Copyright © 1998, American Institute of Aeronautics and Astronautics, Inc.

A98-39736 AIAA-98-4751

A MODIFICATION TO JONES' GLOBAL OPTIMIZATION ALGORITHM FOR FAST LOCAL CONVERGENCE

Sigurd A. Nelson II [*†]
Research Fellow
Member AIAA
Dept. of Mech. Engineering and Applied Mechanics
University of Michigan
Ann Arbor, Michigan 48109

Panos Y. Papalambros
Professor
Member AIAA
Dept. of Mech. Engineering and Applied Mechanics
University of Michigan
Ann Arbor, Michigan 48109

ABSTRACT

The DIRECT algorithm of Jones et al. is a simple and effective Lipschitzian-based global optimization algorithm which does not need the evaluation of gradients. However, the local convergence rate is relatively slow when compared to popular quasi-Newton techniques. By incorporating the judicious use of gradients, modifications are proposed that result in a new algorithm. This new algorithm still covers effectively the design space and eliminates regions which are non-optimal but has the benefit or a fast local convergence rate.

A comparison is made using six standard test problems showing that utilization of gradients is beneficial not only for convergence but also during the first few iterations.

1 Introduction

We consider the unconstrained optimization problem with simple bounds, i.e.:

$$\begin{array}{ll} \min\limits_{\mathbf{x} \in \Re^n} & f(\mathbf{x}) \\ \text{subject to} & \begin{array}{c} x_{\mathrm{l}i} \leq x_i \leq x_{\mathrm{u}i} \\ i = 1, \ldots, n \end{array} \end{array} \quad (1)$$

A *local optimizer* is a point $\mathbf{x}^\dagger$ such that $f(\mathbf{x}^\dagger) \leq f(\mathbf{x})$ for all $\mathbf{x}$ in some neighborhood around $\mathbf{x}^\dagger$. For example, consider the one dimensional minimization problem

$$\begin{array}{ll} \min\limits_{x \in \Re} & \begin{array}{c} \sin(x) + \sin(\frac{10}{3}x) \\ + \log(x) - 0.84x + 3 \end{array} \\ \text{subject to} & 2.7 \leq x \leq 7.5 \end{array} \quad (2)$$

taken from Timenov[15] and Hansen et al.[7] Figure 1 shows that points $\mathbf{x}^A$, $\mathbf{x}^B$, and $\mathbf{x}^C$ are all local optimizers of (2).

Many methods, including the popular quasi-Newton and conjugate-gradient methods (see, for example, Dennis and Schnabel[1] or Gill et al.[4]) are fast, efficient, and explicitly designed to find local optimizers.

[*]Address all correspondence to this author.
[†]Copyright ©1998 by Sigurd A. Nelson II and Panos Y. Papalambros. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

Copyright © 1998, American Institute of Aeronautics and Astronautics, Inc.

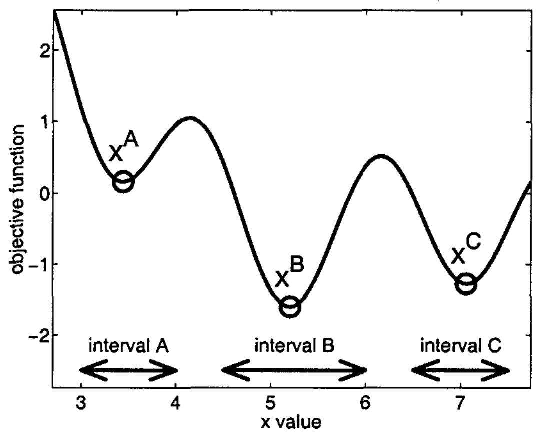

Figure 1. On the interval $[2.7, 7.5]$, the unconstrained minimization problem (2) is an example of an unconstrained minimization problem with several local solutions.

However, a *global optimizer* is a point $\mathbf{x}^*$ such that $f(\mathbf{x}^*) \le f(\mathbf{x})$ for all feasible $\mathbf{x}$. (In Figure 1, $\mathbf{x}^B$ is the global optimizer.) In general, finding the global solution is more difficult than finding a local solution. Genetic algorithms,[5] simulated annealing,[10,12] and multi-start techniques[2,6,14] require many more function calls than local techniques and do not necessarily guarantee that the global minimizer is found in a finite number of function calls. Deterministic methods such as Lipschitzian algorithms, space-covering algorithms, and branch-and-bound algorithms (see, for example, Horst and Tuy[8]) are guaranteed to find the global minimizer, but still require a large number of function calls.

One approach to balance the advantages of different algorithms is to switch between two or more algorithms (see Kleinmichel et al.[11]). For example, one may start with a convex approximation method or a genetic algorithm and switch later to sequential quadratic programming. This approach can be successful when tuned to particular applications, but there is often little mathematical reasoning when to switch algorithms.

In this paper, a global optimization algorithm is combined with a Quasi-Newton trust region algorithm so that the steps performed at every iteration are identical. In the resulting algorithm, a global minimizer is obtained and the local convergence rate to that minimizer is q-superlinear.

An overview of a global optimization algorithm by Jones et al.,[9] and a trust-region based quasi-Newton algorithm is given in Sections 2 and 3, respectively An algorithm that combines the speed of the quasi-Newton algorithm and the robustness of the global one is presented in Section 4. In Section 5 numerical examples are given followed by a discussion and closing comments in Section 6.

2 The DIRECT Algorithm

In the DIRECT algorithm of Jones et al.,[9] the design space is divided into rectangles. A decision process then selects rectangles which show promise for containing the global optimizer, and those rectangles are further subdivided. Here we explain how *potentially optimal* rectangles are chosen without the explicit use of a Lipschitz constant and then subdivided. The DIRECT algorithm is then outlined.

A Lipschitz constant is essentially a bound on the rate of change (i.e. the derivative) of a function, and knowledge of the Lipschitz constant can be used to place bounds on the both the distance from a solution and the extremal value of a function within a given region. A function f is said to be *Lipschitz continuous* over a region $X \subset \Re^n$ with Lipschitz constant K if for any two points $\mathbf{x}$ and $\mathbf{y}$ in X,

$$|f(\mathbf{x}) - f(\mathbf{y})| \le K\|\mathbf{x} - \mathbf{y}\| \tag{3}$$

In global optimization the utility of a Lipschitz constant stems from rearranging (3) in order to establish a lower bound of the objective function over a predefined region. For example, consider a rectangle with the lower left-most point being $\mathbf{x}_l$ and the upper-right-most point $\mathbf{x}_u$. Assume for the moment that the Lipschitz constant K is known, and that the value of the function is also known at a point $\mathbf{x}$ somewhere within the rectangle. The lower bound on the value of the objective function within the rectangle is then:

$$f \ge f(\mathbf{x}) - K \max_{x_{li} \le y_i \le x_{ui}} \{\|\mathbf{y} - \mathbf{x}\|\} \tag{4}$$

In one dimension, a rectangle can be thought of as an interval along a line, so that (4) can be graphically portrayed as in Figure 2. Starting from the point where the function is known, the lines with slopes K and $-K$ form lower bounds, so that

$$f \ge f(\mathbf{x}) - K\alpha \tag{5}$$

Copyright © 1998, American Institute of Aeronautics and Astronautics, Inc.

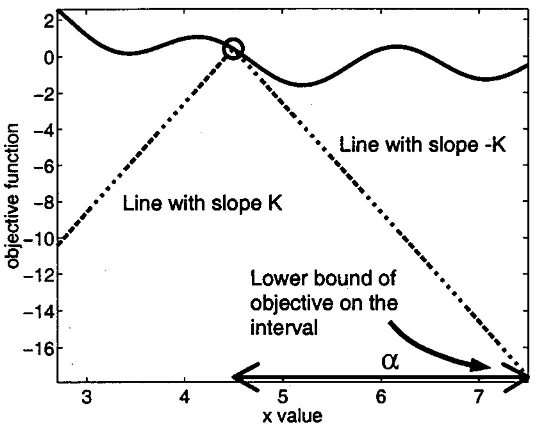

Figure 2. Graphical depiction of how a Lipschitz constant can be used to calculate a lower bound of a function on an interval.

where α is the distance of the farthest point in the rectangle from **x**:

$$\alpha = \sqrt{\sum_{i=1}^{n} \max\{(x_{li} - x_i)^2, (x_{ui} - x_i)^2\}} \qquad (6)$$

For each rectangle, it will be necessary to estimate a lower bound on the objective function, so that rectangles showing the promise of containing the global minimizer can be further subdivided.

Consider now Figure 3 where the interval $[x_l, x_u]$ is divided into four intervals $[x_l^1, x_u^1]$, $[x_l^2, x_u^2]$, $[x_l^3, x_u^3]$, $[x_l^4, x_u^4]$, and one function value $f^j = f(\mathbf{x}^j)$ is given within each interval. Using inequality (5), if the Lipschitz constant were known then a bound for the minimum value within each interval could be computed.

In engineering it is rare for a Lipschitz constant to be known a-priori, so the concept of *potential optimality*[9] is introduced. A rectangle is *potentially optimal* if there exists a Lipschitz constant $\bar{K}$ that will make its lower bound lower than the lower bound of any other rectangle. More precisely, if there is a total of m rectangles in the design space, the rectangle with index j is said to be potentially optimal if there exists a constant $\bar{K} > 0$ such that

$$\begin{gathered} f^j - \bar{K}\alpha^j \le f^k - \bar{K}\alpha^k \\ \text{for all } k = 1, \dots, m \end{gathered} \qquad (7)$$

An intuitive graphical method for identifying potentially optimal rectangles is shown in Figure 3. A point is plotted for each interval in Figure 3. For each point, the ab-

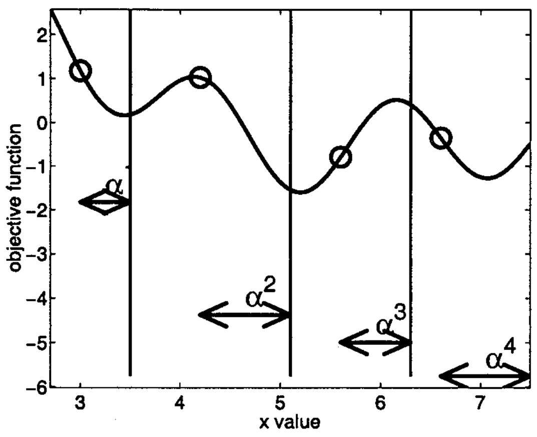

Figure 3. A plot of the one-dimensional minimization problem (2) on the interval $[2.7, 7.5]$ subdivided into several subintervals.

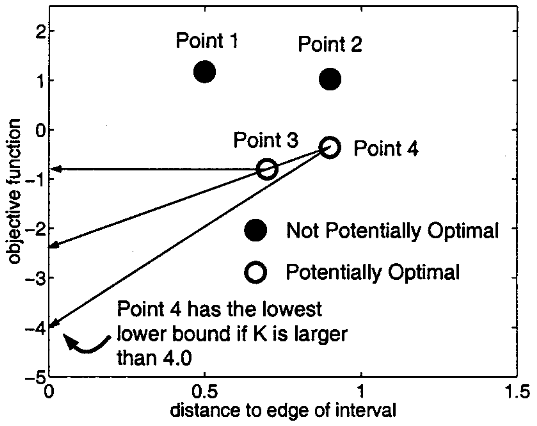

Figure 4. Potentially optimal rectangles can be determined graphically by plotting α^j along the abscissa and the function value along the ordinate. The potentially optimal rectangles have points in the lower convex hull.

scissa is given by α^j and the ordinate is the known function value within that interval. For a given value of $\bar{K}$, the lower bound of the objective function for each rectangle is found by drawing a line with slope $\bar{K}$ from the corresponding point (α, f) to the y-intercept. As discussed in Jones et al.,[9] a rectangle is potentially optimal if it is in the lower convex hull of the points in Figure 4.

Once a set of potentially optimal rectangles has been selected, each rectangle is subdivided into smaller rectangles. In the original DIRECT algorithm, the manner in which rectangles are subdivided is based on a few key insights: First, a function call is made as close as possible to the center of any rectangle. Second, in the absence

Copyright © 1998, American Institute of Aeronautics and Astronautics, Inc.

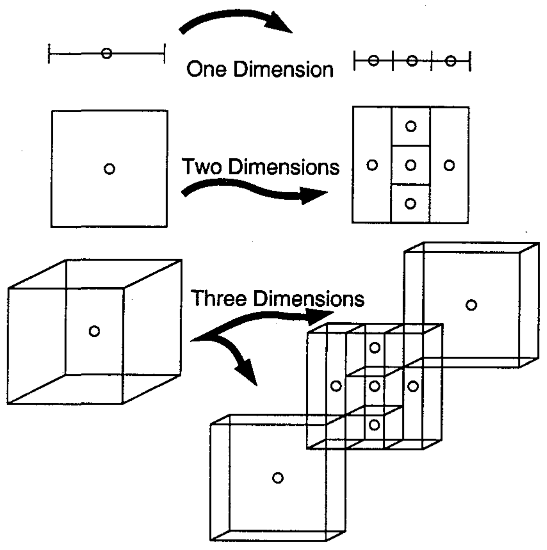

Figure 5. Examples of how rectangles are subdivided and sampled in one, two, and three dimensions.

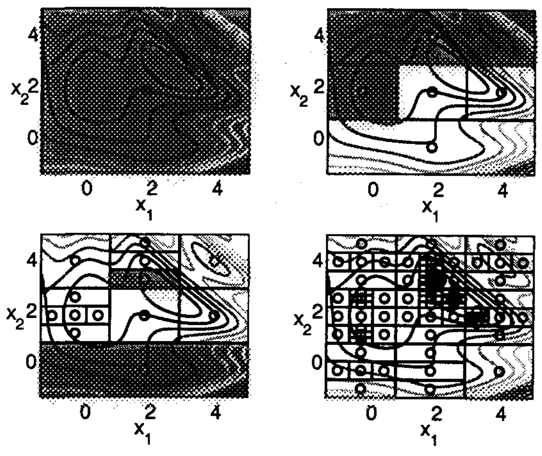

Figure 6. The first, second, third and eighth iterations of the DIRECT algorithm when applied to the optimization problem (8). The potentially optimal rectangles are shaded and are subdivided during the next iteration. The respective number of function calls is 1, 5, 13, and 113.

of any directional (i.e., gradient) information, rectangles are subdivided as evenly as possible. Thus, any potentially optimal rectangle is subdivided into $2n+1$ rectangles, with each new point being along the coordinate directions from the point in the center of the original rectangle. This process is shown in Figure 5.

The DIRECT algorithm is now summarized as follows:

Algorithm 1 (The DIRECT Algorithm)

1. Start with a set of points $\{\mathbf{x}^j\}$ and corresponding upper and lower bounds $\mathbf{x}_u^j$ and $\mathbf{x}_l^j$ representing the surrounding rectangle for each $\mathbf{x}^j$.
2. Identify those points whose rectangles are *potentially optimal.*
3. For every potentially optimal point, subdivide the corresponding rectangle into $2n+1$ new rectangles.
4. Check stopping criteria. If appropriate, go to Step 2.

In Figure 6, a few steps of the DIRECT algorithm are portrayed for the two-dimensional minimization problem:

$$\begin{array}{ll} \min & 2+\frac{(x_2-x_1^2)^2}{100}+(1-x_1)^2 \\ & +2(2-x_2)^2+7\sin\frac{x_1}{2}\sin\frac{7x_1x_2}{10} \\ \mathbf{x}\in\Re^n & \\ \text{subject to} & -\sqrt{2}\le x_1\le 5 \\ & -\sqrt{2}\le x_2\le 5 \end{array} \tag{8}$$

The concept of potential optimality allows for a balance between smaller rectangles with good function values and large unexplored rectangles, so during every iteration many rectangles can be subdivided. The result is a simultaneous covering of the design space and a heavy concentration of points near the global optimizer.

The DIRECT algorithm does have some drawbacks. First, the local convergence rate is slower than bisection because no gradient information is used. Second, unless some upper limit on the Lipschitz constant is known, the stopping criterion is simply a limit on the number of function calls. The net result is that DIRECT becomes slow and expensive in higher dimensions and near the solution.

3 Quasi-Newton Trust Region Algorithms

In the popular quasi-Newton trust region methods, at each iteration the function f is quadratically estimated around the current point $\mathbf{x}^j$ (the superscript j denotes the iteration) by f_{aprx} using the gradient $\nabla f(\mathbf{x}^j)$ and an estimate of the Hessian $\mathbf{B}^j$. Subsequent iterations are calculated by minimizing the approximate model:

$$\begin{array}{lll} \min & f_{\text{aprx}}=f^j+\nabla f^j\mathbf{s}+\frac{1}{2}\mathbf{s}^T\mathbf{B}^j\mathbf{s} \\ \mathbf{s}\in\Re^n & \\ \text{subject to} & \|\mathbf{s}\|_\infty\le\Delta \end{array} \tag{9}$$

The trust region radius Δ is necessary because (i) it is

Copyright © 1998, American Institute of Aeronautics and Astronautics, Inc.

possible for $\mathbf{B}^j$ to be indefinite, so that (9) may be unbounded if there were no restriction on $\mathbf{s}$; and (ii) the approximate model (9) is only useful in the region surrounding $\mathbf{x}^j$ where f_{aprx} gives reasonable predictions for descent directions.

At every iteration, the trust region is modified when the approximate model f_{aprx} gives a good prediction for the change in the objective function using the rule:

$$\Delta^{j+1} = \begin{cases} \max\{2\Delta, 4\|\mathbf{s}^j\|_\infty\} & \text{if } 0.9 < r \\ \Delta^j & \text{if } 0.1 \le r \le 0.9 \\ \min\{\Delta/4, \|\mathbf{s}^j\|_\infty/2\} & \text{if } r < 0.1 \end{cases} \qquad (10)$$
$$\text{where } r = \frac{f(\mathbf{x}^j) - f(\mathbf{x}^j + \mathbf{s}^j)}{f_{\text{aprx}}(\mathbf{0}) - f_{\text{aprx}}(\mathbf{s}^j)}$$

The resulting algorithm is summarized as:

Algorithm 2 (Quasi-Newton Trust Region)

1. Set the iteration counter $j = 1$. Choose some $\mathbf{x}^1$ as an initial point and $\mathbf{B}^1$ as an initial Hessian estimate. Define a trust region radius $\Delta^1 > 0$.
2. Solve the approximate problem (9) and denote the solution $\mathbf{s}^j$. If $\mathbf{s}^j = \mathbf{0}$ then stop.
3. Calculate the objective function f, the approximate penalty function f_{aprx}, and ratio r at the point $\mathbf{x}^j + \mathbf{s}^j$. If $r > 0$ (i.e., the step produces descent) go to Step 4; otherwise set $\Delta^j = \|\mathbf{s}^j\|_\infty/4$, $\mathbf{x}^{j+1} = \mathbf{x}^j$, $j = j+1$, and go to Step 2.
4. Alter the trust region radius according to (10).
5. Generate $\mathbf{B}^{j+1}$. Set $\mathbf{x}^{j+1} = \mathbf{x}^j + \mathbf{s}^j$, $j = j+1$ and go to Step 2.

Because the Hessian estimate need not be positive definite, a straightforward BFGS update without the Powell safeguard for positive definiteness[13] is used.

Quasi-Newton methods are powerful and fast, usually characterized by q-superlinear convergence. That is, if a point $\mathbf{x}^j$ is close to a local minimizer $\mathbf{x}^*$ and the Hessian estimate $\mathbf{B}$ is close enough to the Hessian $\nabla^2 f(\mathbf{x}^*)$, then every iteration will be closer to the local minimizer in the following manner:

$$\|\mathbf{x}^{j+1} - \mathbf{x}^*\| \le c^j \|\mathbf{x}^j - \mathbf{x}^*\| \quad \text{for all } j \text{ greater than some } j_0 \qquad (11)$$

where each c^j is not only less than one but also vanishes as j tends to infinity. Even though q-superlinear behavior is a strong statement, the algorithm will only converge to a nearby local minimizer because there is no mechanism to search elsewhere in the design space.

4 A Faster Global Optimization Algorithm

The global optimization algorithm discussed in Section 2 and the quasi-Newton algorithm discussed in Section 3 have weaknesses and strengths that are complementary: The DIRECT algorithm can find a global optimizer but has very slow local convergence, whereas the quasi-Newton algorithm has no mechanism to find the global optimizer but has fast local convergence. One is tempted then to examine how to blend these two algorithms in a manner that preserves the strengths of each.

In the proposed algorithm, the design space is subdivided into rectangles, and potentially optimal rectangles are further subdivided. The difference is that not all potentially optimal rectangles are divided in the same manner. Most rectangles are divided in the manner described in the original DIRECT algorithm. However the point corresponding to the best known function value is used to calculate a single quasi-Newton step by solving:

$$\begin{aligned} \min_{\mathbf{s} \in \Re^n} \quad & f_{\text{aprx}} = \nabla f^j \mathbf{s} + \tfrac{1}{2}\mathbf{s}^T \mathbf{B}^j \mathbf{s} \\ \text{subject to} \quad & \|\mathbf{s}\|_\infty \le \Delta^j \\ & x_{\text{l}i} \le x_i^j + s_i \le x_{\text{u}i} \\ & i = 1, \ldots, n \end{aligned} \qquad (12)$$

The quantities $\mathbf{B}^j$ and Δ^j depend upon how the point $\mathbf{x}^j$ was originally determined. If $\mathbf{x}^j$ was the result of a quasi-Newton calculation during some previous iteration, then it is possible to estimate $\mathbf{B}^j$ using previously obtained gradient information. Additionally, an appropriate value for Δ^j is also available based on past calculations. If $\mathbf{x}^j$ was not the result of a quasi-Newton calculation during some past iteration, reasonable default values can be used such as the identity matrix for $\mathbf{B}^j$ and $\Delta^j = 1$.

Two procedures must be performed with the new point $\mathbf{x}^{j+} = \mathbf{x}^j + \mathbf{s}^j$. First, $\mathbf{x}^{j+}$ may not necessarily reside in the same rectangle as $\mathbf{x}^j$, so the rectangle containing $\mathbf{x}^{j+}$ must be identified and divided into two new rectangles, with $\mathbf{x}^{j+}$ as close to the center of one of the new rectangles as possible.

Second, if $f(\mathbf{x}^{j+}) < f(\mathbf{x}^j)$, then $\mathbf{x}^j$ has successfully been used to calculate a new point, it is reasonable to assume that $\mathbf{x}^{j+}$ is closer to an optimum than $\mathbf{x}^j$. Therefore, if during future iterations a rectangle containing $\mathbf{x}^j$ is ever deemed *potentially optimal*, then it is only subdivided into thirds along the longest side, as opposed to being divided into $2n+1$ new rectangles.

The newly proposed algorithm is outlined below:

Algorithm 3 (The Modified DIRECT Algorithm)

1. Start with a set of points $\{\mathbf{x}^j\}$. For each point $\mathbf{x}^j$

Copyright © 1998, American Institute of Aeronautics and Astronautics, Inc.

there is a corresponding upper and lower bounds $\mathbf{x}_l^j$ and $\mathbf{x}_u^j$.

2. Identify those points whose rectangles are *potentially optimal.*
3. Of the potentially optimal rectangles, identify the rectangle containing the best function value. For the moment, denote this point $\mathbf{x}_{\text{bst}}^j$.
4. For the point $\mathbf{x}_{\text{bst}}^j$:
   (a) define a new point $\mathbf{x}^{j+} = \mathbf{x}_{\text{bst}}^j + \mathbf{s}^j$ by through the use of (12);
   (b) Find the rectangle which contains $\mathbf{x}^{j+}$ and divide it into two rectangles so that $\mathbf{x}^{j+}$ is as close as possible to the center of the new rectangle.
5. For every potentially optimal rectangle that has not been used to successfully produce a point via (12), divide it into $2n+1$ new rectangles as described in the original DIRECT algorithm.
6. For every potentially optimal rectangle which has been used to successfully produce a point via (12), divide it into 3 new rectangles along the longest edge.
7. Check stopping criteria. If appropriate, go to Step 2.

For most quasi-Newton algorithms, if the current point is close enough to a local minimum, then every iteration produces a point with a better objective function. With respect to the modified DIRECT algorithm, this means that if $\mathbf{x}_{\text{bst}}^j$ is close to the global minimizer $\mathbf{x}^*$, then $f(\mathbf{x}^{j+}) < f(\mathbf{x}_{\text{bst}}^j)$. Therefore, during the next iteration the rectangle containing $\mathbf{x}^{j+}$ will not only be potentially optimal, but the point will be used as the basis of another quasi-Newton step. The result is that once the modified DIRECT algorithm finds a point close enough to the global minimizer, a sequence of points is calculated which converges q-superlinearly to the global minimum.

In contrast to multi-start and a number of other global optimization algorithms, in the modified DIRECT algorithm every iteration consists of the same steps. There are no separate phases for the local and global search, yet a fast local convergence rate is still achieved.

In order to compare with the original DIRECT algorithm, Figure 7 portrays a few steps of the modified DIRECT algorithm when applied to (8). Although intuitive, it is difficult to rigorously explain how the use of a gradient helps during the first few iterations, long before q-superlinear convergence is evident. However, with the computational results discussed in Section 5, the improvement was indeed evident.

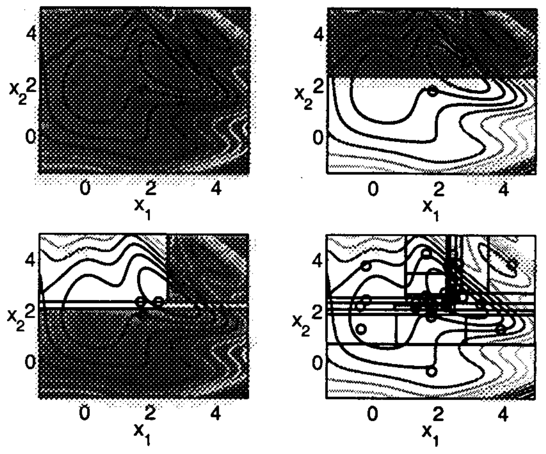

Figure 7. The first, second, fourth and eleventh iterations of the modified DIRECT algorithm when applied to the optimization problem (8). The respective number of function calls (including those used for finite differences) is 3, 6, 12, and 31.

5 Numerical Examples

When comparing the DIRECT and modified DIRECT algorithms, we report the number of function calls required to find a point within some tolerance ε_f.

$$\varepsilon_f = |f(\mathbf{x}) - f^*| \tag{13}$$

In all cases, the algorithms were run for several more iterations than were necessary and we report the first point which satisfies the tolerances of $\varepsilon_f < 0.01$ and $\varepsilon_f < 0.0001$. The six unconstrained minimization problems used in this study were originally proposed by Dixon and Szegö,[3] so the locations of the optima are known. The number of dimensions and local minima are summarized in Table 1. The number of function calls (including those needed for finite difference estimations of gradients) required to meet or exceed set limits of ε_f is summarized in Table 2.

It should be noted that there is a minor difference between the implementation reported here and the original DIRECT implementation of Jones et al. The DIRECT implementation of Jones et al. uses a more sophisticated method of subdividing rectangles by examining the function values prior to dividing the rectangles.

From Table 2, it can be seen that the modified DIRECT algorithm represents a substantial improvement over the original DIRECT algorithm. In nearly all cases, the number of function calls required to locate the global optimizer was reduced by a factor of two or more. However, the results presented here can be somewhat deceiv-

Copyright © 1998, American Institute of Aeronautics and Astronautics, Inc.

Table 1. Number of dimensions and local minima for the test problems. Every test function had one unique global minimizer.

	Dimension	Local Minima
Shekel 5	4	5
Shekel 7	4	7
Shekel 10	4	10
Hartman 3	3	4
Hartman 6	6	4
Goldstein & Price	2	4

Table 2. Number of function evaluations required for the DIRECT (D) and modified DIRECT (mD) algorithms to find points whose objective function value is within a set tolerance of the global optimum. These figures reflect the additional function calls necessary to evaluate finite differences.

	$\varepsilon_f < 0.01$		$\varepsilon_f < 0.0001$	
	D	mD	D	mD
Shekel 5	371	55	707	90
Shekel 7	187	85	278	174
Shekel 10	187	76	286	180
Hartman 3	194	111	1833	139
Hartman 6	33559	81	> 50000	81
Goldstein & Price	113	45	201	65

ing because the criteria do not address the robustness of the algorithms. For both the DIRECT and modified DIRECT algorithms, it is possible that the algorithms found the global optimizer quickly, but that more function calls would be necessary to verify that the global optimizer is indeed found. This is most noticeable for the Hartman 6 test problem, where the modified DIRECT algorithm reached the global optimum in 75 function calls, but needed to run hundreds more to verify that the optimum was found. In effect, more reliable and prudent stopping criteria are needed for both algorithms.

With that said, the results presented in Table 2 imply that the use of gradients can speed up a global optimization algorithm.

6 Conclusions

In this paper, we have modified the DIRECT global optimization algorithm to speed up the local convergence rate by combining a trust region step. The resulting algorithm retains the strong qualities of both algorithms in that in that the global optimizer is found and the convergence rate to the global optimizer is fast.

The computational results verify that the additional use of gradients and a local search mechanism does indeed make the algorithm faster, but work is still needed in order find more satisfactory stopping criteria.

Acknowledgments

This research has been partially supported by the Automotive Research Center at the University of Michigan, a US Army Center of Excellence in Modeling and Simulation of Ground Vehicles, under Contract No. DAAE07-94-C-R094. This support is gratefully acknowledged.

The authors also would like to thank Dr. Don Jones of General Motors Research, whose insights on the original DIRECT algorithm where very helpful.

REFERENCES

[1] J. Dennis and R. Schnabel. *Numerical methods for unconstrained optimization*. Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

[2] L. C. W. Dixon, J. Gomulka, and G. P. Szegö. Reflections on the global optimizatoin problem. In L. C. W. Dixon, editor, *Global Optimization in Action*. Academic Press, New York, New York, 1976.

[3] L. C. W. Dixon and G. P. Szegö. The global optimization problem: An introduction. In L. C. W. Dixon and G. P. Szegö, editors, *Towards Global Optimization 2*. North - Holland, New York, New York, 1978.

[4] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, New York, 1981.

[5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison - Wesley, 1989.

[6] A. A. Groenwold, J. A. Snyman, and N. Stander. Modified trajectory method for pratctical global optimization problems. *AIAA Journal*, 34(10):2126 – 2131, October 1996.

[7] P. Hansen, B. Jaumard, and S. H. Lu. Global optimization of univariate lipschitz functions: II. new algorithms and computational comparison. *Mathematical Programming*, 55(3):273 – 292, 1992.

Copyright © 1998, American Institute of Aeronautics and Astronautics, Inc.

[8] R. Horst and H. Tuy. *Global Optimization, deterministic approaches*. Springer-Verlag, New York, 1990.

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

[10] S. Kirkpatrick, C. D. Gellat, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[11] H. Kleinmichel, C. Richter, and K. Schönefeld. On a class of hybrid methods for smooth constrained optimization. *Journal of Optimization Theory and Applications*, 73(3), June 1992.

[12] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[13] M. J. D. Powell. The convergence of variable metric methods for nonlinear constrained optimization calculations. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 3*, pages 27 – 64. Academic Press, New York, 1978.

[14] J. A. Snyman and L. P. Fatti. A multi-start global minimization algorithm with dynamic search trajectories. *Journal of Optimization Theory and Applications*, 54(1):121 – 141, July 1987.

[15] L. N. Timenov. An algorithm for search of a global extremum. *Engineering Cybernetics*, 15, 38 – 44 1977.