# DIRECT Optimization Algorithm User Guide

Daniel E. Finkel
Center for Research in Scientific Computation
North Carolina State University
Raleigh, NC 27695-8205
definkel@unity.ncsu.edu

March 2, 2003

**Abstract**

The purpose of this brief user guide is to introduce the reader to the DIRECT optimization algorithm, describe the type of problems it solves, how to use the accompanying MATLAB program, direct.m, and provide a synopis of how it searches for the global minium. An example of DIRECT being used on a test problem is provided, and the motiviation for the algorithm is also discussed. The Appendix provides formulas and data for the 7 test functions that were used in [3].

## 1 Introduction

The DIRECT optimization algorithm was first introduced in [3], 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 1.1 (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* $\alpha$. *Find* $x_{opt} \in \Omega$ *such that*

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

*where* $\epsilon$ *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 [1].

The DIRECT algorithm will globally converge to the minimal value of the objective function [3]. 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 [3]. 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 the email address above.

Section 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 3 introduces the reader to the DIRECT algorithm, and describes how it moves towards the global optimum.

# 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.

## 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.

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

- opts - An optional vector argument to customize the options for the program.
  - opts(1) - Jones factor. See Definition 3.2. 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)`.

## 2.2 Example of `DIRECT`

Here we provide an example of `DIRECT` being used on the Goldstein-Price (GP) test function [2]. The function is given by the equation:

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

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 1 shows the function plotted over its domain.

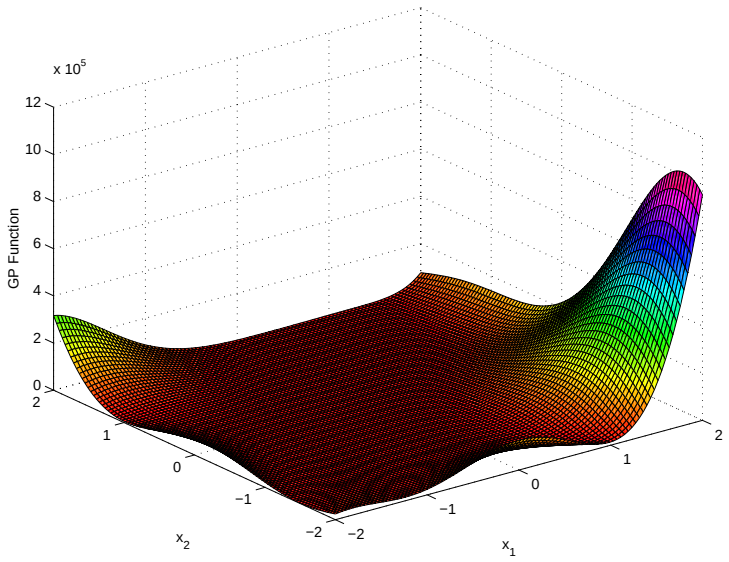


Figure 1: The Goldstein-Price test function.

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];
>> [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 2. The x-axis is the function count, and the y-axis is the smallest value `DIRECT` had found.

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

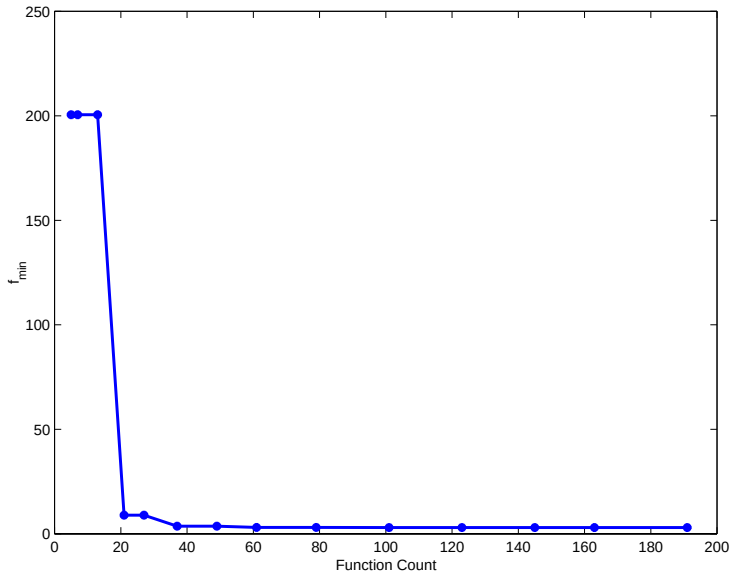


Figure 2: `DIRECT` iteration for GP function.

# 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 [3]. 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.

## 3.1 Lipschitz Optimization

Recall the definition of Lipschitz continuity on $R^1$:

**Definition 3.1 Lipschitz** *Let $M \subseteq R^1$ and $f : M \to R$. The function $f$ is called Lipschitz continuous on $M$ with Lipschitz constant $\alpha$ if*

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

If the function $f$ is Lipschitz continuous with constant $\alpha$, 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. [9].

For the moment, let us assume that $M = [a, b] \subset R^1$. From Equation 3, it is easy to see that $f$ must satisfy the two inequalities

$$f(x) \geq f(a) - \alpha(x - a) \tag{4}$$
$$f(x) \geq f(b) + \alpha(x - b) \tag{5}$$

for any $x \in M$. The lines that correspond to these two inequalities form a V-shape below $f$, as Figure 3 shows. The point of intersection for the two lines is easy to calculate, and

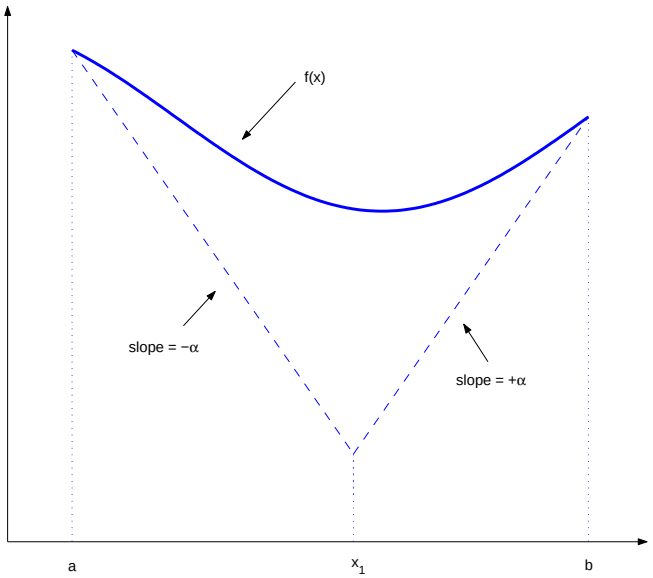


Figure 3: An initial Lower bound for $f$ using the Lipschitz Constant.

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 4 visualizes this process for a couple of iterations.

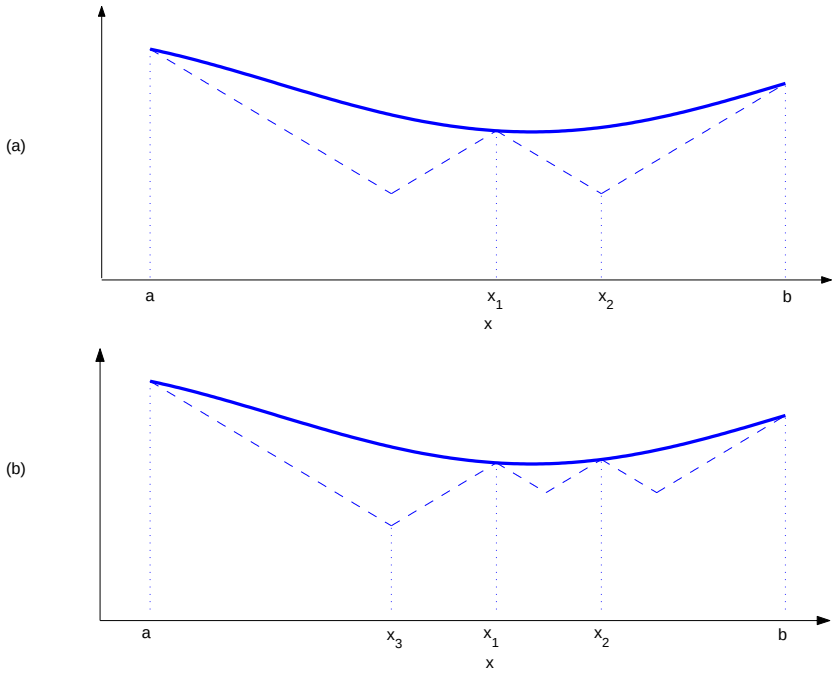


Figure 4: The Shubert Algorithm.

There are several problems with this type of algorithm. Since the idea of endpoints 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 [3].

### 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 $\delta$ is one-third the side-length of the hyper-cube, and $\mathbf{e}_i$ is the $i$th unit vector (i.e., a vector with a one in the $i$th 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 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 5 shows this process being performed on the GP function.

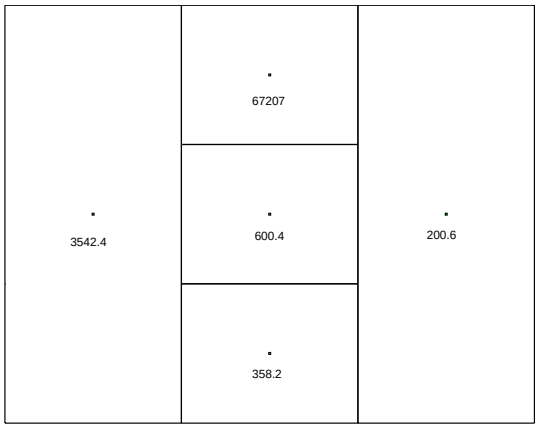


Figure 5: Domain Space for GP function after initialization.

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

### 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 3.2.

**Definition 3.2** *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. [3] 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 [6]. The parameter $\epsilon$ 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 $\epsilon$ has a negligible effect on the calculations [3]. A good value for $\epsilon$ is $1 \times 10^{-4}$.

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 3.2 can be done by utilizing the following lemma:

**Lemma 3.3** *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*

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

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

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

*and*

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

*or*

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

The proof of this lemma can be found in [5]

For the example we are examining, only one rectangle is potentially optimal in the first iteration. The shaded region of Figure 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.

### 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 evaluating the function at the points $c_i \pm \delta_i e_j$, where $e_j$ is the $j$th unit vector, and $\delta_i$ 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

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

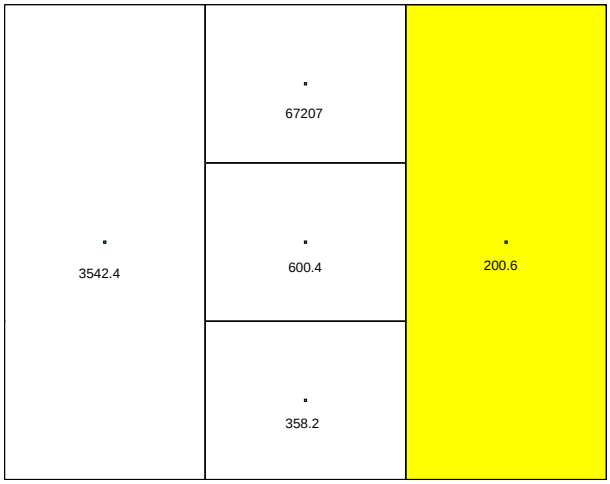


Figure 6: Potentially Optimal Rectangle in 1st Iteration.

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 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.

### 3.5 The Algorithm

We now formally state the `DIRECT` algorithm.

| | **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** |

Figure 8 shows the domain of the GP function after the Direct algorithm terminated.

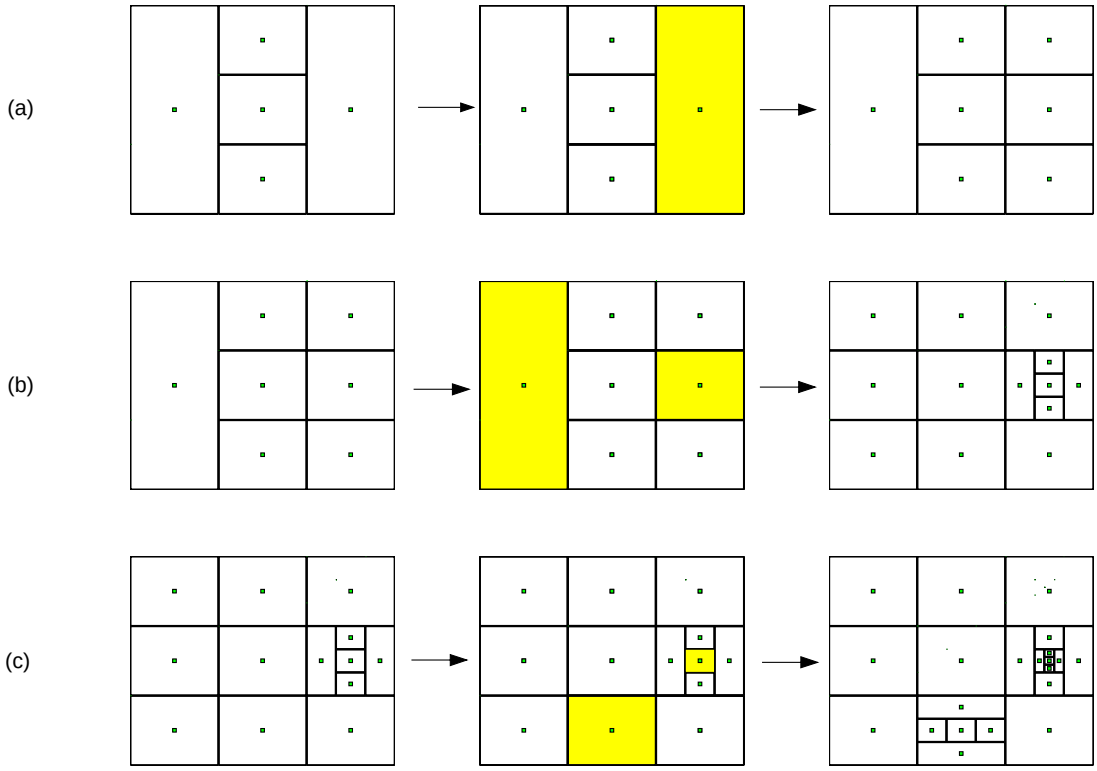


Figure 7: Several Iterations of `DIRECT`.

The termination occured when `DIRECT` was within 0.01% of the global minimum value. `DIRECT` used 191 function evaluations in this calculation.

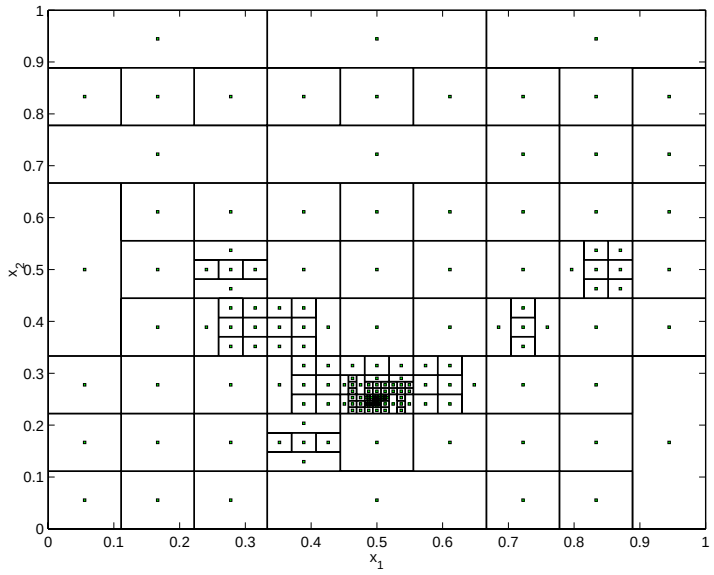


Figure 8: Domain Space for GP function after 191 function evaluations.

Table 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 |

# 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 and help in preparing this document and the accompanying code.

# A Appendix

Here we present the additional test problems that were used in [3] to analyze the effectiveness of the `DIRECT` algorithm. These functions are available at the web-page listed in Section 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.1 The Shekel Family

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

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 1. The domain of all the Shekel functions is

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

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 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 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.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 2. In `H3`, $N = 3$, while in the `H6` function, $N = 6$. The domain for both of these problems is

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

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 $x^* = (0.2017, 0.15, 0.4769, 0.2753, 0.3117, 0.6573)^T$.

### A.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

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

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

### A.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:

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

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

### A.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:

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

The optimal value of the function is -186.7309.

## References


[1] 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.

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

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

[4] J.M. Gablonsky. Direct version 2.0 userguide. *Technical Report, CRSC-TR01-08, Center for Research in Scientific Computation, North Carolina State University*, April 2001.

[5] J.M. Gablonsky. *Modifications of the Direct Algorithm*. PhD Thesis. North Carolina State University, Raleigh, North Carolina, 2001.

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

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

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

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