ones.

All the other parameters have default values. If the user changes a default value and enters an invalid value, the algorithm runs using the default value and informs the user why their choice was invalid in the MATLAB command window. Once all the parameters are entered, the user is required to click the red `Optimize` button to start the optimization.

6.5 Global CARTopt algorithm GUI

The global CARTopt algorithm is designed to find an essential global minimizer in a bound constrained search region, Ω. The algorithm partitions Ω into low subregions and sets Ω as the high region. The user selects `Random` to sample the high region using MATLAB's random number generator. Otherwise `Halton` is selected and the high region is sampled using the deterministic Halton sequence.

The algorithm has seven parameters that control its implementation. `Batch`, `# low points` and `Minimum subregion size` are the same as those described for CARTopt (see Section 6.2). The `Fraction of samples in high region` parameter determines how much computational effort is performed in the high region. This value must be between zero and one. Choosing a small value (e.g. 0.2) focuses the search effort locally and choosing a large value (e.g. 0.8) focuses the search effort globally. `Maximum training data size` determines how many points are retained by the algorithm to form its training data set. Choosing this value greater than `Max # f evals` means all the points will be retained.

The algorithm also has two restart features that attempt to prevent the algorithm becoming trapped at a local solution. Firstly, the algorithm restarts if the minimum 1-norm radius of a low subregion is less than `1norm restart`. Secondly, a restarts happens if the

minimum subregion diameter is less than `-inf norm restart`. In both cases the algorithm restarts using the elements from the training data set that were drawn from the high region and the best point as an initial training data set.

The algorithm terminates when `Max # f evals` is reached and outputs an approximation to the essential global minimum, `Minimum value`, the total number of objective function evaluations, `# f evaluations`, and an approximation to an essential global minimizer, `x value`. The user can also choose to output the objective function values after 50, 100, 200,... function evaluations in the command window by checking `Display f after 50, 100, 200,... evals`. The points generated by the algorithm can also be plotted (two dimensions only) by checking `Plot points generated`.

6.5.1 User defined parameters

The user is required to enter two parameter values; `Problem` and `Dimension`, and it is recommended that the user chooses `Max # f evals` because this is the stopping rule.

`Problem`: This is where the user enters the name of the M-file that computes the objective function to be minimized (see Section 4.2). The name is not entered as a string, just simply typed into the cell.

`Dimension`: The user enters the number of independent variables in the objective function to be minimized.

All the other parameters have default values. If the user changes a default value and enters an invalid value, the algorithm runs using the default value and informs the user why their choice was invalid in the MATLAB command window. Once all the parameters are entered, the user is required to click the red `Optimize` button to start the optimization.

7 Glossary of GUI Parameters

This section is a glossary of all the parameters used in the CARToptimizer package.

`# low points`: This is the number of points that are classified as having low objective function values in the training data set. This integer value is greater than zero and less than or equal to `batch size`.

`# f evaluations`: This is the total number of objective function evaluations required to give the approximate solution to the problem.

`1norm restart`: This is the minimum 1-norm restart radius parameter. If the minimum 1-norm radius over all the low subregions of the CART partition is less than `1norm restart`, the algorithm restarts. At each restart, a new CART partition is formed using the best point and the points drawn from the high region as an input training data set. If this restart is used, `1norm restart` is replaced with `(1norm restart)^3/2` for the next iteration. This allows the algorithm to focus down on promising subsets of the search region.

`Accuracy in f`: This parameter specifies the required accuracy in the final objective function value. If f_* is the essential local minimum the algorithm is converging to, then

$$|\text{ best objective function value } - f_*| = \texttt{Accuracy in f}.$$

This value must be positive and finite.

`Batch size`: This is the number of points that are used at each iteration. This integer value must be finite and greater than two.

`Bound Constrained`: This implements the bound constrained version of the algorithm.

`Box centre`: This is the centre of the initial search region for the algorithm.

`Box radius`: This is the radius of the initial search region for the algorithm.

`Constraint function`: This is the name of the M-file that is used to compute the constraint violation function.

`Constraint violation`: This is the value of the constraint violation function at the final iterate, `x value`. If `Constraint violation` is equal to zero, then `x value` is a feasible point.

`Deterministic`: This implements the deterministic version of the algorithm using the deterministic Halton sequence.

`Dimension`: This is the number of independent variables in the objective function to be minimized.

`Display f after 50, 100, 200,... evals`: This displays the best objective function value after 50, 100, 200,... function evaluations, in the MATLAB command window.

`Downhill steps`: If `Downhill steps` is selected, the Hooke and Jeeves algorithm generates a sequence of iterates with decreasing objective function values. This is how the classical Hooke and Jeeves algorithm operates.

`Equality constraints`: This box is selected if there are equality constraints present in the particular optimization problem. The algorithm uses a `Feasibility parameter` if equality constraints are present, otherwise the `Feasibility parameter` is zero.

`f known to be smooth`: This box is checked if the objective function is known to be smooth. This forces a Hooke and Jeeves mesh reduction after each CARTopt phase and potentially increases the rate of convergence. If the objective function is not known to be smooth, the mesh will not necessarily be reduced after each CARTopt phase. This allows the Hooke and Jeeves algorithm to search on the same mesh size if sufficient descent was found by the CARTopt algorithm.

`Feasibility parameter`: This parameter is only used if `Equality constraints` is selected and must be finite and positive. A point is considered feasible if its `Constraint violation` value is less than `Feasibility parameter`. This parameter ensures there exists a feasible region with positive Lebesgue measure.

`Fraction of samples in high region`: This is the fraction of samples to be drawn from the high region (search region Ω) at each iteration. This value must be between zero and one. Choosing a small value (e.g. 0.2) focuses the search effort locally. Choosing a large value (e.g. 0.8) focuses the search effort globally.

`Halton`: If `Halton` is selected, the high region (search region Ω) is sampled using the deterministic Halton sequence.

`-inf norm restart`: This is the minimum subregion diameter restart parameter. If the minimum subregion diameter over all the low subregions of the CART partition is less than `-inf norm restart`, the algorithm restarts. This indicates a dimension on a particular subregion has become numerically degenerate. At each restart, a new CART partition is formed using the best point and the points drawn from the high region (search region Ω) as an input training data set.

`Initial point`: This is the initial point from which the algorithm starts its search for an essential local minimizer. It is required that the initial point has a finite objective function value.

`Initial mesh`: This is the initial Hooke and Jeeves mesh size. This value is required to be finite and greater than `Minimum mesh size`.

`Max # f evals`: This is the maximum number of objective function evaluations. Once reached, the algorithm terminates and the point with the lowest function value is taken as an approximation to the essential global minimum. This parameter must be finite and non-zero.

`Maximum # of f evaluations`: This is the maximum number of objective function evaluations. The algorithm terminates if the number of objective function evaluations exceeds `Maximum # of f evaluations` and outputs its final values. This parameter must be finite and non-zero.

`Maximum mesh reduction`: This parameter specifies the maximum reduction factor in the Hooke and Jeeves mesh update. For example, setting a value of two means the mesh cannot be reduction by more than a factor of two at each reduction. This value must be finite and greater than one.

`Maximum training data size`: This is the maximum number of points that are retained in the training data set. The training data set consists of low and high points, and is used to form the CART partition on the search region Ω. This value must be finite and greater than or equal to twice the `Batch size`. Choosing `Maximum training data size` greater than `Max # f evals` means that all the points generated by the algorithm will be retained in the training data set.

`Minimum f value` The objective function value at the final iterate.

`Minimum mesh size`: This is the minimum Hooke and Jeeves mesh size. This value is required to be non-zero and less than `Initial mesh`. If `Minimum mesh size` is reached, the Hooke and Jeeves — CARTopt hybrid algorithm terminates and `x value` is an approximation to an essential local minimizer of the objective function. Note: The Hooke and Jeeves — CARTopt hybrid algorithm can also terminate if CARTopt's stopping conditions are satisfied or if `Maximum # of f evaluations` is reached.

`Minimum subregion size`: This is the minimum subregion radius for each low subregion of the CART partition. At each iteration, CART partitions $\mathbb{R}^n$ into a set of hyper-rectangular low subregions. If the radius of a particular low subregion is less

than `Minimum subregion size`, the subregion bounds are extended — post-partition — until each low subregion radius is sufficiently large.

`Minimum value`: The objective function value at the final iterate.

`No equality constraints`: This box is selected if there are no equality constraints in the particular optimization problem. The algorithm uses a `Feasibility parameter` if equality constraints are present, otherwise the `Feasibility parameter` is zero.

`Number of f evaluations`: This is the total number of objective function evaluations required to give the approximate solution to the problem.

`Optimize`: This is the button that runs the optimizer. Before the algorithm can run successfully, the user required parameters must be entered.

`Pattern move scalar`: This is a positive integer that is used to scale the pattern move in the Hooke and Jeeves algorithm. Choosing values larger than one allows larger pattern moves to be taken.

`Penalty parameter`: This is the initial penalty parameter, σ_k, used to generate the sequence of iterates that minimize the penalty function

$$\texttt{Problem} + \sigma_k * \texttt{Constraint function}.$$

This value must be positive and finite.

`Plot points generated`: This box is checked if the user wants to plot all the points generated by the algorithm. Points classified as low are shown in red and the remaining points in black. When this option is selected, the algorithm pauses temporarily between iterations so the user can see where the new points are generated. This option only works when the dimension of the objective function is equal to two.

`Probability of reducing f`: This is the probability of reducing the best objective function value below

$$\text{best objective function value} - \texttt{Accuracy in f}. \tag{8}$$

If the probability of reducing the objective function below (8) is less than `Probability of reducing f`, the CARTopt algorithm terminates.

`Problem`: This is the name of the M-file that is used to compute the objective function to be minimized.

`Random`: If `Random` is selected, the high region (search region Ω) is sampled using MATLAB's random number generator.

`Run`: This is the button that selects a particular optimization algorithm.

`Stochastic`: This implements the stochastic version of the algorithm using MATLAB's random number generator.

`Unconstrained`: This implements the unconstrained version the algorithm.

`Uphill steps`: If `Uphill steps` is selected, the Hooke and Jeeves algorithm generates a sequence of iterates with not necessarily decreasing objective function values. Allowing for uphill steps can prevent the algorithm from becoming trapped.

`x value`: Final iterate generated by the algorithm.

References

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

[2] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern classification*, Wiley-Interscience, New York, 2001.

[3] J. H. Halton, *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals*, Numerische Mathematik, **2** (1960), pp.84–90.

[4] W. Hock and K. Schittkowski, *Test examples for nonlinear programming codes*, Lecture Notes in Economics and Mathematical Systems (**187**), Springer-Verlag, 1981.

[5] R. Hooke and T. A. Jeeves, *Direct search solution of numerical and statistical problems*, Journal of the Association for Computing Machinery, **8** (1961), pp. 212–229.

[6] A. H. G. Rinnooy Kan and G. T. Timmer, *Stochastic global optimization methods part II: multi-level methods*, Mathematical Programming, **39** (1987), pp. 57–78.

[7] B. L. Robertson, *Direct search methods for nonsmooth problems using global optimization techniques*, PhD thesis, University of Canterbury, Christchurch, New Zealand, 2010.

[8] B. L. Robertson, C. J. Price and M. Reale, *A Hooke and Jeeves — CARTopt hybrid method for nonsmooth optimization*, Advanced Modeling and Optimization 13 (3), 2011, pp.381-401.

[9] B. L. Robertson, C. J. Price and M. Reale, *CARTopt: A random search method for nonsmooth unconstrained optimization*, submitted to COAP, 2011.

[10] B. L. Robertson, C. J. Price and M. Reale, *A CARTopt filter method for nonsmooth constrained optimization*, submitted to COAP, 2011.

[11] B. L. Robertson, C. J. Price and M. Reale, *A CARTopt method for bound constrained global optimization*, submitted, 2012.

[12] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Transactions on Mathematical Software, **7** (1981), pp.17–41.