Derivative-Free Pattern Search Methods for Multidisciplinary Design Problems

J.E. Dennis, Jr., Virginia J. Torczon

CRPC-TR94470
August, 1994

Center for Research on Parallel Computation
Rice University
6100 South Main Street
CRPC - MS 41
Houston, TX 77005

Derivative-Free Pattern Search Methods for Multidisciplinary Design Problems

J. E. Dennis* Virginia J. Torczon*

Rice University

Houston, Texas 77251-1892

Abstract

There have been interesting recent developments in methods for solving optimization problems without making use of derivative (sensitivity) information. While calculus based methods that employ derivative information can be extremely efficient and very effective, they are not applicable to all MDO problems, for instance, when the function to be optimized is nondifferentiable, when sensitivity information is not available or is not reliable, or when the function values are inaccurate. In these settings, we have found that the multidirectional search method, a derivate-free method we have developed for solving nonlinear optimization problems, can be used effectively. Our analysis of the multidirectional search algorithm has led us to discover that its algebraic structure and resulting convergence theory can be related to an entire class of derivative-free methods, which we now call pattern search methods, that have been in use for decades.

The goal of this paper is to give an introduction to pattern search methods, to describe the features they share by using coordinate search, one of the earliest and best-known (if not as effective) pattern search methods, as an example, and to review some recent developments that suggest that these methods can be extended to handle problems with a mix of continuous and discrete variables—another situation that can arise in MDO problems. Finally, we will discuss when these methods are an appropriate choice for solving MDO problems.

*Department of Computational and Applied Mathematics

1. Introduction

Pattern search methods form a class of optimization methods designed to solve the problem

$$\min_{x \in S} f(x)\,,$$

where $f : I\!R^n \rightarrow I\!R$, without computing derivatives (sensitivities). To simplify the discussion, we will assume that $S = I\!R^n$ (i.e., the problem is *unconstrained*), though computational efforts have been directed at extending these methods in a reliable fashion to handle problems with constraints [15].

Pattern search methods date back to at least the 1950s [6] and have been used for parameter estimation in a wide variety of scientific and engineering applications. They have remained popular because they are simple, easy to understand, easy to program, widely applicable, and have proven to be robust in practice [23], where by "robust" we mean that the sequence of iterates produced typically converges to a stationary point of the function to be optimized.

Our investigation of pattern search methods was prompted by our interest in developing useful general-purpose parallel optimization methods [8, 21]. However, during the course of our investigations we discovered that pattern search methods have many important features that are independent of the computing environment in which they are used. These are the features we will discuss in this paper.

We reintroduce pattern search methods in the context of MDO for two reasons. First, pattern search methods have long proven to be useful for problems

that exhibit many of the difficulties found in solving MDO problems:

- they can be used when the function is either non-differentiable or the derivative (sensitivity) information proves to be unreliable,
- they can be used when the function is computed to low accuracy,
- they can be effective when the function is highly nonlinear, and
- they can be used even when only rank (order) information is available.

Good arguments can also be made for their effectiveness when a problem has many local minimizers [16]. This is because pattern search methods search in multiple directions, some of which may be "up-hill" directions, rather than limiting the search to a single descent (down-hill) direction, as is more typical of classical higher-order methods based on Taylor's series approximations to the function to be minimized. Second, recent theoretical developments [22] show that given a precise definition of pattern search methods these methods can be related analytically to classical higher-order methods based on a Taylor's series approximation to the function to be minimized. This analysis is significant for several reasons. It provides a theoretical underpinning to years of experience that suggests pattern search methods are robust in practice. But of even more importance for MDO problems, an understanding of the structure that makes this analysis possible suggests constructive ways to develop robust methods for dealing with problems that have discrete variables (such as variables based on table look-ups) and function values that are subject to random error (as in settings where not all the environmental factors can be controlled).

Of course, all these advantages do not come without a price. Because the pattern search methods are *direct search methods* (i.e., methods that neither require nor estimate derivatives to drive the search procedure but rely solely on function values), they are not as efficient as higher-order methods can be when using reliable sensitivity information [10]. Furthermore, because pattern search methods can be viewed as sampling methods, they suffer from the so-called "curse of dimensionality," though they have been used successfully on problems with up to twenty parameters [11].

The purpose of this paper is to define pattern search methods and their many features while giving some guidelines as to when their use is most appropriate.

2. Pattern Search Methods

We have yet to define pattern search methods. We begin by presenting some informal examples and then give the formal abstraction that both serves as a framework for rigorous analysis and gives guidelines for designing new, possibly application-specific, pattern search methods.

2.1 Some Informal Examples

In the most general sense, a pattern search method samples the function at a predetermined *pattern* of points centered about the current "best" point, using *exploratory moves* that satisfy certain minimal conditions to ensure the robust behavior of the method. If this sampling is successful (i.e., produces a new "best" point), then the process is repeated with the pattern centered about the new best point. If not, the size of the pattern is reduced (typically, by halving) and the function is again sampled about the current "best" point.

To make this more concrete, we will consider the method of coordinate search with fixed step length, perhaps the simplest and most obvious of the pattern search methods (used, for example, forty years ago by Enrico Fermi and Nicholas Metropolis to determine which values of certain theoretical parameters (phase shifts) best fit experimental data (scattering cross sections) [6]). This simple idea conforms with classical notions of good experimental design: vary one factor at a time and observe the effect of that variation on the result of the associated experiment. When no increase or decrease in any one parameter further improves the fit to the experimental data, halve the step size and repeat the process until the steps are deemed sufficiently small. If we consider the *pattern* of points from which the function can be sampled (shown in Figure 1 for problems with only two

variables), we see something that looks much like a two-factor composite design as described by G. E. P. Box and K. B. Wilson [4] on the use of experimental designs to attain optimum conditions.

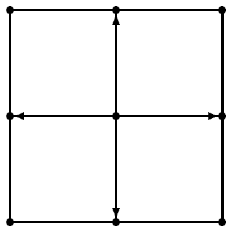

FIGURE 1: Pattern for coordinate search in $\mathbb{R}^2$.

In fact, the patterns associated with pattern search methods share many features with some of the early orthogonal designs suggested for experimental design. Our own work for the multidirectional search algorithm is based on the use of simplex designs first proposed by Spendley, Hext, and Himsworth [18] and subsequently developed by Nelder and Mead [17]. (See Figure 2 for a possible pattern for problems with only two variables.) Curiously, neither the simplex

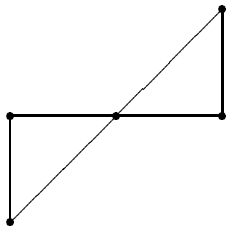

FIGURE 2: Pattern for multidirectional search in $\mathbb{R}^2$.

algorithm of Spendley, Hext and Himsworth nor the simplex method of Nelder and Mead satisfy the strict definition of a pattern search method. With suitable modifications, the analysis for pattern search methods can be extended to the simplex algorithm of Spendley, Hext, and Himsworth. However, similar extensions cannot be made for the simplex algorithm of Nelder and Mead and numerical experiments show that the Nelder-Mead algorithm can fail on trivial problems [19]. Note that one advantage of working with simplex designs is that the number of function evaluations required at each iteration is $O(2n)$, the same as would be required for centered finite-difference approximations to the gradient, as opposed to the $O(2^n)$ function values that would be required for a full two-level factorial design.

This loose definition of pattern search methods captures a variety of other long-standing direct search methods. These include optimization methods based on the response surface methodology work first introduced by G. E. P. Box and K. B. Wilson [4] and subsequently developed by Box and other researchers [2, 3, 5], the original pattern search algorithm of Robert Hooke and T. A. Jeeves [14], and more recently the multidirectional search algorithm and its variants developed by the authors [8, 19, 20].

2.2 A Formal Abstraction

The analysis of the pattern search methods is based on a general unifying abstraction. A simplified version of this abstraction is given here. A complete version, with the accompanying analysis, can be found in [22].

The two key components of a pattern search method are the *generating matrix* and the *exploratory moves* algorithm. The generating matrix designates the set of points that can be sampled at any given iteration; i.e., the generating matrix is used to define the pattern from which the function is sampled. The exploratory moves algorithm specifies how this sampling is to be conducted.

2.2.1 The Generating Matrix

We will denote the generating matrix as $C_k \in \mathbb{R}^{n \times p}$, where $p > 2n$. The generating matrix must have full row rank. We use C_k to generate a pattern of points associated with a given pattern search method. We must be able to partition the columns of the generating matrix into two components: a fixed component F, which is composed of a nonsingular core matrix $\Gamma \in \mathbb{R}^{n \times n}$ and its negative, and any additional columns A_k. Thus:

$$\begin{aligned} C_k &= [\ \underbrace{\Gamma}_{n} \quad \underbrace{-\Gamma}_{n} \quad \underbrace{A_k}_{p-2n}\ ] \\ &= [\ \underbrace{F}_{2n} \quad \underbrace{A_k}_{p-2n}\ ]. \end{aligned} \tag{1}$$

For most pattern search methods, the core matrix

is simply the identity matrix. The columns of A_k may or may not vary across iterations; however, A_k always contains at least one column—the column of all zeros. There are certain restrictions, discussed fully in [22], on the form of the additional columns A_k, but there is a great deal of flexibility in the choice of these columns. For example, the generating matrix for the pattern given in Figure 1 is

$$C_k = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 1 & -1 & -1 & 1 & 0 \end{bmatrix}, \quad (2)$$

while the one associated with the pattern given in Figure 2 is

$$C_k = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \end{bmatrix}.$$

Again we suppress some of the many nuances found in [22] to simplify the presentation. In its most basic form, the pattern can be represented by the generating matrix C_k. We define a *trial step* s_k to be any vector of the form $s_k = \Delta_k c_k$, where $\Delta_k \in I\!R$ is used to control the length of the step (which is equivalent to controlling the size of the pattern) and, introducing a convenient abuse of notation, $c_k \in C_k$, i.e., the vector c_k must be a a column of the generating matrix C_k.

Now, given a current "best" point x_k, a trial point can be any point of the form

$$x'_k = x_k + s_k.$$

The question then becomes, how do we choose from a potentially large number of trial points? That is the role of the exploratory moves algorithm.

2.2.2 The Exploratory Moves

The goal of the exploratory moves algorithm is to sample the function about the current iterate x_k in a well-defined deterministic fashion with the goal of producing a new, better iterate. For the unconstrained minimization problem, a better iterate is simply one with a lower function value. Thus, given a current iterate x_k and its function value $f(x_k)$, we would like to find a step s_k such that

$$f(x_k + s_k) < f(x_k).$$

For example, the exploratory moves for coordinate search can be specified as in Algorithm 1 (where e_i denotes a column of the identity matrix, i.e., a unit coordinate vector).

ALGORITHM 1. Exploratory Moves Algorithm for Coordinate Search.
Given x_k, Δ_k, and $f(x_k)$, set $s_k = \mathbf{0}$ and $min = f(x_k)$.
For $i = 1, \cdots, n$ do

a) $s_k^i = s_k + \Delta_k e_i$ and $x_k^i = x_k + s_k^i$. Compute $f(x_k^i)$.

b) If $f(x_k^i) < min$ then $min = f(x_k^i)$, and $s_k = s_k^i$. Otherwise,

   i) $s_k^i = s_k - \Delta_k e_i$ and $x_k^i = x_k + s_k^i$. Compute $f(x_k^i)$.

   ii) If $f(x_k^i) < min$ then $min = f(x_k^i)$, and $s_k = s_k^i$.

Return.

Note that we are polling each of the coordinate directions in turn and building a final step s_k based on the cumulative result after we have considered each of the n coordinate directions. For $n = 2$, all of the 3^n possible outcomes are contained in the pattern shown in Figure 1 and defined by the columns of the generating matrix given in (2). However, the exploratory moves algorithm samples—*at most*—$2n$ of these 3^n columns at any single iteration. The choice of which columns of C_k to use is made during the course of the iteration and cannot be predicted in advance, even if C_k captures all possible outcomes. Furthermore, this is only one of many ways in which the coordinate search could proceed. One possible modification we could make to this process would be to build in a memory function. For instance, we could keep track of which orientation of each of the coordinate directions was successful at the previous iteration and try this orientation first at the next step (rather than always trying the positive coordinate direction first) in an effort to save function evaluations—a most reasonable consideration. The pattern we have defined for our example with $n = 2$ would still capture all possible outcomes, but the exploratory moves algorithm would be modified as a consequence, as would the likely sequence of iterates. This should begin to give

some indication of the flexibility one has in creating specialized pattern search methods since other modifications that take advantage of knowledge about the problem to be solved can also be incorporated into the exploratory moves algorithm.

To ensure that we can say something substantive about the quality of the solution produced by the choice of columns from the generating matrix, we must place the following two conditions on the conduct of the exploratory moves. First, the direction of any step s_k used to construct a new iterate x_{k+1} must be defined by the pattern and its length must be determined by Δ_k; it is critical that the structure of the pattern be preserved. Second, if simple decrease on the function value at the current iterate can be found among *any* of the $2n$ trial steps defined by $\Delta_k F$ (the fixed portion of the generating matrix), then the exploratory moves must produce a step s_k that also gives simple decrease on the function value at the current iterate. In particular, $f(x_k + s_k)$ need not be the minimum of $f(x_k + \sigma)$ for all $\sigma \in \Delta_k F$; s_k only need satisfy $f(x_k + s_k) < f(x_k)$.

The point of this second restriction is more subtle. We do not wish to force the exploratory moves algorithm to examine all $2n$ steps defined by $\Delta_k F$ at every iteration. For instance, it is possible to construct examples for which an iteration of coordinate search would be successful with at most n function values, as shown in Figure 3. However, it is also possible that for the same example $f(x_k^2) < f(x_{k+1})$ (where x_k^2 is as shown in Figure 4), but x_k^2, while part of the core pattern for coordinate search, was not considered during the course of the exploratory moves.

On the other hand, we do not want to give up the

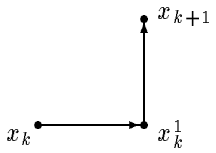

FIGURE 3: A parsimonious iteration of coordinate search in $I\!R^2$.

search prematurely just because improvement is not found immediately. Thus in the "worst case" for coordinate search, we must consider all $2n$ directions

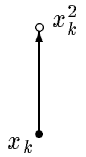

FIGURE 4: An alternate possibility for coordinate search in $I\!R^2$.

defined by the core matrix and its negative, as shown in Figure 5.

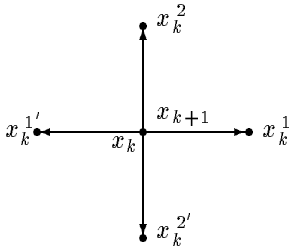

FIGURE 5: A "worst case" iteration of coordinate search in $I\!R^2$.

The question remains, what to do if the iteration is unsuccessful? What if the exploratory moves algorithm considers, at a minimum, the $2n$ steps defined by $\Delta_k F$ without finding a step s_k such that $f(x_k + s_k) < f(x_k)$? In this case, we require that the size of the step be reduced and we continue the search at a new iteration, with a smaller step size.

There is some flexibility in the way that the step length is updated from iteration to iteration, but again certain restrictions must be satisfied (see [22]). In fact, the multidirectional search algorithm of Dennis and Torczon also allows the step size to be increased if an iteration has been successful. But, in general, a very simple update algorithm (Algorithm 2) is used. Here the critical point is that the size of the step is reduced only if the exploratory moves algorithm could not produce a better iterate. If the exploratory moves algorithm was successful, then the

ALGORITHM 2. Updating Δ_k.

a) If $f(x_{k+1}) = f(x_k)$ then $\Delta_{k+1} = \frac{1}{2}\Delta_k$.

b) If $f(x_{k+1}) < f(x_k)$ then $\Delta_{k+1} = \Delta_k$.

size of the step remains the same (or may actually be increased).

3. The Convergence Theory

We have tried to demonstrate, albeit with a very simple example, that there is a great deal of flexibility for deriving pattern search methods within the restrictions imposed by the definition for pattern search methods found in [22]. However, it is important to stress that these restrictions are not arbitrary. The goal is to try to define methods that are robust.

Within the numerical optimization community, "robustness" is usually considered analogous to first-order stationary point convergence. In other words, the goal is not simply to find methods that produce a sequence of iterates that can be shown to have an accumulation point, but to say something stronger about the accumulation points of the sequence. In particular, the goal is to demonstrate that, at the very least, the algorithm produces a sequence of iterates satisfying both that $f(x_{k+1}) \leq f(x_k)$ and that

$$\lim_{k \to +\infty} \|\nabla f(x_k)\| \quad = \quad 0,$$

given some reasonable hypotheses on the function (for instance, that f is bounded below). Such an algorithm is considered to be a "globally convergent algorithm," by which is meant that the method is designed to converge to a *local* minimizer of the function *from almost any starting point.* The terminology is somewhat unfortunate in that convergence to a *global* minimizer of the function is *not* implied. However, "locally convergent" is reserved by tradition for another use [7].

The so-called global convergence theory for classical unconstrained minimization methods based on Taylor's series approximations, such as the method of steepest-descent or Newton's method, have been well-studied and are well-understood. The challenge when analyzing pattern search methods using a similar approach is that we have no explicit approximation of the gradient with which to work.

As it turns out, one of the surprising results [10] is that pattern search methods can be viewed as gradient-related methods. Thus, some of the analysis techniques made possible by the use of Taylor's series approximations can by applied. If we assume that the function to be optimized is continuously differentiable, even though no explicit approximation to the directional derivative is constructed, it can be shown [22] that the sequence of iterates satisfies both that $f(x_{k+1}) \leq f(x_k)$ and that

$$\liminf_{k \to +\infty} \|\nabla f(x_k)\| \quad = \quad 0.$$

In other words, we can show *weak* first-order stationary point convergence for these methods. This weaker result is not surprising given that we are working with no explicit higher-order information about the function. Practically, this result means that the norm of the gradient goes to zero. While this result is perhaps interesting—and surprising—within the framework of optimization theory, it is also important—if less surprising—to practitioners. In particular, it helps solidify the claim that pattern search methods are consistently robust in practice.

Much about the structure imposed upon the pattern search methods contributes to this analysis. The goal here is not to review the details of the analysis but to point out a few salient features.

3.1 Descent Methods

One of the first things to observe is the key role played by the core matrix Γ and its negative. If $\|\nabla f(x_k)\| \neq 0$, then we are guaranteed that at least one of the directions defined by the fixed columns $F = [\Gamma \ -\Gamma]$ of the generating matrix $C_k = [F \ A_k]$ is a *descent direction.* By this we mean that for at least one $s_k \in \Delta_k F$, $\nabla f(x_k)^T s_k > 0$. This is why in the "worst case" scenario demonstrated in Figure 5 we may be required to poll all $2n$ directions; if we have no luck finding decrease by looking at some subset of directions, we force consideration of all the directions defined by $\Delta_k F$ since we know that at least one of these is necessarily a descent direction.

The interaction between the second restriction on the exploratory moves algorithm and the algorithm for updating the step length Δ_k also plays a critical role: by halving the length of the step Δ_k when no decrease is found among any of the directions defined by $\Delta_k F$, we introduce a so-called backtracking mechanism. This guarantees that in a finite number

of iterations we will necessarily find a Δ_k for which at least one of the steps $s_k \in \Delta_k F$ is guaranteed to satisfy the simple decrease condition $f(x_k + s_k) < f(x_k)$.

3.2 Gradient-Related Methods

The second critical role played by the core matrix Γ and its negative, and the reason that they remain *fixed columns* of the generating matrix across all iterations of the pattern search algorithm, is that it allows us to say something stronger about the quality of the descent direction we are guaranteed exists. In particular, it gives us a nonzero uniform lower bound on the cosine of the angle between the direction of descent contained in F and the (negative) gradient at x_k whenever $\nabla f(x_k) \neq 0$. We suppress many of the details in this discussion (see [22]), but in effect it can be shown that if θ is the angle between the gradient and a guaranteed direction of descent, then

$$|\cos\theta| \quad \geq \quad \frac{1}{\kappa(\Gamma)\sqrt{n}},$$

where $\kappa(\Gamma)$ is the *condition number* of the core matrix Γ.

For the simple examples we have shown, the core matrix Γ is the identity matrix, so that $\kappa(\Gamma) = 1$. Thus, for the coordinate search algorithm we have presented, it can be shown [22] that

$$|\cos\theta| \quad \geq \quad \frac{1}{\sqrt{n}} \ .$$

When $n = 2$, even in the worst case (shown in Figure 5), at least one of the search directions defined by $\Delta_k F$ must be within 45° of the negative gradient, a claim easily verified by inspection for this example.

Note that this points to two features that explain much about the behavior of pattern search methods. First, we should be careful in our choice of core matrices; as the condition number of the core matrix increases, our bound on the angle between the search direction and the gradient deteriorates, which can lead to very poor progress during the course of the exploratory moves. Second, this bound also deteriorates as the dimension n of the problem increases. This certainly would explain, at least in part, the long-standing observation that the effectiveness of pattern search methods tends to decrease as the dimension of the problem increases [1].

We close by noting that this lower bound on the cosine of the angle between our guaranteed direction of descent contained in F and the gradient at x_k whenever $\nabla f(x_k) \neq 0$ is what allows us to claim that pattern search methods are gradient-related, even though we have no explicit representation of the gradient. This means that we can use at least some of the analytic tools that have been developed for the classical optimization methods based on Taylor's series expansions to help develop a global convergence theory for pattern search methods.

3.3 Adaptive Grid Search Methods

The third key observation concerning pattern search methods, and the point at which the analysis takes a very different turn from that developed for classical optimization methods, is that pattern search methods could be considered to be *adaptive* grid search algorithms. The restrictions, to which we have alluded without detailed comments, on the form of the generating matrix C_k, the restrictions on the exploratory moves algorithm, and the algorithm for updating the step length Δ_k are all designed to preserve the critical algebraic structure underlying the iterates.

In essence, the pattern used to drive the search conforms to an underlying grid structure. The goal is to sample the function at only a very small subset of the points on this grid, and to know when it is necessary to refine this grid for the search to make further progress.

We return to our simple example, coordinate search for $n = 2$. The pattern shown in Figure 1 could be seen as being a small part of a much larger grid, as shown in Figure 6.

Now let us consider what happens in a single parsimonious iteration (shown in Figure 3) as it affects the number of points in the grid at which we must evaluate the function, as shown in Figure 7.

Assume that at the next iteration, we encounter the so-called "worst case" (shown in Figure 5). The points that are sampled are shown in Figure 8. Notice that although we are now on a subsequent iteration, iteration $k + 1$, we are sampling points *on the same grid*.

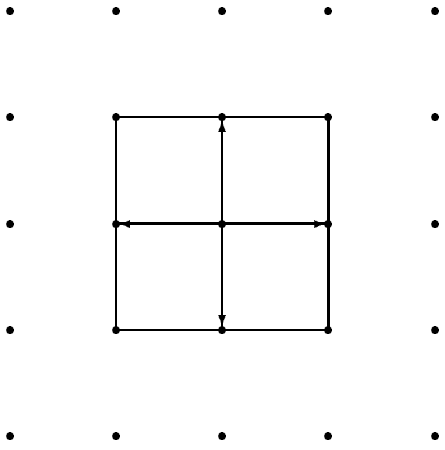

FIGURE 6: The pattern for coordinate search, in $I\!R^2$, as part of a larger grid.

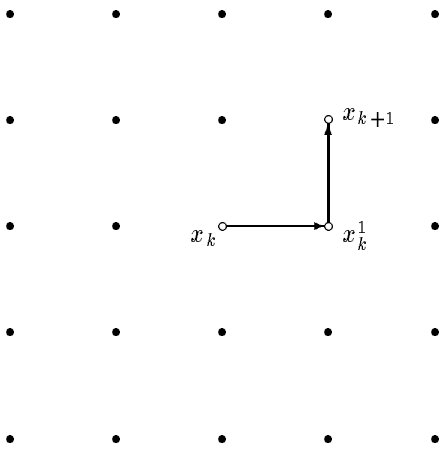

FIGURE 7: Iteration k.

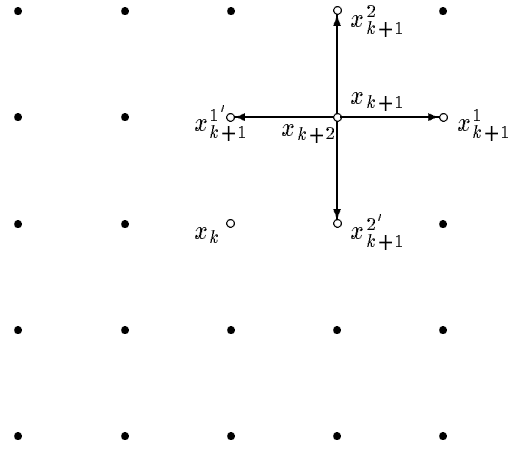

FIGURE 8: Iteration $k+1$.

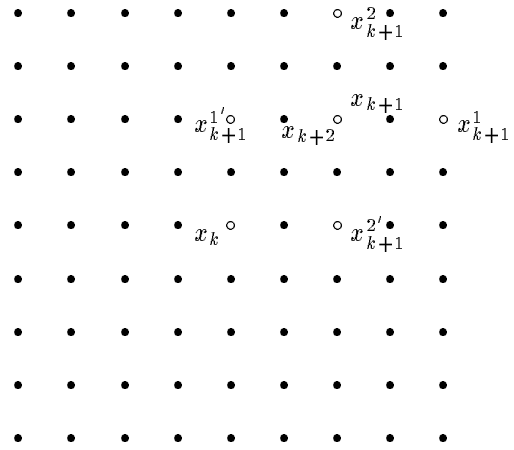

FIGURE 9: The grid for iteration $k+2$.

Now the algorithm for updating the step length comes in to play, by reducing the size of the mesh with the goal of making further progress in the search. However, the rules governing the generating matrix, as well as the reduction factor for the step size, ensure that the refined grid is consistent with the grid used in earlier iterations, as seen in Figure 9. And, again, the goal is to sample only a small subset of the total number of points on the new, refined, grid.

Thus, we consider pattern search methods to be a form of adaptive grid search methods, where we mean "adaptive" in two senses. First, that the exploratory moves algorithm associated with any particular pattern search method decides, adaptively, which subset of points on the current grid needs to be sampled at any given iteration. Second, "adaptive" in the sense that the pattern search methods have a mechanism for deciding when the mesh size of the grid needs to be reduced in order for the search to make further progress.

4. Using Pattern Search Methods

We believe that the recent development in the analysis of pattern search methods makes it possible to substantiate suggestions on when and how to use pattern search methods. Of course, in our experience most people will use them whenever possible. While

our examples centered around coordinate search, because it is the simplest and probably best-known of the pattern search methods, our comments on the effectiveness of pattern search methods are based on our extensive use of the more effective multidirectional search algorithm and its parallel variants [8, 20, 21, 22].

We begin by noting that there is no question but that higher-order methods, when they are applicable, are generally more efficient in their use of function values than pattern search methods. However, when the derivative is unavailable, difficult or expensive to obtain, or is likely to be unreliable, higher-order methods, which typically construct a single search direction based on the gradient, can get into difficulties. The advantage of pattern search methods is that while they are gradient-related, they do not rely explicitly on the gradient *and* they consider multiple directions. This may not be as efficient, but it can be much more reliable in such settings [9].

Pattern search methods can also be used for nondifferentiable functions [13]. The global convergence theory can be extended to handle problems of this sort, but the resulting conclusion is further weakened. (One can show that any accumulation point must be either a point where the norm of the gradient is zero, or a point where the function is nondifferentiable, or, for technical reasons, a point where the gradient exists but is not continuous.)

Another feature of pattern search methods is that because they search in multiple directions, some of which may be up-hill directions, they can be useful in situations where there are a great many local minimizers [16]. While we can give no theoretical justification for this claim, this is a useful feature of pattern search methods that distinguishes them from methods based on higher-order information.

Pattern search methods also make useful exploratory tools, even when it is possible to obtain higher-order information. Because these methods require so little information about the function, they are easy to program and thus can be applied to problems quickly to obtain preliminary results. If the answer they produce is satisfactory, there may be no need to use a higher-order method. If a better (perhaps more accurate) solution is required, the solution obtained by using a pattern search method can be used to provide a good initial guess (a "hot start") to a higher-order method.

Our more recent efforts have focussed on developing effective extensions to the multidirectional search algorithm and its parallel variants [15] and monitoring the applications of these extensions to optimization problems based on engineering simulations [11, 12]. We are also investigating the use of the theory to guide the development of additional robust variants to handle constraints.

Finally, we are intrigued by the possibility of using the grid structure which underlies the pattern search methods to develop variants of the multidirectional search/parallel direct search algorithms that can handle nonlinear optimization problems with a mix of continuous and discrete variables. While we have only begun a preliminary investigation, we believe that such extensions would further enhance the usefulness of pattern search methods in the context of solving MDO problems.

Acknowledgments

Research supported by the State of Texas under contract #1059, the Air Force Office of Scientific Research under F49629-9310212, the Department of Energy under grant DE–FG005–86ER25017, and the National Science Foundation under cooperative agreement CCR-9120008.

References

[1] Mordecai Avriel. *Nonlinear Programming: Analysis and Methods*. Prentice-Hall, Englewood Cliffs, NJ, 1976.

[2] G. E. P. Box. The exploration and exploitation of response surfaces: Some general considerations and examples. *Biometrics*, 10:16–60, March 1954.

[3] G. E. P. Box. Evolutionary operation: A method for increasing industrial productivity. *Appl. Statist.*, 6(2):81–101, June 1957.

[4] G. E. P. Box and K. B. Wilson. On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society, Series B*, XIII(1):1–45, 1951.

[5] M. J. Box, D. Davies, and W. H. Swann. *Non-Linear Optimization Techniques*. ICI Monograph No. 5. Oliver & Boyd, Edinburgh, 1969.

[6] William C. Davidon. A belated preface for ANL 5990. *SIAM J. Optimization*, 1(1):1–17, February 1991. The belated preface was received by the editors June 4, 1990; accepted for publication August 10, 1990. The rest of the article was originally published as "Variable Metric Method for Minimization," Argonne National Laboratory Research and Development Report 5990, May 1959 (revised November 1959).

[7] J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, 1983.

[8] J. E. Dennis, Jr. and Virginia Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4):448–474, November 1991.

[9] Samuel K. Eldersveld. Part nesting for just-in-time manufacturing. *TechNet: Briefs on Computing Technology*, 7(3):7–8, June 1993.

[10] P. D. Frank, A. J. Booker, T. P. Caudell, and M. J. Healy. A comparison of optimization and search methods for multidisciplinary design. In *Proceedings of the Fourth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization*, Cleveland, OH, September 21–23, 1992.

[11] Roland Glowinski and Anthony J. Kearsley. On the simulation and control of some friction constrained motions. Technical Report 93–21, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77251–1892, 1993. To appear in *SIAM Journal on Optimization*.

[12] Roland Glowinski, Anthony J. Kearsley, Tsorn-Whay Pan, and Jacques Peraux. Numerical simulation and optimal shape for viscous flow by a fictitious domain method. *Special Edition: The International Journal of Numerical Methods in Engineering*, 1994. To appear.

[13] Nicholas J. Higham. Optimization by direct search in matrix computation. *SIAM J. Matrix Anal. Appl.*, 14(2):317–333, April 1993.

[14] Robert Hooke and T. A. Jeeves. "Direct search" solution of numerical and statistical problems. *J. Assoc. Comput. Mach.*, 8(2):212–229, April 1961.

[15] A. J. Kearsley, R. A. Tapia, and V. Torczon. On the use of parallel direct search methods for nonlinear programming problems. Technical Report 93-33, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77251-1892, 1993. In revision.

[16] J. C. Meza and M. L. Martinez. Direct search methods for the molecular conformation problem. *Journal of Computational Chemistry*, 15(6):627–632, 1994.

[17] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7(4):308–313, January 1965.

[18] W. Spendley, G. R. Hext, and F. R. Himsworth. Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461, November 1962.

[19] Virginia Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Department of Mathematical Sciences, Rice University, Houston, TX, 1989; also available as Tech. Report 90–7, Department of Computational and Applied Mathematics, Rice University, Houston, TX 77251–1892.

[20] Virginia Torczon. On the convergence of the multidirectional search algorithm. *SIAM Journal on Optimization*, 1(1):123–145, February 1991.

[21] Virginia Torczon. PDS: Direct search methods for unconstrained optimization on either sequential or parallel machines. Technical Report 92–9, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77251–1892, 1992. In revision; to appear in *acm Transactions on Mathematical Software*.

[22] Virginia Torczon. On the convergence of pattern search algorithms. Technical Report 93–10, Department of Computational and Applied Mathematics, Rice University, Houston, Texas 77251–1892, 1993. In revision; to appear in *SIAM Journal on Optimization*.

[23] Garret N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design: With Applications*. McGraw-Hill, Inc., 1984.