Abstract

FINKEL, DANIEL EDWIN. Global Optimization with the DIRECT Algorithm. (Under the direction of C.T. Kelley.)

This work presents theoretical results, and practical improvements to the DIRECT Algorithm, a direct search global optimization algorithm for bound-constrained problems. We rigorously show that a sub-sequence of the points sampled by the algorithm satisfy first order necessary conditions for both smooth and non-smooth problems. We show linear convergence of the algorithm for linear problems, and demonstrate why our analysis cannot be extended to higher dimensional problems.

We analyze a parameter of DIRECT, and show that it negatively affects the performance of the algorithm. A modified version of the DIRECT is presented. Test examples are used to demonstrate the effectiveness of the modified algorithm.

We apply DIRECT to six well-field optimization problems from the literature. We collect data on the problems with DIRECT, and utilize statistical methods to glean information from the data about the well-field problems.

January 18, 2005

GLOBAL OPTIMIZATION WITH THE DIRECT ALGORITHM

BY

Daniel E. Finkel

A DISSERTATION SUBMITTED TO THE GRADUATE FACULTY OF

NORTH CAROLINA STATE UNIVERSITY

IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

OPERATIONS RESEARCH, MATHEMATICS MINOR

RALEIGH, NORTH CAROLINA

January 2005

APPROVED BY:

C. T. Kelley CHAIR OF ADVISORY COMMITTEE	S. Ghosal
R.C. Smith	H.T Tran

Biography

Daniel Edwin Finkel was born in 1976 in Tampa, FL. and grew up in Grand Island, N.Y. Daniel received his undergraduate degree from SUNY Fredonia State in mathematics (1998), his M.S. in applied mathematics from North Carolina State University (2000), and his Ph.D. in operations research also from North Carolina State University (2005).

Daniel is married to Nancy Hawkins, has two younger brothers, Benjamin and Aaron, and is the son of Anna and Sidney Finkel. Daniel plans to continue his career in the quantitative sciences as a technical staff member at MIT Lincoln Labs in Lexington, M.A.

Acknowledgments

I would like to acknowledge my advisor, Tim Kelley. Professor Kelley's intelligence and knowledge of numerical optimization is well documented, but the success of this project is a product of his patience, mentoring, friendship, intensity, loyalty and persistence. I will be grateful for Tim's guidance for the rest of my life.

I would also like to acknowledge the members of my committee: Dr. H.T. Tran, Dr. R. Smith, and Dr. S. Ghosal. All three committee members played invaluable roles in my education, and I am indebted to all of them for their support.

I would like to thank my research group: Jill Reese, Matthew Lasater, Katie (Kavanaugh) Fowler, Kelly Dickson, Karen Dillard and Robert Darwin. I would also like to thank Professor Ilse Ipsen and the rest of the NA seminar group for all the help with my writing and presentation skills. I would like to thank Evin Cramer, Joerg Gablonsky and John Dennis of The Boeing Company for introducing me to the issues concerning industrial optimization.

I would not have attended graduate school without the support of Nancy Boynton, Robert Rogers, Albert Polimeni, William Leslie, Fawzi Yaqub and Joseph Straight.

My sincere thanks to them and the rest of the mathematics department at SUNY Fredonia.

Kathie Zink, Brenda Currin, Rory Schnell, and Barbara Walls were incredibly helpful and patient with me. Thank you. Finally, I wish to thank my family: Mom, Dad, Nancy, Ben, Aaron, and my friends: Josh Aaron, Brian Scozzaro, Brian Lewis, Mike Malinowski, Toby Trowbridge, Sabrina Papes, Joe Foegen , Tim Richthammer and many others. They have all been incredibly supportive, and played an important role in the success of my research.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The DIRECT algorithm, a direct search (in the sense of [30] and [54]) global optimization method, was first introduced in [35]. In this thesis, we describe rigorous new analysis and algorithmic improvements to DIRECT.

The global optimization that we consider is:

Problem 1.1 (G). *Let $\Omega \subseteq \mathbb{R}^N$, and let $f : \Omega \to \mathbb{R}$. Find f^* and x^* such that*

$$f(x^*) = f^* = \min_{x \in \Omega} f(x).$$

Global optimization problems arise in many different applications in the fields of mathematics, operations research, engineering, biology and biomathematics, chemistry, finance, etc...(for example, [9, 21, 58, 10, 38]).

Problem G can be difficult to solve. First, descriptive information about f may not exist. Assumptions about its smoothness, continuity, and differentiability often

cannot be made. Objective functions of this type are typically referred to as "black-box" functions because of the uncertainty. The objective function may also be noisy, and have many undesirable local minima. Hence, traditional gradient based methods are often unable to solve Problem G. In addition, an evaluation of f may be computationally expensive, and require minutes, hours, or even days to calculate. Therefore, an algorithm which requires a lot of function evaluations to find the next iterate, or does not lend itself to parallel processor programming is not practicable. One of the attractive features of `DIRECT` is that it is an embarrassingly [42] parallelizable algorithm.

The design space, Ω, can also present difficulties. Many industrial problems contain constraints that cannot be explicitly stated (see, for example [21]). These are typically referred to as "hidden constraints." Since most optimization algorithms are not designed to treat hidden constraints, a heuristic may be needed.

In this work, a global optimization method known as `DIRECT` is presented and analyzed. `DIRECT` was developed to solve a subset of Problem G, which is referred to as Problem P:

Problem 1.2 (P). *Let* $-\infty \leq l_i \leq u_i \leq \infty$, $\Omega = \left\{x \in \mathbb{R}^N | l_i \leq x_i \leq u_i\right\}$ *and* $f : \Omega \rightarrow \mathbb{R}$ *be Lipschitz continuous with constant* K*. Find* f^* *and* x^* *such that*

$$f(x^*) = f^* = \min_{x \in \Omega} f(x).$$

DIRECT searches for a solution to Problem P by carving Ω into hyperrectangles, and evaluating f at their centers. In each iteration, DIRECT chooses several of the hyperrectangles that compose Ω to be further sampled and divided. The name DIRECT comes from this description (DIViding RECTangles).

In the next section, we motivate and introduce the DIRECT algorithm.

1.1 Lipschitz Optimization

DIRECT evolved as a solution to the shortcomings of a Lipschitz optimization algorithm developed by S.A. Piyavskii [50], and independently by B. Shubert [52]. In this section, we describe the Piyavskii algorithm.

The Piyavskii algorithm was developed to solve Problem P for univariate functions. Let $f : [a, b] \to \mathbb{R}$, where $a < b$ and $a, b \in \mathbb{R}$. If f is Lipschitz continuous, then its rate of change is bounded by a finite value. That is,

Definition 1.3. *Let $[a, b] \subseteq \mathbb{R}^1$ and $f : [a, b] \to \mathbb{R}$. The function f is said to be Lipschitz continuous on $[a, b]$ with Lipschitz constant K if*

$$|f(x) - f(x')| \leq K|x - x'|, \qquad \forall x, x' \in [a, b]. \tag{1.1}$$

Piyavskii's algorithm searches for the global minimum of f by bounding it with a piece-wise linear function. Knowledge of the Lipschitz constant allows this piece-wise linear function to be an accurate approximation to the function f. This next theorem,

reproduced from [31] formally states this fact.

Theorem 1.1 (Piyavskii). *After evaluating function f at k points $x_1, x_2, ..., x_k$ of $[a, b]$, the tightest upper bounding function F_k for f on this interval is defined as*

$$F_k(x) = \min_{i=1,2,...,k} \{f(x_i) - K|x - x_i|\}. \tag{1.2}$$

Figure 1.1 illustrates the function $F_k(x)$ applied to a test function. Due to the shape of $F_k(x)$, Piyavskii's algorithm is sometimes referred to as the saw-tooth method. The minimum of (piyaeqn) is found at the low peak location. At each iter-

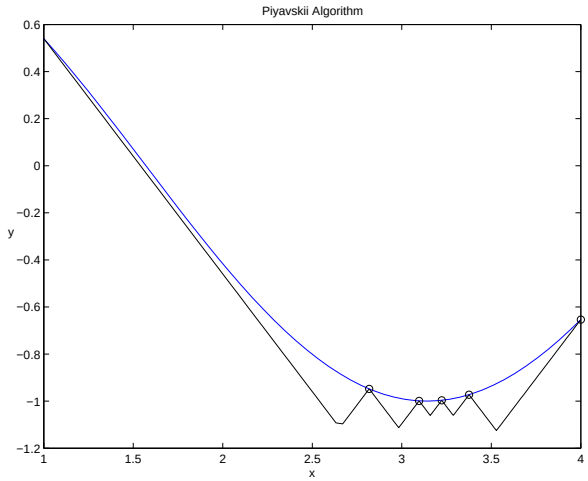

Figure 1.1: An example of the Piyavskii Algorithm.

ation, Piyavskii's algorithm chooses to sample f the minimum peak, and then split it's tooth into two smaller teeth. A formal description of the algorithm is described in Table 1.1.

Equation 1.2 demonstrates the local/global nature of Piyavskii's algorithm. When

Table 1.1: Piyavskii's Algorithm

Algorithm Piyavskii('myfcn',a,b,K,ϵ)

Set $k \leftarrow 1$
Set $x_1 \leftarrow (a+b)/2$
Set $x_{opt} \leftarrow x_1$, $f_{opt} \leftarrow f(x_{opt})$,
$F_{opt} \leftarrow f_{opt} + L(b-a)/2$, $F_1(x) \leftarrow f(x_1) + L|x - x_1|$
while $F_{opt} - f_{opt} > \epsilon$ **do**
    $x_{k+1} \leftarrow \min_{x \in [a,b]} F_k(x)$
    if $f(x_{k+1}) > f_{opt}$ **then**
        $f_{opt} \leftarrow f(x_{k+1})$
        $x_{opt} \leftarrow x_{k+1}$
    endif
    $F_{k+1}(x) \leftarrow \min_{i=1,\ldots,k+1} \{f(x_i) + L|x - x_i|\}$
    $F_{opt} \leftarrow \min_{x \in [a,b]} F_{k+1}(x)$
    $k \leftarrow k + 1$
end while

trying to minimize $F_k(x)$, the first term, $f(x_i)$ will reward locations that occur at low function values. This will lead the algorithm to search locally near good points. The second term, $|x - x_i|$, will encourage the selection of intervals that are are large, and hence contain unexplored territory. This is the global search portion of the algorithm. The Lipschitz constant, K, provides a weight between the global search and the local search.

The saw-tooth equation, $F_k(x)$, also demonstrates a potential pitfall of Piyavskii's algorithm. If the Lipschitz constant is not explicitly known, as is frequently the case, it must be estimated. A poor estimate can place an improper emphasis on the global search, and significantly decelerate the speed of convergence. Figure 1.2 shows an example of the Piyavskii algorithm performed on with an over-estimated Lipschitz

constant.

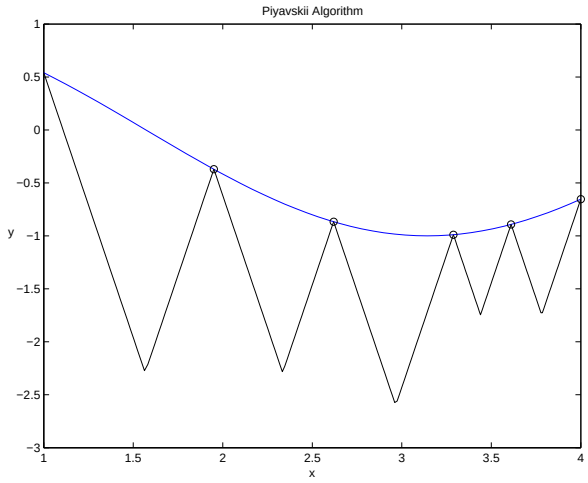

Figure 1.2: Piyavskii Algorithm with a poor estimate for the Lipschitz Constant.

In Figure 1.2, the poor estimate for the Lipschitz constant results in the search becoming more global. Another problem occurs when Piyavskii's algorithm is extended to higher dimensions. In the initialization phase, the vertices of the region are evaluated. This is not an issue in $\mathbb{R}^1$, but, in higher dimensions the number of vertices grows exponentially. Several attempts have been made to address the computational complexities that are created in higher dimensions [48, 45, 25], but the results are inconclusive.

In the next section, the `DIRECT` algorithm is introduced. `DIRECT` (first introduced in [35]) is a global optimization algorithm designed to overcome the problems associated with Lipschitz optimization previously discussed. Although `DIRECT` can use a lot of evaluations of the objective function in its search, the algorithm is designed to be as frugal as possible in this regard.

1.2 DIRECT

The DIRECT algorithm (DIViding RECTangles) is a deterministic sampling algorithm developed by D. Jones, C.D. Perttunen and B.E. Stuckman, [35]. The algorithm was created in the spirit of Lipschitz optimization, and designed to overcome some of the shortcomings of traditional algorithms, like Piyavskii's.

Recall that one problem of Piyavskii's algorithm was its reliance on an accurate estimation of the Lipschitz constant. DIRECT will solve this problem by replacing the Lipschitz constant with a varying, and adaptive internal parameter. This strategy is appealing for several reasons.

- First, the Lipschitz constant is a global attribute of a function, and may misrepresent pieces of a function. For example, the one dimensional function $f(x) = x^2$ has a Lipschitz constant of ten over the interval $[-5, 5]$. The rate of change of the function is not ten, however, near the point $x = 0$. The adaptive variable used in place of the Lipschitz constant allows DIRECT to react to different neighborhoods of the objective function properly.

- Next, replacing the Lipschitz constant with an adaptive variable provides DIRECT with the flexibility to simultaneously conduct local and global searches for the optimal solution; we will explore this fact in the next sections.

- Finally, the DIRECT algorithm determines what values the approximate Lipschitz constant takes on, relieving the burden on a priori knowledge of the actual

Lipschitz constant from the user.

DIRECT seeks a solution to Problem P. Throughout this work, it is assumed that the hyper-rectangular domain is in fact the unit-hypercube. If the bound constraints on x are not the unit cube, then a scaling transformation can be done. This transformation will not affect the results of DIRECT.

1.2.1 Initialization of DIRECT

DIRECT initiates its search for x^* by evaluating f in the center of Ω. Figure 1.3 provides a visual representation for a two-dimensional problem. Throughout DIRECT's

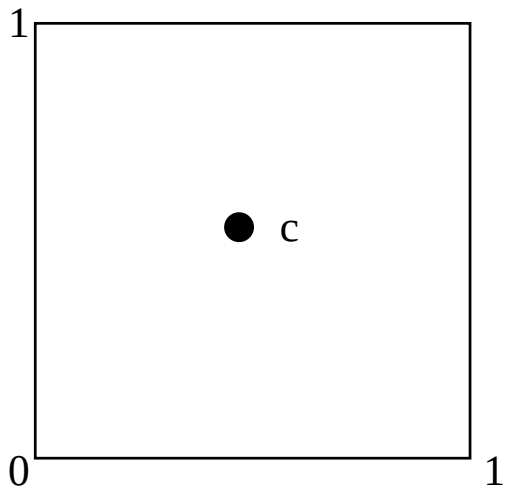

Figure 1.3: The initial sample of f at the center of the design space.

search, it will continue to sample in the center of hypercubes and hyperrectangles. Rectangles and hyperrectangles will only have one center, regardless of the dimension, (as opposed to vertices, which grow exponentially as N increases), therefore DIRECT has a simple and coherent strategy for higher dimensional problems.

This is an important deviation that DIRECT takes from Piyavskii's method. Others have modified Piyavskii's algorithm, see [27], but have maintained a sampling strategy of evaluating the objective function at edges and corners of the partitions. This strategy seems to add confusion to the process, and also increases the complexity issues of the algorithm (both of these arguments are pointed out in [35]). By creating a sampling strategy, independent of the problem's dimension, DIRECT has successfully addressed an important implementation issue.

DIRECT is now ready to divide the design space into smaller hyperrectangles. The strategy described is for the division of a hypercube, because in this initiation phase, only one hypercube exists. Later on, we will discuss a general division strategy.

Let $c = (1/2, 1/2, ..., 1/2)^T$ be the center of the unit hypercube. DIRECT begins by evaluating the objective function at the $2N$ points $c \pm 1/3e_i$, where $1 \leq i \leq N$ and e_i is the ith unit vector (i.e. a vector with 1 in the ith position and zeros elsewhere). The aim of the next step is to make each of these polled points centers of their own hyperrectangles.

There are different ways that this can be done. Figure 1.4 shows, for a two dimensional problem, two different ways the hypercube can be divided. In part (a), the horizontal dimension is divided first, and then the vertical dimension, while in part (b) the order is reversed. DIRECT implements a strategy so that dimensions with low function values are divided first. This strategy places the lowest function values in the largest hyperrectangles created by the division. As will be shown in the

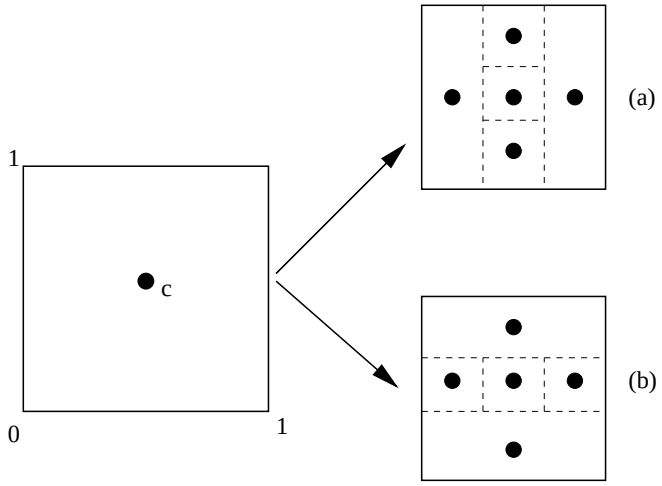

Figure 1.4: Different division strategies for DIRECT.

next section, DIRECT is biased towards large hyperrectangles, and therefore this will make the locations of low function values more attractive to DIRECT. Figure 1.5 is an example of a three dimensional hypercube that has been divided by DIRECT. The

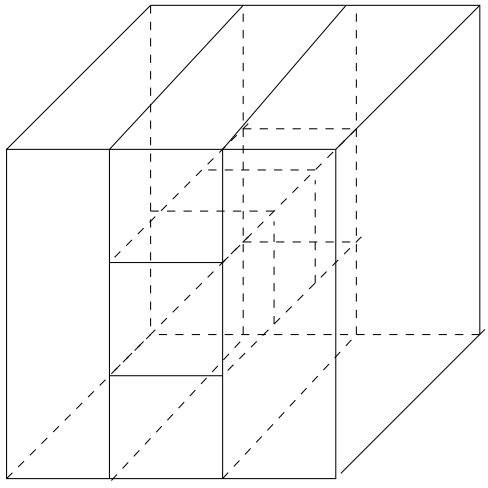

Figure 1.5: Division technique for a three-dimensional hypercube in DIRECT.

design space has now been divided into $2N + 1$ hyperrectangles. In it's next step, DIRECT will determine which of these hyperrectangles should be divided.

1.2.2 Potentially Optimal Rectangles

When DIRECT decides to divide a hyperrectangle and further sample inside of it, that rectangle is referred to as *potentially optimal*. Recall that Piyavskii's algorithm would minimize the piecewise linear surrogate $F_k(x)$ to determine where to sample next. DIRECT will attempt to duplicate this process, only assuming a broad range of values for the Lipschitz constant.

DIRECT will search globally and locally for the global minimum by dividing all hyperrectangles that meet the criteria of Definition 1.4 [35].

Definition 1.4. *Suppose that we have a partition of the unit hypercube into m hyperrectangles. Let $\mathbf{c}_i$ denote the center point of the ith hyperrectangle, and let d_i denote the distance from the center point to the vertices. Let $\epsilon > 0$ be a positive constant. A hyperrectangle j is said to be potentially optimal if there exists some $\tilde{K} > 0$ such that*

$$f(\mathbf{c}_j) - \tilde{K} d_j \leq f(\mathbf{c}_i) - \tilde{K} d_i, \qquad \text{for all } i = 1, 2, ..., m, \tag{1.3}$$

$$f(\mathbf{c}_j) - \tilde{K} d_j \leq f_{\min} - \epsilon |f_{\min}|. \tag{1.4}$$

In this definition, $\tilde{K}$ can be thought of as the Lipschitz approximation value. Figure 1.6 will aide in understanding Definition 1.4. In Figure 1.6, each rectangle that exists in the design space is plotted as a single point. The horizontal axis represents the distance from the center of the rectangle to one of its corners (it does not matter which corner, since all are equidistant from the center). The vertical axis

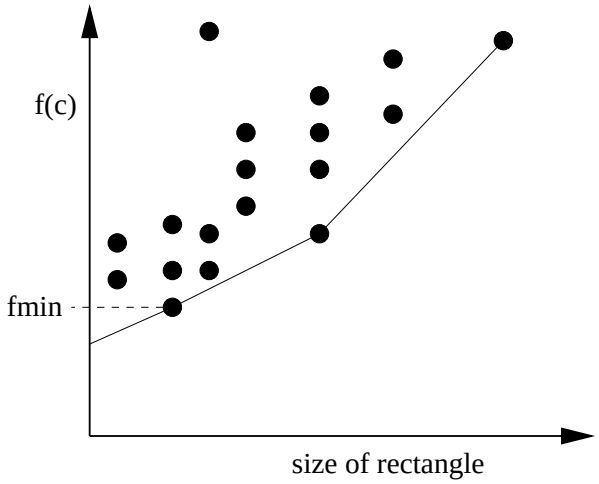

Figure 1.6: Interpretation of Definition 1.4.

is the value of the function f evaluated at the hyperrectangle's center.

If a line with slope $\tilde{K}$ is drawn through one of the points, say $(d_i, f(c_i))$, then its intercept along the vertical axis is calculated to be $f(c_i) - \tilde{K}d_i$. Thererfore, (1.3) and (1.4) can be interpreted as a description of points that comprise the lower convex hull of the graph shown in Figure 1.6.

This understanding of potentially optimal hyperrectangles leads to several lemmas:

Lemma 1.5. *Hyperrectangle i can not be potentially optimal if there exists a hyperrectangle j such that*

$$d_i = d_j \tag{1.5}$$

$$f(c_i) > f(c_j) \tag{1.6}$$

Proof. The is a direct result of Definition 1.4. Let $\hat{K}$ be any real number. From

Equations 1.5 and 1.6, we see that

$$f(c_i) - \hat{K}d_i > f(c_j) - \hat{K}d_j,$$

and thus hyperrectangle i does not satisfy Equation 1.3. □

This next lemma was first shown in [24].

Lemma 1.6. *Let $S_{\max}$ be the set of largest hyperrectangles. Then, `DIRECT` will always divide one of the hyperrectangles in $S_{\max}$.*

Proof. Let $S \in S_{\max}$ be the hyperrectangle in $S_{\max}$ with the smallest function value at its center. Let c be the center of S and let its size be d. Let $\hat{S} \in \mathcal{S}$, be any other hyperrectangle in the current state of `DIRECT`. Let $\hat{d}$ be the size of $\hat{S}$, and let $\hat{c}$ be its center. We show that Equations 1.3 and 1.4 are satisfied with respect to $\hat{S}$.

Case 1: $\hat{d} = d$ (i.e. $\hat{S}$ is the same size as S).

In this case, choose $\tilde{K}$ large enough so that

$$f(c) - \tilde{K}d \leq f_{\min} - \epsilon|f_{\min}|.$$

By definition, we know that $f(c) \leq f(\hat{c})$, thus

$$f(c) - \tilde{K}d \leq f(\hat{c}) - \tilde{K}\hat{d},$$

and S is potentially optimal with respect to $\hat{S}$.

Case 2: $\hat{d} > d$ (i.e. $\hat{S}$ is smaller than S).

We split this scenario into two sub-cases

(a) $f(c) \leq f(\hat{c})$

The argument for this case is identical to the argument posed in Case 1. Choose $\tilde{K}$ large enough to satisfy Equation 1.3, and Equation 1.4 follows immediately.

(b) $f(c) > f(\hat{c})$

Choose $\tilde{K}$ so that

$$\tilde{K} \geq \max\left\{\frac{f(\hat{c}) - f(c)}{\hat{d} - d}, \frac{f(c) - f_{\min} + \epsilon|f_{\min}|}{d}\right\}.$$

This choice of $\tilde{K}$ ensures that Equations 1.3 and 1.4 are satisfied, and hyperrectangle S is therefore potentially optimal.

□

The parameter ϵ is used to ensure that `DIRECT` does not become too local in its search. Figure 1.7 shows an example where a hyperrectangle might not be selected because of ϵ. In the figure, the rectangle X_1 will not be divided because a line that is steep enough to advance its vertical intercept below the threshold $f_{\min} - \epsilon|f_{\min}|$ is not capable of being drawn. This is due to the larger hyperrectangle X_2 having a comparable value of f at its center. We will further examine the effects of ϵ in Chapter 3.

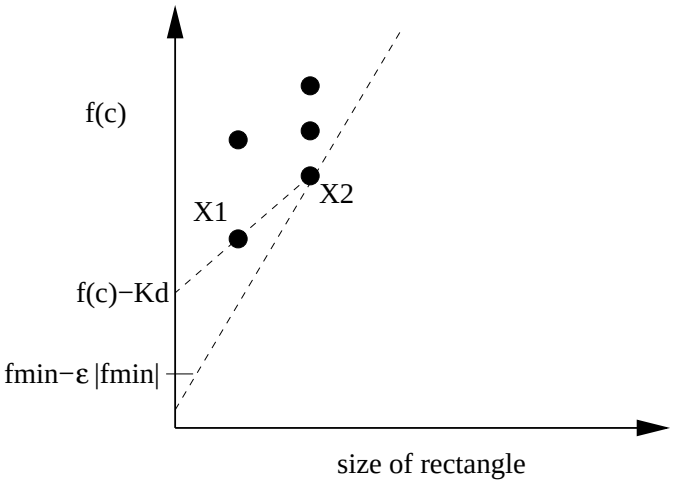

Figure 1.7: A situation where a rectangle is not divided because of ϵ.

DIRECT builds a list of hyperrectangles which it will divide in the current iteration by using the criteria set forth in Definition 1.4. The next step is to divide these rectangles into smaller rectangles, and sample at their centers. The next section outlines this procedure.

1.2.3 Division Procedure

The creators of DIRECT have offered two strategies for dividing potentially optimal hyperrectangles (see [35, 34]). This document is focused on the original implementation, although, we will describe the alternative strategy that was suggested in [34].

In the initialization step, DIRECT trisected the hyperrectangle that contained the center, c, along every dimension. In general, DIRECT divides only along the long sides of a potentially optimal hyperrectangle. This strategy ensures that no dimension is ignored by DIRECT. Figure 1.8 demonstrates what can happen if DIRECT ignores a

dimension. In Figure 1.8, the algorithm is repeatedly dividing the rectangle which contains a global minima, x^*, but is not converging to it.

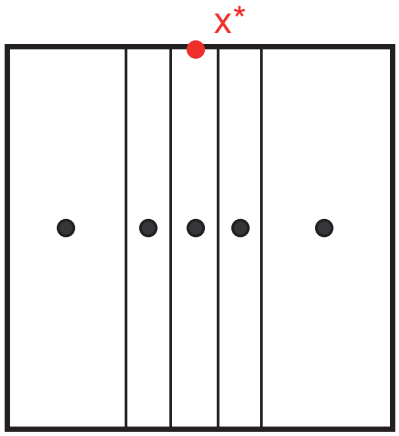

Figure 1.8: A poor division strategy

The decision to trisect sides, along with the assumption that Ω is a unit hypercube, ensures that every hyperrectangle created by DIRECT will have side lengths of a power of $\frac{1}{3}$. This fact can simplify the analysis of the algorithm, and allows an implementation of DIRECT to calculate the side lengths of hyperrectangles a priori.

The order in which DIRECT divides the long sides of a hyperrectangle alters the structure of the hyperrectangles that occupy Ω, as Figure 1.8 shows. In [35], a heuristic was proposed to determine the order in which long sides are divided. The heuristic was designed to leave low function values in large hyperrectangles. Since the two most appealing attributes of a hyperrectangle, in the eyes of DIRECT, are a large (relative) size, and a low (relative) objective function value in its center, this strategy encourages DIRECT to divide the hyperrectangles which contain the best points found

so far. Table 1.2 formally outlines the division procedure, and was first reported in [35].

Table 1.2: Steps taken by DIRECT to divide a hyperrectangle

How DIRECT divides a hyperrectangle

1:	Let ξ be the maximal side length.
2:	Let I be the set of dimensions of the hyperrectangle with length ξ.
3:	Evaluate the objective function at the points $c \pm \frac{1}{3}\xi e_i$, for all $i \in I$, where c is the center of the hyperrectangle, and e_i is the ith unit vector
4:	Let $w_i = \min\left\{f(c \pm \frac{1}{3}e_i)\right\}$
5:	Divide the hyperrectangle containing c into thirds along the dimensions in I, starting with the dimension with smallest w_i and continuing to the dimension with the largest w_i.

In [34], an alternative division strategy was suggested by the creators of DIRECT. Instead of dividing every long side of a potentially optimal hyperrectangle, as was suggested in [35], it is recommended that DIRECT only divide **one** of the long sides of a potentially optimal hyperrectangle in a given iteration. Although no evidence was produced, the authors in [34] claim that this strategy will improve the results of the algorithm on higher dimensional problems.

This thesis and the accompanying MATLAB code use the original division strategy from [35].

1.2.4 Summary of the Algorithm

Figure 1.9 shows the first three iterations of DIRECT on a sample problem, where $\Omega \subset \mathbb{R}^2$. The first column represents the state of DIRECT at the beginning of an

iteration. The second column shows which hyperrectangles are potentially optimal, and the third column shows the state of the domain after the potentially optimal hyperrectangles are divided. DIRECT initiates its search by sampling the objective

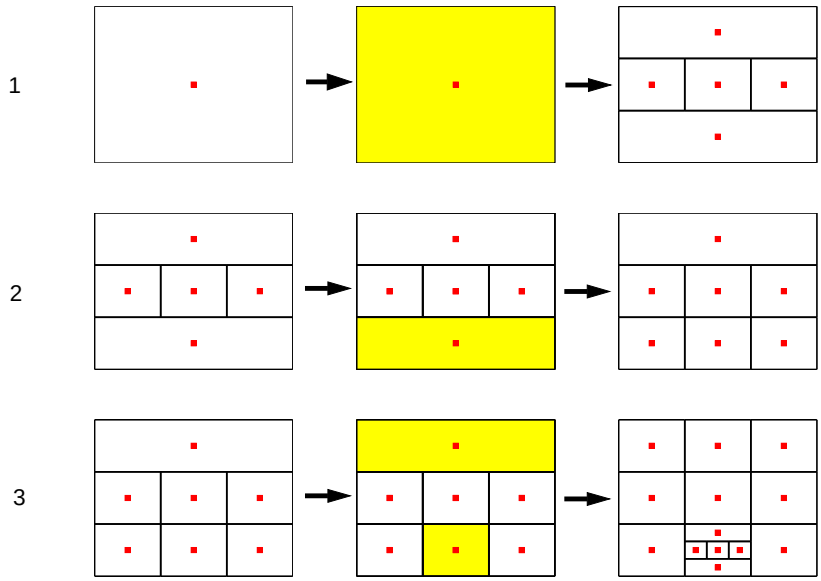

Figure 1.9: First three iterations of DIRECT on a sample problem.

function in the center of Ω, and treating the entire design space as a potentially optimal hyperrectangle. The search terminates when a user-supplied budget of function evaluations has been exhausted. Since DIRECT will not stop in the middle of a iteration, the algorithm may exceed its budget by a slight amount. An open question for DIRECT, (see [40, 23, 14]) is when should the algorithm terminate? This uncertainty is illustrated when one examines a plot of the convergence of DIRECT. Figure 1.10 is a typical example of the convergence of DIRECT Figure 1.10 is a graph of the function counter vs. the best objective value found by DIRECT on a typical test problem. As the figure suggests, the early iterations of DIRECT provide a rapid rate

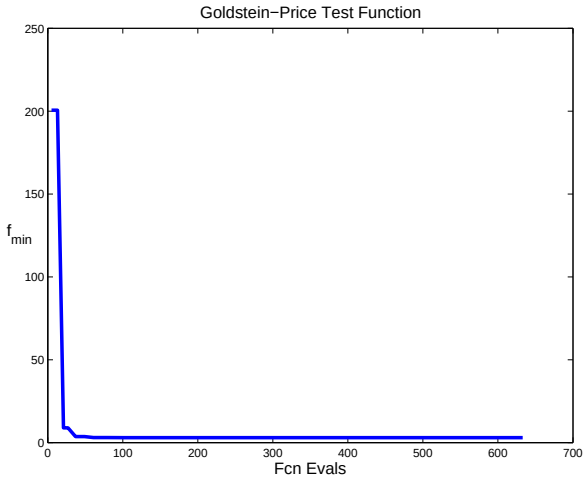

Figure 1.10: Results of DIRECT on a sample problem.

of improvement, while the later iterations do not improve the solution very much.

1.3 Known Convergence of DIRECT

The points sampled by DIRECT form a dense subspace of the domain, Ω, a fact pointed out in [35]. Thus, every point sampled by DIRECT is an accumulation point, and DIRECT eventually samples arbitrarily close to a global minima. In [35], this fact is used to argue that DIRECT eventually reports a satisfactory solution of Problem P to the user.

The set of points sampled by DIRECT becomes dense because of two important characteristics of the algorithm. First, every hyperrectangle will eventually be divided. This is due to the method of choosing potentially optimal hyperrectangles, and a consequence of Lemma 1.6, which we formally state as a corollary.

Corollary 1.1. *Let $\mathcal{S}^n$ be the set of hyperrectangles created by* `DIRECT` *after n iterations. Then, any $R \in \mathcal{S}^n$ will be potentially optimal in a finite number of iterations.*

Proof. This is a direct results of Lemma 1.6. Let

$$\hat{\mathcal{S}}^n = \{S \in \mathcal{S}^n : \alpha(S) \geq \alpha(R)\},$$

where $\alpha(\cdot)$ is the size of a hyperrectangle. The cardinality of the set $\hat{\mathcal{S}}^n$ is finite, say $|\hat{\mathcal{S}}^n| = q$, and at least one hyperrectangle in $\hat{\mathcal{S}}^n$ is potentially optimal in every iteration. Potentially optimal hyperrectangles are divided, and reduced in size. Since it will take a finite number of iterations to divide a hyperrectangle in $\hat{\mathcal{S}}^n$ so that it is smaller than R, after a finite number of iterations the set $\hat{\mathcal{S}}^n$ will be empty. In the next iteration, by Lemma 1.6, it follows that R must be potentially optimal and divided. □

The division strategy used by `DIRECT` ensures that sampling occurs in every dimension. By dividing only along the longest side, the algorithm prevents the situation described in Figure 1.8 from happening.

Thus, since `DIRECT` eventually divides every hyperrectangle (shown in Corollary 1.1), and samples in every dimension, the set of points evaluated by `DIRECT` becomes dense in the domain. As stated above, this fact was used to show convergence of the algorithm in [35].

Figure 3.8 shows the points sampled by `DIRECT` on a sample test problem, the

Branin test function. The density of the points sampled appears centered around the global minima, and growing outward. If we allow DIRECT to run even longer on this problem, the dense cloud around the global minima grows.

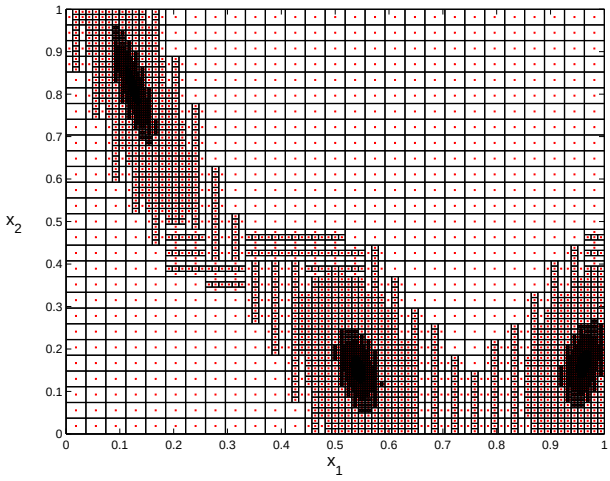

Figure 1.11: Points sampled by DIRECT on the Branin Test Function.

Although DIRECT convergence is due to the dense set produced, this strategy of the algorithm slows the convergence. Table 1.3 shows the cost of improving the error of the optimal value, $|f(x_{\min} - f(x^*)|$, on the Branin test function. As the table and

Table 1.3: Convergence of DIRECT slows down

Branin Test Function	
$\lvert f_{\min} - f^* \rvert$	Function Evals.
0.1	41
0.01	63
10^{-3}	117
10^{-4}	195
10^{-6}	377
10^{-7}	1295
10^{-8}	38455

Figure 3.8 suggest, improving the best point found by DIRECT can grow substantially in cost. In Chapter 3, we will show that one of the contributing factors to this characteristic of DIRECT is the balance parameter ϵ.

1.4 Modifications to DIRECT

Researchers have studied ways to improve the DIRECT algorithm (see, for example, [23, 24, 14, 29, 34]). The algorithm has been extended to handle both nonlinear and hidden constraints. A localized version of the algorithm has been introduced, as well as an aggressive form, designed for use on parallel computing systems. Different types of stopping criteria have also been explored. This section presents the motivation and descriptions for the modifications that have been made to DIRECT.

1.4.1 DIRECT-l

A locally biased version of DIRECT, named DIRECT-l was introduced in [24] and analyzed in [23]. DIRECT-l places a stronger emphasis on the objective function values at the hyperrectangle centers, and prioritizes these hyperrectangles for division.

Recall that, in the original implementation of DIRECT, hyperrectangles were measured by the distance from their center to a corner. In DIRECT-l, hyperrectangles are instead measured by the length of their longest side. This measure is traditionally known as the infinity norm, [26], and allows the algorithm to group more hyperrectangles at the same size.

For example, let the current state of the design space, Ω, is given in Figure 1.12.

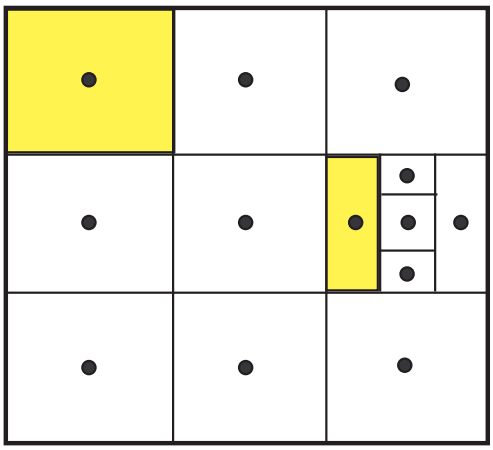

Figure 1.12: The highlighted hyperrectangles are the same size in DIRECT-l

The two highlighted hyperrectangles in Figure 1.12 are of different size, with respect to the original algorithm. Thus, theoretically, both could be deemed potentially optimal and divided in the current iteration. Dividing both of these hyperrectangles would create six new hyperrectangles, and would require six immediate calls to the objective function.

The modified version, DIRECT-l, acts differently. The two highlighted hyperrectangles have the same size, with respect to the infinity norm used by DIRECT-l. The rules for determining potentially optimal hyperrectangles allow for at most one hyperrectangle to be chosen for a given size; the one that has the smallest center value (recall Lemma 1.5). Therefore, DIRECT-l can only choose one of the highlighted hyperrectangles to be potentially optimal. This reduces the number of objective function calls for the current iteration (in our example, by either two or four, depending on which

hyperrectangle has a smaller objective function value), but still allows the algorithm to search areas that have promise.

In [24], `DIRECT-l` was compared to to the original `DIRECT` by running it on the same test problems that were used in [35]. Before we present the results, we define each of the nine test problems. This table was first introduced in [35].

Table 1.4: Characteristics of the Test problems

Test Function	Abbreviation	Number of dimensions	Number of local minima	Number of global minima
Shekel 5	S5	4	5	1
Shekel 7	S7	4	7	1
Shekel 10	S10	4	10	1
Hartman 3	H3	3	4	1
Hartman 6	H6	6	4	1
Branin RCOS	BR	2	3	3
Goldstein and Price	GP	2	4	1
Six-Hump Camel	C6	2	6	2
Two-Dimensional Shubert	SHU	2	760	18

The results are reprinted in Table 1.5, and were first reported in [24]. The algorithms were tested by comparing the number of function evaluations needed to find a solution whose function value was within 0.01% of the optimal value.

The results in Table 1.5 favor the local version, `DIRECT-l`. As the authors in [24] point out, this may be due to the relative simplicity of these problems. Clearly though, `DIRECT-l` seems to improve the efficiency of `DIRECT` on the tested problems.

The results in Table 1.5 also demonstrate the effects of the modifications in `DIRECT-l`. The convergence occurs in fewer function evaluations, but can take more

Table 1.5: Comparison of DIRECT and DIRECT-l

		S5	S7	S10	H3	H6	BR	GP	C6	SHU
DIRECT	Its.	15	15	15	14	21	15	14	13	135
	Fcn. Evals.	155	145	145	199	571	195	191	285	2967
DIRECT-l	Its.	15	15	15	14	21	17	14	20	280
	Fcn. Evals.	147	141	139	111	295	159	115	191	2043

sweeps (see, for example, the C6 problem). This is due to fewer hyperrectangles being chosen in each iteration, as was previously discussed.

1.4.2 Aggressive DIRECT

An aggressive version of DIRECT is described in [57, 4, 5]. In this version of DIRECT, the authors discard the idea of potentially optimal hyperrectangles, and divide a hyperrectangle of every size in each iteration.

Table 1.6 shows a comparision of DIRECT, DIRECT-l, and aggressive DIRECT on the same nine test problems described in [35]. Again, convergence is defined as $|f_{\min} - f^*| < 0.01\%$, where $f_{\min}$ is the best value found by the algorithm, and f^* is the global minimum value.

The numbers in Table 1.6 show that the aggressive version of DIRECT uses many more function evaluations than the other versions. This is to be expected, since the criteria for choosing hyperrectangles to be divided has been relaxed. In [4], it is described that this cost is offset by the use of a large parallel processing supercomputer. Although this approach does not appear to be favorable on these simple test prob-

Table 1.6: Comparision of DIRECT, DIRECT-l and aggressive DIRECT

		S5	S7	S10	H3	H6	BR	GP	C6	SHU
DIRECT	Its.	15	15	15	14	21	15	14	13	135
	Fcn. Evals.	155	145	145	199	571	195	191	285	2967
DIRECT-l	Its.	15	15	15	14	21	17	14	20	280
	Fcn. Evals.	147	141	139	111	295	159	115	191	2043
Aggressive	Its.	15	15	15	14	21	14	12	13	??
DIRECT	Fcn. Evals.	2615	2021	2037	1031	72531	513	375	785	$> 10^7$

lems, more difficult problems may be more easily solved by this strategy. A good example is [5].

Another interesting attribute of the aggressive version of DIRECT is that the cost of an iteration grows much faster than either of the other versions of DIRECT. This is because of the large number of hyperrectangles created by the algorithm. Figure 1.13 shows a comparison of the cost per iteration of DIRECT and the aggressive DIRECT on the H6 test problem.

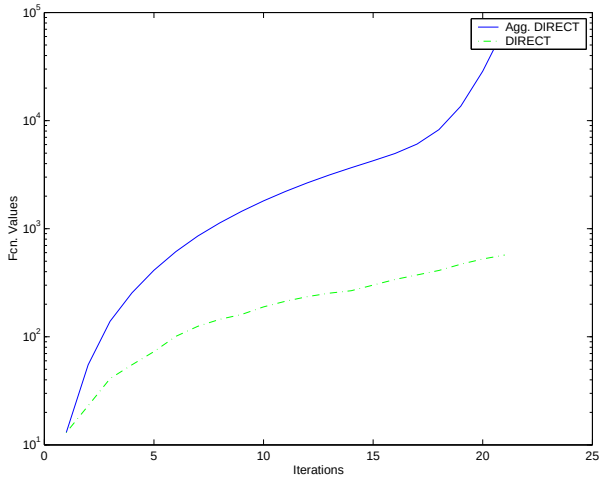

Figure 1.13: Cost per iteration of aggressive DIRECT grows quickly

The H6 test problem is a six dimensional problem. On problems of a larger dimension, the cost per iteration will grow even faster for aggressive DIRECT. In order for a thorough search to be performed, a large supercomputer may be necessary to overcome the high cost of later iterations.

1.4.3 DIRECT Modifications for Constraints

Many real-world optimization applications contain constraints [10, 47]. DIRECT requires constraints on the variables, but does not naturally address additional linear and nonlinear constraints. In this section, we review some constraint-handling methods from the literature that have been developed for use with DIRECT.

Jones Approach

The first method we describe was created by one of the original authors of DIRECT [34]. The method involves a non-traditional penalty function, and a heuristic for determining the penalty parameters. It is designed to solve problems of the form

$$(C)\qquad \begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g_1(x) \leq 0 \\ & \vdots \\ & g_m(x) \leq 0 \\ & l \leq x \leq u \end{aligned}$$

The Jones strategy for handling constraints begins by defining an "auxiliary" function:

$$\mathcal{A}(x; f^*) = \max(f(x) - f^*, 0)) + \sum_{j=1}^{m} c_j \max(g_j(x), 0), \tag{1.7}$$

where f^* is the global minimum value of f over the feasible region. The first term of (1.7) penalizes deviation from the global minimum, while the second term penalizes infeasibility. Clearly, the minimum value of $\mathcal{A}(x; f^*)$ is zero, and is obtained when $x = x^*$. Thus, minimizing $\mathcal{A}(x; f^*)$ is equivalent to minimizing the original objective function. In this scheme, potentially optimal hyperrectangles are determined based on the rate of change of $\mathcal{A}(x; f^*)$ for different values of f^*.

The penalty weights, c_j, are updated dynamically throughout the optimization. Every time a hyperrectangle is divided, the variable s_0 and s_j for $j = 1, ...m$ are updated such that

$$s_0 = s_0 + \sum_{\text{child=left}}^{\text{child=right}} \frac{|f(x^{\text{child}}) - f(x^{\text{mid}})|}{x^{\text{child}} - x^{\text{mid}}},$$

and

$$s_j = s_j + \sum_{\text{child=left}}^{\text{child=right}} \frac{|g_j(x^{\text{child}}) - g_j(x^{\text{mid}})|}{x^{\text{child}} - x^{\text{mid}}}.$$

Once the values of s_0 and s_j are updated, the penalty weights are calculated to be

$$c_j = \frac{s_0}{\max(s_j, 10^{-30})}.$$

The variables s_0 and s_j are meant to approximate rates of change of the objective function and the constraint violations, while c_j are meant to convert the constraint violation, $g_j(c)$, to potential increases in the objective function [34].

In this version of `DIRECT`, potentially optimal hyperrectangles are chosen based on the rate of change of $\mathcal{A}(x; f^*)$ for different values of f^*. The rate of change function for $\mathcal{A}(c; f^*)$ of hyperrectangle R is given by

$$h_R(c; f^*) = \frac{\mathcal{A}(c; f^*)}{\alpha(R)} = \frac{\max(f(c) - f^*, 0) + \sum_{j=1}^{m} c_j \max(g_j(c), 0)}{\alpha(R)}. \tag{1.8}$$

Note that (1.8) is piece-wise linear with respect to f^*. It is a flat line for $f^* > f(c)$, and a line with slope $(f(c) - f^*)/\alpha(R)$. Figure 1.14 illustrates how $h_R(c; f^*)$ is used to find potentially optimal hyperrectangles. Six hyperrectangles are compared in the Figure 1.14. Three of the hyperrectangles (2, 5, 4) are on the lower envelope of the picture; these are the potentially optimal hyperrectangles. For a specified value of f^*, one of the potentially optimal hyperrectangles has a minimal rate of change, $h_R(c; x^*)$, and is therefore selected for further search. If there are no constraints, this strategy is identical to the original method for choosing potentially optimal hyperrectangles [34]. If no feasible point has been found (and the point $f_{\min} - \epsilon|f_{\min}|$ cannot be determined), then the hyperrectangle that minimizes

$$\frac{\sum_{i=1}^{m} c_j \max(g_j(c), 0)}{\alpha(R)}$$

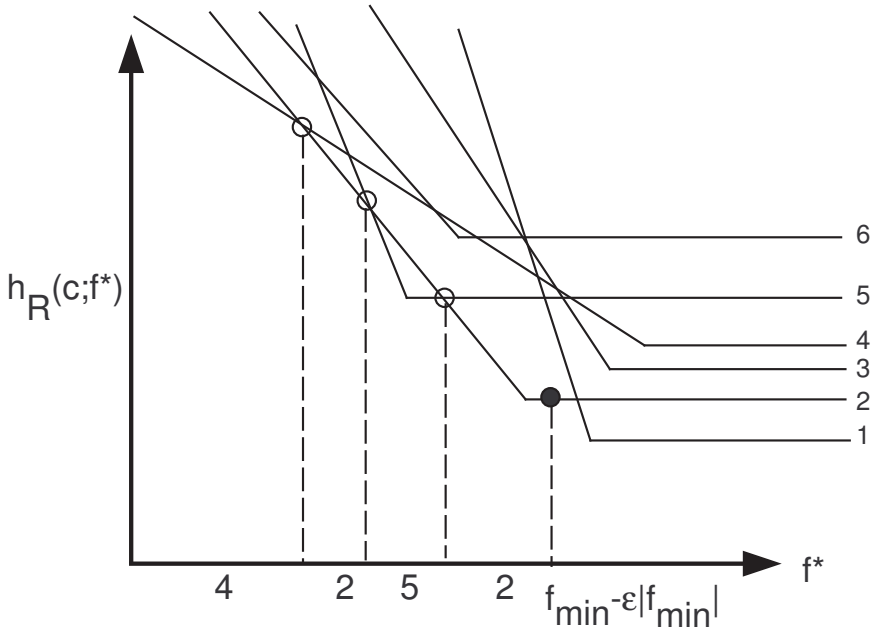

Figure 1.14: Plot of $h_R(c; f^*)$ vs f^*.

is selected as potentially optimal. A detailed description of this method can be found in [34].

Exact L1 penalty functions

A traditional exact L1 penalty function approach [19] has been implemented in our Matlab software[18]. An L1 penalty approach is a transformation of problem (C) to the form

$$(C') \quad \begin{array}{ll} \min & f(x) + \sum_{i=1}^{m} c_j g_j(x) \\ \text{subject to} & l \leq x \leq u, \end{array}$$

where c_j are penalty parameters. In `direct.m`, the user provides the software with values for the penalty parameters.

In [19, 16], theoretical convergence results are shown for L1 penalty problems, provided a sufficiently large penalty vector, c, is chosen. In practice, choosing penalty parameters is very important. On some problems, an extremely large penalty parameter is necessary for the algorithm to converge to a feasible point. Large penalty parameters are an issue for `DIRECT`, since large penalties could bias the algorithm away from hyperrectangles near the boundary of feasibility.

Barrier Approach

A barrier approach assigns a very large number to infeasible points. In [2, 3], barrier approaches are shown to be effective for generalized pattern search (GPS), and mesh adaptive direct search (MADS) methods.

Barrier approaches are not a good strategy for `DIRECT` [23]. A hyperrectangle with a large feasible area, but an infeasible center, will not be explored by `DIRECT` in a reasonable amount of time. A barrier approach overly biases `DIRECT` to feasible regions.

Neighborhood Assignment Strategy

In [23], a novel approach for addressing constraints with `DIRECT` was proposed. The neighborhood assignment strategy (NAS) assigns values to an infeasible point, x, relative to feasible values already found in the neighborhood around x. The NAS strategy was recommended by R. Carter [23].

For this approach, we look at problems formulated:

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & l \leq x \leq u \\ & x \in \mathcal{D}, \end{aligned}$$

where $\mathcal{D} \subset \mathbb{R}^n$. We do not assume any information about $\mathcal{D}$; for example, the space may be defined by implicit, "hidden" constraints unknown to the user. An oracle may return a yes/no answer about the feasibility of a point. In general, very little is known about $\mathcal{D}$.

Let's assume that hyperrectangle R has an infeasible center c. The NAS approach expands R to double it's size in every dimension. If this neighborhood has any feasible points in it, the value $F + \delta|F|$ is assigned as a surrogate function value to c. In this example, F is the low function value in the neighborhood around c, and δ is a parameter provided by the user. If no feasible points exist in the neighborhood around c, then $F_{\max} + 1$ is assigned as a surrogate function value at c, where $F_{\max}$ is the largest function value found so far. Figure 1.15 illustrates the two scenarios described.

The NAS approach prioritizes subdivision of hyperrectangles with infeasible centers based on function values in the nearby neighborhood. The hope is that `DIRECT` will not be biased away from the edges of feasible regions. The NAS approach does not need constraint violation values, unlike the Jones approach, or exact L1 penalty

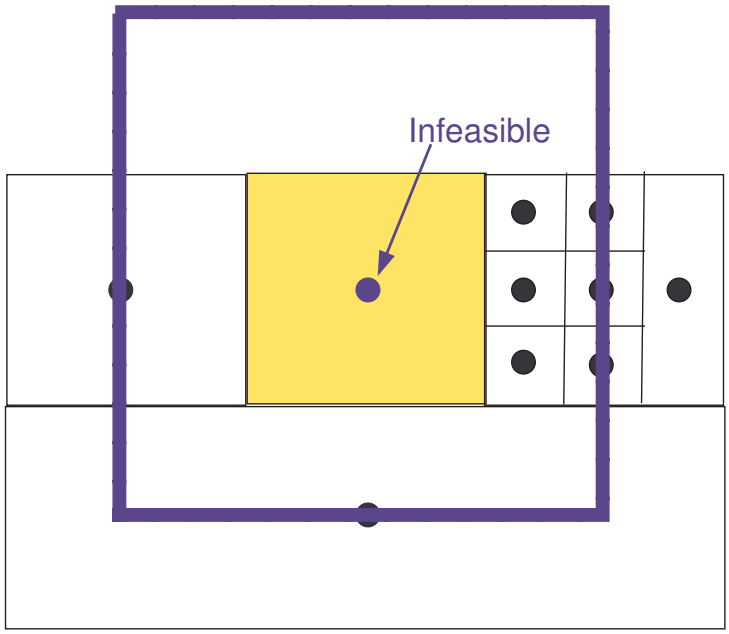

Figure 1.15: An illustration of the NAS strategy. Every feasible point within the enlarged rectangle is used to determine the surrogate value assigned to the infeasible point.

functions. Thus, NAS will work on problems with hidden constraints. In fact, the next comparison will show that NAS should only be used on hidden constraints.

We attempt to solve:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i^2 \leq 6, \end{aligned}$$

with `DIRECT`. The problem was suggested by John Dennis Jr. We compare three different constraint handling approaches (barrier, exact L1 penalty, and NAS) on the problem, with varying dimension n. The results are reported in Table 1.7, while

Figure 1.16 illustrates the points DIRECT samples with an exact L1 penalty approach.

Table 1.7: Function Evaluations needed by DIRECT to solve Problem (T1) with different tactics for handling constraints.

direct.m				
$n =$	2	3	4	5
Penalty Functions	129	935	3757	11385
NAS	225	3323	13211	59433
Barrier	911	55147	$> 10^5$	$> 10^5$

DIRECT converges much faster when exact L1 penalty functions are used to handle the constraint. The NAS does not use all the available information (constraint violations), thus it does not converge very fast. This example illustrates that NAS should only be used when constraint violation information cannot be ascertained. Other methods, like the Jones constraint approach, or exact L1 penalty functions should be used when the constraints are explicit, or violation information is known. Barrier functions should never be used with DIRECT.

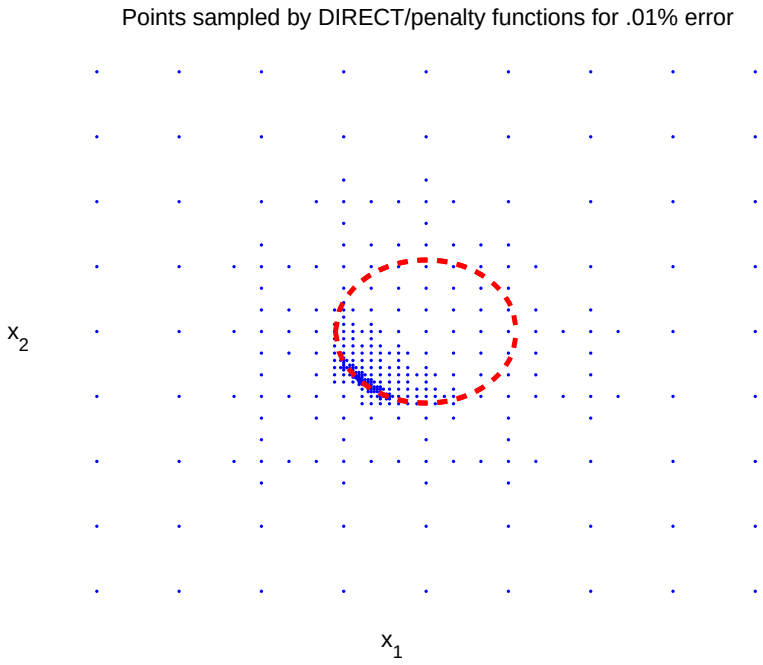

Figure 1.16: Points sampled by `DIRECT` with penalty functions on the test problem for $n = 2$.

Chapter 2

Convergence Analysis of DIRECT

A byproduct of the complex mathematical models that are used in scientific and engineering based applications is nonsmooth behavior (see [21] for example). Nonsmooth equations occur naturally in these models, and create additional challenges for optimization, as traditional gradient based methods may no longer produce accurate solutions. As was discussed in the introduction, DIRECT can be a viable alternative to traditional optimization methods when a nonsmooth problem needs to be solved. In this chapter, we use nonsmooth calculus techniques developed in [12, 51, 33] to describe the convergence of DIRECT. We will rigorously show that a subsequence of the points sampled by DIRECT cluster at nonsmooth KKT points.

Our analysis is built upon the observation made in [35] that the points sampled by DIRECT form a dense subset of Ω. We also follow the techniques laid out in [2, 3] for generalized pattern search (GPS) and mesh adaptive direct search (MADS)

algorithms. Our results mimic the results found for MADS, despite the fact that DIRECT behaves differently from MADS.

The next section provide background information about pertinent nonsmooth analysis results from the literature. In Section 2.2, we present our results. In the last section, § 2.3, we explore the rate of convergence of DIRECT on some simple examples.

2.1 Background

2.1.1 Directional Cones

Directional cones play an important role in our analysis. There are three important cones that we will consider.

The first cone that we discuss is the hypertangent cone. The definitions that we introduce are originally from [12, 33, 51].

Definition 2.1. *A vector $v \in \mathbb{R}^n$ is said to be a hypertangent vector to the set $\mathcal{D} \subset \mathbb{R}^n$ at the point $x \in \mathcal{D}$ if there exists a scalar $\epsilon > 0$ such that*

$$y + tw \in \mathcal{D} \quad \text{for all } y \in \mathcal{D} \cap B_\epsilon(x), \quad w \in B_\epsilon(v) \text{ and } 0 < t < \epsilon. \tag{2.1}$$

The set of hypertangent vectors to $\mathcal{D}$ at x is called the hypertangent to $\mathcal{D}$ at x and is denoted by $T_{\mathcal{D}}^H(x)$.

The definition of the hypertangent cone is technical, but the cone is important for the general results. Figure 2.1 illustrates the hypertangent cone on a sample problem. On the right side of Figure 2.1, we see the hypertangent cone for x over $\mathcal{D}$. Note that the $T_{\mathcal{D}}^{H}(x)$ is an open set; directions that lie on its boundary do not satisfy Definition 2.1.

The next cone we consider is the Clarke cone.

Definition 2.2. *A vector $v \in R^n$ is said to be a Clarke tangent vector to the set $\mathcal{D} \subset \mathbb{R}^n$ at the point x in the closure of $\mathcal{D}$ if for every sequence $\{y_k\}$ of elements of $\mathcal{D}$ that converges to x and for every sequence of positive real numbers $\{t_k\}$ converging to zero, there exists a sequence of vectors $\{w_k\}$ converging to v such that $y_k + t_k w_k \in \mathcal{D}$. The set $T_{\mathcal{D}}^{Cl}(x)$ of all Clarke tangent vectors to $\mathcal{D}$ at x is called the Clarke tangent cone to $\mathcal{D}$ at x.*

If $\mathcal{D} = \Omega$, then the Clarke cone is equivalent to every feasible direction at a point x. We state this formally as a lemma.

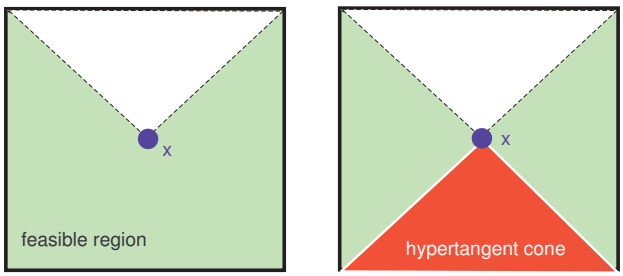

Figure 2.1: On the left, the shaded region is the feasible domain. The right illustrates the hypertangentone cone to $\mathcal{D}$ at x

Lemma 2.3. *Suppose that $\mathcal{D} = \Omega$, i.e. the set $\mathcal{D}$ is defined by simple bound constraints. If $x \in \mathcal{D}$, then*

$$T_{\mathcal{D}}^{Cl}(x) = \bar{T} = \{v \in \mathbb{R}^n | x + tv \in \mathcal{D} \text{ for all } t > 0 \text{ sufficiently small}\}. \tag{2.2}$$

Proof. This result follows from the fact that linear constraints define $\mathcal{D}$. Clearly, $\bar{T} \subseteq T_{\mathcal{D}}^{Cl}(x)$. Next, assume that $v \in T_{\mathcal{D}}^{Cl}(x)$. If no $\bar{t} > 0$ exists such that $x + tv \in \mathcal{D}$ for $0 < t < \bar{t}$, then there exists an $\epsilon > 0$ such that $x + t\bar{w} \notin \mathcal{D}$ for all $w \in B_\epsilon(v)$ (since $\mathcal{D}$ is closed). Since $y_k + t_k w_k = x + tw$ for some small t and $w \in B_\epsilon(v)$ we arrive at a contradiction. □

Figure 2.2 is an illustration of the Clarke cone. in this example, the Clarke cone is the closure of the hypertangent cone. This is true whenever $T_{\mathcal{D}}^{H}(x)$ is non-empty [51].

The last directional cone we consider is the contingent cone.

Definition 2.4. *A vector $v \in \mathbb{R}^n$ is said to be a tangent vector to the set $\mathcal{D} \subset \mathbb{R}^n$*

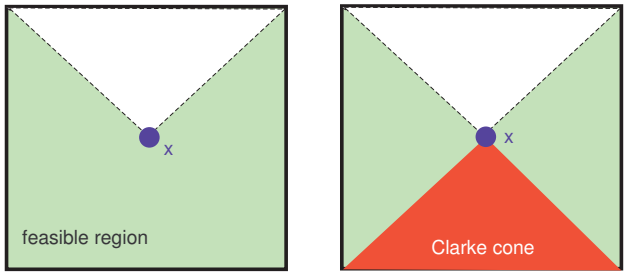

Figure 2.2: On the left, the shaded region is the feasible domain. The right illustrates the Clarke cone to $\mathcal{D}$ at x

at the point x in the closure of $\mathcal{D}$ if there exists a sequence $\{y_k\}$ of elements of $\mathcal{D}$ that converges to x and a sequence of positive real numbers $\{\lambda_k\}$ for which $v = \lim_k \lambda_k(y_k - x)$. The set $T_{\mathcal{D}}^{Co}(x)$ of all tangent vectors to $\mathcal{D}$ at x is called the contingent cone (or sequential Bouligand tangent cone) to $\mathcal{D}$ at x.

The contingent cone is the most general of the three cones; in fact, it was pointed out in [3] that

$$T_{\mathcal{D}}^{H}(x) \subseteq T_{\mathcal{D}}^{Cl}(x) \subseteq T_{\mathcal{D}}^{Co}(x).$$

Figure 2.3 is an illustration of the contingent cone for our ongoing example.

Regularity is defined next.

Definition 2.5. *The set $\mathcal{D}$ is said to be regular at x provided $T_{\mathcal{D}}^{Cl}(x) = T_{\mathcal{D}}^{Co}(x)$.*

Note that our example in this section (Figures 2.1, 2.2 and 2.3) is not regular. It was shown in [12] that any convex set is regular.

The three cones that we have described play important roles in our analysis. Next, we review Clarke derivatives and review the corresponding necessary conditions.

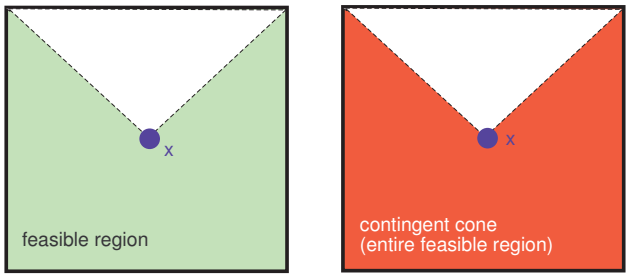

Figure 2.3: On the left, the shaded region is the feasible domain. The right illustrates the contingent cone to $\mathcal{D}$ at x

2.1.2 Nonsmooth Analysis

This section is concerned with the generalized directional derivative and the generalized gradient, which were first introduced in [12]. We provide a small example to aid in understanding of the subject, and then review necessary conditions for convergence.

Following [12], we assume $f : X \subset \mathbb{R}^N \to \mathbb{R}$ where X is a Banach space. We assume f to be locally Lipschitz continous near x. The *generalized directional derivative* of f at x in direction v is given by

$$f^o(x; v) = \limsup_{y \to x; t \downarrow 0} \frac{f(y + tv) - f(y)}{t}. \tag{2.3}$$

Note that the above definition differs from a traditional directional derivative in that the base point of the difference quotient, y, varies. The generalized directional derivative of f in direction v chooses a sequence that converges to x and converges to a maximum difference quotient.

The *generalized gradient* of f at x is the subset of the dual space of X, X^*, given by

$$\left\{\zeta \in X^* : f^o(x; v) \geq \zeta^T v \text{ for all } v \text{ in } X\right\}, \tag{2.4}$$

We denote the generalized gradient of f at x by $\partial f(x)$. In [12], it is shown that the above definition is equivalent to describing the generalized gradient as

$$\partial f(x) = conv\left\{\lim_{n \to \infty} \nabla f(x_n) : x_n \in D_f, x_n \to x\right\}, \tag{2.5}$$

where *conv* denotes the convex hull, and D_f is the set of points where f is differentiable. In [12], it was shown that $\partial f(x)$ is a nonempty, convex, compact set. When the generalized gradient is a singleton, the function is smooth and differentiable, in the traditional sense, at that point.

This next theorem, from [12], generalizes the smooth first order necessary conditions.

Theorem 2.1. *If f attains a local minimum at x^*, then $0 \in \partial f(x^*)$.*

Once again, if f is differentiable at x^*, then $\partial f(x^*)$ is the singleton 0, and the above theorem reduces to the normal first order optimality conditions.

The problems we consider our constrained, therefore we will use a modified version of (2.3), proposed in [33]:

$$f^o(x;d) = \limsup_{\substack{y \to x, y \in \Omega \\ t \downarrow 0, y+tv \in \Omega}} \frac{f(y+td) - f(y)}{t}. \tag{2.6}$$

Unlike directional derivatives, generalized directional derivatives are not linear in the direction; that is

$$f^o(x;u+v) \neq f^o(x;u) + f^o(x;v) \tag{2.7}$$

Before we present our results, we give a short example.

2.1.3 Example

Let $f(x) = |x|$. A plot is shown in Figure 2.4. Note that f is Lipschitz (with Lipschitz

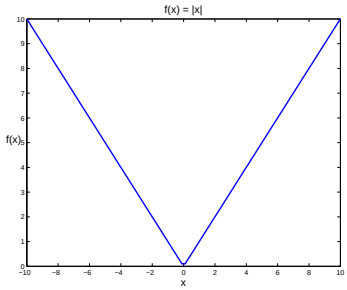

Figure 2.4: Construction of $\hat{x}$.

constant $K = 1$), and is differentiable everywhere except $x = 0$. We now calculate the generalized directional derivatives. If x is strictly positive then

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

The generalized gradient at x is

$$\{\zeta \in \mathbb{R} : v \geq \zeta v\},$$

or the set containing the single value of 1. This is consistent with our discussion above; that is, $\nabla f(x) = \partial f(x)$, when f is differentiable at x.

Similarly, if x is strictly negative then

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

The generalized gradient at x, when x is strictly negative is the set containing the singleton -1. Again, the generalized gradient is equivalent to the gradient of f at x

because the function f is differentiable there.

Finally, we consider the last case, when $x = 0$. Note that f has no gradient at $x = 0$. However, we calculate that

$$f^0(x; v) = \begin{cases} v & \text{if } v \geq 0 \\ -v & \text{if } v \leq 0 \end{cases} = |v|,$$

and that

$$\partial f(0) = \{\zeta \in \mathbb{R} : |v| \geq \zeta v\} = \{-1 \leq \zeta \leq 1\}.$$

Finally, we verify that generalized directional derivatives are not linear in directions. Note that $f^o(x; -1/2) = f^o(x; 1 + \frac{3}{2}(-1)) \neq f(x; 1) + \frac{3}{2} f(x; -1) = \frac{5}{2}$.

2.2 Convergence Analysis of DIRECT

In previous sections, we introduced some important concepts in nonsmooth analysis. We now use the concepts to state convergence conditions for DIRECT. Our results are based on the observation made in [35] that DIRECT samples from a dense subset of Ω. In [23], it was shown that this set is

$$\mathcal{S} = \left\{ x | x = \sum_{i=1}^{N} \frac{2n_i + 1}{2 \cdot 3^{k_i}} e_i \right\}, \tag{2.8}$$

where e_i is the ith coordinate vector, $k_i > 0$, and $0 \leq n_i \leq 2 \cdot 3^{k_i} - 1$.

We now introduce some notation. Let $\mathcal{S}_k$ be the set of points sampled by DIRECT

after k iterations, and let

$$\mathcal{B}_k = \{x \in \mathcal{S}_k | f(x) \leq f(z) \text{ for all } z \in \mathcal{S}_k\}, \tag{2.9}$$

that is, $\mathcal{B}_k$ is the set of best points found by DIRECT after k iterations. Since DIRECT has an exhaustive search property [35],

$$\mathcal{S} = \bigcup \mathcal{S}_k.$$

We define $\mathcal{B} = \bigcup \mathcal{B}_k$. Our analysis focuses on cluster points of $\mathcal{B}$.

2.2.1 Simple Bound Constraints

We begin our analysis by examining the case when $\mathcal{D} = \Omega$. The formal statement of our convergence result is

Theorem 2.2. *Let f be Lipschitz continuous on $\mathcal{D} = \Omega$, and let x^* be any cluster point of $\mathcal{B}$. Then,*

$$f^o(x; d) \geq 0 \text{ for all } d \in T_{\mathcal{D}}^{Co}(x^*) \tag{2.10}$$

Proof. We will show that (2.10) holds with an indirect proof. Assume that $f^o(x^*; v) < 0$ for some $v \in T_{\mathcal{D}}^{Cl}(x^*)$. We will exhibit K and $\Delta > 0$ such that

$$\inf_{k \geq K} \text{dist } (x^*, \mathcal{B}_k) \geq \Delta, \tag{2.11}$$

contradicting the assumption that x^* is a limit point of $\mathcal{B}$. Since $\mathcal{D}$ is regular (by convexity), our result follows.

Since $f^o(x^*; v) < 0$, and $v \in T_{\mathcal{D}}^{Cl}(x^*)$, there is $\delta > 0$ such that

$$y^* = x^* + \delta v \in \mathcal{D},$$

and $f(y^*) < f(x^*)$.

Let L_f denote the Lipschitz constant of f. Let

$$\Delta = \min \left\{ \delta/2, \frac{f(x^*) - f(x^* + \delta v)}{2L_f} \right\} \tag{2.12}$$

and let

$$\mathcal{N} = \{x | \|x - x^*\| \leq \Delta\} \cap \mathcal{D}. \tag{2.13}$$

For all $x \in \mathcal{N}$,

$$f(x) - f(y^*) \geq f(x^*) - L_f \Delta - f(y^*) \geq \frac{f(x^*) - f(y^*)}{2} > 0. \tag{2.14}$$

Since $\mathcal{S}$ is dense in $\mathcal{D}$, there is $K > 0$ and $\hat{x} \in \mathcal{S}_K$ such that

$$\|\hat{x} - y^*\| \leq \Delta/2,$$

and hence, for all $x \in \mathcal{N}$,

$$f(\hat{x}) \leq f(y^*) + L_f \Delta/2 \leq f(y^*) + \frac{f(x^*) - f(y^*)}{4} < f(x). \tag{2.15}$$

Hence,

$$\mathcal{N} \cap \mathcal{B}_k = \emptyset,$$

for all $k \geq K$, as asserted. Again, since $\mathcal{D}$ is a regular set, the result is verified. □

When the objective function, f, is differentiable at the limit point,x^*, of centers of strongly optimal hypercubes, we see that x^* is a KKT point. We show this formally in Corollary 2.1.

Corollary 2.1. *Let x^* be any limit point of a sequence of centers of strongly optimal hypercubes. If f is differentiable in a neighborhood about x^*, then x^* satisfies the Karush-Kuhn-Tucker first order optimality conditions.*

Proof. Recall that if x^* is a KKT point, then the negative gradient of f at x^* lies in a cone of the gradients of the active constraints. That is, if we are trying to solve

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad 1 \leq i \leq M \end{aligned}$$

and x is a KKT point, then

$$-\nabla f(x) = \sum_{i \in \mathcal{A}} \lambda_i \nabla g_i(x), \tag{2.16}$$

where $\mathcal{A} = \{i \in (1, M) : g_i(x) = 0\}$ is the set of active constraints, and each $\lambda_i \geq 0$. Recall that our problem is constrained so that

$$x - u \leq 0, \qquad l - x \leq 0,$$

thus Equation 2.16 reduces to

$$-\nabla f(x) = \sum_{i \in \mathcal{A}} \pm \lambda_i e_i, \tag{2.17}$$

where the sign in front of e_i depends on which constraint is active.

We let x^* be a limit point of centers of strongly optimal hypercubes.

If x^* is in the interior of Ω, we show that $\nabla f(x^*) = 0$. The Taylor expansion about x^* shows that

$$f(x^* + tv) = f(x^*) + t\nabla f(x^*)^T v + o(t), \tag{2.18}$$

for sufficiently small t. In Theorem 2.2 we showed that $\nabla f(x^*)^T v \geq 0$, for all feasible v. Since x^* is in the interior of Ω, all v are feasible. It follows $f(x^* + tv) \geq f(x^*)$ and

that

$$t\nabla f(x^*)^T v + o(t) \geq 0.$$

Thus, Equation 2.16 is satisfied when x^* is in the interior of Ω.

Next, we assume that x^* is not in the interior; that is, the set $\mathcal{A}$ is non-empty. Assume that $-\nabla f(x^*) \neq \sum_{i\in\mathcal{A}} \pm\lambda_i e_i$; that is,

$$-\nabla f(x^*) = \sum_{i\in\mathcal{A}} \pm\lambda_i e_i + \sum_{j\in\bar{\mathcal{A}}} \pm\lambda_j e_j,$$

where $\bar{\mathcal{A}}$ is the set of inactive constraints, and λ_j are not all zero for $j \in \bar{\mathcal{A}}$. Say $\lambda_j > 0$ for some j. Then,

$$-\nabla f(x^*)^T e_j = \lambda_j,$$

which contradicts the results of Theorem 2.2. Thus, the KKT condition is satisfied for a limit point of centers of strongly optimal hypercubes. □

2.2.2 Constrained Case

In this section, we extend our results to a more general domain, i.e. when $\mathcal{D} \subset \Omega$.

Bound constraints are part of DIRECT's sampling strategy, and are incorporated automatically into the optimization. DIRECT can address more general constraints by assigning an artificial value to the objective function at an infeasible point. One way to do this is the barrier method, i.e. assigning a very large value to infeasible centers. Another approach is the neighborhood assignment strategy that was used

in [10], and incorporated into our code [18]. Here the artificial value assigned to an infeasible point changes as the optimization progresses; this method was described in § 1.4.3. An exact L1 penalty function could also be used, provided the penalty parameter was large enough [19]. For the following results, we will assume very little about the heuristic used to handle general constraints on the problem. We will only assume that (*i*) $\mathcal{D} \subset \Omega$, and that if $z \notin \mathcal{D}$, then the value assigned by DIRECT to z is larger than $f_{\min}$. That is, we assume

$$\mathcal{B}_k \subset \mathcal{D} \text{ for all } k.$$

Our first result demonstrates that $f^o(x^*; v) \geq 0$ for all $v \in T_{\mathcal{D}}^H(x^*)$.

Theorem 2.3. *Let f be Lipschitz continuous on $\mathcal{D}$ and let x^* be any cluster point of $\mathcal{B}$. Then $f^o(x^*; v) \geq 0$ for all $v \in T_{\mathcal{D}}^H(x^*)$.*

Proof. The proof is a simple extension of Theorem 2.2. Assume that $f^o(x^*; v) < 0$ for some $v \in T_{\mathcal{D}}^H(x^*)$. Then, there exists $0 < \delta < \epsilon$ such that

$$y^* = x^* + \delta v \in \mathcal{D},$$

and $f(y) < f(x^*)$.

Define Δ and $\mathcal{N}$ by (2.12) and (2.13) and let $\mathcal{D}^C$ denote the compliment of $\mathcal{D}$. We have

$$\left(\mathcal{N} \cap \mathcal{D}^C\right) \cap \mathcal{B}_k = \emptyset \tag{2.19}$$

because $\mathcal{B}_k \subset \mathcal{D}$.

Since $\mathcal{S}$ is dense in $\mathcal{D}$, it follows that there exists K and $\hat{x} = x^* + tw \in \mathcal{S}_K$, for some $0 \leq t \leq \delta$ and $w \in B_\epsilon(v)$. Since $v \in T_{\mathcal{D}}^H(x^*)$ it follows that $\hat{x} \in \mathcal{D}$. Furthermore, we choose $\hat{x}$ so that $\|w - v\|$ and $\|t - \delta\|$ are small enough to ensure that

$$\|\hat{x} - y^*\| = \|tw - \delta v\| \leq \Delta/2.$$

As already shown in (2.14) and (2.15), $f(\hat{x}) < f(x)$ for all $x \in \mathcal{N} \cap \mathcal{D}$, so

$$(\mathcal{N} \cap \mathcal{D}) \cap \mathcal{B}_k = \emptyset, \tag{2.20}$$

and thus (from (2.19) and (2.20)),

$$\mathcal{N} \cap \mathcal{B}_k = \emptyset,$$

for all $k \geq K$, which proves our assertion. □

In general, the hypertangent cone and the contingent cone are not the same; thus, the Theorem 2.3 is, in general, different from Theorem 2.2. To argue results as strong as Theorem 2.2 for the general case, we need to assume properties about $\mathcal{D}$.

Theorem 2.4. *Let x^* be a cluster point of $\mathcal{B}$. If $T_{\mathcal{D}}^H(x^*) \neq \emptyset$, then $f^o(x^*; v) \geq 0$ for all $v \in T_{\mathcal{D}}^{Cl}(x^*)$.*

Proof. By Theorem 2.3, $f^o(x^*; w) \geq 0$ for all $w \in T_{\mathcal{D}}^H(x^*)$. If $T_{\mathcal{D}}^H(x^*)$ is non-empty,

then [3],

$$f^o(x^*; v) = \lim_{\substack{w \to v \\ w \in T_{\mathcal{D}}^H(\hat{x})}} f^o(\hat{x}; w) \geq 0,$$

as asserted. □

We conclude this section by extending our constrained result to differentiable functions. Our final result relies on the simple observation regarding contingent cones and DIRECT. A similar observation was made in [3] regarding MADS algorithms.

Corollary 2.2. *Let x^* be a cluster point of $\mathcal{B}$. If $T\mathcal{D}^H(x^*) \neq \emptyset$, and if $\mathcal{D}$ is regular at x^*, then $f^o(x^*; v) \geq 0$ for all $v \in T_{\mathcal{D}}^{Co}(x^*)$.*

Proof. This is a direct result of Theorem 2.4 and the definition of a regular cone of directions. □

Our final results is an extension of Corollary 2.2. If we assume strict differentiability of f, then we can state the constraint qualifications needed to show that cluster points of $\mathcal{B}$ are KKT points.

Theorem 2.5. *Let f be strictly differentiable, and let x^* be a cluster point of $\mathcal{B}$. If $T_{\mathcal{D}}^H(x^*) \neq \emptyset$, and if $\mathcal{D}$ is regular at x^*, then x^* is a contingent KKT stationary point of f over $\mathcal{D}$.*

Proof. As pointed out in [3, 12], strict differentiability of f at x^* implies that $\nabla f(x^*)^T v = f^o(x^*; v)$ for all $v \in T_{\mathcal{D}}^{Co}(x^*)$. Thus, it follows from the previous corollary that $-\nabla f(x^*)^T v \leq 0$ for all v in the contingent cone. □

2.3 Convergence Rates and DIRECT

The rate of convergence for DIRECT is highly dependent on both the objective function being minimized, and also the geometry of the problem. In this section, we explore these ideas on very trivial examples. The results show how difficult determining a rate of convergence for DIRECT can be.

For the remainder of this section, we assume that ϵ, the balance parameter , is set to zero. In the next chapter, we see the impact that ϵ has on the convergence of DIRECT. By setting $\epsilon = 0$, DIRECT always divides the hyperrectangle with contains the smallest value found.

2.3.1 Linear Objective functions

The DIRECT algorithm's ability to solve optimization problems with a linear objective function were studied in [23]. Here, we are able to determine the rate of convergence that DIRECT by using strongly optimal hypercubes.

Recall that a linear function being minimized over a box-constrained domain has a global minima in one of the corners of the box, and this minimum may or may not be unique. If the contours of the linear function, $f(x) = a^T x$, are parallel to one of the constraints, then the minimum value is not unique, and it located along one of the boundaries. We define

$$X^* = \{x^* \in \Omega : f(x^*) = f^*\},$$

where f^* is the optimal minimum value of f over Ω.

Our next result is fundamental to determining the rate that DIRECT converges to the global minimum of a linear function.

Lemma 2.6. *Let $f(x) = a^T x$, where a is a $N \times 1$ constant vector. Let $\Omega = \left\{x \in \mathbb{R}^N : 0 \leq x_i \leq 1\right\}$ be the domain over which we wish to minimize f with DIRECT. Then,*

- *DIRECT finds a strongly optimal hypercube every N iterations.*
- *X^*, the set of optimal solutions, lie on the boundary of every strongly optimal hypercube.*

Proof. This result is due to the linear contours of a linear function. We begin by defining our notation, and making some assumptions.

The number of zero elements of a determine whether or not the optimal solution is unique. Figure 2.5 shows a two dimensional example. In each part, the number of zero elements is varied, and the red part shows the impact on the global minimum. In Figure 2.5 (a), no elements of the vector a are zero thus the global minimum is unique and lies in one of the corners. In (b) and (c), the global minimum is no longer unique, because elements of a are zero.

We let

$$J = \{i : a_i = 0, 1 \leq i \leq N\},$$

and assume that $|J| = j$, where $0 \leq j \leq N$. Without loss of generality, we assume

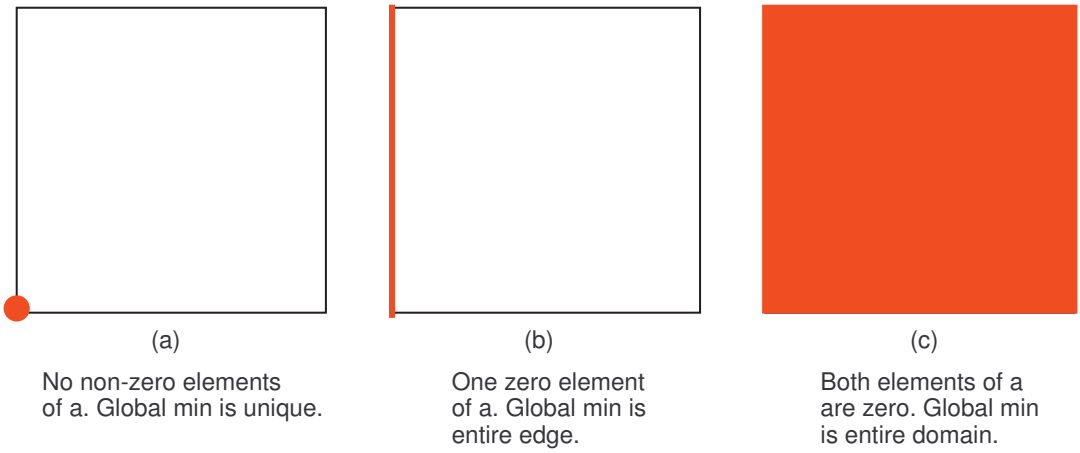

Figure 2.5: Unique and non-unique global minima of a linear function.

that $a_i > 0$, for $i \notin J$, and we also assume that

$$a_1 \geq a_2 \geq ... \geq a_{N-j} > a_{N-j+1} = ... = a_N = 0.$$

This assumption implies that if

$$x^* = (0, 0, ..., 0, x_{N-j+1}, ..., x_N)^T,$$

where $x_i^* \in [0, 1]$ for $N - j + 1 \leq i \leq N$, then $x^* \in X^*$. We define d_i to be the ith unit vector (i.e. a one in the ith position, and zeros everywhere else). We let $\mathcal{S}_k$ be the set of points sampled by DIRECT after k iterations, and we define

$$f_{\min} = \min_{x \in \mathcal{S}_k} f(x).$$

We will prove the Lemma by induction. Our first step is to show that in the first N iterations of DIRECT, a strongly optimal hypercube is found, and it contains a global minimum.

Iteration 1:

In the first iteration, DIRECT samples the objective function at the points $c \pm \frac{1}{3} d_i$, where

$$c = \left(\frac{1}{2}, ..., \frac{1}{2} \right)^T.$$

Due to our assumptions on the structure of the vector a,

$$f_{\min} = f(x_{\min}) = \begin{cases} f(c - \frac{1}{3} d_1), & \text{if } j \neq N \\ f(c), & \text{if } j = N. \end{cases}$$

and we say that $f_{\min}$ is found in the center of hyperrectangle R_1. If $j = N$, (i.e f is the constant zero function), then we will satisfy the first step of induction in the next iteration, as rectangle R_1 will be

- the smallest hypercube in the domain
- has $f_{\min}$ as its center
- X^* lies along the hypercube boundary (actually, it's the entire space Ω).

Thus, we assume $j \neq N$, and continue.

Because of the division rules of DIRECT, the hyperrectangle R_1 has one short side of length $\frac{1}{3}$, $N - 1$ long sides of length one, and X^* lies along a border.

Also, every other hyperrectangle created by DIRECT in the first iteration will have a short side in dimension d_1. Again, this is due to the division strategy of DIRECT

Iteration 2:

No strongly optimal hypercubes are found in this iteration if $j < N$ since $f_{\min}$ is not in the smallest hyperrectangle.

Hyperrectangle R_1 is potentially optimal in the second iteration, since its center is $f_{\min}$. When it's divided, the points sampled are

$$c - \frac{1}{3}d_1 \pm \frac{1}{3}d_k, \quad 2 \leq k \leq N.$$

After the second iteration, we see that

$$f_{\min} = f(x_{\min}) = \begin{cases} f(c - \frac{1}{3}d_1 - \frac{1}{3}d_2), & \text{if } j < N-1 \\ f(c - \frac{1}{3}d_1), & \text{if } j = N-1 \end{cases},$$

and we say that $f_{\min}$ is found in the center of hyperrectangle R_2. Note that R_2 has two short sides, in dimensions d_1 and d_2, and $N-2$ long sides. Again, no strongly optimal hypercube is found, since $f_{\min}$ was not in the smallest hypercube.

Iteration $N - j$:

No strongly optimal hypercube has been found up to this point, since up to this

point $f_{\min}$ has not been in a hyperrectangle with all short sides.

After $N - j$ iterations, we see that

$$f_{\min} = f(x_{\min}) = f(c - \frac{1}{3}\sum_{i=1}^{N-j} d_i),$$

and we say that $f_{\min}$ is located at the center of hyperrectangle R_{N-j}. Hyperrectangle R_{N-j} has $N - j$ short sides, in dimensions $d_1, ..., d_{N-j}$, and j long sides in the remaining dimensions.

If $j = 0$, then R_{N-j} has no long sides, has the smallest value found as its center, and X^*, which is a singleton, is in one of R_{N-j} corners; thus, we have shown the hypothesis. If $j > 0$, hyperrectangle R_{N-j} is divided in the current iteration (since it's center is the smallest value found), and we move on to the next iteration.

When R_{N-j} is divided, the points

$$c - \frac{1}{3}\sum_{i=1}^{N-j} e_i \pm \frac{1}{3}e_k, \quad N - j + 1 \leq k \leq N$$

are sampled by DIRECT. Since f is linear, and $a_k = 0$, we know that

$$f(c - \frac{1}{3}\sum_{i=1}^{N-j} e_i \pm \frac{1}{3}e_k) = f(c - \frac{1}{3}\sum_{i=1}^{N-j} e_i).$$

This fact, combined with the fact that every other point sampled must be larger

than $f(c - \frac{1}{3}\sum_{i=1}^{N_j} e_i)$ draws us to the conclusion that $x_{\min}$ from the previous iteration is still the location of $f_{\min}$. The hyperrectangle in which $x_{\min}$ resides now has no long sides, and is therefore strongly optimal. The set of global minima, X^*, is along one of the boundaries, thus we have shown the hypothesis of the Lemma is true for the first strongly optimal hypercube.

Our next step is to state the induction hypothesis. We assume that the kth strongly optimal hypercube was found after at most $2kN$ iterations, and the global minimum of f lays along one of the boundaries of the strongly optimal hypercube.

Implicit in our induction assumption is that the center of the kth strongly optimal hypercube, say c_k, is the closest point to any of the boundaries of the problem. This is due to the "center- out" sampling strategy of DIRECT. If there was a point, say $\tilde{c}_k$, that was closer to a boundary of Ω, then the hyperrectangle in which the point $\tilde{c}_k$ resided in would have a side with length less than 3^{-k}. If this were true, then there would exist a hyperrectangle smaller than R_k, a fact due to the division strategy of DIRECT. Since no hyperrectangle is smaller than R_k, we know that no point $\tilde{c}_k$ can exist.

Once we divide the strongly optimal hypercube R_k, our argument proceeds as it did for the initial step. No points from another hyperrectangle can interfere with this process, since they are farther away from any of the active boundaries, thus we have the identical process as we did in the first iteration. □

Lemma 2.6 leads us to the next result, which gives a rate of convergence for

DIRECT in terms of strongly optimal hypercubes. A numerical example follows. Since the optimal solution of a linear problem need not be unique, we define

$$X^* = \{x^* \in \Omega : f(x^*) = f^*\},$$

where f^* is the optimal value of f over the design space, Ω. We then define, for some $x \in \Omega$,

$$\|x - X^*\| = \min_{x* \in X^*} \|x - x^*\|.$$

Theorem 2.6. *Let $f(x) = a^T x$, i.e. f is a linear function, let x_k be the center of a strongly optimal hypercube with length 3^{-k}, and let $\{x_k\}$ be the corresponding sequence. Finally, let $e_k = \|x_k - X^*\|$. Then,*

$$e_{k+1} = \frac{1}{3} e_k.$$

Proof. In Lemma 2.6, we showed that every strongly optimal hyperrecgtangle has the global minimum on one of its boundaries. If x^* is a unique global minimum, then it lies in the corner of each strongly optimal hyperrectangle; if it is non-unique then it is along a edge, side, facet, etc..., depending on the dimension and the number of zero elements of a.

If x^* is unique, then it's in the corner, and we know (from Lemma 3.2) that

$$e_k = \|x_k - X^*\| = \frac{3^{-k}}{2}\sqrt{N}.$$

Consequently,

$$e_{k+1} = \|x_{k+1} - X^*\| = \frac{3^{-k-1}}{2}\sqrt{N} = \frac{1}{3}e_k.$$

If x^* is not unique (i.e. say there are j zero elements of a), then

$$e_k = \|x_k - X^*\| = \frac{3^{-k}}{2}\sqrt{N-j}.$$

This result is due to the fact that the closest point to x_k is simply the center of the face of X^*. Figure 2.6 shows a 3-D example. Consequently,

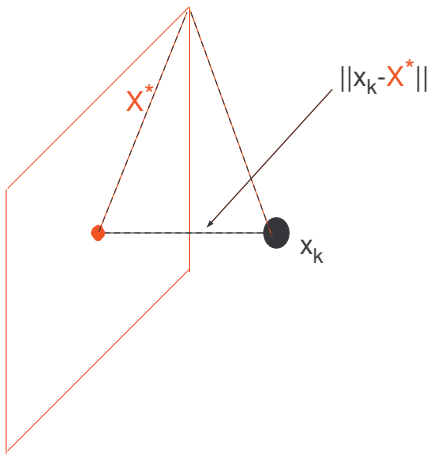

Figure 2.6: A visual explanation of $\|x_k - X^*\|$.

$$e_{k+1} = \|x_{k+1} - x^*\| = \frac{3^{-k-1}}{2}\sqrt{N-j} = \frac{1}{3}e_k.$$

□

Theorem 2.6 was stated in this way so that the convergence can be described as

N step, $q-$linear. Here, N is the dimension, and $q = \frac{1}{3}$. We can observe the results of Theorem 2.6 with a simple two dimensional example. We solve the problem:

$$\min_{\Omega=[0 \quad 1]^2} [4 \quad 5]\,\mathbf{x}.$$

Table 2.1 describes the results, while Figure 2.7 shows the strongly optimal squares that are created on every even iteration.

Table 2.1: Table of strongly optimal squares for a linear function. Data shows N step,$q-$linear convergence.

It.	fcn. evals.	e_k
2	7	0.236
4	19	0.079
6	37	0.026
8	65	0.009
10	91	0.003
12	121	9.70×10^{-4}
14	161	3.23×10^{-4}
16	203	1.07×10^{-4}
18	253	3.59×10^{-5}
20	313	1.19×10^{-5}

This result shows that DIRECT is N step, q-linear for linear functions, *with respect to the iterations of DIRECT* when the balance parameter $\epsilon = 0$. This result does not include information about function evaluations. This is because the number of function evaluations that DIRECT uses in a particular iterations varies. Later iterations are much more expensive, in terms of function evaluations, than early ones. This is seen in Table 2.1.

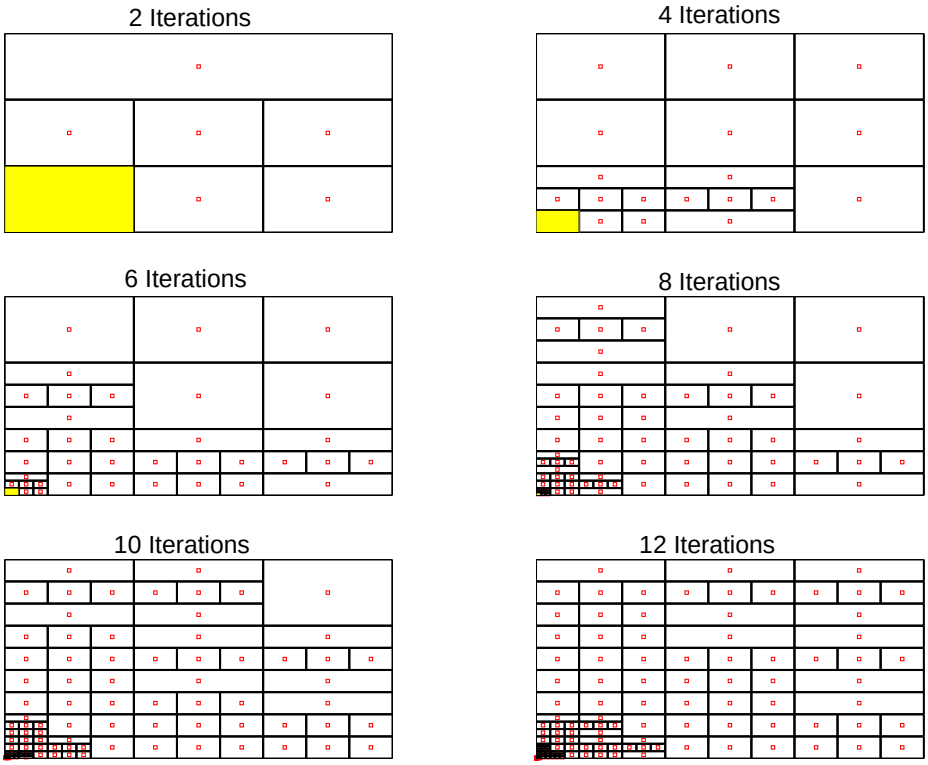

Figure 2.7: Rate of Convergence for linear function with global minimum at $x^* = (0 \;\; 0)^T$.

We have shown that DIRECT has a linear rate of convergence for linear functions. In the next section, we show, numerically, that a similar result is not as easily attained for another class of trivial functions. In general, a rate of convergence is difficult to calculate.

2.3.2 Convex Quadratic Functions

In this section, we illustrate the difficulties of determining a convergent rate for DIRECT, by exploring the class of convex quadratic functions.

Our first example is to minimize the function

$$f(x) = x^T \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} x = x^T x,$$

over the domain $\Omega = \{x \in \mathbb{R}^2 : -4.2 \leq x_1 \leq 1.3, -2.1 \leq x_2 \leq 5.2\}$. The domain was randomly chosen so that the geometric impact on convergence is not a factor. Figure 2.8 shows the gradient contours of f.

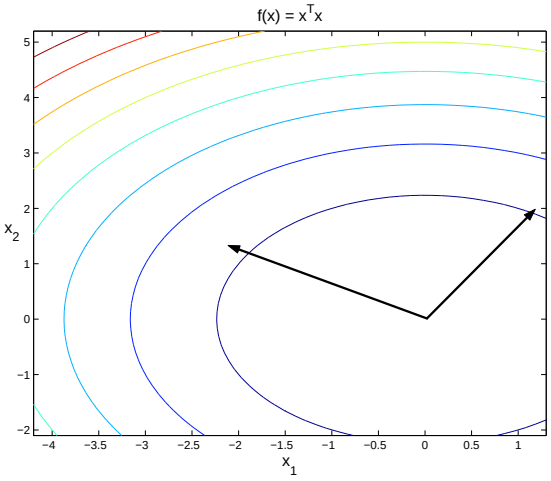

Figure 2.8: Contours of $f(x) = x^T x$.

Table 2.2 shows the convergence results for centers of strongly optimal hypercubes on the problem $f(x) = x^T x$ with the balance parameter, ϵ, is set to zero. We let DIRECT run for twenty iterations, and we only report results for the iterations in which strongly optimal hypercubes occur.

The convergence of centers of strongly optimal hypercubes on the objective function $f(x) = x^T x$ is similar to the convergence shown in the previous section on linear

Table 2.2: Results for $f(x) = x^T x$.

It.	fcn. evals.	e_k
2	7	0.963
4	19	0.234
6	41	0.076
8	67	0.030
10	111	0.012
12	161	2.57×10^{-3}
14	217	1.62×10^{-3}
16	291	2.56×10^{-4}
18	373	1.08×10^{-4}
20	455	4.97×10^{-5}

functions. First, it appears that strongly optimal hypercubes are found every N iterations (tests were also run on higher dimension versions of this problem). Also, the error is monotonically decreasing. These results could indicate that DIRECT also solves convex quadratics at a predictable rate using some type of modified version of Lemma 2.6.

Our next example shows that the rate of convergence for DIRECT is not predictable on convex quadratic functions. In this example, we try to minimize a function of the form $f(x) = x^T Ax$ over the same domain, Ω, as in the last problem. Unlike the previous example, we choose a matrix, A, that has a large condition number. Our two dimensional example has eigenvalues $\lambda_1 = 1$, $\lambda_2 = 10^7$, and randomly chosen eigenvectors. Figure 2.9 shows the contours for this problem. Note that the problem has the same location for the global minimum as $f(x) = x^T x$, and same optimal value.

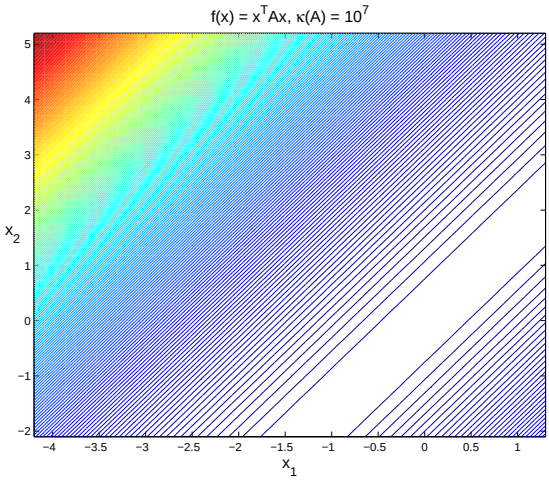

Figure 2.9: Contours of $f(x) = x^T Ax$, with large condition number for A.

We give DIRECT a budget of twenty iterations, and report the results in Table 2.3. The iterations which produce a strongly optimal hypercube are shown, along with the error, e_k at that iteration.

Table 2.3: Results for $f(x) = x^T Ax$, where $\kappa(A) = 10^7$.

It.	fcn. evals.	e_k
2	7	1.600
4	19	1.842
5	29	1.842
10	97	1.990
11	117	1.990
17	215	1.50

The numbers in Table 2.3 show that DIRECT has a difficult time with this problem. Only six strongly optimal hypercubes were found (as opposed to ten on the first problem), and the error is much higher. The error is also not monotonic. This fluctuation is due to the confusion that is created by the elliptic contours. In general,

DIRECT can behave poorly on any problem for which the objective function has small function values far away from the global minimum. Figure 2.10 shows a comparison of the points sampled by DIRECT on the two problems explored in this section. Figure 2.10 shows why the convergence was not as good for the second problem; DIRECT is confused by the long valley created by the large condition number of the second example.

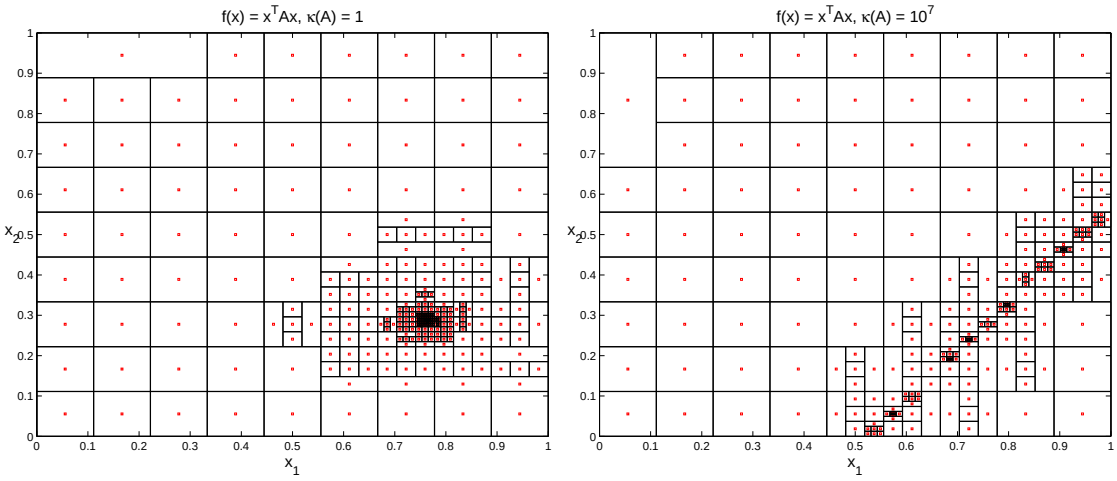

Figure 2.10: Points sampled by DIRECT on two different convex quadratic functions.

We find that we are unable to determine a rate of convergence for convex quadratic functions. In this short, simple example, we see that the condition number of a convex, quadratic problem will alter the rate of convergence for DIRECT. Similarly, we could alter the size of Ω so that the rate of convergence for DIRECT would not be altered by the condition number of A. In general, there are many factors which can determine how fast DIRECT converges.

Chapter 3

Algorithmic Analysis of DIRECT

In this chapter, we explore the sensitivity of DIRECT to the balance parameter, ϵ. Our analysis shows that DIRECT is sensitive to additive scaling, and has slow asymptotic convergence. Simple examples are used to illustrate the analysis.

We use a subset of potentially optimal hyperrectangles, named *strongly optimal hyperrectangles*, to analyze DIRECT. In the next section of this chapter, we introduce the idea of strongly optimal hyperrectangles, and assert some of their characteristics. In § 3.2, we illustrate DIRECT's slow convergence and additive scaling problems by observing strongly optimal hyperrectangles. An important inequality is presented that explains the behavior of DIRECT on these examples. We derive a modification to DIRECT in § 3.2.4, and present test results that illustrate our modifications effectiveness.

3.1 Strongly Optimal Hypercubes

We begin this section by reviewing some fundamental analysis of DIRECT. Many of the initial results that are shown can be found in [23], however it is convenient for the reader to be reminded of these results. Following the notation of [23], we describe the minimum value of f in the nth iteration of DIRECT

$$f_{\min}^n = \min_{S \in \mathcal{S}^n} f(c(S)).$$

If $\alpha(S)$ is the size of hyperrectangle S, then we let

$$v_n = \min_{S \in \mathcal{S}^n} \alpha(S).$$

Recall that the division strategy DIRECT employs only divides hyperrectangles along their long sides. Since the entire domain is the unit hypercube (and hence has equal length sides), any hyperrectangle created by DIRECT will never have side lengths that differ by more than $\frac{1}{3}$. This fact was explored in [23]. Thus, if a hyperrectangle in the current state of DIRECT has a long side of 3^{-l}, then the shortest side will either be 3^{-l}, or $3^{-(l+1)}$.

The next definition is taken from [23].

Definition 3.1. *Let S be an N-dimensional hyperrectangle created by DIRECT. Assume that S has p sides of length 3^{-l-1} and $N-p$ sides of length 3^{-l}. Then we say*

that S is of level l and stage p, where $0 \leq p \leq N-1$.

The definition above quantifies an important property of DIRECT. Whenever a hyperrectangle is chosen as potentially optimal, only the longest sides are divided. Therefore, a hyperrectangle of level l and stage p was created when a hypercube of level l was divided.

The next Lemma (which is repeated from [23]) shows how we can express the size of a hyperrectangle in terms of its level and stage.

Lemma 3.2. *Let S be a hyperrectangle of level l and stage p. Then the size of S (in the sense of [35]) is given by*

$$\alpha(S) = \frac{3^{-l}}{2}\sqrt{N - 8p/9}. \tag{3.1}$$

Proof. Recall that, in [35], the size of a hyperrectangle is it's l^2 radius. Let a and b be opposite corners of M. Then

$$\alpha(S) = \frac{1}{2}\|a-b\|_2 = \sqrt{\sum_{i=1}^{N}(a_i - b_i)^2}.$$

Since p sides of S have length of 3^{-l-l}, there exists $I \subset \{1, ..., N\}$, $|I| = p$ such that

$b_i - a_i = 3^{-l-1}$, for all $i \in I$ and $b_i - a_i = 3^{-l}$, for all $i \in \{1, ..., N\} / I$. Therefore,

$$\begin{aligned}
\alpha(S) = \frac{1}{2}\|a - b\|_2 &= \sqrt{p(3^{-l-1})^2 + (N-p)(3^{-l})^2} \\
&= \frac{3^{-l}}{2}\sqrt{\frac{p}{3^2} + (N-p)} \\
&= \frac{3^{-l}}{2}\sqrt{N - p + p/9} \\
&= \frac{3^{-l}}{2}\sqrt{N - 8p/9},
\end{aligned}$$

□

The next lemma follows directly from the previous result.

Lemma 3.3. *After n iterations of DIRECT, if $S \in \mathcal{S}^n$ is a hyperrectangle created by DIRECT such that $\alpha(S) = v_n$, then S is a hypercube.*

Proof. Assume not. Let S be a hyperrectangle created by DIRECT such that $\sigma(S) = v_n$, and let it's level and stage be l and p, respectively. Since S is not a hypercube, it follows that $0 < p \leq N - 1$. Note that

$$\alpha(S) = \frac{3^{-l}}{2}\sqrt{N - 8p/9}.$$

Also, S was created because a hypercube at level l was found to be potentially optimal, and was divided. When this hypercube was divided, every dimension was divided, and a hypercube, say $\bar{S}$, at level $l + 1$ was created. We now argue that $\alpha(\bar{S}) < \alpha(S)$.

From Lemma 3.2,

$$\begin{aligned}
\alpha(S) &= \frac{3^{-l}}{2}\sqrt{N - 8p/9} \\
&> \frac{3^{-l}}{2}\sqrt{N - \frac{8N}{9}} \\
&= \frac{3^{-l}}{2}\sqrt{N/9} \\
&= \frac{3^{-l-1}}{2}\sqrt{N} = \alpha(\bar{S})
\end{aligned}$$

Thus, since we have found a hypercube smaller than S, it follows that the smallest hyperrectangle in $\mathcal{S}^n$ must be a hypercube. □

With Lemma 3.3 complete, we are now ready to define a subsequence of the potentially optimal hyperrectangles. This subsequence is used to study the effects of the balance parameter.

Definition 3.4. *Let $\mathcal{S}^n$ be the state of DIRECT after n iterations. A potentially optimal hyperrectangle $R \in \mathcal{S}^n$ is* **strongly optimal** *if it is the smallest hyperrectangle in $\mathcal{S}^n$.*

Since one of the requirements for a hyperrectangle to be strongly optimal is for it to be one of the smallest hyperrectangles created, it follows from Lemma 3.3 that only hypercubes can be strongly optimal. Also, a necessary condition for a hypercube to be strongly optimal is that its center must be the location of the best objective value found so far. We state this formally as a corollary.

Corollary 3.1. *Let R be a strongly optimal hypercube found by DIRECT after n iterations. Then $f(c(R)) = f^n_{\min}$.*

Proof. Assume not. Then, R is a strongly optimal hypercube, and $f(c(R)) \neq f^n_{\min}$. Let $\mathcal{S}^n$ be the state of DIRECT at the time of discovery of R. Recall that, since R is potentially optimal, there exists $\tilde{K}$ such that

$$f(c(R)) - \tilde{K}\alpha(R) \leq f(c(S)) - \tilde{K}\alpha(S) \tag{3.2}$$

$$f(c(R)) - \tilde{K}\alpha(R) \leq f_{\min} - \epsilon|f_{\min}| \tag{3.3}$$

for all $S \in \mathcal{S}^n$. Let S' be in $\mathcal{S}^n$ such that

$$f_{\min} = f(c(S'))$$

. If $\alpha(S') = \alpha(R)$, then Equation 3.2 yields

$$f(c(R)) \leq f(c(S')),$$

which is an immediate contradiction to our assumption. Therefore, since R is strongly optimal, we consider the case where $\alpha(S') > \alpha(R)$.

From Equation 3.2 we see that

$$\tilde{K} \leq \frac{f(c(S')) - f(c(R))}{\alpha(S') - \alpha(R)} < 0.$$

The last inequality is due to our assumption that $f(c(R) \neq f_{\min} = f(c(S'))$. But, since Equation 3.3 can be rewritten as

$$\tilde{K} \geq \frac{f_{\min} - f(c(R)) - \epsilon|f_{\min}|}{-\alpha(R)} > 0,$$

we again arrive at a contradiction.

Thus, our assumption that $f(c(R)) \neq f_{\min}$ is false, and the lemma is verified. □

Figure 3.1 provides an intuitive portrait of Corollary 3.1 using the familiar representation of potentially optimal hyperrectangles from Chapter A.1. The only way a smallest hypercube can lie on the lower convex hull of our diagram is for its center to be the smallest value found so far, as Figure 3.1 indicates. Strongly optimal hy-

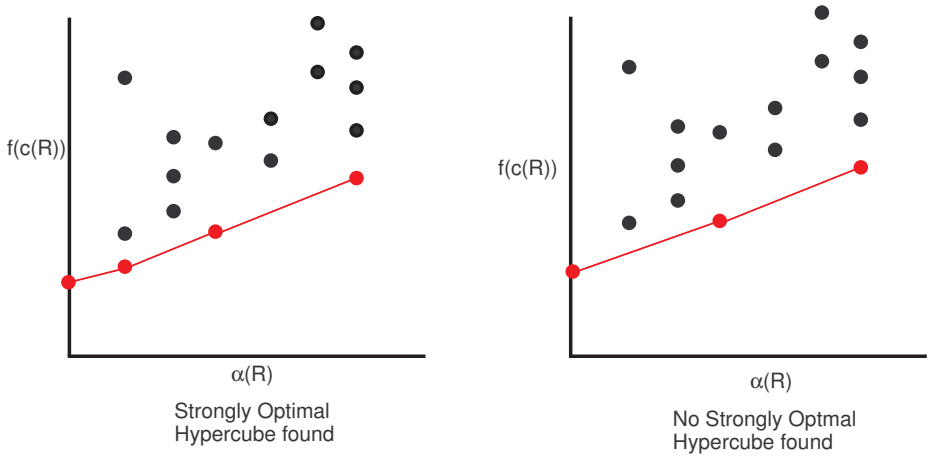

Figure 3.1: Verification of Lemma 3.1.

percubes are characterized by the length of their sides. We say that R is a strongly optimal hypercube at level l if the length of its side is 3^{-l}. Thus, when DIRECT divides

the entire domain in its initial step, DIRECT has actually found a strongly optimal hypercube at level 0.

The next corollary continues this discussion, and shows that DIRECT finds strongly optimal hypercubes sequentially, with respect to the level.

Corollary 3.2. *Strongly optimal hypercubes are found in sequential order; i.e. a strongly optimal hypercube of level l will be found before one of level $l+1$ is discovered.*

Proof. Let S be a strongly optimal hypercube at level $l+1$. As described in [35, 23], S can be traced to a parent hypercube at level l, say S'. This is due to the division strategy that DIRECT uses. By only dividing along the longest side, the children of a hyperrectangle divided by DIRECT are either at the same level as the parent, or else increased by one.

Thus, when S' was divided, it was either the smallest hypercube in existence, or else there was another hyperrectangle that was smaller. Let Θ be the set of hyperrectangles who are smaller than S' at its time of division. As already described, every hyperrectangle in Θ can be traced back to a parent hypercube at level l. Since the cardinality of Θ is finite, eventually we will find one hyperrectangle whose parent hypercube at level l was the smallest at its time of division. Figure 3.2 describes this parent/child relationship of hypercubes. □

The next lemma ensures that DIRECT will eventually find a strongly optimal hypercube at any level.

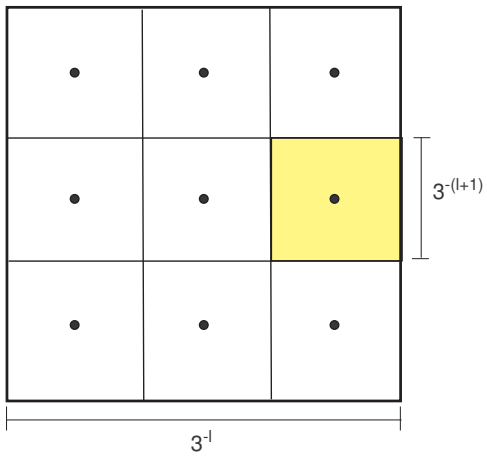

Figure 3.2: An example of the "parent" of a strongly optimal hypercube.

Lemma 3.5. *DIRECT will find a strongly optimal hypercube at level k, for any $k > 0$, in a finite number of iterations.*

Proof. We proceed by induction. Clearly, the initial case is true since DIRECT finds a strongly optimal hypercube at level $l = 0$ in its first iteration.

We assume the hypothesis for the case l; that is, DIRECT has found a strongly optimal hypercube at level l after n iterations. Lemma 3.2 ensures that if a strongly optimal hypercube is found, it will be of level $l + 1$. We now argue that a strongly optimal hypercube is found.

When DIRECT found a level l strongly optimal hypercube, it was divided and a hypercube at level $l + 1$ was created. As Corollary 1.1 indicates, this hypercube will eventually be divided. If it is the first level $l + 1$ hypercube divided, it is strongly optimal; otherwise whichever hypercube of level $l + 1$ was divided first becomes was the strongly optimal hypercube at level $l + 1$. □

3.2 Practical Implications of Strongly Optimal Hypercubes

In the previous sections of this chapter, we introduced the idea of strongly optimal hypercubes, and proved fundamental results regarding their behavior.

In this next section, we shift our attention from theoretical results to applied and practical ideas. We will explore the sensitivity of DIRECT to the balance parameter, ϵ, with simple examples and some analysis. We will discuss some improvements to the DIRECT algorithm that we have made, and provide some numerical results to verify our improvements.

3.2.1 The balance parameter ϵ

As we have seen, the discovery of a strongly optimal hypercube causes DIRECT to refine the mesh upon which it samples; in effect, strongly optimal hypercubes provide an indicator for the performance of DIRECT. In Figure 3.3, we observe the contrast potentially optimal hyperrectangles with strongly optimal hypercubes on a two dimensional sample problem. On the left is a plot of all the points sampled by DIRECT, when it is given a budget of 200 function evaluations. On the right, only the centers of strongly optimal hypercubes are shown. As this example shows, when you look only at the strongly optimal hypercubes, you focus on the movement towards a minimum of the objective function. This observation provides the motivation for the

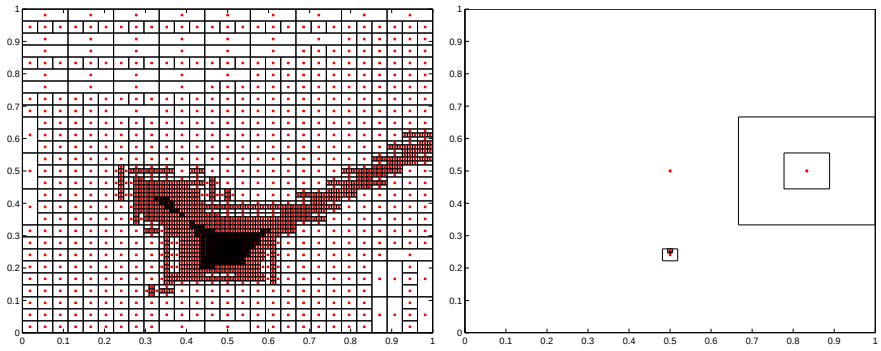

Figure 3.3: All points sampled by DIRECT vs. centers of strongly optimal hypercubes.

forthcoming discussions; is there a way to find strongly optimal hypercubes in a more efficient way?

DIRECT was created with a balance parameter, ϵ. The balance parameter aims to prevent the algorithm from "becoming too local in its orientation, and wasting precious function evaluations in pursuit of extremely small improvements" [35]. The balance parameter ϵ alters Equation 1.4 in the definition of potentially optimal hyperrectangles, Definition 1.4.

The parameter ϵ provides a balance between the global and local searches that DIRECT uses. An effect of the balance parameters is it regulates the occurrence of strongly optimal hypercubes. In [35], ϵ was shown to provide improved results for DIRECT on a series of test problems when set to the value $\epsilon = 10^{-4}$. An example from [35] is the Shubert test problem [35, 55]. This problem is described in § A. Figure 3.4 illustrates the effectiveness of the balance parameter. When $\epsilon = 0$, DIRECT gets trapped in a local minimum, and cannot find the global solution in a reasonable amount of function evaluations (in this example, a budget of 10,000 function evalu-

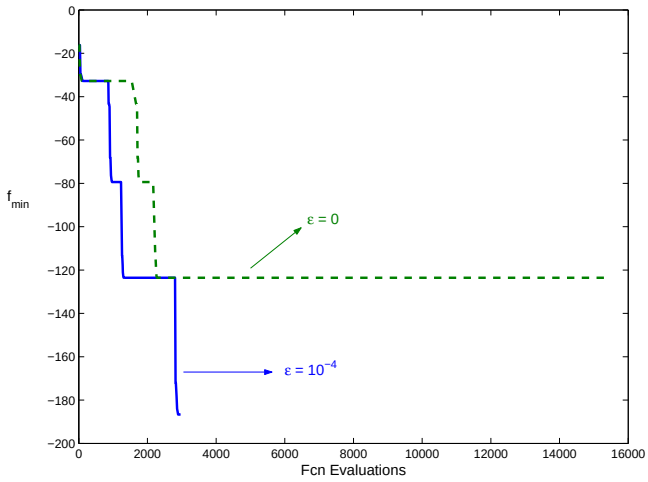

Figure 3.4: A comparison of different values of ϵ on the Shubert test problem.

ations was used). A larger value of ϵ improves the convergence speed of DIRECT on the Shubert problem. Essentially, a large balance parameter causes DIRECT to reject strongly optimal hypercubes in lieu of dividing larger, unexplored hyperrectangles. Rejecting strongly optimal hyperrectangles also maintains the number of function evaluations used in an iteration of DIRECT, as Figure 3.5 illustrates.

When $\epsilon = 0$, the smallest hypercube is constantly divided, thus creating many new sizes of hyperrectangles. The pool of potentially optimal hyperrectangle candidates grows, and the number of potentially optimal hyperrectangles increases. This increase leads to more function evaluations being used in an iteration of DIRECT. The Shubert test problem demonstrates the effectiveness of the balance parameter.

However, under certain circumstances, the balance parameter ϵ can compromise the efficiency of DIRECT. The next lemma quantifies a situation when this type of inefficient behavior will occur.

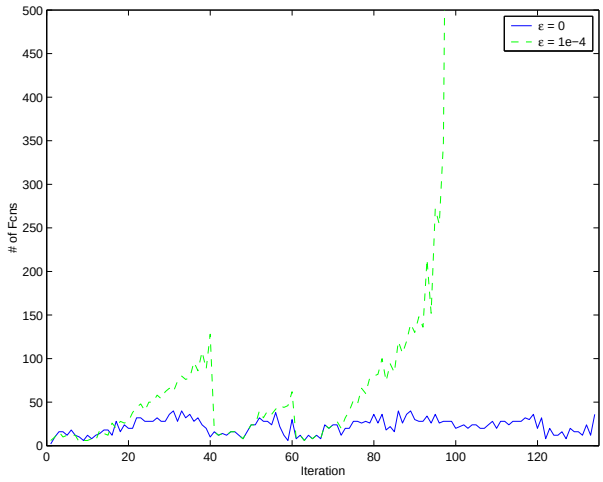

Figure 3.5: When $\epsilon = 0$, the number of function evaluations spent in an iteration of DIRECT explodes.

Lemma 3.6. *Let f be a Lipschitz continuous function with constant K, and $N > 1$, where N is the dimension of the problem. Let R be a hypercube with a center c at level l such that*

- $f(c) = f_{\min} \neq 0$ *(i.e. $f(c)$ is the best value found so far)*
- $d \leq d_i$, *for all i (i.e. R is the smallest hypercube)*

If

$$l \quad > \quad \log_3 \left(\frac{K \left(\sqrt{N^2 + 8N} + N \right)}{8\epsilon |f(c)|} \right) + 1 \tag{3.4}$$

then R cannot be potentially optimal until all the hyperrectangles in the current state of DIRECT whose centers are on the stencil $c \pm 3^{-l} e_i$ for $i = 1, ..., N$ are the same size as R.

Proof. Say Inequality 3.4 holds. Recall that, for hypercube R to be potentially optimal there must exist $\tilde{K}$ such that

$$f(c) - \tilde{K}d \leq f(c_i) - \tilde{K}d_i \tag{3.5}$$

$$f(c) - \tilde{K}d \leq f_{\min} - \epsilon|f_{\min}| \tag{3.6}$$

From 3.5, it is clear that

$$\tilde{K} \leq \frac{f(c_i) - f(c)}{d_i - d}$$

for all i. We define $\tilde{K}$ to be as large as possible; that is, we let

$$\tilde{K} = \min_i \frac{f(c_i) - f(c)}{d_i - d} = \frac{f(c_{\tilde{i}}) - f(c)}{d_{\tilde{i}} - d}, \tag{3.7}$$

and show that Inequality 3.6 cannot be satisfied.

Recall that $d = \frac{3^{-l}\sqrt{N}}{2}$ (from Lemma 3.2) . With the aid of elementary algebra, Inequality 3.4 can be re-written as,

$$3^{-(l-1)} < \frac{\epsilon|f(c)|\left(\sqrt{N+8} - \sqrt{N}\right)}{K\sqrt{N}}. \tag{3.8}$$

In turn, Equation 3.8 can also be re-written as

$$\frac{3 \cdot 2K}{\sqrt{N+8} - \sqrt{N}} \leq \frac{\epsilon|f(c)|}{\frac{3^{-l}}{2}\sqrt{N}}. \tag{3.9}$$

Now, we are ready to show Inequality 3.6 is false.

The results of Lemma 3.2 allow us to compute the difference in size of hyperrectangles R and $\tilde{i}$.

$$d_{\tilde{i}} - d = \frac{3^{-l}}{2}\left(\sqrt{N - 8\tilde{p}/9} - \sqrt{N}\right) \geq \frac{3^{-(l+1)}}{2}\left(\sqrt{N+8} - \sqrt{N}\right). \tag{3.10}$$

The last inequality is found by setting $\tilde{p} = N - 1$. Therefore,

$$\frac{1}{d_{\tilde{i}} - d} \leq \frac{1}{\frac{3^{-(l+1)}}{2}\left(\sqrt{N+8} - \sqrt{N}\right)}. \tag{3.11}$$

Also note that

$$f(c_{\tilde{i}}) - f(c) = f(c \pm 3^{-l}e_{\tilde{i}}) - f(c) \leq K3^{-l}. \tag{3.12}$$

due to the Lipschitz continuity of f. Therefore,

$$\tilde{K} = \frac{f(c_{\tilde{i}}) - f(c)}{d_{\tilde{i}} - d} < \frac{K3^{-l}}{\frac{3^{-(l+1)}}{2}\left(\sqrt{N+8} - \sqrt{N}\right)} = \frac{3 \cdot 2K}{\sqrt{N+8} - \sqrt{N}}. \tag{3.13}$$

Combining 3.13 and 3.9, we see that

$$\tilde{K} < \frac{\epsilon|f(c)|}{\frac{3^{-l}}{2}\sqrt{N}}, \tag{3.14}$$

and the it follows that (recall that $d = \frac{3^{-l}}{2}\sqrt{N}$)

$$f(c) - \tilde{K}d > f_{\min} - \epsilon|f_{\min}|. \tag{3.15}$$

Recall that in Equation 3.7 that we chose $\tilde{K}$ as large as possible. Therefore, hypercube R cannot be potentially optimal. □

With Lemma 3.6, we can identify a situation where DIRECT does not behave as intended. In the following sections, we describe two scenarios which lead to the situation described in Lemma 3.6, and conclude with an improvement to DIRECT and numerical results to support.

3.2.2 Termination of DIRECT

Lemma 3.6 describes when a strongly optimal hypercube is prevented from being found by one its immediate neighbors. This attribute may be desirable in the early iterations of DIRECT, when the algorithm has just begun searching the design space. But, in later iterations, when the mesh on which DIRECT is sampling is now fine, this type of behavior may be an indicator that DIRECT is no longer behaving as intended. In this scenario, the rate of change of the objective function does not overcome the magnitude of the fixed parameter ϵ (recall that in [35], it is recommended to hardwire the value of ϵ to 1^{-4}), and DIRECT spends function evaluations dividing neighboring hyperrectangles, before finally choosing to divide the hypercube with the smallest

function value. In these later iterations, the parameter ϵ no longer balances the local and global search; instead, it prevents DIRECT from improving the minimum value found.

To demonstrate this situation, we will solve the following problem with DIRECT:

$$\begin{aligned} \min \quad & f(x) = \sum_{i=1}^{2} |x_i| + 7 \\ \text{subject to} \quad & -1 \leq x_i \leq 1, \quad i = 1, 2. \end{aligned}$$

Note that the geometry of the constraints is such that DIRECT will find the global minimum ($x_1 = x_2 = 0$) in its first function evaluation. We also perturb the problem by 7 units so that $f_{\min} \neq 0$. We will use this problem to observe strongly optimal hypercubes.

We give DIRECT a budget of 1500 function evaluations to try and solve this problem. Figure 3.6 shows the domain of the problem, and the points sampled therein. As expected, the Figure shows that DIRECT is sampling around the global minimum of the problem. Figure 3.6 shows nothing surprising; the sample points cluster around

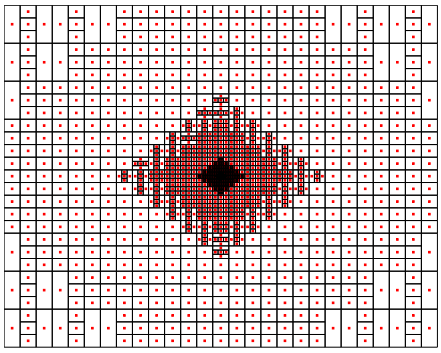

Figure 3.6: Results of DIRECT on the objective function $f(x) = \|x\|_1 + 7$.

the global minimum. DIRECT appears to be working properly.

Figure 3.7 tracks the iterations it takes DIRECT to find strongly optimal hypercubes in the above problem. As shown in the figure, it takes seven iterations to find the

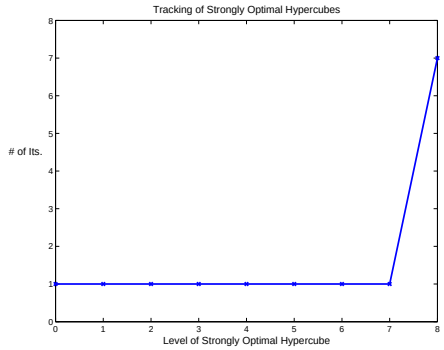

Figure 3.7: Tracking the strongly optimal hypercubes for $f(x) = \|x\|_1 + 7$.

first seven strongly optimal hypercubes; i.e. one is found in each of the first seven iterations. This behavior is expected, since the necessary conditions for a strongly optimal hypercube (i.e. the smallest function value found occurring in the center of the smallest hypercube) are always satisfied, due to the global minimum being immediately found by DIRECT.

However, this behavior changes after the seventh level of strongly optimal hypercubes is found. Instead of taking a single iteration, DIRECT needs 7 iterations before it finds the level 8 strongly optimal hypercube. This is due to the behavior described by Lemma 3.6. The Lipschitz constant for the objective function, f, is 1, therefore Equation 3.4 reduces to

$$l \geq \log_3 \left(\frac{K(\sqrt{N^2 + 8N} + N)}{8\epsilon |f(c)|} \right) + 1 = 7.42.$$

Thus, when DIRECT attempts to find the level 8 strongly optimal hypercube, it finds that it must first divide the neighboring hyperrectangles. In Table 3.1, the number of function evaluations needed to find successive levels of strongly optimal hypercubes are shown. As Table 3.1 suggests, finding a level 8 hypercube becomes very expensive

Table 3.1: Statistics for Finding Strongly Optimal Hypercubes

$f(x) = \|x\|_1$		
Level	Iterations	Fcn. Evals
0	0	1
1	1	9
2	2	33
3	3	65
4	4	97
5	5	121
6	6	193
7	7	273
8	14	1481

for DIRECT. Although this may seem like extreme behavior for this example, we note that a level 8 hypercube lies on a mesh of $3^{-8} \approx 10^{-4}$. The original authors of DIRECT (see [35]) terminated DIRECT on test problems when best objective value found by DIRECT was less than .01% away from the known solution. In more practical applications, when the true global solution is not known, an analgous stopping criteria might be to terminate when a level 8 hypercube is found. As we have just shown, DIRECT will deviate from its normal behavior and become much less efficient at finding this higher level strongly optimal hypercubes.

One possible solution to this problem is reduce the value of the parameter ϵ from

the recommended value of 10^{-4}. This would raise the value of the right-hand side of Inequality 3.4, and allow the efficient behavior of DIRECT to continue. Another possibility would be to simply set $\epsilon = 0$, and remove its effects entirely (this would cause the right-hand side of Inequality 3.4 to become infinite).

These are both viable solutions, and would improve the efficiency of DIRECT in this, and many other examples. However, in [35], test problems were used where the reduction of the parameter ϵ results in poorer performance by the algorithm. In addition, reducing the balance parameter to zero substantially increases the number of function evaluations spent on a DIRECT iteration. Instead of removing the parameter entirely, we instead seek to find a way to better utilize it in the sense of the original authors intent; to provide a proper balance between the local and global search. We describe our improvements in the next section.

We conclude from this example that Inequality 3.4 shows that there is a limit to the efficient behavior of DIRECT. Unless the parameter ϵ is set to zero, eventually DIRECT will become inefficient in its search for strongly optimal hypercubes, and will begin to act in a way, we believe, was not intended.

3.2.3 Scaling

Poor scaling can have a major impact on the results of an optimization algorithm. The Implicit Filtering algorithm [39], for example, relies on scaling in its iteration process. In this section, we will discuss various types of scaling, and their impact on

DIRECT.

As shown in the last section, DIRECT will begin to exhibit inefficient search behavior when the mesh size reduces beyond a threshold determined by the ratio of the Lipschitz constant and the parameter ϵ. Another situation that can disrupt the efficiency of DIRECT is when the objective function values are large. The right-hand side of Inequality 3.4 clearly shows this; when $f_{\min}$ is large, the inefficient behavior will begin at a much lower level.

This next example illustrates this type of behavior. For this example, we use the Branin test problem, a common test problem for global optimization, and found in [17]. We compare the results of DIRECT on this test problem, and an additively perturbed variation which creates an objective function on a large scale. The second problem is essentially the same as the first, except that $k = 10^6$ has been added to the objective function. The global minima (there are 3 such locations) have not been moved, and other properties of the function remain the same. More information on the Branin test problem is in the Appendix of this document and [17]. For both problems, DIRECT is given a budget of 200 function evaluations.

An intuitive feel for the results of this experiment can be obtained by observing the points that DIRECT samples in the design space. Figure 3.8 presents the results of DIRECT on the original Branin test problem, and Figure 3.9 shows the results for DIRECT on the perturbed problem. Visually, a difference is already seen. Despite the objective functions being nearly identical, the clusters of points sampled by DIRECT

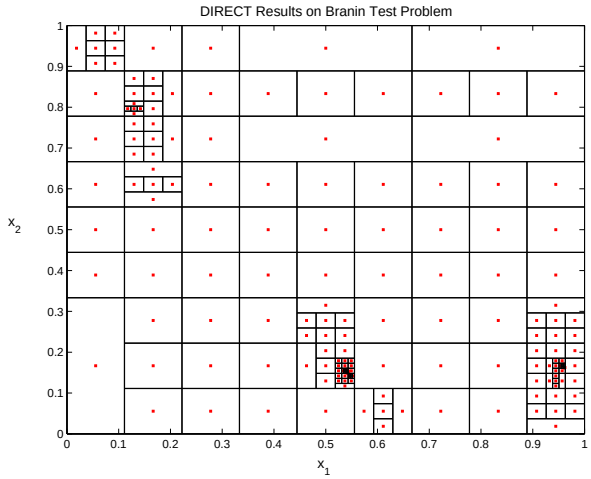

Figure 3.8: DIRECT sampling locations for Branin Test Problem.

on the perturbed problem are not nearly as well defined as they are on the original problem. It appears that the additive perturbation has overly biased DIRECT into a more global search.

Table 3.2 verifies these suspicions numerically. When DIRECT tries to solve the

Table 3.2: DIRECT results on Branin Test Problem, and perturbed variation

	True Solns.		DIRECT Results		Error	
	$x_{\min}$	$f_{\min}$	x^*	$f(x^*)$	$\|x_{\min} - x^*\|$	$\|f_{\min} - f(x^*)\|$
Branin	(3.14,2.28)	0.398	(3.14, 2.27)	0.3979	1.5×10^{-3}	3.5×10^{-6}
Pert. Branin	(3.14,2.28)	0.398	(3.06, 2.5)	$0.4580 + 10^6$	2.4×10^{-1}	6.0×10^{-2}

perturbed problem, it loses two orders of magnitude of accuracy with respect to the optimal solution, and four orders of magnitude of accuracy with respect to the optimal value.

The lack of well defined clusters in Figure 3.9 is quantified by the number of

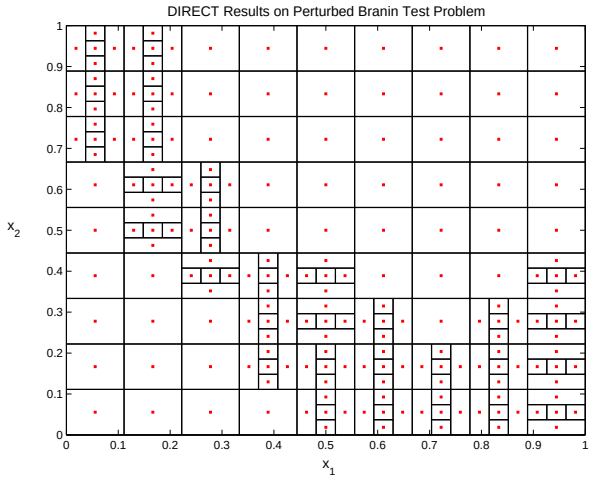

Figure 3.9: DIRECT sampling locations for the Additively perturbed Branin Test Problem.

strongly optimal hypercubes found. For the original problem, DIRECT was able to find a level 7 strongly optimal hypercube. With the same budget on the perturbed problem, the highest level of strongly optimal hypercube was 2. Table 3.3 summarizes these results. Thus, when the magnitude of the objective value is large, DIRECT's

Table 3.3: Strongly Optimal Hypercube Statistics for Branin Test fcns.

	Branin		Pert. Branin	
Level	It.	Fcn. Evals	It.	Fcn. Evals
0	0	1	0	1
1	2	9	3	9
2	3	19	30	81
3	5	37		
4	7	59		
5	9	93		
6	11	137		
7	14	191		

ability to balance the global and local search becomes compromised. Although the

test problems in [35] did not have cost functions that are this large in magnitude, it is not uncommon for practical applications to have costs on the order of 10^6, or larger. The problems in [21], for example, are on this order. Also, [10], has costs that are larger. This type of behavior is an undesirable consequence of using the balancing parameter,ϵ. As described in the last section, one solution to this problem is to set $\epsilon = 0$. Another strategy, which we develop in the next section, is to adaptively update the balance parameter, ϵ, during the optimization process. Although this will create additional parameters for DIRECT, it will also allow the balance parameter to be used in the way it was intended, and also will protect DIRECT from additive scaling problems.

3.2.4 DIRECT-ϵ

In this section, we present a modified version of DIRECT that was developed to overcome some of the problems described previously. We call our modified version DIRECT-ϵ, in the spirit of [23]. Our strategy will be to adaptively change the value of the balance parameter ϵ so that the algorithm will use the balance parameter in the way the creators of DIRECT intended.

Our strategy will be to set the parameter $\epsilon = 0$ when iterations of DIRECT are producing improved function values. However, when the values start to stagnate, indicating that a local minimum has been found, we will increase the value of ϵ so that DIRECT becomes biased globally in its search. This strategy is designed to encourage

Table 3.4: Algorithm DIRECT-ϵ

	Algorithm DIRECT-ϵ
1:	Normalize the domain to be the unit hyper-cube with center c_1
2:	**Set** $\epsilon = 0$
3:	Find $f(c_1)$, $f_{min} = f(c_1)$, $i = 0$, $m = 1$
4:	Evaluate $f(c_1 \pm \frac{1}{3}e_i)$, $1 \leq i \leq n$, and divide hyper-cube
5:	**WHILE** $i \leq maxits$ and $m \leq maxevals$ **DO**
6:	**Determine value for** ϵ
6:	Identify the set S of all pot. optimal rectangles/cubes
7:	**FOR** all $j \in S$
8:	Identify the longest side(s) of rectangle j
9:	Evaluate f at centers of new rectangles, and divide j into smaller rectangles
10:	Update f_{min}, $xatmin$, and m
11:	**END FOR**
12:	$i = i + 1$
13:	**END WHILE**

DIRECT to search globally after it identifies a local minimum. If the algorithm stagnates again, when $\epsilon \neq 0$, then we reset ϵ to zero, and allow DIRECT to search locally for the corresponding local minimum there. We formally describe the new algorithm in Table 3.4 and Table 3.5, where our new modification is in bold-face. The hard-coded values can be modified; the default values are the ones shown.

Although this modification introduces new parameters to the problem, we feel that the test results in the next section will show that the default values are pretty robust, and should work for most problems. Our implementation (see [18]) gives the user the ability to modify any of these parameters.

Table 3.5: Algorithm for determining ϵ for DIRECT-ϵ

	Algorithm Determine ϵ
1:	**IF** $\epsilon = 0$ **THEN**
2:	**IF** $\lvert f_{\min new} - f_{\min old}\rvert < 10^{-4}$ **THEN**
3:	stagnation = stagnation + 1
4:	**IF** stagnation $\geq$ maxstagnation **THEN**
5:	$\epsilon = 10^{-2}$
5:	startupvalue = $f_{\min new}$
6:	stagnation = 0
7:	**END IF**
8:	**ELSE**
8:	stagnation = 0
8:	**END IF**
9:	**ELSE**
10:	**IF** $\lvert f_{\min new} - startupvalue\rvert \leq 10^{-2}$ **THEN**
11:	stagnation = stagnation + 1
12:	**ELSE**
13:	stagnation = 0
14:	**END IF**
15:	**IF** stagnation $\geq 2\times$ maxstagnation **THEN**
16:	**IF** $\lvert f_{\min new} - f_{\min old}\rvert / f_{\min old} \geq 0.03$ **THEN**
17:	$\epsilon = 0$
18:	**END IF**
19:	stagnation = 0
20:	**END IF**
21:	**END IF**
22:	$f_{\min old} = f_{minnew}$

3.2.5 Test Results

In this section, we analyze the DIRECT-ϵ algorithm by comparing it to DIRECT on various test problems.

The first set of optimization test problems chosen are traditional problems taken from the global optimization community. Seven of the test problems are taken from [17], while the other two are reproduced from [56]. In [35], the results of DIRECT

algorithm's performance on the test problems was presented.

Table 3.6 introduces the problems and provides some information about each problems basic characteristics. The information is reprinted from [35].

Table 3.6: Characteristics of the Test problems

Test Function	Abbreviation	Number of dimensions	Number of local minima	Number of global minima
Shekel 5	S5	4	5	1
Shekel 7	S7	4	7	1
Shekel 10	S10	4	10	1
Hartman 3	H3	3	4	1
Hartman 6	H6	6	4	1
Branin RCOS	BR	2	3	3
Goldstein and Price	GP	2	4	1
Six-Hump Camel	C6	2	6	2
Two-Dimensional Shubert	SHU	2	760	18

Table 3.7 shows a comparison of the results of DIRECT and DIRECT-ϵ on the nine test problems. In each problem, the algorithm is terminated when it finds a point whose function value is within 0.01% of the known optimal value. DIRECT uses the default value of $\epsilon = 10^{-4}$, while DIRECT $-\ \epsilon$ uses $\epsilon = 10^{-2}$. Because ϵ is only non-zero when we wish to perform a global search, we are able to increase it's value and bias the search even more in a global sense.

Table 3.7: Comparison of DIRECT and DIRECT-ϵ on the nine test problems.

	S5	S7	S10	H3	H6	BR	GP	C6	SHU
DIRECT	155	145	145	199	571	195	191	285	2967
DIRECT-ϵ	179	145	145	199	571	195	191	285	1711

Table 3.7 shows that, for the most part, DIRECT-ϵ performs similarly to DIRECT. On the Shubert problem, however, our modified version of the algorithm significantly outperforms the original implementation. This is due to our ability to increase the parameter ϵ to a higher value than the original implementation uses. Thus, our global search is truly global.

Our next test set is the same problems, but additively perturbed by 10^6. In the last section, we showed that DIRECT has trouble with functions with large values. These results will show that our modified version does not encounter these problems. We modified the termination criterium for this set of problems; for these tests, DIRECT and DIRECT-s were given identical function evaluation budgets, and we report the results after each algorithm has exhausted its budget. This type of termination accurately resembles the way DIRECT is used by optimization practitioners. The budgets given to DIRECT and DIRECT-s are based on the results found in Table 3.7. The results are presented in Table 3.8.

The results shown in Table 3.8 describe the difficulties that DIRECT encounters when faced with problems with relatively large function values. In particular, strongly optimal hypercubes occur much less frequently for DIRECT in these examples. This indicates that DIRECT is spending its budget of function evaluations globally searching on the current mesh without locally attacking solutions which may show promise. The "H6" problem, in particular, exposes this potential pitfall.

The H6 perturbed test problem has six input variables, and each dimension has a

Table 3.8: Comparison of results on additively perturbed test problems.

		DIRECT			DIRECT-ϵ		
Prob.	Budget	$\|x^* - x_{\min}\|$	$\|f(x^*) - f_{\min}\|$	S.O.	$\|x^* - x_{\min}\|$	$\|f(x^*) - f_{\min}\|$	S.O.
S5	154	8.67	8.52	1	1.76E-2	3.01E-2	5
S7	144	8.67	8.75	1	2.70E-3	9.73E-4	5
S10	144	8.67	8.84	1	2.70E-3	1.00E-3	5
H3	198	8.90E-2	1.34E-1	1	1.75E-2	3.10E-4	5
H6	570	1.00	1.28	0	3.80E-2	2.94E-4	4
BR	194	2.41E-1	6.01E-2	2	3.40E-3	4.81E-5	6
GP	190	1.23E-2	6.50E-2	3	4.57E-4	9.04E-5	6
C6	284	4.39E-2	9.60E-3	2	3.71E-5	1.13E-8	6
SHU	2966	6.23	12.70	3	2.02	12.71	11

lower bound of zero, and an upper bound of one. In the H6 perturbed test problem, DIRECT is not able to move past the first level in its search. Nearly 600 function evaluations are used without the algorithm refining the mesh. As a result, the best point found in this example by DIRECT has an error of 1, which is large relative to the size of the domain of this problem.

The results of DIRECT-ϵ on the perturbed problems remain consistent with the results found on the original problems. This appears to be an advantage of our modified version of the algorithm. The performance of the algorithm does not seem to deteriorate, and the changes improve DIRECT's ability on some problems.

Chapter 4

Groundwater Supply Optimization

In this section, we present clustering results found with DIRECT on a suite of groundwater test problems. We used a Fortran implementation of DIRECT [22] to obtain the results. For this project, we utilized clustering analysis and statistical techniques to post-process the data collected by DIRECT. Motivation and a detailed explanation are presented, along with preliminary results.

In 2002, a large set of groundwater test problems was proposed in [44]. The test problems aim to provide environmental engineers and optimization experts with a better understanding of the current issues facing current optimization algorithms used on groundwater remediation problems. Five of the test problems have been developed into independent packages, and are posted on the internet [20]. The web repository of problems is expected to expand in the future.

4.1 Groundwater Optimization

The objective in the groundwater test-set is to minimize the installation and operational cost of wells in an underground water-bearing region, commonly referred to as an aquifer. A well costs approximately $20,000 to install, and can extract and inject water into the aquifer. On four of the problems, the goal is to extract a fixed amount of drinking water over five years; on the other problem, the goal is to contain a contaminant plume in the aquifer.

The location of each well, and the rate at which its injects/extracts water are the decision variables for this set of problems. Some regions in the aquifer contain more water than others, thus the operational cost of a well will depend on its location in the region.

The optimization problem posed is to find the number of wells, the (x, y, z) locations and the corresponding pumping rates of wells in an aquifer so as to minimize the cost of installation and operation.

The operational cost of a well is a function of its pumping rate, and the *hydraulic head* at the well location. Hydraulic head can be thought of as the amount of water available at a location in the water bearing region. Computing the hydraulic head requires a simulation of the subsurface. The simulation models the interaction of a well in an aquifer with the natural groundwater levels and flow patterns. The equation

$$C\frac{\partial h}{\partial t} = \nabla \cdot (H_c \nabla h) + P \tag{4.1}$$

describes the flow of water through the domain. The left side of (4.1) represents the rate of change in head level at a point in our domain. The constant C is the compressibility of water, a measure of how much the volume of water changes in response to a change in pressure. The left-hand-side depends on how the head levels disperse spatially within the domain, and how much water is extracted at that point by a pumping well. H_c is the hydraulic conductivity, which is the ease with which water can move through the soil. Finally, P is the pumping rate of the well.

The hydraulic heads are calculated by a computer simulation using the MODFLOW software from the U.S. Geological Survey [28]. MODFLOW outputs the steady-state hydraulic head levels at each point in the domain. The objective function is $f^T = f^c + f^o$, where f^c is the capital (installation) cost, and f^o is the operational cost. The simulation time is $t_f = 5$ years. The objective function, as proposed in [44] is given by

$$f^T = \underbrace{\sum_{i=1}^{n} c_0 d_i^{b0} + \sum_{i,P_i<0.0} c_1 |P_i|^{b_1} (z_{gs} - h^{min})^{b_2} +}_{f^c} \tag{4.2}$$

$$\underbrace{\int_0^{t_f} \left(\sum_{i,P_i<0.0} c_2 P_i (h_i - z_{gs}) + \sum_{i,P_i>0.0} c_3 P_i \right) dt}_{f^o} .$$

Here, c_j and b_j are cost coefficients and exponents, $d_i = z_{gs}$ is the depth of well i, P_i is the design pumping rate h^{min} is the minimum allowable head, and h_i is the hydraulic head in well i. In f^c the first term accounts for drilling and installing all the wells and the second term is an additional cost for pumps for the extraction wells. In f^o,

the term pertaining to the extraction wells includes a lift cost to raise the water to the surface.

The hydraulic head is constrained so that $10 \leq h_i \leq 30$. This ensures the well does not dry up or flood the land. The pumping rates are also constrained, based on the problem. For example, on the extraction problems, a net pumping rate constraint is imposed on the wells to ensure enough water is extracted from the aquifer. Also, the well locations are constrained so that two wells do not occupy the same location. A detailed description of the groundwater test suite is found in [37, 44].

4.2 Using DIRECT to Find Clusters

The results obtained by DIRECT are compared to five other optimization algorithms in [21]. In general, DIRECT did not perform well when compared to the other algorithms.

In addition to solving the groundwater test suite, we are also using DIRECT to support a comparison project that involves the test problems. Five other algorithms (Nomad [1], APPS [32], Design Explorer [8], IFFCO [11], and a genetic algorithm [15]) have been used to solve the groundwater problems, with their performances compared. The other algorithms require an initial iterate, and we are using DIRECT to find a small subset of feasible points that can be used by the other algorithms.

The strategy is to re-formulate each of the groundwater problems so that the

objective is feasibility. For example, if we the original problem is of the form

$$\begin{array}{lll} \min & f(x) & \\ \text{subject to} & g_i(x) \leq 0 & \forall i \in [1, m] \\ \text{and} & x \in \mathcal{D} \subset \Omega, & \end{array} \tag{4.3}$$

where Ω is defined by bound-constraints on x, then our re-formulation is of the form

$$\begin{array}{ll} \min & \sum_{i=1} \mu_i \max\left(g_i(x), 0\right) + \begin{cases} 0 & x \in \mathcal{D} \\ M & x \notin \mathcal{D} \end{cases} \\ \text{subject to} & x \in \Omega. \end{array} \tag{4.4}$$

The global minimum of Problem (4.4) is zero, and the solution is located at any feasible solution to (4.3). Our strategy is to solve (4.4) with `DIRECT`, and observe where the global minima cluster.

This strategy is appealing because of the natural clustering capabilities of `DIRECT`. Figure 4.1 illustrates the clustering of points sampled by `DIRECT` on a two-dimensional example.

The points sampled by `DIRECT` can be analyzed by a clustering algorithm. The clustering analysis typically returns a dendrogram which describes the clustering behavior of the data. Figure 4.2 is the dendrogram created using the Agglomerative Nesting (AGNES) [36] clustering algorithm on the data in Figure 4.1.

AGNES systematically fuses the data together by merging dissimilar objects to-

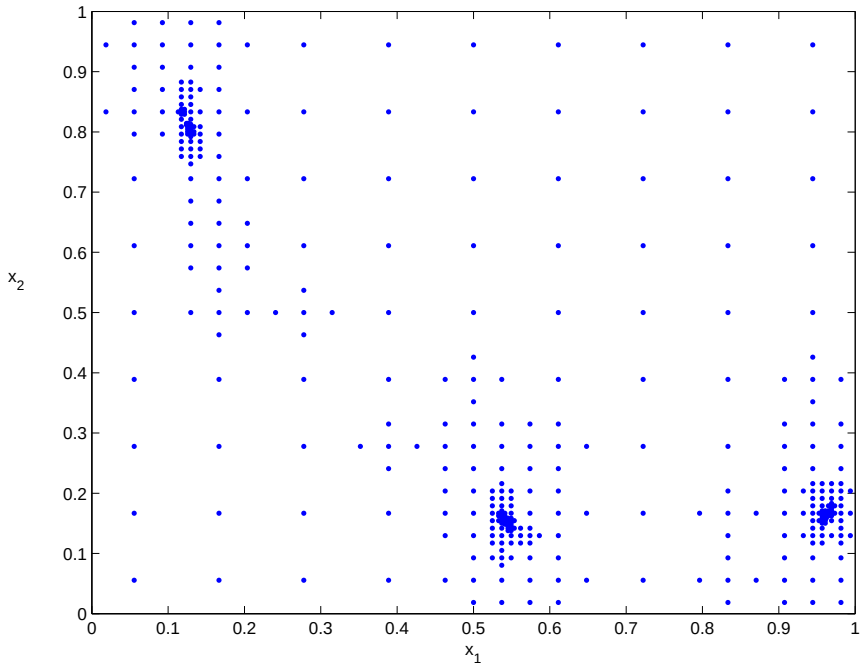

Figure 4.1: `DIRECT` sampling locations on a two-dimensional test problem.

gether into clusters. For example, let's assume we want to cluster the data set

$$\{x_1, x_2, ..., x_P\}.$$

AGNES begins by creating P clusters, i.e. each data point is its own cluster. In the first iteration, AGNES merges the two data points that are closest together, i.e. that have minimal dissimilarity. Thus, x_i and x_j that satisfy

$$\min_{i,j \in [1,...,P]} d(x_i, x_j) = \|x_i - x_j\|$$

are merged into a cluster. There are now $P-1$ clusters, (x_i, x_j) and then the remaining

$P-2$ individual data points.

In the second iteration (and those thereafter), AGNES merges the next two closest clusters together. The method for measuring dissimilarity between two clusters varies [36]; we use "Wards" method for measuring dissimilarity. If R and S are two clusters, then the dissimilarity between them is a weighted measure of the Euclidean distance between their centroids:

$$d(R,S) = \left(\frac{2|R||S|}{|R|+|S|} \|\bar{x}(R) - \bar{x}(S)\| \right)^{1/2}. \tag{4.5}$$

In (4.5), $|\cdot|$ is the cardinality of the cluster, and the i-th coordinate of the centroid of a cluster C is

$$\bar{x}(C)^i = (1/|C|) \sum_{j=1}^{|C|} x_j^i.$$

The coefficient in (4.5) is due to a relationship between (4.5) and the total sum-of-squares error. More precisely, if the sum-of-squares error for a cluster C is defined by

$$ESS(C) = \|x_i - \bar{x}(C)\|^2,$$

and ESS is the sum-of-squares error for all clusters, then it can be shown when clusters R and S are joined

$$\Delta ESS = \frac{1}{2} d^2(R,S).$$

A derivation can be found in [36].

In the $P-1$ iteration, AGNES merges the remaining two clusters together to form one compete dataset. This process is graphed with a dendrogram.

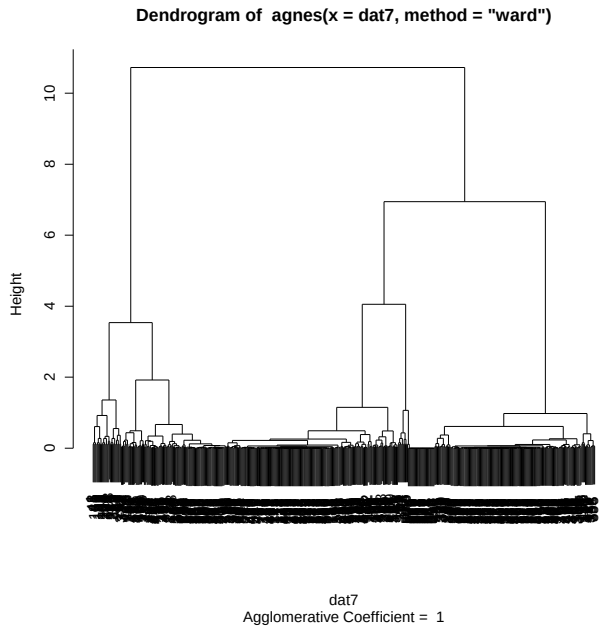

Figure 4.2: `DIRECT` sampling locations on a two-dimensional test problem.

A dendrogram, sometimes called a cluster-tree, is a branching diagram that represents hierarchial relationships among members of a dataset. Each piece of data lives at the bottom of the tree, and represents its own cluster. The vertical lines extending out of each data point are connected with a horizontal line when they are merged together by AGNES. The merges are graphed at the measure of their distance away

from each other on the horizontal axis.

A dataset with well-defined, distinct clusters would have long branches, and horizontal connections at the top of the diagram. A dendrogram with short branches indicates data that is not clustered. Because the data in Figure 4.1 appears to have good clusters, the dendrogram in Figure 4.2 has long branches. We "cut" the dendrogram at a point where three clusters are formed, and graph each cluster set individually. Figure 4.3 is an illustration.

A shortcoming of the AGNES method (and many other clustering methods [53]) is that no recommendation about how many clusters exist in the data is made. In the simple two dimensional example just outlined, it is easy to see that three clusters exist; in general it is so trivial.

We implemented the gap statistic [53] to find the number of clusters found by AGNES with the data collected by `DIRECT`. The gap statistic is a method to determine the "optimal" number of clusters in a dataset. The procedure compares the fitness of a clustered dataset to the fitness of a reference set of clustered data . The number of clusters that leads to the largest discrepancy between the real data and the reference data fitness values is deemed optimal.

More precisely, let's assume that dataset X has been divided into k clusters (or subsets) $r_1, ..., r_k$, i.e.

$$r_1 \cup ... \cup r_k = X \quad \text{and} \quad r_1 \cap ... \cap r_k = \emptyset.$$

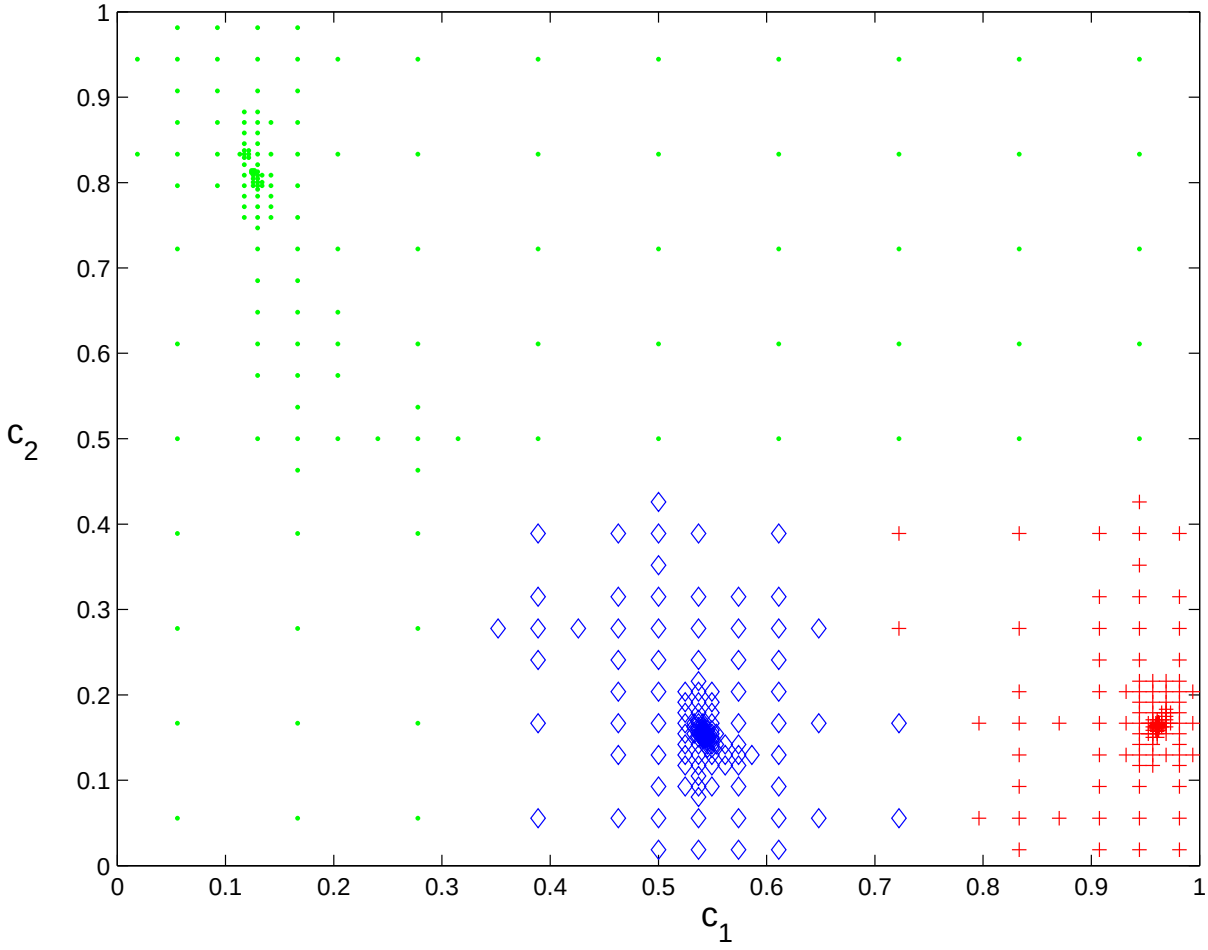

Figure 4.3: Data that was found by `DIRECT` was divided into three distinct clusters by the AGNES method of clustering.

We define

$$\mathcal{D}_r = \sum_{x,y \in r} \|x - y\|,$$

and let

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} \mathcal{D}_r.$$

The value W_k is called the "pooled within-cluster sum of squares around the cluster

means" [53]. W_k is monotonic in k, and represents how well a given clustering of the data "fits" [53]. The method of gap statistic is to compare $\log(W_k)$ to its expectation under an appropriate null reference distribution of the data [53]. The formula is

$$\text{Gap}_n(k) = E_n^* \{\log(W_k)\} - \log(W_k),$$

where E_n^* is the expectation under a sample size n from the reference distribution [53]. The optimal number of clusters is the value, $\hat{k}$, that maximizes $\text{Gap}_n(k)$ after the sampling distribution is considered. For our results, we used the uniform distribution as the reference distribution; this is consistent with [53].

4.3 Groundwater Results

The procedure outlined in Section 4.2 has been applied to the groundwater test problems. `DIRECT` was given a budget of 50,000 function evaluations, and the first 8,000 feasible points were used in the process. The value 8,000 was chosen because of our computational abilities. Clustering 8,000 data points took around five hours. Table 4.3 is a summary of our results.

On four of the groundwater problems, our method found two clusters; on the other problem three clusters were detected. At the time of writing this document, the method for choosing representatives of the clusters has not been determined.

Table 4.1: Number of clusters in the groundwater problems

Problem		
No.	Name	Cluster Count
1	CON5	2
2	UNC5	3
3	CON6	2
4	UNC6	2
5	CAPT	2

Chapter 5

Conclusion

This aim of this work was to develop new understanding of the DIRECT algorithm. DIRECT is a derivative-free optimization algorithm that aggressively searches for global minima of a real-valued function over a bound-constrained domain.

Our work focused on stating and proving convergence results regarding DIRECT. We showed, in its limit, that a subset of the points sampled by DIRECT cluster near Clarke stationary points. Our result only requires Lipschitz continuity of the objective function. If stronger assumptions are made on the objective function, i.e. differentiability, then we are able to prove corresponding properties of the cluster points.

We have extended this result to problems with additional constraints. For problems with general constraints, DIRECT requires a heuristic to assign surrogate function values to infeasible points. Our research has uncovered that DIRECT responds well

when these surrogate values are calculated based on information about their infeasibility, e.g. penalty functions.

The results we developed are dependent on the cone of feasible directions at the cluster points. We used three different directional cones in our analysis, and asserted conditions regarding the Clarke derivative with respect to each directional cone at a cluster point. We assume very little about the heuristic used for infeasible points; only that it does not interfere with the asymptotic behavior of `DIRECT` in the feasible regions of the domain.

In addition, this document focused on the algorithmic behavior of `DIRECT`. We uncovered a significant scaling problem with the algorithm, and illustrated its effect with simple examples. Our analysis provides an explanation for this scaling problem, and also explains the slow asymptotic convergence observed for `DIRECT`. We present a modification to `DIRECT` that alleviates scaling problems with the original version, and improves performance on some global optimization problems. Test results presented verify these claims.

Finally, we have used `DIRECT` to support a comparison project involving six different optimization algorithms on a suite of groundwater test problems. We have utilized the global search ability of `DIRECT`, and it's clustering tendencies to search the design space of the test problems for feasible regions. Representatives of the feasible regions will be fed back to the other algorithms to compare the robustness of each optimization package.

References

[1] M.A. Abramson. *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. PhD thesis, Computational and Applied Mathematics Department, Rice University, Houston, TX, 2002.

[2] C. Audet and J.E. Dennis Jr. Analysis of generalized pattern searches. *SIAM J. Optim.*, 13:889–903, 2003.

[3] Charles Audet and J. E. Dennis. Mesh adaptive direct search algorithms for constrained optimization. Technical Report TR04-02, Department of Computational and Applied Mathematics, Rice Univeristy, 2004.

[4] C.A. Baker, L.T. Watson, B. Grossman, R.T. Hafka, and W.H. Mason. Parallel global aircraft configuration design space exploration. In *8th IAAA/USAF/NASA/IMMO Symp. on Multidiscplinary Analysis and Optimization*, Long Beach, CA, 2000. AIAA Paper 2000-4763-CP. CDROM.

[5] C.A. Baker, L.T. Watson, B. Grossman, R.T. Hafka, and W.H. Mason. Parallel

global aircraft configuration design space exploration. Technical Report TR-00-07, Virginia Polytechnic Institute and State University, Blacksburg, VA, 2000.

[6] A. Batterman, J.M. Gablonsky, A. Patrick, C.T. Kelley, and K. Kavanaugh. Solution of a groundwater flow problem with implicit filtering. *Optimization and Engineering*, 3:189–199, 2002.

[7] A. Battermann, J. M. Gablonsky, A. Patrick, C. T. Kelley, K. R. Kavanagh, T. Coffey, and C. T. Miller. Solution of a groundwater flow problem with implicit filtering. Technical Report CRSC-TR00-30, North Carolina State University, Dec 2000.

[8] A.J. Booker, J.E. Dennis Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17(1):1–13, 1999.

[9] J. Burkardt, M. Gunzburger, and J. Peterson. Insensitive functionals, inconsistent gradients, spurious minima, and regularized functionals in flow optimization problems. *Int. J. Comput. Fluid Dyn.*, 16:171–195, 2002.

[10] R.G. Carter, J.M. Gablonsky, A. Patrick, C.T. Kelley, and O.J. Eslinger. Algorithms for noisy problems in gas transmission pipeline optimization. *Optimization and Engineering*, 2:139–157, 2002.

[11] T. D. Choi, O. J. Eslinger, P. Gilmore, A. Patrick, C. T. Kelley, and J. M. Gablonsky. IFFCO: Implicit Filtering for Constrained Optimization, Version 2.

Technical Report CRSC-TR99-23, Center for Research and Scientific Computation, July 1999.

[12] F.H. Clarke. *Optimization and Nonsmooth Analysis*. Canadien Mathematical Society Series of Monographs and Advanced Texts. Wiley-Interscience, New York, 1983.

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

[14] S.E. Cox, R.T. Haftka, C.A. Baker, B. Grossman, W.H. Mason, and L.T. Watson. Global multidisciplinary optimization of a high speed civil transport. In *Aerospace Numerical Simulation Symposium*, pages 23–28, Tokyo, Japan, June 16-18 1999.

[15] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer methods in Applied Mechanics and Engineering*, 186:311–338, 2000.

[16] G. DiPillo and L. Grippo. Exact penalty functions in constrained optimization. *SIAM J. Control and Opt.*, 27(6):1333–1360, November 1989.

[17] L.C.W. Dixon and G.P. Szego. *Towards Global Optimisation 2*. North-Holland, New York, NY, first edition, 1978.

[18] D.E. Finkel. DIRECT optimization algorithm user guide. Center for Research

and Scientific Computation CRSC-TR03-11, North Carolina State University, Raleigh, NC, March 2003.

[19] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, New York, second edition, 1987.

[20] K.R. Fowler and C.T. Kelley. Community problems and solutions. http://www4.ncsu.edu/eos/users/c/ctkelley/www/community.html.

[21] K.R. Fowler, J.P. Reese, C.E. Kees, J.E. Dennis, C.T. Kelley, C.T. Miller, C. Audet, A.J. Booker, G. Coutoure, R.W. Darwin, M.W. Farthling, D.E. Finkel, J.M. Gablonsky, G. Gray, and T.G. Kolda. A comparison of optimization methods for problems involving flow and transport phenomena in saturated subsurface systems. To Appear in —.

[22] J.M. Gablonsky. DIRECT version 2.0 userguide. Technical Report CRSC-TR01-08, North Carolina State University, Raleigh, NC, April 2001.

[23] J.M. Gablonsky. *Modifications of the DIRECT Algorithm.* PhD thesis, North Carolina State University, 2001.

[24] J.M. Gablonsky and C.T. Kelley. A locally-biased form of the DIRECT algorithm. *Journal of Global Optimization*, 21:27–37, 2001.

[25] E. Galperin. The cubic algorithm. *Journal of Mathematical Analysis and Applications*, 112:635–640, 1985.

[26] G.H. Golub and C.F Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, third edition, 1983.

[27] P. Hansen, B. Jaumard, and S. Lu. Global optimization of univariate lipschitz functions: I. survey and properties. *Mathematical Programming*, 55:251–272, 1992.

[28] A. Harbaugh and M. McDonald. User's documentation for MODFLOW-96, an update to the US Geological Survey modular finite-difference ground-water flow model. Technical Report OFR 96-485, US Geological Survey, 1996.

[29] J. He, L.T. Watson, N. Ramakrishnan, C.A. Shaffer, A. Verstak, J. Jian, K. Bae, K. Bae, and W.H. Tranter. Dynamic data structures for a direct search algorithm. *Computational Optimization and Applications*, 23(1):5–25, October 2002.

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

[31] R. Horst. *Handbook of Global Optimization*. Kluwer Academic Pub., Dordrecht, 1995.

[32] P.D. Hough, T.G. Kolda, and V.J. Torczon. Asynchronous parallel pattern search for nonlinear optimization. *SIAM J. Scientific Computing*, 23(1):134–156, June 2001.

[33] J. Jahn. *Introduction to the Theory of Nonlinear Optimization.* Springer, Berlin, 1994.

[34] D.R. Jones. *The DIRECT Global Optimization Algorithm.* The Encyclopedia of Optimization. Kluwer Academic, 1999.

[35] D.R. Jones, C.D. Perttunen, and B.E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Application*, 79(1):157–181, October 1993.

[36] L. Kaufman and P.J. Rousseew. *Finding groups in data.* John Wiley, New York, 1990.

[37] K. Kavanaugh. *Nonsmooth nonlinearities in applications from hydrology.* PhD thesis, Department of Mathematics, North Carolina State University, Raleigh, NC, 2004.

[38] A.J. Kearsley. *The Use of Optimization Techniques in the Solution of Partial Differential Equations from Science and Engineering.* PhD thesis, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 1996.

[39] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*, volume 16 of *Frontiers in Applied Mathematics.* SIAM, Philadelphia, PA, first edition, 1995.

[40] C.T. Kelley. *Iterative Methods for Optimization.* Frontiers in Applied Mathematics. SIAM, Philadelphia, PA, first edition, 1999.

[41] C.T. Kelley. *Solving Nonlinear Equations with Newton's Method*. Fundamentals of Algorithms. SIAM, Philadelphia, PA, first edition, 2003.

[42] T.G. Kolda, R.M. Lewis, and V. Torczon. Optimization by direct search: New perspective on some classical and modern methods. *SIAM Review*, 45(3):385–482, 2003.

[43] T.G. Kolda, R.M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482, September 2003.

[44] A.S. Mayer, C.T. Kelley, and C.T. Miller. Optimal design for problems involving flow and transport phenoma in saturated subsurface systems. *Advance in Water Resources*, 25:1233–1256, 2002.

[45] R. Mladineo. An algorithm for finding the global maximum of a multimodal, multivariate function. *Mathematical Programming*, 34:188–200, 1986.

[46] J.A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7:308–313, 1965.

[47] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Ser. Oper. Res. Springer-Verlag, New York, 1999.

[48] J. Pinter. Globally convergenct methods for n-dimensional multiextremal optimization. *Optimization*, 17:187–202, 1986.

[49] J.D. Pinter. *Global Optimization in Action*. Kluwer Academic Pub., Netherlands, 1996.

[50] S.A. Piyawksii. An algorithm for finding the absolute extremum of a function. *USSR Computational Mathematics and Mathematical Physics*, 12:57–67, 1972.

[51] R.T. Rockafellar. Generalized directional derivatives and subgradients of non-convex functions. *Canadien Journal of Mathematics*, 32:157–180, 1980.

[52] B. Shubert. A sequential method seeking the global maximum of a function. *SIAM J. Numer. Anal.*, 9:379–388, 1972.

[53] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a dataset via the gap statistic. *J.R. Statist. Soc. B*, 63:411–423, 2001.

[54] M.H. Wright. Direct search methods: Once scorned, now respectable. In D.F. Griffiths and G.A. Watson, editors, *Proceedings of the 1995 Dundee Bennial Conference in Numerical Analysis*, pages 191–208, CRC Press, Boca Raton, FL, 1996. Pirman Res. Notes Math. Ser. 344.

[55] Y. Yao. Dynamic tunneling algorithm for global optimization. *IEEE Transactions on System, Man and Cybernetics*, 19:1222 – 1230, 1989.

[56] Y. Yao. Dynamic tunneling algorithm for global optimization. *IEEE Transactions on Systems, Man and Cybernetics*, 19:1222–1230, 1989.

[57] Z.B. Zabinsky, G.R. Wood, M.A. Steel, and W.P. Baritompa. Pure adaptive search for finite global optimization. *Mathematical Programming*, 69:443–448, 1995.

[58] H. Zhu and D.B. Bogy. DIRECT algorithm and its application to slider air bearing surface optimization. CML Technical Report 001-003, University of California at Berkeley, 2001.

Appendix A

Direct UserGuide

A.1 Introduction

The DIRECT optimization algorithm was first introduced in [35], motivated by a modification to Lipschitzian optimization. It was created in order to solve difficult global optimization problems with bound constraints and a real-valued objective function. More formally stated, DIRECT attempts to solve:

Problem A.1. (P) *Let* $a, b \in R^N$, $\Omega = \left\{x \in R^N : a_i \leq x_i \leq b_i\right\}$, *and* $f : \Omega \rightarrow R$ *be Lipschitz continuous with constant* α. *Find* $x_{opt} \in \Omega$ *such that*

$$f_{opt} = f(x_{opt}) \leq f^* + \epsilon, \tag{A.1}$$

where ϵ *is a given small positive constant.*

DIRECT is a sampling algorithm. That is, it requires no knowledge of the objective function gradient. Instead, the algorithm samples points in the domain, and uses the information it has obtained to decide where to search next. A global search algorithm like DIRECT can be very useful when the objective function is a "black box" function or simulation. An example of DIRECT being used to try and solve a large industrial problem can be found in [10].

The DIRECT algorithm will globally converge to the minimal value of the objective function [35]. Unfortunately, this global convergence may come at the expense of a large and exhaustive search over the domain. The name DIRECT comes from the shortening of the phrase "DIviding RECTangles", which describes the way the algorithm moves towards the optimum.

The DIRECT method has been shown to be very competitive with existing algorithms in its class [35]. The strengths of DIRECT lie in the balanced effort it gives to local and global searches, and the few parameters it requires to run.

The accompanying MATLAB code for this document can be found at:

http://www4.ncsu.edu/ definkel/research/index.html.

Any comments, questions, or suggestions should be directed to Dan Finkel at definkel@ncsu.edu or tim_kelley@ncsu.edu.

Section A.2 of this guide is a reference for using the MATLAB function direct.m. An example of DIRECT being used to solve a test problem is provided. Section A.3

introduces the reader to the DIRECT algorithm, and describes how it moves towards the global optimum.

A.2 How to use direct.m

In this section, we discuss how to implement the MATLAB code direct.m, and also provide an example of how to run direct.m.

A.2.1 The program direct.m

The function direct.m is a MATLAB program that can be called by the sequence:

```
[minval,xatmin,hist] = Direct('myfcn',bounds,opts);.
```

Without the optional arguments, the function can be called

```
minval = Direct(‘myfcn’,bounds);.
```

The input arguments are:

- 'myfcn' - A function handle the objective function that one wishes to be minimized.

- `bounds` - An $n \times 2$ vector which describes the domain of the problem.

$$\texttt{bounds(i,1)} \leq x_i \leq \texttt{bounds(i,2)}.$$

- `opts` - An optional vector argument to customize the options for the program.
  - `opts(1)` - Jones factor. See Definition A.3. The default value is .0001.
  - `opts(2)` - Maximum number of function evaluations. The default is 20.
  - `opts(3)` - Maximum number of iterations. The default value is 10.
  - `opts(4)` - Maximum number of rectangle divisions. The default value is 100.
  - `opts(5)` - This parameter should be set to 0 if the global minimum is known, and 1 otherwise. The default value is 1.
  - `opts(6)` - The global minimum, if known. This parameter is ignored if `opts(5)` is set to 1.

The output arguments are:

- `minval` - The minimum value that `DIRECT` was able to find.
- `xatmin` - The location of minval in the domain. An optional output argument.
- `hist` - An optional argument, `hist` returns an array of iteration history which is useful for plots and tables. The three columns returned are iteration, function evaluations, and minimal value found.

The program termination criteria differs depending on the value of the `opts(5)`. If the value is set to 1, `DIRECT` will terminate as soon as it exceeds its budget of iterations, rectangle divisions, or function evaluations. The function evaluation budget is a tentative value, since `DIRECT` may exceed this value by a slight amount if it is in the middle of an iteration when the budget has been exhausted.

If `opts(5)` is set to 0, then `DIRECT` will terminate when the minimum value it has found is within 0.01% of the value provided in `opts(6)`.

A.2.2 Example of `DIRECT`

Here we provide an example of `DIRECT` being used on the Goldstein-Price (GP) test function [17]. The function is given by the equation:

$$\begin{aligned} f(x_1, x_2) &= [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \\ &\quad [30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)] \end{aligned} \quad \text{(A.2)}$$

The domain of the GP function is $-2 \leq x_i \leq 2$, for $i \in \{1, 2\}$, and it has a global minimum value of $f_{min} = 3$. Figure A.1 shows the function plotted over its domain.

A sample call to `DIRECT` is provided below. The following commands were made in the MATLAB Command Window.

```
>> opts = [1e-4 50 500 100 0 3];
>> bounds = [-2 2; -2 2];
```

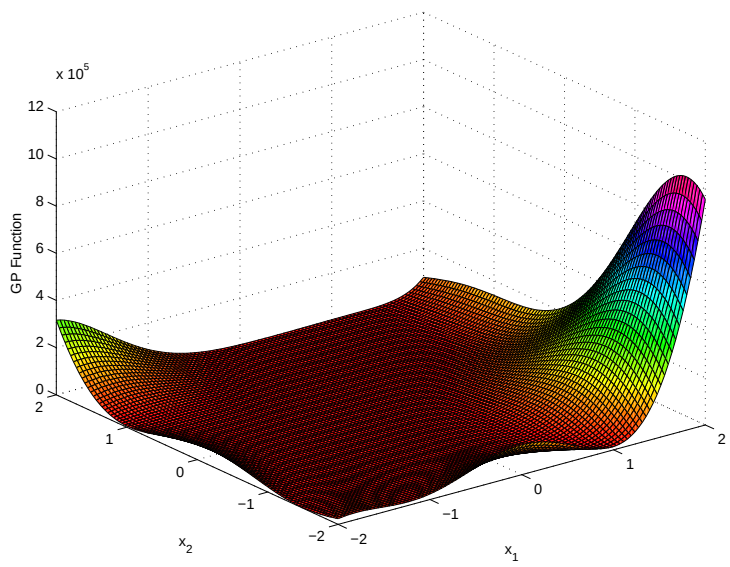

Figure A.1: The Goldstein-Price test function.

```
>> [minval,xatmin,hist] = Direct('myfcn',bounds,opts);
```

Since `opts(4)` is set to 0, `DIRECT` ignores the bounds on function evaluations, and iterations. Instead it terminates once it has come within 0.01% of the value given in `opts(6)`. The iteration history would look like:

```
>> hist


hist =


    1.0000     5.0000   200.5487

    2.0000     7.0000   200.5487

    3.0000    13.0000   200.5487

    4.0000    21.0000     8.9248
```

```
    5.0000    27.0000     8.9248

    6.0000    37.0000     3.6474

    7.0000    49.0000     3.6474

    8.0000    61.0000     3.0650

    9.0000    79.0000     3.0650

   10.0000   101.0000     3.0074

   11.0000   123.0000     3.0074

   12.0000   145.0000     3.0008

   13.0000   163.0000     3.0008

   14.0000   191.0000     3.0001
```

A useful plot for evaluating optimization algorithms is shown in Figure A.2. The x-axis is the function count, and the y-axis is the smallest value DIRECT had found.

```
>> plot(hist(:,2),hist(:,3),'-*')
```

A.3 The DIRECT Algorithm

In this section, we introduce the reader to some of the theory behind the DIRECT algorithm. For a more complete description, the reader is recommended to [35]. DIRECT was designed to overcome some of the problems that Lipschitzian Optimization encounters. We begin by discussing Lipschitz Optimization methods, and where these problems lie.

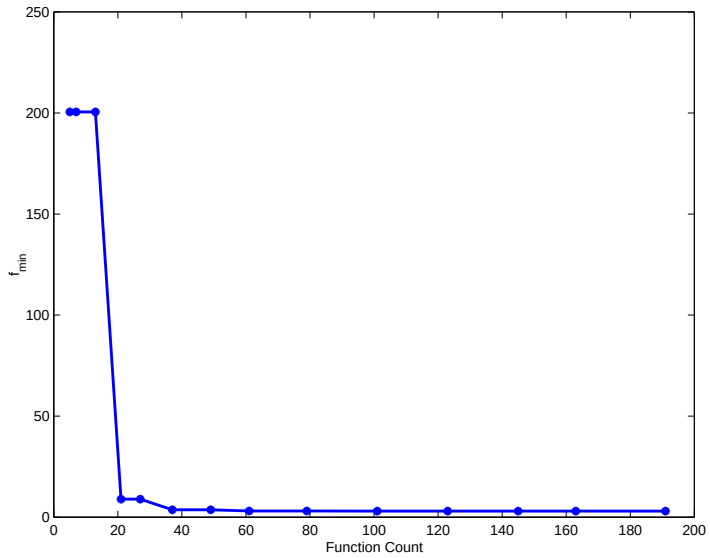

Figure A.2: `DIRECT` iteration for GP function.

A.3.1 Lipschitz Optimization

Recall the definition of Lipschitz continuity on R^1:

Definition A.2. Lipschitz *Let $M \subseteq R^1$ and $f : M \to R$. The function f is called Lipschitz continuous on M with Lipschitz constant α if*

$$|f(x) - f(x')| \leq \alpha |x - x'| \quad \forall x, x' \in M. \tag{A.3}$$

If the function f is Lipschitz continuous with constant α, then we can use this information to construct an iterative algorithm to seek the minimum of f. The Shubert algorithm is one of the more straightforward applications of this idea. [52].

For the moment, let us assume that $M = [a, b] \subset R^1$. From Equation A.3, it is

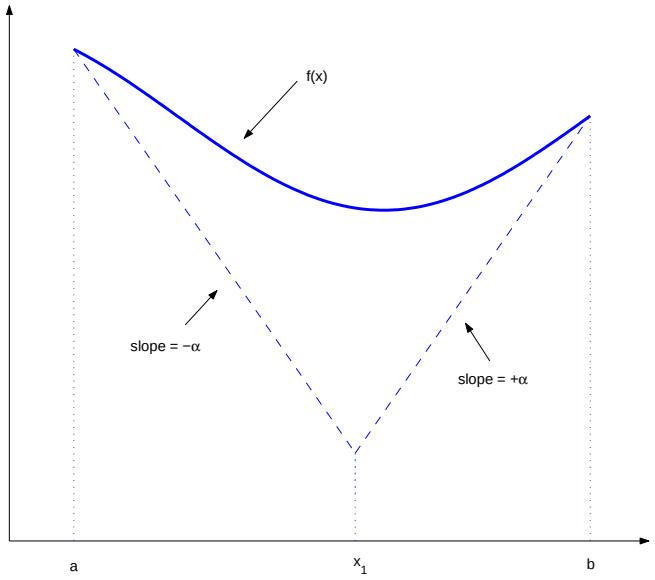

Figure A.3: An initial Lower bound for f using the Lipschitz Constant.

easy to see that f must satisfy the two inequalities

$$f(x) \geq f(a) - \alpha(x - a) \tag{A.4}$$

$$f(x) \geq f(b) + \alpha(x - b) \tag{A.5}$$

for any $x \in M$. The lines that correspond to these two inequalities form a V-shape below f, as Figure A.3 shows. The point of intersection for the two lines is easy to calculate, and provides the 1st estimate of the minimum of f. Shubert's algorithm continues by performing the same operation on the regions $[a, x_1]$ and $[x_1, b]$, dividing next the one with the lower function value. Figure A.4 visualizes this process for a couple of iterations.

There are several problems with this type of algorithm. Since the idea of end-

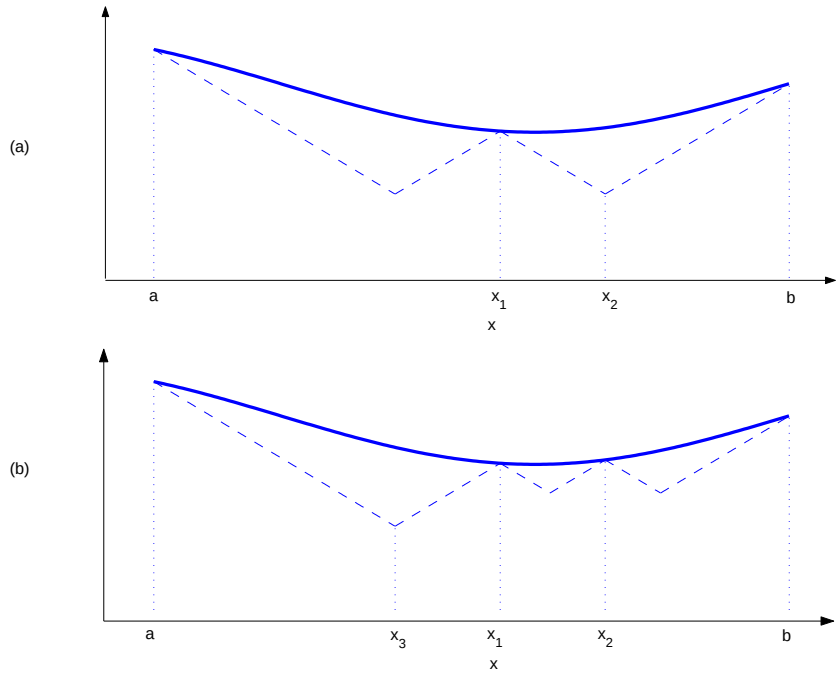

Figure A.4: The Shubert Algorithm.

points does not translate well into higher dimensions, the algorithm does not have an intuitive generalization for $N > 1$. Second, the Lipschitz constant frequently can not be determined or reasonable estimated. Many simulations with industrial applications may not even be Lipschitz continous throughout their domains. Even if the Lipschitz constant can be estimated, a poor choice can lead to poor results. If the estimate is too low, the result may not be a minimum of f, and if the choice is too large, the convergence of the Shubert algorithm will be slow.

The `DIRECT` algorithm was motivated by these two shortcomings of Lipschitzian Optimization. `DIRECT` samples at the midpoints of the search spaces, thereby removing any confusion for higher dimensions. `DIRECT` also requires no knowledge of the Lipschitz constant or for the objective function to even be Lipschitz continuous.

It instead uses all possible values to determine if a region of the domain should be broken into sub-regions during the current iteration [35].

A.3.2 Initialization of DIRECT

DIRECT begins the optimization by transforming the domain of the problem into the unit hyper-cube. That is,

$$\bar{\Omega} = \left\{ x \in R^N : 0 \leq x_i \leq 1 \right\}$$

The algorithm works in this normalized space, referring to the original space only when making function calls. The center of this space is c_1, and we begin by finding $f(c_1)$.

Our next step is to divide this hyper-cube. We do this by evaluating the function at the points $c_1 \pm \delta \mathbf{e}_i$, $i = 1, ..., n$, where δ is one-third the side-length of the hyper-cube, and $\mathbf{e}_i$ is the ith unit vector (i.e., a vector with a one in the ith position and zeros elsewhere).

The DIRECT algorithm chooses to leave the best function values in the largest space; therefore we define

$$w_i = min(f(c_1 + \delta e_i), f(c_1 - \delta e_i)), \quad 1 \leq i \leq N$$

and divide the dimension with the smallest w_i into thirds, so that $c_1 \pm \delta e_i$ are the

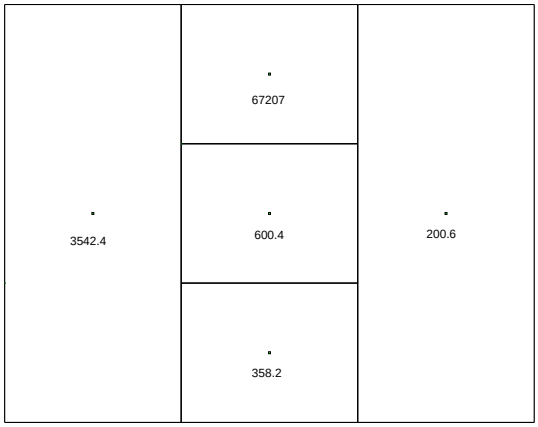

Figure A.5: Domain Space for GP function after initialization.

centers of the new hyper-rectangles. This pattern is repeated for all dimensions on the "center hyper-rectangle", choosing the next dimension by determining the next smallest w_i. Figure A.5 shows this process being performed on the GP function.

The algorithm now begins its loop of identifying potentially optimal hyper-rectangles, dividing these rectangles appropriately, and sampling at their centers.

A.3.3 Potentially Optimal Hyper-rectangles

In this section, we describe the method that `DIRECT` uses to determine which rectangles are potentially optimal, and should be divided in this iteration. `DIRECT` searches locally and globally by dividing all hyper-rectangles that meet the criteria of Definition A.3.

Definition A.3. *Let $\epsilon > 0$ be a positive constant and let f_{min} be the current best function value. A hyperrectangle j is said to be potentially optimal if there exists*

some $\hat{K} > 0$ *such that*

$$\begin{aligned} f(c_j) - \hat{K}d_j &\leq f(c_i) - \hat{K}d_i, \quad \forall i, \quad \textit{and} \\ f(c_j) - \hat{K}d_j &\leq f_{min} - \epsilon|f_{min}| \end{aligned}$$

In this definition, c_j is the center of hyper-rectangle j, and d_j is a measure for this hyper-rectangle. Jones et. al. [35] chose to use the distance from c_j to its vertices as the measure. Others have modified `DIRECT` to use different measures for the size of the rectangles [24]. The parameter ϵ is used so that $f(c_j)$ exceeds our current best solution by a non-trivial amount. Experimental data has shown that, provided $1 \times 10^{-2} \leq \epsilon \leq 1 \times 10^{-7}$, the value for ϵ has a negligible effect on the calculations [35]. A good value for ϵ is 1×10^{-4}.

A few observations may be made from this definition:

- If hyper-rectangle i is potentially optimal, then $f(c_i) \leq f(c_j)$ for all hyper-rectangles that are of the same size as i (i.e. $d_i = d_j$).
- If $d_i \geq d_k$, for all k hyper-rectangles, and $f(c_i) \leq f(c_j)$ for all hyper-rectangles such that $d_i = d_j$, then hyper-rectangle i is potentially optimal.
- If $d_i \leq d_k$ for all k hyper-rectangles, and i is potentially optimal, then $f(c_i) = f_{min}$.

An efficient way of implementing Definition A.3 can be done by utilizing the following lemma:

Lemma A.4. *Let $\epsilon > 0$ be a positive constant and let f_{min} be the current best function value. Let I be the set of all indices of all intervals existing. Let*

$$\begin{aligned} I_1 &= \{i \in I : d_i < d_j\} \\ I_2 &= \{i \in I : d_i > d_j\} \\ I_3 &= \{i \in I : d_i = d_j\}. \end{aligned}$$

Interval $j \in I$ is potentially optimal if

$$f(c_j) \leq f(c_i), \quad \forall i \in I_3, \tag{A.6}$$

there exists $\hat{K} > 0$ such that

$$\max_{i \in I_1} \frac{f(c_j) - f(c_i)}{d_j - d_i} \leq \hat{K} \leq \min_{i \in I_2} \frac{f(c_i) - f(c_j)}{d_i - d_j}, \tag{A.7}$$

and

$$\epsilon \leq \frac{f_{min} - f(c_j)}{|f_{min}|} + \frac{d_j}{|f_{min}|} \min_{i \in I_2} \frac{f(c_i) - f(c_j)}{d_i - d_j}, \quad f_{min} \neq 0, \tag{A.8}$$

or

$$f(c_j) \leq d_j \min_{i \in I_2} \frac{f(c_i) - f(c_j)}{d_i - d_j}, \quad f_{min} = 0. \tag{A.9}$$

The proof of this lemma can be found in [23]

For the example we are examining, only one rectangle is potentially optimal in the

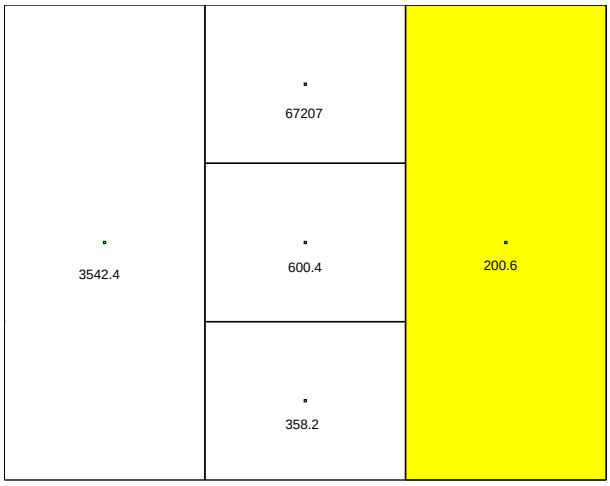

Figure A.6: Potentially Optimal Rectangle in 1st Iteration.

first iteration. The shaded region of Figure A.6 identifies it. In general, there may be more than one potentially optimal rectangle found during an iteration. Once these potentially optimal hyper-rectangles have been identified, we complete the iteration by dividing them.

A.3.4 Dividing Potentially Optimal Hyper-Rectangles

Once a hyper-rectangle has been identified as potentially optimal, `DIRECT` divides this hyper-rectangle into smaller hyper-rectangles. The divisions are restricted to only being done along the longest dimension(s) of the hyper-rectangle. This restriction ensures that the rectangles will shrink on every dimension. If the hyper-rectangle is a hyper-cube, then the divisions will be done along all sides, as was the case with the initial step.

The hierarchy for dividing potentially optimal rectangle i is determined by eval-

uating the function at the points $c_i \pm \delta_i e_j$, where e_j is the jth unit vector, and δ_i is one-third the length of the maximum side of hyper-rectangle i. The variable j takes on all dimensions of the maximal length for that hyper-rectangle. As was the case in the initialization phase, we define

$$w_j = \min \{f(c_i + \delta_i e_j), f(c_i - \delta_i e_j)\}, \quad j \in I.$$

In the above definition, I is the set of all dimensions of maximal length for hyper-rectangle i. Our first division is done in the dimension with the smallest w_j, say $w_{\hat{j}}$. `DIRECT` splits the hyper-rectangle into 3 hyper-rectangles along dimension $\hat{j}$, so that c_i, $c_i + \delta e_{\hat{j}}$, and $c_i - \delta$ are the centers of the new, smaller hyper-rectangles. This process is done again in the dimension of the 2nd smallest w_j on the new hyper-rectangle that has center c_i, and repeated for all dimensions in I.

Figure A.7 shows several iterations of the `DIRECT` algorithm. Each row represents a new iteration. The transition from the first column to the second represents the identifying process of the potentially optimal hyperrectangles. The shaded rectangles in column 2 are the potentially optimal hyper-rectangles as identified by `DIRECT`. The third column shows the domain after these potentially optimal rectangles have been divided.

A.3.5 The Algorithm

We now formally state the `DIRECT` algorithm.

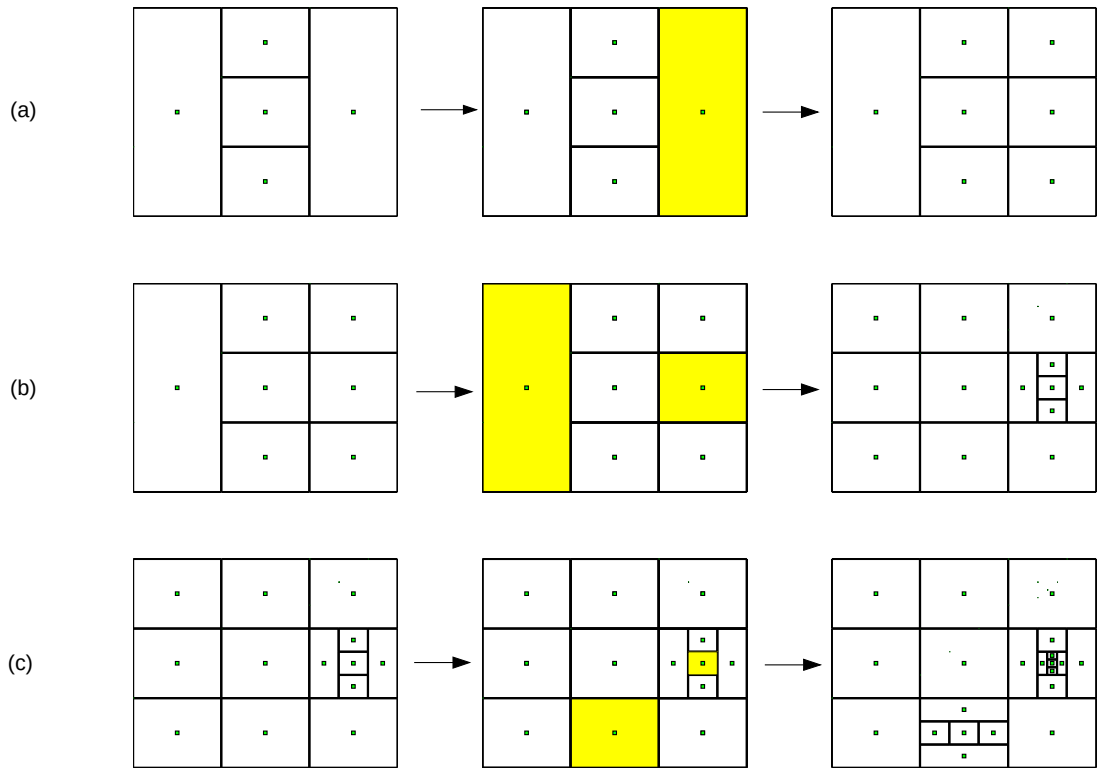

Figure A.7: Several Iterations of DIRECT.

Figure A.8 shows the domain of the GP function after the Direct algorithm terminated. The termination occured when DIRECT was within 0.01% of the global minimum value. DIRECT used 191 function evaluations in this calculation.

A.4 Acknowledgements

This research was supported by National Science Foundation grants DMS-0070641 and DMS-0112542.

In addition, the author would like to thank Tim Kelley of North Carolina State University, and Joerg Gablonksy of the the Boeing Corporation for their guidance

	Algorithm `DIRECT`('myfcn',bounds,opts)
1:	Normalize the domain to be the unit hyper-cube with center c_1
2:	Find $f(c_1)$, $f_{min} = f(c_1)$, $i = 0$, $m = 1$
3:	Evaluate $f(c_1 \pm \delta e_i$, $1 \leq i \leq n$, and divide hyper-cube
4:	**while** $i \leq maxits$ and $m \leq maxevals$ **do**
5:	Identify the set S of all pot. optimal rectangles/cubes
6:	**for** all $j \in S$
7:	Identify the longest side(s) of rectangle j
8:	Evaluate myfcn at centers of new rectangles, and divide j into smaller rectangles
9:	Update f_{min}, $xatmin$, and m
10:	**end for**
11:	$i = i + 1$
12:	**end while**

and help in preparing this document and the accompanying code.

A.5 Test Problems

Here we present the additional test problems that were used in [35] to analyze the effectiveness of the `DIRECT` algorithm. These functions are available at the web-page listed in Section A.1. If x^*, the location of the global minimizer, is not explicitly stated, it can be found in the comments of the MATLAB code.

A.5.1 The Shekel Family

$$f(x) = -\sum_{i=1}^{m} \frac{1}{(x - a_i)^T (x - a_i) + c_i}, \quad x \in R^N.$$

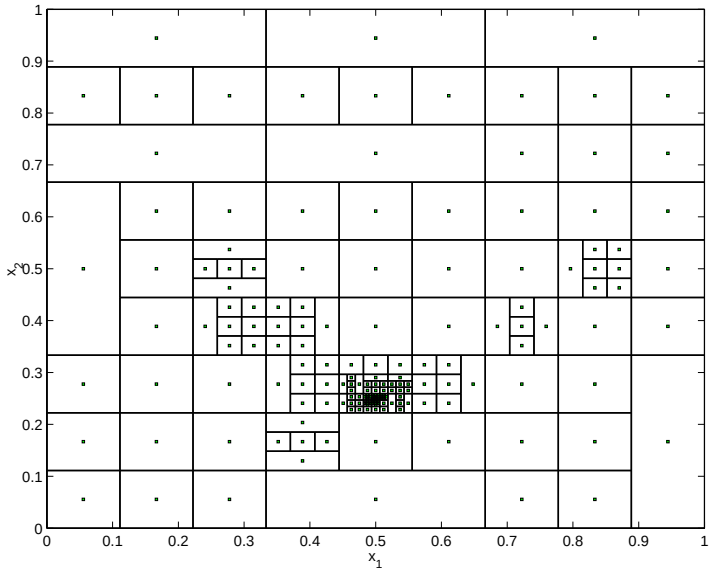

Figure A.8: Domain Space for GP function after 191 function evaluations.

There are three instances of the Shekel function, named `S5`, `S7` and `S10`, with $N = 4$, and $m = 5, 7, 10$. The values of a_i and c_i are given in Table A.1. The domain of all the Shekel functions is

$$\Omega = \left\{ x \in R^4 : 0 \leq x_i \leq 10, \;\; 1 \leq i \leq 4 \right\}.$$

All three Shekel functions obtain their global minimum at $(4, 4, 4, 4)^T$, and have optimal values:

$$\begin{aligned} \texttt{S5}^* &= -10.1532 \\ \texttt{S7}^* &= -10.4029 \\ \texttt{S10}^* &= -10.5364 \end{aligned}$$

Table A.1: Parameters for the Shekel's family of functions.

i	a_i^T				c_i
1	4.0	4.0	4.0	4.0	.1
2	1.0	1.0	1.0	1.0	.2
3	8.0	8.0	8.0	8.0	.2
4	6.0	6.0	6.0	6.0	.4
5	3.0	7.0	3.0	7.0	.4
6	2.0	9.0	2.0	9.0	.6
7	5.0	5.0	3.0	3.0	.3
8	8.0	1.0	8.0	1.0	.7
9	6.0	2.0	6.0	2.0	.5
10	7.0	3.6	7.0	3.6	.5

A.5.2 Hartman's Family

$$f(x) = -\sum_{i=1}^{m} c_i \exp\left(-\sum_{j=1}^{N} a_{ij}\left(x_j - p_{ij}\right)^2\right), \quad x \in R^N.$$

There are two instances of the Hartman function, named `H3` and `H6`. The values of the parameters are given in Table A.2. In `H3`, $N = 3$, while in the `H6` function, $N = 6$. The domain for both of these problems is

$$\Omega = \left\{x \in R^N : 0 \le x_i \le 1, \;\; 1 \le i \le N\right\}.$$

The `H3` function has a global optimal value $\texttt{H3}^* = -3.8628$ which is located at $x^* = (0.1, 0.5559, 0.8522)^T$. The `H6` has a global optimal at $\texttt{H6} = -3.3224$ located at $x^* = (0.2017, 0.15, 0.4769, 0.2753, 0.3117, 0.6573)^T$.

Table A.2: Parameters for the Hartman's family of functions. First case: $N = 3, m = 4$.

i	a_i			c_i	p_i		
1	3	10	30	1	0.3689	0.1170	0.2673
2	.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	1	0.1091	0.8732	0.5547
4	.1	10	35	3.2	0.0382	0.5743	0.8828

Table A.3: Second case: $N = 6, m = 4$.

i	a_i						c_i
1	10	3	17	3.5	1.7	8	1
2	0.05	10	17	0.1	8	14	1.2
3	3	3.5	1.7	10	17	8	3
4	17	8	0.05	10	0.1	14	3.2

i	p_i					
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.6650
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

A.5.3 Six-hump camelback function

$$f(x_1, x_2) = (4 - 2.1x_1^2 + x_1^4/3)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2.$$

The domain of this function is

$$\Omega = \left\{x \in R^2 : -3 \leq x_i \leq 2, \quad 1 \leq i \leq 2\right\}$$

The six-hump camelback function has two global minimizers with values -1.032.

A.5.4 Branin Function

$$f(x_1, x_2) = \left(x - 2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10.$$

The domain of the Branin function is:

$$\Omega = \left\{x \in R^2 : -5 \leq x_1 \leq 10, \;\; 0 \leq x_2 \leq 15, \;\; \forall i\right\}.$$

The global minimum value is 0.3978, and the function has 3 global minimum points.

A.5.5 The two-dimensional Shubert function

$$f(x_1, x_2) = \left(\sum_{j=1}^{5} j\cos[(j+1)x_1 + j]\right)\left(\sum_{j=1}^{5} j\cos[(j+1)x_2 + j]\right).$$

The Shubert function has 18 global minimums, and 760 local minima. The domain of the Shubert function is:

$$\Omega = \left\{x \in R^2 : -10 \leq x_i \leq 10, \;\; \forall i \in [1, 2]\right\}.$$

The optimal value of the function is -186.7309.

A.6 Addendum to Userguide

This short document provides a description of the updates to the `direct.m` software. The software, this document, and several examples can be found at

`http://www4.ncsu.edu/~definkel/research/index.html`

The code `direct.m` is now designed to solve problems of the form

$$(P) \quad \begin{array}{ll} \min & f(x) \\ \text{subject to} & c(x) \leq 0, \quad l \leq x \leq u. \end{array}$$

The ability of the code to address constraints is a new feature. The software still uses the `DIRECT` algorithm to solve problems. When a point is sampled that violates the constraints, it assigns an artificial function value to that point. There are two approaches to determining the artificial value, and we describe them both below.

Change in Calling Structure: There is a subtle change to the calling structure of `direct.m`. In older versions, the objective function was passed to the program as a character array. In the new version, it is passed as part of a strucutre. The calling sequence is still of the form

```
>> [fmin,xmin,history] = Direct(Problem,bounds,options);
```

Here, `Problem` is a MATLAB structure. For a simple, bound-constrained problem, you only need to set one field of the `Problem` structure

```
>> Problem.f = 'MyTestProblem';
```

where your objective function is the MATLAB file `MyTestProblem.m`. The variable `bounds` is still an $n \times 2$ vector of the lower and upper bounds for each variable, and `options` is a MATLAB structure which contains specific options for the program. The options structure is described in the sample files, and by typing

```
>> help Direct
```

at the command prompt.

Choices for handling additional constraints The new version of `direct.m` has two methods built into it for solving problems which have constraints in addition to those on the bounds.

- $L1$ Penalty Functions.

  One approach is to transform the constrained problem into an unconstrained one. If we are trying to solve Problem (P), then we can instead try to solve

$$\min f(x) + \mu^T \max(c(x), 0)$$

where $l \leq x \leq u$, and μ is a user-supplied penalty parameter. This method can be very effective because information about the feasibility of a point is included in the objective function. It does, however, require explicit knowledge of the constraint, and an educated guess at the appropriate value for the penalty parameter. For example, say we are trying to solve

$$\begin{array}{lll} \min & f(x,y) = x + y & \\ \text{subject to} & c_1(x,y) = x^2 + y^2 - 6 & \leq \quad 0 \\ & c_2(x,y) = 2x + y & \leq \quad 0 \end{array}$$

and $-10 \leq x, y \leq 10$. We assume that the objective function, and the two constraints are in three separate files named `objfcn.m, c1.m` and `c2.m`, respectively. The following shows a sample call to `direct.m` for this problem:

```
>> bounds = [-10 10;-10 10];
>> Problem.f = 'objfcn';
>> Problem.constraint(1).func = 'c1';
>> Problem.constraint(1).penalty = 1;
>> Problem.constraint(2).func = 'c2';
>> Problem.constraint(2).penalty = 1;
>> Problem.numconstraints = 2;
>> options. ... (set your options)
>> [fmin,xmin,hist] = Direct(Problem,bounds,options);
```

For each constraint, you are allowed to choose a unique penalty parameter for it. NOTE, if you do not wish to use penalty function capabilities, simply leave the `Problem.constraint`, and `Problem.numconstraints` fields empty.

- Neighborhood Assignment Strategy (NAS)

  This approach was first described in [23], and was suggested by R. Carter. We do not go into the details, and instead refer the reader to [23, 22] for more information on the details of this approach.

  If the constraints $c(x)$ are not explicitly known, or you do not wish to use penalty approach for addressing your c, `direct.m` has a feature which allows the algorithm to assign a value to infeasible points based on the values of feasible points in it's neighborhood.

  To use this approach, add the additional option:

```
>> options.impcons = 1;
```

  By setting this option, `direct.m` will now expect the objective function to return two values. A sample header for the objective function is now

```
 [retval, fflag ] = testfcn(x);
```

  A sample call to the program is:

```
>> bounds = [-10 10;-10 10];
>> Problem.f = 'objfcn';
```

```
>> options. ... (set your options)

>> options.impcons = 1;

>> [fmin,xmin,hist] = Direct(Problem,bounds,options);
```

Please see the sample files on the web-site for complete examples.

Additionally, a modified version of `DIRECT`, named `DIRECT`-ϵ, is now available in the code. A description of this method can be found in Chapter 3. To use this version of `DIRECT`, update the options:

```
>> options.method = 1;
```

The default is the original version of `DIRECT`.