A hybrid Hooke and Jeeves—Direct method for non-smooth optimization.

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

Abstract

A direct search optimization method for finding local optima of non-smooth unconstrained optimization problems is described. The method is a combination of that of Hooke and Jeeves, and the global optimization algorithm DIRECT of Jones, Perttunen, and Stuckman. The method performs modified iterations of Hooke and Jeeves until no further progress is forthcoming. A modified form of the DIRECT algorithm is then applied in a neighbourhood of the current iterate in order to make further progress. Once such progress has been obtained the method reverts to that of Hooke and Jeeves. The method is applicable to non-smooth and some discontinuous problems. On a discontinuous function it is guaranteed to find a point which is minimal on a dense subset of a region containing that point under mild conditions. If the function is continuous at that point, then it is a local minimizer. Additionally, the method is able to determine empirically if the objective function is partially separable. If so, partial separability is exploited by selecting the order in which the Hooke and Jeeves and DIRECT subalgorithms poll each dimension. Numerical results show that the method is effective in practice.

keywords: direct search, Hooke and Jeeves, non-smooth, numerical results, Jones, Perttunen, and Stuckman.

1 Introduction

In this paper we are interested in direct search methods for solving the unconstrained optimization problem

$$\min_x f(x) \tag{1}$$

where the objective function f maps R^n into $R \cup \{+\infty\}$. Our interest lies particularly with objective functions which are not smooth or discontinuous. Examples of such functions are exact penalty functions and functions with deterministic noise of, for example, numerical origin. The inclusion of $+\infty$ as a possible value for f means that the algorithm can be applied to extreme barrier functions and functions which are not defined everywhere by assigning f the value $+\infty$ in regions where it is not otherwise defined [1].

Direct search methods for (1) were popular in the 1960's, but then fell out of favour with researchers [17] until the late 1990's. In 1997, Torczon [16] provided a general convergence theory for Generalized Pattern Search (GPS) methods. These are direct search methods which employ nested sequences of meshes to solve (1) when f is C^1. The theory behind GPS methods has since been extended in a variety of ways. Audet and Dennis [1] showed that convergence is still guaranteed provided f is strictly differentiable only at each cluster point of the sequence of iterates. Price and Coope [13] extended this to the case when the grids are no longer nested. Audet and Dennis [2] further extended GPS to yield a method called Mesh Adaptive Direct Search (MADS) which has partial convergence results when f is non-smooth or discontinuous. These results are partial in the sense that they guarantee non-negativity of the Clarke derivative [4] in all relevant directions at each cluster point of the sequence of iterates, but do not guarantee non-existence of descent directions at these cluster points.

The Clarke derivative of a locally Lipschitz function f at x in the direction v is

$$f^\circ(x,v) = \limsup_{y\to x, t\downarrow 0} \frac{f(y+tv)-f(y)}{t}.$$

Consider a directionally differentiable function f. Using $y \equiv x$ it is clear that $f^\circ(x,v) < 0$ implies the directional derivative $D_v f(x)$ along v at x is also negative. Hence $f^\circ(x,v) \geq 0$ is a necessary, but not sufficient condition for x to be a local minimizer of f. An illustrative example is the function

$$\Psi = \begin{cases} 3(2|x_2| - x_1) + (0.9 + \sqrt{5}/2)x_1 & x_1 \geq 2|x_2| \\ 0.9x_1 + \sqrt{x_1^2 + x_2^2} & \text{otherwise.} \end{cases}$$

This function has a cone of descent directions at the origin centred on the direction $e_1 = (1,0)^T$, as is shown in Figure 1. Nevertheless the Clarke derivative $\Psi^\circ(0,v)$ is positive for all non-zero directions v. The directional derivative at the origin is positive for all $v \neq 0$ pointing into the region where $x_1 \leq 2|x_2|$. For the remaining directions we have $v_1 > 2|v_2|$, with $v_1 > 0$. Evaluating the directional derivative $D_v\Psi$ of Ψ at a point $(0,x_2)$ with $x_2 > 0$ we get $0.9v_1 + v_2$, which is positive for all $v_1 > 2|v_2|$.

It has been shown [14] that finding a descent step for a nonsmooth optimization problem is closely linked to solving a global optimization problem. We replace the Clarke derivative approach with one using the global optimization algorithm DIRECT of Jones et al [11]. Our method alternates between two modes of operation. Preferentially it executes iterations of a modified Hooke and Jeeves [10] algorithm until descent is no longer forthcoming. At this point standard Hooke and Jeeves reduces the stepsize. Instead our algorithm (hereafter HJDIRECT) applies DIRECT until a lower point is found, then reverts to the altered Hooke and Jeeves. This repeated restarting of DIRECT is an effective strategy [3].

A partially separable function [9] is a function of the form

$$f(x) = \sum f_i(x)$$

where each element f_i depends only on a small number of the elements in x. There are several ways to exploit partial separability in optimization. For example [5, 6] use it to minimize the effort required to form an approximation to the Hessian. A somewhat different approach occurs in [15], which exploits partial separability by

(a) calculating function values more cheaply by altering one variable at a time and only recalculating function elements containing that variable; and

(b) combining changes from several variables, no two of which appear in the same element, to get extra function values for free.

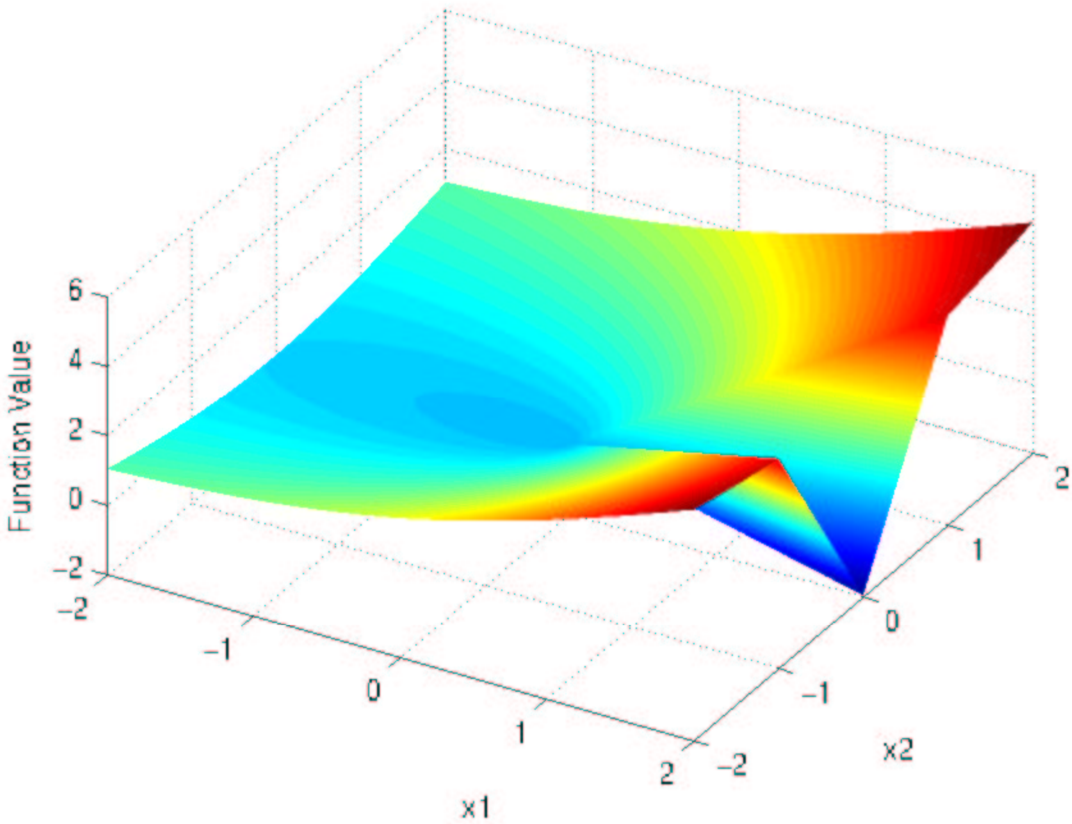

Figure 1: Graph of a two dimensional function Ψ with negative directional derivative at the origin along $e_1 = (1, 0)^T$. The Clarke derivative Ψ° is positive for all directions at $x = 0$.

Strategy (a) requires explicit knowledge of the partial separability structure, and access to the individual function elements. We use strategy (a) only in generating some numerical results, so that comparison with the results in [15] is fair. For strategy (b), a set of decision variables is called non-interacting if at most one of those variables appears in any single element f_i [15]. This means changes to these variables are independent of one another. So if one looks at changes to each of a set of non-interacting variables in turn, and accepts each change which reduces f, one automatically gets the combination of these changes which gives the smallest possible f value. Hooke and Jeeves does this, which means HJDIRECT is capable of exploiting partial separability structure provided it can order the elements of x so that sets of non-interacting variables are searched contiguously. Since partial separability information is not assumed to be available, this must be estimated. To do this Hooke and Jeeves is modified so that function values are calculated for a 'square' of four points lying in the plane of two consecutively polled coordinate directions. The fourth point of each square is an additional function evaluation, and if the two relevant variables do not interact this fourth point is valueless. This presents two possibilities: one could order for minimal interaction between consecutive variables, and suffer the cost of the worthless fourth points; or order for maximum interaction and try and extract as much value out of the fourth points as possible. Both options are explored. This estimation process is useful even when partial separability is only approximate or local.

The main part of our algorithm is described in detail in the next section. Section 3 describes the modified DIRECT subalgorithm. Convergence results are given in Section 4. These apply when f either belongs to a class of non-smooth functions, or is strictly differentiable at point(s) the algorithm identifies as solution(s) to (1). Numerical results are presented in Section 5, and concluding remarks are in Section 6.

2 The Algorithm

The algorithm searches over a succession of grids, where the m^{th} grid $\mathcal{G}_m$ is defined by one point (y_m) on it and the grid size h_m:

$$\mathcal{G}_m = \left\{ y_m + h_m \sum_{i=1}^{n} \eta_i e_i : \eta_1, \ldots, \eta_m \text{ integer} \right\}.$$

Here e_i is the i^{th} column of the identity matrix. The algorithm uses Hooke and Jeeves iterations to search over $\mathcal{G}_m$ for a point z_m satisfying

$$f(z_m \pm h_m e_i) \geq f(z_m) \quad \forall i = 1, \ldots, n. \tag{2}$$

Any point satisfying (2) is called a grid local minimizer of $\mathcal{G}_m$ [7]. For the first grid $y_1 = x_0$. For subsequent grids, $y_m = z_{m-1}$. A precise statement of the algorithm is given in Figure 2.

The algorithm consists of an initialization phase (step 1) and two loops. Step 1 selects an initial point x_0, and sets the initial Hooke and Jeeves pattern move v to zero. It also initializes the iteration and grid counters, k and m.

An iteration of the outer loop (steps 2–6) begins by calling the inner loop (steps 2–4). The inner loop performs iterations of the modified Hooke and Jeeves method on $\mathcal{G}_m$ until it is not able to make further progress in reducing f. The inner loop then halts with best known point z_m. A modified DIRECT algorithm is applied in step 5. DIRECT searches over

1. `Initialize: Set` $k = 0$ `and` $m = 1$. `Choose` x_0 `and` $h_0 > 0$. `Set` $v_0 = 0$.
2. `Calculate` $f(x_k + v_k)$, `and form the modified Hooke and Jeeves exploratory step` E_k `from` $x_k + v_k$.
3. `Pattern move: If` $f(x_k + v_k + E_k) < f_k$ `then`
   (a) `set` $x_{k+1} = x_k + v_k + E_k$ `and` $v_{k+1} = v_k + E_k$.
   (b) `Conduct an Armijo linesearch along the ray` $x_{k+1} + \alpha v_{k+1}$, $\alpha > 0$.
   (c) `Increment` k `and go to step 2.`
4. `If` $v_k \neq 0$ `set` $v_k = 0$ `and go to step 2.`
5. `Set` $z_m = x_k$. `Execute the modified DIRECT method about` z_m `until a point` x_{k+1} `lower than` z_m `is found or stopping conditions are satisfied. Select` h_{m+1} `and set` $v_{k+1} = x_{k+1} - z_m$.
6. `If stopping conditions do not hold, increment` m `and` k, `and goto step 2.`

Figure 2: The main Hooke and Jeeves—DIRECT algorithm HJDIRECT.

a hypercube shaped box centred on $z_m = x_k$, with edge length $2h_{\rm d}$. It searches for a point lower than z_m, and halts when successful or if the maximum number of function evaluations has been reached. If successful, this lower point becomes x_{k+1}, and h_{m+1} is chosen so that z_m and x_{k+1} both lie on $\mathcal{G}_{m+1}$. This permits $v_{k+1} = x_{k+1} - z_m$ to be used. A new outer loop iteration is started at step 2 unless the stopping conditions are satisfied. These halt the algorithm when h_m falls below a minimum grid size $H_{\min}$. Additionally the algorithm stops if an upper limit $N_{\max}$ on the number of function evaluations is exceeded.

In the inner loop, step 2 performs a modified Hooke and Jeeves exploratory phase about the point $x_k + v_k$, yielding the exploratory step E_k. This exploratory phase has $n - 1$ more points than that of standard Hooke and Jeeves. These extra points are designed to extract partial separability information about the function. The exploratory phase is described in detail in subsection 2.1.

Step 3 is executed if $x_k + v_k + E_k$ is lower than x_k. This step performs an Armijo ray search (see subsection 2.2) along the ray $x_{k+1} + \alpha v_{k+1}$, $\alpha \geq 0$, and then a new iteration of the inner loop is started at step 2. Otherwise (step 4), if v_k is non-zero, the algorithm sets v to zero and performs exploratory step about x_k by going to step 2. Finally, if $v_k = 0$, then the algorithm has located a grid local minimizer, the inner loop terminates, and the method proceeds to step 5.

2.1 The exploratory phase

The method performs a modified version of the exploratory phase of Hooke and Jeeves. There are three differences between the modified and original forms of this phase. First, the order in which the decision variables are perturbed is changed each iteration in order to best exploit the known levels of interactions between the decision variables. Second, extra function values are used in order to estimate the interactions between the decision variables. Third, a decision variable x_i is decremented first if the last successful change to x_i was a decrement.

2.1.1 Interaction

The measure of interaction between two decision variables x_i and x_j is calculated using four function values $f_a, \ldots, f_d$ at points x_a, $x_b = x_a + he_i$, $x_c = x_a + he_j$ and $x_d = x_a + he_i + he_j$. If the variables do not interact then the changes in f due to altering x_i and x_j are independent of one another. This implies $f_a + f_d = f_b + f_c$. The interaction H_{ij} measures the degree of departure from this condition, and is given by

$$H_{ij} = \frac{|f_a + f_d - f_b - f_c|}{\epsilon + \max\{f_a, f_b, f_c, f_d\} - \min\{f_a, f_b, f_c, f_d\}}$$

where ϵ is a small positive constant used to avoid divide by zero problems. It also prevents round-off error producing spurious interaction values when the four function values are very nearly equal. Clearly

$$|f_a + f_d - f_b - f_c| \leq 2\left(\max\{f_a, f_b, f_c, f_d\} - \min\{f_a, f_b, f_c, f_d\}\right),$$

and so $H_{ij} \in [0, 2)$. The interactions are stored in a matrix H, with the convention that $H_{ii} = 2$ for all i.

The decision variables are re-ordered to either maximize or minimize the interaction between consecutive variables. The intentions behind these two choices are quite different, but the processes are quite similar. Interactions are generated for pairs of variables x_i x_j which are polled consecutively during the exploratory phase. Three of the values $f_a, \ldots, f_d$ are generated by standard Hooke and Jeeves exploratory search. Our exploratory search also calculates the fourth. The order in which the decision variables are polled changes each iteration in order to obtain as many H_{ij} values as possible, and to take advantage of known interaction information. These two aims are reasonably compatible. For example if one wishes to maximize interaction, then initially $H_{ij} = 2$ is used for all distinct i and j. At each iteration the variables are placed in an order that maximizes interaction between consecutive variables. This yields the interactions for each such consecutive pair. If the calculated H_{ij} is small, then x_i and x_j will not be consecutive in the following iteration.

2.1.2 Maximizing Interaction

When interaction is maximized the aim is to get maximum value out of the extra point used to estimate the interaction between consecutive variables. There is no advantage in maximizing the interaction between larger groups of consecutive variables, and so this is not done. A greedy algorithm is used to determine the order of the decision variables. An ordered list of decision variables is formed, and the exploratory phase polls these variables in that order. The list is initialized as $\{x_\ell\}$, where $\ell = (k \bmod n) + 1$. The algorithm then finds the unlisted variable with the largest interaction with the last member on the list. This new variable is placed on the end of the list. The process is repeated until all variables have been listed. All interactions are initially set at 2, which ensures that all possible pairs of variables are eventually used.

1. `Set` $G = H$.
2. `Let` $\ell = (k \bmod n) + 1$ `and set` $s_1 = \ell$.
3. `For` $i = 2$ `to` n `do`
   (a) `Let` $j \notin \{s_1 \dots s_{i-1}\}$ `be the smallest value which minimizes` $G_{\ell j}$. `Set` $s_i = j$.
   (b) `If` $G_{\ell j} \leq \tau$ `set` $G_{\ell r} = \max\{G_{\ell r}, G_{jr}\}$ `for` $r = 1$ `to` n. `Otherwise set` $\ell = j$.

Figure 3: The subalgorithm for selecting a minimally interacting order $\{s_1, \dots, s_n\}$ for the decision variables.

2.1.3 Minimizing Interaction

Here the aim is to exploit the (local or global) partial separability of f and get extra function values for free [15]. In this case there are definite advantages to grouping as many non-interacting variables together as possible. Once again the algorithm selects x_ℓ first, with ℓ as before. This forms the start of the first group of variables. The variable which minimizes the maximum of its interactions with variables already in the group is selected next. If this maximum is less than a preset value τ then this variable is added to the first group, and the process is repeated. Otherwise this variable is considered to be the first in a new group. In either case this selected variable becomes the next in the list of variables after x_ℓ. The process is repeated until all variables have been placed in the list. A more precise statement is given in the algorithm listed in Figure 3. When the interactions are being estimated all interactions are initially set to zero. This ensures the method will consider a large number of possible consecutive pairs of variables.

2.2 The ray search

A standard forward-tracking ray search was used. The ray search looked along the ray $x_{k+1} + \alpha v_{k+1}$, and considered values $\alpha = 1, 2, 4, 8, \dots, \alpha_{\max}$, accepting the largest value for which the corresponding sequence of function values is strictly decreasing. In the implementation, $\alpha_{\max}$ was chosen as the smallest power of 2 greater than 10^6. Any integer value greater than 1 is acceptable for $\alpha_{\max}$. The ray search was included as it was found to significantly improve the algorithm's performance.

3 The Direct subalgorithm

The DIRECT algorithm of Jones, Perttunen, and Stuckman [11] is designed to locate a global minimizer of a piecewise smooth function on an n dimensional interval (or box) Ω of the form

$$\Omega = \{x \in R^n : \ell \leq x \leq u\}$$

where $\ell, u \in R^n$ are finite and satisfy $\ell < u$. At each iteration DIRECT has a list of boxes which cover Ω. The function values at the centre points of these boxes are known. Using these function values and the relative sizes of the boxes, some boxes are selected for subdivision. Once subdivided the function values at the centre points of each new box are calculated unless they are already known. Stopping conditions are then checked, and if not satisfied, a new iteration is begun.

The standard form of DIRECT differs substantially from ours in how boxes are selected for subdivision, subdivided, and in the stopping criteria, and so the modified version of DIRECT is discussed from now on. The subdivision process is straightforward. Let $B = \{x : a \le x \le b\}$ be a box selected for subdivision. Let i be a value for which the length of the i^{th} box edge $e_i^T(b-a)$ is maximal. The box B is subdivided into three identically shaped boxes using the two cutting planes $e_i^T(x-(2a+b)/3) = 0$ and $e_i^T(x-(a+2b)/3) = 0$. When more than one edge has maximal length, the edge along which subdivision occurs is selected as follows. If the variables have been ordered for maximum interaction, the first edge in that order which is of maximal length is chosen. Otherwise the variables are placed in the order $\rho, \ldots n, 1, \ldots, \rho - 1$ and the first maximal edge is chosen, where ρ is half the current number of boxes. This choice avoids favouring lower numbered dimensions by not always exploring them first.

The DIRECT subalgorithm searches over the region $z_m + h_{\text{d}}[-1,1]^n$ for a point x_{d}, where $f(x_{\text{d}}) < f(z_m)$. Once successful, DIRECT halts and returns x_{d} as the new iterate x_{k+1}. It also returns a new grid size h_{m+1}, which is defined by

$$h_{m+1} = \min\{|(z_m)_i - (x_{\text{d}})_i| : i \in 1, \ldots, n \text{ and } (z_m)_i \neq (x_{\text{d}})_i\} \tag{3}$$

where $(z_m)_i$ denotes the i^{th} component of z_m, and similarly for x_{d}. This choice of h_m ensures that both z_m and x_{k+1} lie on the new grid. This allows DIRECT to set $v_{k+1} = x_{k+1} - z_m$.

The size h_{d} of DIRECT's search region is chosen as follows. If f is known to be smooth, or if $h_m > h_{\text{macro}}$ then $h_{\text{d}} = 3h_m/2$. This choice means that the $2n+1$ known function values at x_k and $x_k \pm h_m e_i$, $i = 1, \ldots, n$ from the most recent Hooke and Jeeves iteration can be used as the first $2n+1$ points for the DIRECT subalgorithm. These n pairs of points $x_k \pm h_m e_i$, $i = 1, \ldots, n$ are placed in ascending order of the quantity

$$\min\{f(x_k + h_m e_i), f(x_k - h_m e_i)\}.$$

The DIRECT algorithm proceeds as if it had generated these n pairs of points initially in ascending order. If f is potentially non-smooth and $h_m \le h_{\text{macro}}$ then h_{d} is chosen as

$$h_{\text{d}} = \tfrac{3}{2}\min\{h_{\text{macro}}, \max\{81h_m, h_{\text{meso}}\}\}. \tag{4}$$

In this case the $2n$ extra points from the most recent Hooke and Jeeves iteration are not useable as they are in the wrong positions, and so DIRECT begins with the single point x_k in the centre of its search region.

The parameters h_{macro} and h_{meso} are the upper and lower limits of what is termed 'the mesoscale.' When $h_m > h_{\text{macro}}$ DIRECT searches in a box of the same scale as the Hooke and Jeeves increments. In the mesoscale DIRECT searches on a larger scale than Hooke and Jeeves, and DIRECT's scale is bounded below by h_{meso}. This strictly positive lower bound is crucial to the algorithm's convergence properties on nonsmooth problems. The '$81h_m$' term in (4) allows h_m to increase once in the mesoscale. The quantity $h_{\text{macro}}/h_{\text{meso}}$ is required to be a multiple of 3 for convergence on nonsmooth problems.

Boxes are selected for subdivision according to two values associated with each box. The first is the box's 'height' which is the function value at the centre point of the box. The second is the box's 'level,' which is the number of times Ω has to be subdivided in order to generate the box in question. Any box which is Pareto optimal in terms of height and level is subdivided unless the box is too small to subdivide further. A box is considered Pareto optimal if every other box is higher or has a greater level than that box. The maximum number of times a box can be subdivided is set at

$$\max\left(n(2 + \lceil\log(h_{\text{meso}}/H_{\min})\rceil, 2n\lceil\log(N_{\max} - N_{\text{now}})\rceil\right) \tag{5}$$

where $\lceil x\rceil$ denotes the least integer not less than x, and the logarithms are to the base e. The purpose of this maximum is to force DIRECT to open more low level boxes when the current number of function valuations N_{now} is getting close to the maximum number of function evaluations $N_{\max}$. The first term in (5) ensures that DIRECT is able to perform enough subdivisions to select h_m smaller than $H_{\min}$, guaranteeing that the stopping condition $h_m < H_{\min}$ can actually be satisfied. The second term ensures that enough subdivisions can be done to use up all remaining points $(N_{\max} - N_{\text{now}})$, if needed. When $N_{\max} = \infty$ this limit on the number of subdivisions disappears.

This selection strategy has the property that it subdivides at least one (specifically the lowest) box in the lowest non-empty level at each iteration. The way DIRECT selects boxes for subdivision is a crucial to the convergence theory. Our selection strategy has the following property.

Definition 1 *A selection strategy Γ is called* **valid** *if at least one box of the lowest non-empty level is subdivided in each iteration of* DIRECT*.*

In the convergence analysis valid selection strategies are treated collectively. This allows us to obtain results uniformly applying to *every* execution of the DIRECT subalgorithm.

4 Convergence Results

The algorithm's convergence properties are analyzed when $N_{\max} = \infty$ and $H_{\min} = 0$. First we show the set of points generated by DIRECT is uniformly dense in the search region, for all valid box selection strategies.

Proposition 2 *Let $\xi \in \Xi$, be arbitrary, where $\Xi = z_m + h_{\text{meso}}[-1, 1]^n$. Let $\{\delta_r\}_{r=1}^{\infty}$ be the sequence of points generated by* DIRECT *for an arbitrary valid box selection strategy Γ, and let*

$$\Delta(r) = \max_{\Gamma} \max_{\xi\in\Xi} \left\{ \min_{i=1,\ldots,r} (\|\xi - \delta_i\|) \right\}$$

Then $\Delta(r) \to 0$ as $r \to \infty$.

Proof: Clearly ξ lies in at least one of the r boxes with centres $\delta_1, \ldots, \delta_r$ generated by DIRECT. Hence $\min(\|\xi - \delta_i\|)$ is at most n times the maximum of all edge lengths of the r boxes generated by DIRECT. The number of such maximal boxes is always finite, and the one such box is subdivided each iteration, because the selection strategy Γ is valid. No new maximal sized boxes are created as there are no larger boxes to subdivide, and so after a finite number of iterations (independent of Γ and ξ) all boxes of the maximal size are eliminated. The only possible values edge lengths can take are $2.3^{-s}h_{\text{d}}$, where $h_{\text{d}} \in [h_{\text{meso}}, h_{\text{macro}}]$ for some non-negative integer s, so $\min_r(\|\xi - \delta_i\|)$ goes to zero as $r \to \infty$. Since Γ and ξ were arbitrary, the result follows. $\square$

The convergence result for the smooth version of the algorithm can now be given.

Theorem 3

(a) Assume the sequences $\{z_m\}$ and $\{x_k\}$ are finite. If f is strictly differentiable at the final value z_{m_*} of $\{z_m\}$, then $\nabla f(z_{m_*}) = 0$.

(b) If z_* is a cluster point of the sequence of grid local minimizers $\{z_m\}$ and if f is strictly differentiable at z_*, then $\nabla f(z_*) = 0$.

Proof: If the sequences $\{z_m\}$ and $\{x_k\}$ are finite then the final execution of DIRECT must be an infinite process. Proposition 2 implies $f(x) \geq f(z_{m_*})$ for all x in a dense set in $z_{m_*} + h_{\text{meso}}[-1, 1]^n$. The definition of strict differentiability [4] implies result (a).

Result (b) follows immediately from Corollary 4.5 of [13]. □

It is possible that $\{z_m\}$ is an infinite sequence with no cluster points, in which case $\{z_m\}$ must be unbounded. This can be excluded by assuming that $\{z_m\}$ is bounded, but it is possible for $\{z_m\}$ to be both unbounded and have cluster points. In the latter case Theorem 3 is still valid for these cluster points.

The non-smooth convergence result shows that every cluster point of the sequence $\{z_m\}$ is an open set essential local minimizer of f.

Definition 4 *An essential local minimizer z_* is a point for which the set*

$$L(z_*, \epsilon) = \{z \in R^n : f(z) < f(z_*) \quad \text{and} \quad \|z - z_*\| < \epsilon\}$$

has Lebesgue measure zero for all sufficiently small positive ϵ.

If f is continuous at z_*, then z_* is also a local minimizer of f in the traditional sense. However it is easily seen that $\{z_m\}$ does not always converge to an essential local minimizer. A simple counterexample is the function

$$f = \begin{cases} \|x\|^2 & x \text{ rational} \\ -1 & \text{otherwise.} \end{cases}$$

Clearly $z = 0$ is not an essential local minimizer of f. If grids $\{\mathcal{G}_m\}$ contain only rational points, then HJDIRECT will misidentify $z = 0$ as a local minimizer. Hence the definition of essential local minimizer is modified somewhat, as follows.

Definition 5 *An open set essential local minimizer z_* is a point for which the set $L(z_*, \epsilon)$ contains no open set in the standard topology for all sufficiently small positive ϵ.*

We show the grids become arbitrarily fine, and then give the non-smooth convergence result.

Proposition 6 *Let $\{z_m\}_{m \in \mathcal{M}}$ be a bounded infinite subsequence of $\{z_m\}$. Then*

$$\liminf_{m \in \mathcal{M}} h_m = 0 \qquad \text{and} \qquad \limsup_{m \in \mathcal{M}} N_m = \infty$$

where N_m is the number of function evaluations used by the m^{th} execution of DIRECT.

Proof: The proof is by contradiction. Assume $h_m > H$ for all $m \in \mathcal{M}$, where $H > 0$. Then every member of $\{z_m\}_{m\in\mathcal{M}}$ lies on the union of at most two grids. The first contains x_0 and has a grid size equal to the smallest h_m used before the mesoscale is encountered. The second grid has grid size equal to the smallest h_m used after the mesoscale has been encountered. It also contains the first iterate generated after the mesoscale is reached. The two grids are not necessarily related to one another because h_{meso} need not be a rational multiple of h_0. The fact that $h_{\text{macro}} = 3^s h_{\text{meso}}$ for some positive integer s means the second grid exists. Now $\{f(z_m)\}$ is a strictly decreasing monotonic sequence, so all z_m, $m \in \mathcal{M}$ are distinct. The fact that $\{z_m\}_{m\in\mathcal{M}}$ is bounded means it must be a finite sequence, which yields the contradiction.

For the second limit, the construction of h_{m+1} in (3) means that $h_{m+1} \geq h_{\text{meso}} 3^{\lceil -N_m/n \rceil}$, which yields the required result. □

Theorem 7 *Exactly one of the following possibilities holds:*

(a) $\{z_m\}$ is an infinite sequence and each cluster point z_* of it is an open set essential local minimizer of f; or

(b) both $\{z_m\}$ and $\{x_k\}$ are finite sequences and the final z_m is an open set essential local minimizer; or

(c) $\{z_m\}$ is finite and $\{x_k\}$ is an infinite unbounded sequence.

Proof: Clearly at most one of these possibilities holds because no sequence can be both finite and infinite.

Let $\{z_m\}$ be an infinite sequence, and let z_* be a cluster point of that sequence. By passing to a subsequence if necessary, let $z_m \to z_*$ and $N_m \to \infty$ as $m \to \infty$. Proposition 6 shows that the latter limit is achievable for a suitable subsequence.

Assume z_* is not an open set essential local minimizer. Then there exists an open ball $B(\theta, \eta)$, centre θ, radius η in $z_* + \frac{1}{2} h_{\text{meso}}[-1, 1]^n$ on which $f(x) < f(z_*)$ for all $x \in B(\theta, \eta)$. Now $B(\theta, \eta)$ lies in $z_m + h_{\text{meso}}[-1, 1]^n$ for all m sufficiently large. The closest point generated by the m^{th} use of DIRECT is at most $\Delta(N_m)$ from θ, for m sufficiently large. Proposition 2 implies there exists a sufficiently large m such that $\Delta(N_m) < \eta$. This implies DIRECT finds a lower point than z_* — contradiction. Hence z_* must be an open set essential local minimizer, which is case (a).

Let $\{z_m\}$ be a finite sequence, and let m_* be the final value of m. This can only occur if the final execution of the algorithm's outer loop is an infinite process. There are two ways this can happen: the inner loop can be infinite, or DIRECT can fail to halt. For the former we have $f(x_k) < f(x_{k-1})$ for all k, and $x_k \in \mathcal{G}_{m_*}$ for all k sufficiently large. Hence $\{x_k\}$ must be infinite and unbounded, which is case (c). For the latter, Proposition 2 implies $f(x) \geq f(z_{m_*})$ for all x in a dense set in $z_{m_*} + h_{\text{meso}}[-1, 1]^n$. Hence z_{m_*} is an open set essential local minimizer, which is case (b). □

5 Numerical Results and Discussion

Two versions of our algorithm HJDIRECT were tested: smooth and non-smooth. The former uses $h_{\text{d}} = 3h_m/2$ always, and is provably convergent on C^1 functions via Theorem 3. The

Table 1: Non-smooth results for test set A.

	Results from [14]		Max interact		Min interact			
Function	f	nf	f	nf	f	nf	n	p
Rosenbrock	5E-8	4438	8E-8	897	2E-8	1154	2	2
Brown 2 dim	2E-3	10598	4E-4	950	4E-4	950	2	3
Beale	4E-8	3638	2E-7	1232	2E-8	1119	2	3
Helical val	7E-8	8406	3E-10	1951	1E-9	2773	3	3
Gulf	1E-5	15583	1E-5	19071	6E-6	31306	3	99
Powell 4 dim	4E-7	11074	7E-3	4570	3E-3	3659	4	4
Woods	3E-7	15610	1E-4	7630	5E-4	4682	4	6
Trigonometric	5E-8	14209	2E-7	7235	4E-8	6678	5	5
Variably dim	2E-7	34679	2E-6	35491	5E-7	55647	8	10

Table 2: Non-smooth results for test set B using exact interation. information

	Max interact				Min interact			
Function	f	nf	k	m	f	nf	k	m
Arrowhead	3E-6	853	13	6	6E-5	897	19	5
Boundary value	8E-5	3183	51	6	2E-4	3152	39	6
Broyden 3 dim	2E-4	3163	34	7	2E-4	2672	38	6
Broyden banded	3E-4	5256	35	7	9E-5	5407	48	8
ex Rosenbrock	5E-4	2206	42	12	6E-4	3747	67	22
nzf1	3E-5	3644	38	7	3E-5	3165	32	10
Tridiagonal	5E-6	820	37	7	4E-5	569	29	6
ex Woods	2E-3	6008	85	21	2E-3	8607	80	34
ex Variably dim	2E-2	20013	93	33	4E-1	20003	96	43
Brown almost lin	8E-3	17298	183	16	5E-3	20005	130	27
Powell lin fcn	3E-4	8629	56	6	3E-4	8489	53	6

Table 3: Non-smooth results for test set B using estimated interaction information.

	Max interact				Min interact			
Function	f	nf	k	m	f	nf	k	m
Arrowhead	4E-5	691	14	3	7E-5	902	20	5
Boundary value	7E-5	2864	43	5	3E-4	3406	53	6
Broyden 3 dim	4E-4	2541	36	6	3E-4	2538	34	6
Broyden banded	3E-4	4865	53	7	3E-4	5054	41	8
ex Rosenbrock	5E-4	3010	56	17	1E-3	4301	54	22
nzf1	2E-5	3392	25	8	1E-5	3224	40	8
Tridiagonal	5E-6	812	39	8	4E-5	568	29	6
ex Woods	3E-3	8407	105	35	2E-3	7269	77	33
ex Variably dim	3E-2	20005	130	36	1E-1	20015	138	51
Brown almost lin	1E-1	17067	100	19	9E-4	20001	147	29
Powell lin fcn	1E-4	9238	53	6	3E-4	8634	64	6

Table 4: Comparison with problems from test set B which are smooth at x_*. Estimated interaction information has been used and the variables have been ordered for maximum interaction.

	Results from [15]		HJDIRECT			
Function	f	nf	f	nf	k	m
Arrowhead	6E-16	105	1E-11	266	21	6
Boundary value	2E-7	16994	2E-9	1055	88	8
Broyden 3 dim	5E-9	420	5E-8	460	36	6
Broyden banded	1E-9	1444	2E-9	989	41	8
ex Rosenbrock	4E-6	3268	4E-7	1002	109	6
nzf1	5E-11	155	2E-9	561	34	6
Tridiagonal	2E-10	532	4E-10	331	39	6
ex Woods	2E-8	446	2E-7	529	69	6
ex Variably dim	2E-8	4566	2E-8	2307	95	7
Brown almost lin	5E-8	8674	4E-8	2462	116	6
Powell lin fcn	1E-10	1158	5E-10	959	40	6

Table 5: Results for problems from test set B which are smooth at x_*. The non-smooth version of Hooke and Jeeves—DIRECT with estimated interaction information has been used.

Function	n	ℓ	p	f	nf	k	m
Arrowhead	10	9	9	1E-7	1024	16	3
Boundary value	10	10	10	3E-6	3301	55	6
Broyden 3 dim	10	10	10	1E-6	2754	44	6
Broyden banded	10	10	10	2E-6	4376	33	6
ex Rosenbrock	10	5	10	4E-6	2880	69	11
nzf1	13	5	5	1E-7	5285	34	9
Tridiagonal	3	3	3	2E-7	579	29	5
ex Woods	16	4	24	7E-5	3308	85	11
ex Variably dim	22	6	40	5E-5	20025	251	16
Brown almost lin	22	6	30	3E-6	11041	109	11
Powell lin fcn	22	6	30	2E-7	11715	48	6

Table 6: Test set B results for C^1 problems using estimated interaction information and reordering for maximum interaction.

	Results from [15]		HJDIRECT			
Function	f	nf	f	nf	k	m
Arrowhead	7E-12	105	1E-8	256	20	6
Boundary value	5E-6	10838	6E-7	938	79	6
Broyden 3 dim	5E-7	372	1E-6	454	37	6
Broyden banded	2E-7	1587	4E-7	933	42	6
ex Rosenbrock	0.14	85	3E-3	1424	145	6
nzf1	2E-8	188	6E-7	542	33	6
Tridiagonal	2E-8	536	9E-9	328	39	6
ex Woods	3E-4	3492	9E-6	654	84	7
ex Variably dim	1E-3	74622	3E-5	3836	179	6
Brown almost lin	5E-6	9848	1E-5	2203	98	6
Powell lin fcn	9E-8	1163	2E-7	1094	48	6

latter uses the mesoscale (4) and is provably convergent on smooth and non-smooth functions via Theorem 7.

HJDIRECT was tested on two sets of problems. Test set A is drawn from problems in [12], and has been used previously in [14]. Test set B is drawn from the CUTEr [8] testset and results exploiting partial separability appear in [15]. The problems in test set A are set up as black box functions and are not partially separable. These problems allow comparison with another method for non-smooth optimization. The algorithm in [15] is not provably convergent on non-smooth problems.

In their original forms, all of these test functions are sums of squares of the form

$$f = \sum_{i=1}^{p} |f_i(x)|^\beta \tag{6}$$

with $\beta = 2$, and with the additional property that the optimal function value is zero. This latter detail allows us to modify these functions to get nonsmooth test functions with known global minimizers. Three types of test problems are used. A non-smooth set is obtained by using $\beta = 1$. These problems have multiple kinks, all of which pass through the solution point. The second set uses $\beta = 3/2$ yielding C^1 problems. These problems have discontinuous second derivatives at the solution points and elsewhere. The main effect of this is to exacerbate any ill-conditioning present in their $\beta = 2$ versions. The third set of problems uses the form

$$f = \sum_{i=1}^{p} \min\left(f_i^2(x), |f_i(x)|\right). \tag{7}$$

These problems have multiple kinks, but are smooth near the solution. The nature of these kinks is fairly benign, and these problems are much like their $\beta = 2$ counterparts.

Most of the problems in both test sets are standard. The 'nzf1' problem in test set B is fully defined in [15]. The extended Woods function is simply four non-overlapping copies of the Woods function in 16 dimensions. The extended variably dimensioned, Brown almost linear, and Powell linear functions are constructed by taking five 6–dimensional copies of these functions and overlapping them so that the last two variables in each copy are the first two variables of the following copy. Table 5 contains further information in the final three columns. Here n, ℓ, and p are the dimension, number of elements in the partially separable form, and number of absolute value terms. These last two quantities are different. For example the extended Rosenbrock's function is of the form

$$\sum_{i=1}^{5} f_i\left(x_{2i}, x_{2i+1}\right) \quad \text{where} \quad f_i\left(x_{2i}, x_{2i+1}\right) = \Big(|x_{2i} - 1| + 10|x_{2i+1} - x_{2i}^2|\Big).$$

This has 5 elements containing 10 absolute value terms. Similar information about test set A is listed in the two right hand columns of Table 1.

The algorithm was implemented with a stopping grid size of $H_{\min} = 10^{-5}$. The initial grid size was $h_0 = e/3$. This apparently strange value was chosen in preference to $h_0 = 1$ because the latter placed both the initial point and solution on the initial grid for several test problems. This allowed the algorithm to step *exactly* to the solution in a very small number of iterations, giving a misleading impression of what the method is capable of. The values $h_{\text{macro}} = e/27$ and $h_{\text{meso}} = e/3^7$ were used. The interactions were calculated with $\epsilon = 10^{-10}$ and variables ordered for minimal interaction with $\tau = 0.0005$.

For set B, the partial separability structure of each problem is explicitly available. This is used to calculate function values more cheaply in the same manner used in [15], allowing a fair comparison with the results from [15]. This also allows us to compare the effectiveness of how interactions are estimated. When the partial separability structure of f is used to generate the interactions, the estimated interaction values are replaced with the following. For $i \neq j$, H_{ij} is set to one if x_i and x_j interact, and $H_{ij} = 0$ otherwise. This does not take into account the differing strengths of the interactions between various pairs of variables as this information is not contained in the partial separability structure.

Numerical results are listed in 6 tables. The legend for the tables is as follows. The column headed f lists the function value at the final iterate. Columns headed 'nf', k and m list the numbers of function evaluations, Hooke and Jeeves iterations and DIRECT iterations used in achieving that function value. The multicolumn headings 'Max interact' and 'Min interact' refer to results when the variables are respectively ordered to maximize or minimize interaction between consecutive variables.

Table 1 lists results for the nonsmooth functions ($\beta = 1$) in test set A. A comparison with the results from [14] is given. Our method was superior on 5 of the 9 problems listed. On the Powell 4 dimensional and Woods functions it found less accurate answers in fewer function evaluations. For the Gulf and variably dimensioned problems it was somewhat slower, performing huge numbers of Hooke and Jeeves iterations between locating grid local minima.

Tables 2 and 3 show results for test set B used as nonsmooth problems (i.e. $\beta = 1$). Those which calculated the interactions from known partial separability information are listed in Table 2 and those which estimate the interactions are given in Table 3. Both tables list results for the cases when the variables are ordered for maximal and minimal interaction between consecutive variables. In each case $N_{\max} = 20000$ was used. A final function value of 10^{-3} here corresponds to a final function value of 10^{-6} for the 'sum of squares' case, so any value less than about 10^{-3} is considered acceptable. The results for the extended Woods and Brown almost linear functions are marginal. The method was not able to solve the extended variably dimensioned problem in 20000 function evaluations (along with the Brown almost linear problem in one case). The results show there is little difference between generating the interaction matrix using partial separability information and estimating the interactions numerically. An examination of the interaction matrices H showed that HJDIRECT almost always deduced the correct partial separability structure quickly. The results also show that there is little difference between ordering for minimal and for maximal interaction, with perhaps a slight favouring of the latter. Interestingly, the *smooth* version of HJDIRECT managed to solve all but four of the *non-smooth* test problems in set B: failing on extended Rosenbrock, extended Woods, extended variably dimensioned, and extended Brown almost linear.

Tables 4 and 5 list results for test set B using the problem form (7). In Table 4 comparisons are made with results from [15], which show that our method outperforms that of [15] on average. The problems on which our HJDIRECT is slower are four of the five quickest problems to solve. On the harder problems HJDIRECT is clearly superior. Table 5 lists results for the same problems using the non-smooth form of HJDIRECT. A comparison between the HJDIRECT results for Tables 4 and 5 shows that the number of grids (m) and function evaluations used are higher for the non-smooth algorithm, but the number of Hooke and Jeeves iterations (k) is broadly similar. This is because DIRECT is doing much more work in the non-smooth version once the mesoscale is reached.

Table 6 lists results for the same problems using $\beta = 3/2$. Once again HJDIRECT is faster on most problems and much faster on the harder ones. On three of the easiest problems it is slower. It is noted that the method of [15] failed to solve the extended Rosenbrock's problem with $\beta = 3/2$.

Although results are only presented for an accuracy of $H_{\min} = 10^{-5}$, HJDIRECT has been tested with accuracies down to near machine precision ($H_{\min} = 10^{-15}$). HJDIRECT performed well over the range of $H_{\min}$ values, with an approximately linear rate of convergence.

In retrospect, the bound on the number of subdivisions (5) is too crude. It allows the possibility that DIRECT will find a lower point extremely close to z_m when m is small. This would give an unjustifiably small value for h_{m+1}, which might lead to premature termination of the algorithm. A better approach would be to slowly increase the maximum number of subdivisions as DIRECT progresses. Nevertheless, this weakness did not appear to adversely affect the numerical results.

6 Conclusion

A direct search method for nonsmooth unconstrained optimization has been presented. The method is a hybrid of the classical Hooke and Jeeves method, and the DIRECT algorithm of Jones, Pertunnen, and Stuckman. It can detect and exploit partial separability through choice of the order in which Hooke and Jeeves polls each decision variable. Convergence on non-smooth problems has been demonstrated under mild conditions. Numerical results have been presented for problems of up to 22 dimensions. These verify the convergence theory and show that our method is competitive in practice. Comparisons with two other direct search algorithms show that our method is faster than both.

References

[1] C. Audet and J. E. Dennis Jr., Analysis of generalized pattern searches, *SIAM J. Opt.*, **13** (2003), 889–903.

[2] C. Audet and J. E. Dennis Jr., Mesh adaptive direct search algorithms for constrained optimization, *SIAM J. Opt.*, **17** (2006), 188–217.

[3] M. C. Bartholomew-Biggs, S. C. Parkhurst, and S. P. Wilson, Using DIRECT to solve an aircraft routing problem, *Computational Opt. and Applications*, **21** (2002), 311–323.

[4] F. H. Clarke, Optimization and Nonsmooth Analysis, SIAM classics in Applied Mathematics, 1990.

[5] B.Colson and Ph. L. Toint, Exploiting sparsity in unconstrained optimization without derivatives, *Optimization in Engineering*, **2** (2001), 349–412.

[6] B. Colson, and Ph. L. Toint, Optimizing partially separable functions without derivatives, *Optimization Methods and Software*, **20** (2005), 493–508.

[7] I. D. Coope and C. J. Price, On the convergence of grid-based methods for unconstrained optimization, *SIAM J. Opt.*, **11** (2001), 859–869.

[8] N. I. M Gould, D. Orban, and Ph. L. Toint, CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited, Technical Report RAL-TR-2002-009, Computational Science and Engineering Department, Rutherford Appleton Laboratory, 2002.

[9] A. Griewank and Ph. L. Toint, On the unconstrained optimization of partially separable functions, In: M. J. D. Powell (Ed.), *Nonlinear Optimization 1981*, (London: Academic Press), pp. 301–312.

[10] R. Hooke and T. A. Jeeves, Direct search solution of numerical and statistical problems, *Assoc. Computing Machinery J.*, **8** (1960), 212–229.

[11] D. Jones, C. D. Perttunen, and B. E. Stuckman, Lipschitzian optimization without the Lipschitz constant, *J. Opt. Theory Applic.*, **79** (1993), 157–181.

[12] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, Testing unconstrained optimization software, *ACM Trans. Math. Software*, **7**:1 (1981), 17–41.

[13] C. J. Price and I. D. Coope, Frames and grids in unconstrained and linearly constrained optimization: a non-smooth approach, *SIAM J. Opt.*, **14** (2003), 415–438.

[14] C. J. Price, M. Reale, and B. L. Robertson, A direct search method for smooth and nonsmooth unconstrained optimization, *ANZIAM J.*, **48** (2006), C927–C948.

[15] C. J. Price and Ph. L. Toint, Exploiting problem structure in pattern search methods for unconstrained optimization, *Optimization Methods and Software*, **21**, (2006), 479–491.

[16] V. Torczon, On the convergence of pattern search algorithms, *SIAM J. Opt.*, **7** (1997), 1–25.

[17] M. H. Wright, Direct search methods: once scorned now respectable, in *Proceedings of the 1995 Dundee Biennial Conference in Numerical Analysis*, Longman, Harlow, UK (1996), 191–208.