Why Pattern Search Works*

Robert Michael Lewis† Virginia Torczon‡

Michael W. Trosset§

October 21, 1998

1 Introduction

Pattern search methods are a class of direct search methods for nonlinear optimization. Since the introduction of the original pattern search methods in the late 1950s and early 1960s [2, 5], they have remained popular with users due to their simplicity and the fact that they work well in practice on a variety of problems. More recently, the fact that they are provably convergent has generated renewed interest in the nonlinear programming community.

The purpose of this article is to describe what pattern search methods are and why they work. Much of our past work on pattern search methods was guided by a desire to unify a variety of existing algorithms and provide them with a common convergence theory. Unfortunately, the unification of this broad class of algorithms requires a technical framework that obscures the features that distinguish pattern search algorithms and make them work. We hope here to give a clearer explanation of these ideas. Space does not allow us to do justice to the history of these methods and all the work relating to them; this will be the subject of a lengthier review elsewhere; for a historical perspective, see [17].

2 A Simple Example of Pattern Search

We begin our discussion with a simple instance of a pattern search algorithm for unconstrained minimization: minimize $f(x)$. At iteration k, we have an iterate

*This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while the authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-2199.

†Institute for Computer Applications in Science and Engineering (ICASE), Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681–2199, U.S.A., buckaroo@icase.edu.

‡Department of Computer Science, College of William & Mary, P.O. Box 8795, Williamsburg, Virginia 23187–8795, U.S.A., va@cs.wm.edu.

§Department of Mathematics, College of William & Mary, P.O. Box 8795, Williamsburg, Virginia 23187–8795, U.S.A., trosset@math.wm.edu.

$x_k \in I\!R^n$ and a step-length parameter $\Delta_k > 0$. Let e_i, $i = 1, \ldots, n$, denote the standard unit basis vectors. We successively look at the points $x_+ = x_k \pm \Delta_k e_i$, $i = 1, \ldots, n$, until we find x_+ for which $f(x_+) < f(x_k)$. Fig. 1 illustrates the set of points among which we search for x_+ for $n = 2$. This set of points is an instance of what we call a pattern, from which pattern search takes its name. If we find no x_+ such that $f(x_+) < f(x_k)$, then we reduce Δ_k by a half and continue; otherwise, we leave the step-length parameter alone, setting $\Delta_{k+1} = \Delta_k$ and $x_{k+1} = x_+$. In the latter case we can also increase the step-length parameter, say, by a factor of 2, if we feel a longer step might be justified. We repeat the iteration just described until Δ_k is deemed sufficiently small.

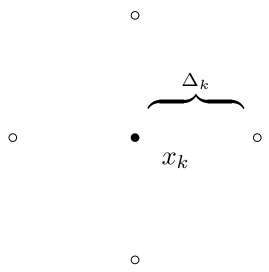

Figure 1: A simple instance of pattern search

This simple example illustrates two attractive features of pattern search algorithms:

- They can be extremely simple to specify and implement.
- No explicit estimate of the derivative nor anything like a Taylor's series appears in the algorithm. This makes these algorithms useful in situations where derivatives are not available and finite-difference derivatives are unreliable, such as when $f(x)$ is noisy.

These qualities have made pattern search algorithms popular with users. Yet despite their seeming simplicity and heuristic nature and the fact that they do not have explicit recourse to the derivatives of $f(x)$, pattern search algorithms possess global convergence properties that are almost as strong as those of comparable line-search and trust-region algorithms. In this article we will attempt to explain this perhaps surprising fact.

Before turning to the discussion of how this can be, we note some further features of pattern search which are manifest in this simple example.

- We require only simple decrease in $f(x)$. In fact, we do not even need to know $f(x)$ as a numerical value, provided we can make the assessment that $f(x_+)$ is an improvement on $f(x_k)$.
- If we are lucky, we need only a single evaluation of $f(x)$ in any given iteration. Once we find an x_+ for which $f(x_+) < f(x_k)$, we can accept it

and proceed. On the other hand, in the worst case we will look in quite a few directions ($2n$, for this example) before we try shorter steps.

- The steps that are allowed are restricted in direction and length. In this example, the steps must lie parallel to the coordinate axes and the length of any step has the form $\Delta_0/2^N$ for some integer N.

This simple example also suggests that there is a great deal of flexibility in pattern search algorithms, depending on how one specifies the pattern of points to be searched for the next iterate. These features will be recurring themes in our discussion.

3 The General Pattern Search Algorithm

For simplicity, our discussion will focus primarily on the case of unconstrained minimization,

$$\text{minimize} \quad f(x).$$

We assume that f is continuously differentiable, but that information about the gradient of f is either unavailable or unreliable. Since the inception of pattern search methods, various techniques have also been used to apply them to solve the general nonlinear programming problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & c(x) \geq 0 \\ & \ell \leq x \leq u. \end{array}$$

More recently, pattern search methods specifically designed for constrained problems with an attendant convergence theory have been developed in [6, 9, 8].

The form of a general pattern search algorithm is quite simple and not all that different from any other nonlinear minimization algorithm: first, find a step s_k from the current iterate x_k; second, determine if that step is acceptable; and finally, update the critical components of the algorithm. At iteration k pattern search methods will consider steps in directions denoted by d_k. We require d_k to be a column of D_k, where D_k is an $n \times p_k$ matrix (i.e., D_k represents the set of directions under consideration).

Generalized pattern search:
Given $x_0 \in I\!R^n$, $f(x_0)$, $D_0 \in I\!R^{n \times p_0}$, and $\Delta_0 > 0$,
for $k = 0, 1, \ldots$ until done do {

1. Find a step $s_k = \Delta_k d_k$ using the procedure `Exploratory_Moves`(Δ_k, D_k).
2. If $f(x_k + \Delta_k d_k) < f(x_k)$, then $x_{k+1} = x_k + \Delta_k d_k$; otherwise, $x_{k+1} = x_k$.
3. `Update`(Δ_k, D_k)

}

In order to establish convergence results for this class of algorithms, we will, by and by, place additional conditions on D_k, the step calculation procedure `Exploratory_Moves()`, and the update procedure `Update()`. The analysis reveals that we do not need to explicitly define `Exploratory_Moves()` or `Update()`; for the purposes of ensuring convergence it suffices to specify conditions on the results they produce. We refer the interested reader to [16] for specific examples of `Exploratory_Moves()` and `Update()` used for some of the more traditional pattern search methods.

4 Global Convergence Analysis

Here we will use global convergence of an optimization algorithm to mean convergence to a stationary point of at least one subsequence of the sequence of iterates produced by the algorithm. A slightly weaker assertion is

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

this is equivalent to the previous property if the iterates $\{x_k\}$ remain in a bounded set.

Classical analyses of such methods as steepest descent and globalized Newton methods rely in a fundamental way on $\nabla f(x)$ to prove global convergence. Moreover, the technical conditions that make the proof of global convergence for these algorithms possible, such as the Armijo-Goldstein-Wolfe conditions for line-search methods, are actually built into the specification of gradient-based algorithms.

On the other hand, no such technical conditions appear in the description of pattern search algorithms (witness the example in §2). The philosophy of pattern search algorithms (and direct search methods in general) is best described by Hooke and Jeeves [5]:

> We use the phrase "direct search" to describe sequential examination of trial solutions involving comparison of each trial solution with the "best" obtained up to that time together with a strategy for determining (as a function of earlier results) what the next trial solution will be. The phrase implies our preference, based on experience, for straightforward search strategies which employ no techniques of classical analysis except where there is a demonstrable advantage in doing so.[1]

[1]It might strike the modern reader as odd that Hooke and Jeeves would question the advantages of employing techniques of "classical analysis"—meaning calculus—given the success of quasi-Newton algorithms. However, direct search methods appeared in the late 1950s and early 1960s, a time at which derivative-based methods were not as efficient as today, and no general convergence analysis existed for any practical optimization algorithm, derivative-based or not. The Armijo-Goldstein-Wolfe conditions [1, 4, 19], which form the basis for designing and analyzing what we now consider to be practical line-search algorithms, were several years in the future; trust region algorithms [14] were further still.

This passage captures the basic philosophy of the original work on direct search algorithms: an avoidance of the explicit use or approximation of derivatives. Instead, the developers of the original direct search algorithms relied on heuristics to obtain what they considered promising search directions.

Nonetheless, we can prove global convergence results for pattern search methods, even though this class of algorithms was not originally developed with convergence analysis in mind. The analysis does ultimately rely on $\nabla f(x)$; hence the assumption that f is continuously differentiable. But because pattern search methods do not compute or approximate $\nabla f(x)$, the relationship between these algorithms and their convergence analysis is less direct than that for gradient-based algorithms.

4.1 The Ingredients of Global Convergence Analysis

We will now review the ideas that underlie the global convergence analysis of line-search methods for unconstrained minimization in order to compare them with those for pattern search. We focus on line-search methods rather than trust-region methods since the comparisons and contrasts with pattern search are simpler for line-search methods.

In order to prove global convergence of a line-search algorithm, at the very least one must show that if the current iterate x_k is not a stationary point, then the algorithm will eventually find an iterate x_{k+1} such that $f(x_{k+1}) < f(x_k)$. This unavoidably leads to the contemplation of the gradient, since the gradient ensures that a direction of descent can be identified: if x_k is not a stationary point of f, then any direction within 90° of $-\nabla f(x_k)$ is a descent direction. For our purposes, this will prove a crucial, if elementary, observation: one does not need to know *the* negative gradient in order to improve $f(x)$, one only needs a direction of descent. Then, if one takes a short enough step in that direction, one is guaranteed to find a point x_{k+1} such that $f(x_{k+1}) < f(x_k)$.

However, descent is not sufficient to ensure convergence: one must also rule out the possibility that the algorithm can simply grind to a halt, converging to a point that is not a stationary point. One begins by requiring at least one search direction to be uniformly bounded away from orthogonality with $-\nabla f(x_k)$. This ensures that the sequence of iterates cannot degenerate into steps along directions that become ever more orthogonal to the gradient while producing an ever diminishing improvement in $f(x)$.

This restriction on the search directions is still not sufficient to prevent the iterates from converging to points that are not stationary points. This unhappy situation can occur in two ways. First, there is the pathology depicted in Fig. 2. The ellipse represents a level set of $f(x)$, which in this case is a convex quadratic. The steps taken are too long relative to the amount of decrease between successive iterates. While the sequence of iterates $\{x_k\}$ produces a strictly decreasing sequence of objective values $\{f(x_k)\}$, the sequence of iterates converges to two nonstationary points.

The other pathology, depicted in Fig. 3, occurs when the amount of decrease between successive iterates is too small relative to the amount of decrease ini-

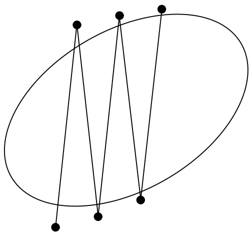

Figure 2: Decrease is too small relative to the length of the step

tially seen in the direction from one iterate to the next. This time the steps between successive iterates become excessively short. This sequence converges

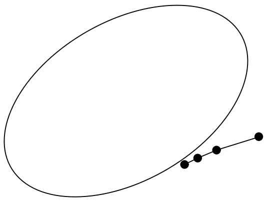

Figure 3: Decrease is too small relative to the norm of the gradient

to a single point which again is not a stationary point.

These pathologies lead to the second standard element of global convergence analysis: a mechanism that controls the length of the step. Both of the preceding pathologies can be avoided, for instance, by requiring that the amount of decrease in $f(x)$ between successive iterates be "sufficient," where sufficient relates the amount of decrease, the length of the step, and the gradient $\nabla f(x)$. This is the purpose of the Armijo-Goldstein-Wolfe conditions for line-search algorithms: given a suitable descent direction d_k, we choose a step length $\Delta_k > 0$ such that for some fixed $\alpha \in (0, 1)$ and fixed $\beta \in (\alpha, 1)$, $x_{k+1} = x_k + \Delta_k d_k$ satisfies both

$$f(x_{k+1}) \quad \leq \quad f(x_k) + \alpha \Delta_k \nabla f(x_k)^T d_k \tag{1}$$

and

$$\nabla f(x_{k+1})^T d_k \quad \geq \quad \beta \nabla f(x_k)^T d_k. \tag{2}$$

The first condition precludes steps that are too long; the second condition precludes steps that are too short.

5 How Pattern Search Does Its Thing

We can summarize the devices that ensure the global convergence of line-search methods for unconstrained minimization as follows:

1. The choice of a suitably good descent direction.

2. Step-length control:

   (a) a mechanism to avoid steps that are too long, and

   (b) a mechanism to avoid steps that are too short, where long and short refer to the sufficient decrease conditions (1) and (2), respectively.

These mechanisms, which are explicitly built into line-search algorithms, all depend on information about the gradient. However, pattern search algorithms do not assume such information, and thus do not and cannot enforce such conditions. What, then, ensures the global convergence of pattern search algorithms?

The answer resembles the classical arguments for establishing the global convergence of line-search methods, but necessarily with novel elements. As we shall see, pattern search algorithms are globally convergent because:

1. At each iteration, they look in enough directions to ensure that a suitably good descent direction will ultimately be considered.

2. They possess a reasonable back-tracking strategy that avoids unnecessarily short steps.

3. They otherwise avoid unsuitable steps by restricting the nature of the step allowed between successive iterates, rather than by placing requirements on the amount of decrease realized between successive iterates.

At the heart of the argument lies an unusual twist: we relax the requirement of sufficient decrease and require only simple decrease ($f(x_{k+1}) < f(x_k)$), but we impose stronger conditions on the form the step s_k may take. Furthermore, this trade-off is more than just a theoretical innovation: in practice, it permits useful search strategies that are precluded by the condition of sufficient decrease.

5.1 Pattern Search as a Crypto-Gradient Method

The analysis begins by demonstrating that a search direction not too far from the negative gradient is always available. This is accomplished by considering a set of step directions D_k sufficiently rich that it necessarily includes at least one acceptable descent direction. In the absence of any estimate of $-\nabla f(x_k)$, pattern search algorithms hedge against the fact that $-\nabla f(x_k)$ could point in any direction.

For the example in §2 the set of directions D_k is $\{\pm e_i,\ i = 1, \ldots, n\}$, so the set of prospective next iterates has the simple form $\{x_k \pm \Delta_k e_i,\ i = 1, \ldots, n\}$. If a step $s_k = \pm\Delta_k e_i$ producing simple decrease on $f(x_k)$ is found, then $x_{k+1} = x_k \pm \Delta_k e_i$; otherwise, reduce Δ_k and try again. Other of the original pattern

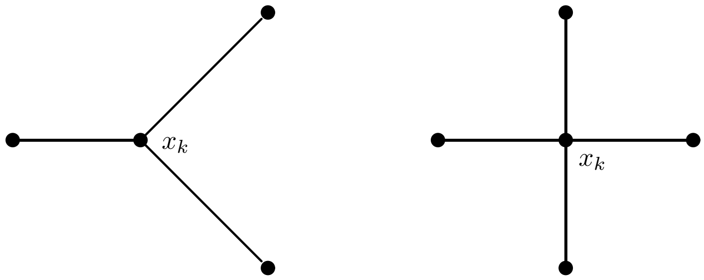

Figure 4: Examples of a minimal and a maximal positive basis for $\mathbf{R}^2$

search methods, such as the method of Hooke and Jeeves [5] or coordinate search [13], also include in D_k the directions $\{\pm e_i,\ i = 1, \ldots, n\}$.

The analysis in [16] allows for more general conditions on the set of directions. In particular, D_k must contain a set of the form $\{\pm p_i,\ i = 1, \ldots, n\}$, where $p_1, \ldots, p_n$ is any linearly independent set of vectors. One can allow this set to vary with k, so long as one restricts attention to a finite collection of such sets.

The discussion in [18] brought to our attention that even less is required: it suffices that the set of directions D_k contain a *positive basis* Π_k for $\mathbb{R}^n$ [7]. In terms of the theory of positive linear dependence [3], the positive span of a set of vectors $\{a_1, \ldots, a_r\}$ is the cone

$$\{a \in \mathbb{R}^n \mid a = c_1 a_1 + \ldots + c_r a_r,\ c_i \geq 0 \text{ for all } i\}.$$

The set $\{a_1, \ldots, a_r\}$ is called positively dependent if one of the a_i's is a nonnegative combination of the others; otherwise the set is positively independent. A positive basis is a positively independent set whose positive span is $\mathbb{R}^n$, i.e., a set of generators for a cone that happens to be a vector space. A positive basis must contain at least $n + 1$ vectors and can contain no more than $2n$ vectors [3]; we refer to the former as minimal and the latter as maximal; Fig. 4 demonstrates examples of both for $\mathbf{R}^2$.

How do we know that at least one of the directions in D_k is not too orthogonal to the direction of steepest descent, regardless of what $-\nabla f(x)$ might be? A proof by picture is given in Fig. 5; see [7] for details. Consider the minimal positive basis $\{(1,1)^T, (1,-1)^T, (-1,0)^T\}$ depicted Fig. 5 as directions emanating from x_k. Notice that the angles between these vectors are 90°, 135°, and 135°. For *any* continuously differentiable function $f : \mathbb{R}^2 \to \mathbb{R}$, if x_k is not a stationary point, then $-\nabla f(x_k)$ can be no more than 67.5° from one of the vectors in the positive basis, as shown in Fig. 5. Thus, including a positive basis Π_k in the set of directions D_k guarantees that we can approximate the negative gradient in a way that cannot be arbitrarily bad. This is the first step towards establishing global convergence.

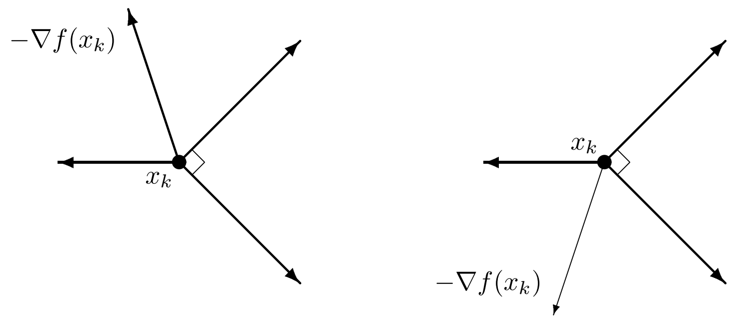

Figure 5: A minimal positive basis for $\mathbf{R}^2$ and the two worst cases for $-\nabla f(x_k)$

5.2 The Underlying Lattice Structure

As it happens—as it was meant to happen—pattern search methods are restricted in the nature of the steps they take. This ultimately turns out to be the reason pattern search methods can avoid the pathologies illustrated in Fig. 2 and Fig. 3 without enforcing a sufficient decrease condition.

Let P_k denote the set of candidates for the next iterate (i.e., $P_k = x_k + \Delta_k D_k$, by abuse of notation). We call P_k the *pattern*, from which pattern search takes its name. Several traditional patterns are depicted in Fig. 6.

Though it does not appear to have been by any conscious design on the part of the original developers of pattern search algorithms, these algorithms produce iterates that lie on a finitely generated rational lattice, as indicated in Fig. 6. More precisely, there exists a set of generators $g_1, \ldots, g_m$, independent of k, such that any iterate x_k can be written as

$$x_k = x_0 + \Delta_0 \sum_{i=1}^{m} c_i^k g_i, \tag{3}$$

where each c_i^k is rational.

Note that this structural feature means that the set of possible steps, and thus the set of possible iterates, is known in advance and is independent of the actual objective $f(x)$. This is in obvious contrast to gradient-based methods.

Furthermore—and this is significant to the convergence analysis of pattern search—by judicious (but not especially restrictive) choice of the factors by which Δ_k can be increased or decreased, we can establish the following behavior of these algorithms. Suppose the set $\{ \, x \mid f(x) \le f(x_0) \, \}$ is bounded. Given any $\Delta_* > 0$, there exists a finite subset (that depends on Δ_*) of the lattice of all possible iterates such that x_k must belong to this subset until $\Delta_k < \Delta_*$. That is, there is only a finite number of distinct values x_k can possibly have until such time as $\Delta_k < \Delta_*$. This means that the only way to obtain an infinite sequence of distinct x_k is to reduce the step length parameter Δ_k infinitely often so that $\liminf_{k\to\infty} \Delta_k = 0$.

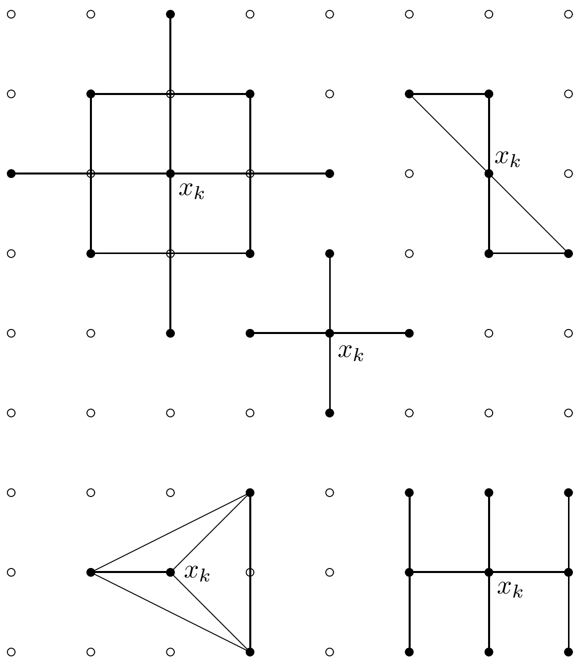

Figure 6: Some possible patterns

This also reveals another role played by the parameter Δ_k. Reducing Δ_k increases the set of candidate iterates by allowing us to search over a finer subset of the rational lattice of all possible iterates. This is shown in Fig. 6 and Fig. 7. In these pictures, halving Δ_k refines the grid over which we are tacitly searching for a minimizer of $f(x)$, while halving the minimum length a step is allowed to have.

5.3 Putting It All Together

We return to the general pattern search algorithm:

Generalized pattern search:
Given $x_0 \in \mathbb{R}^n$, $f(x_0)$, $D_0 \in \mathbb{R}^{n \times p_0}$, and $\Delta_0 > 0$,
for $k = 0, 1, \ldots$ until done do {

1. Find a step $s_k = \Delta_k d_k$ using the procedure `Exploratory_Moves`(Δ_k, D_k).
2. If $f(x_k + \Delta_k d_k) < f(x_k)$, then $x_{k+1} = x_k + \Delta_k d_k$; otherwise, $x_{k+1} = x_k$.
3. `Update`(Δ_k, D_k)

}

The step s_k returned by the `Exploratory_Moves()` algorithm must satisfy two simple conditions:

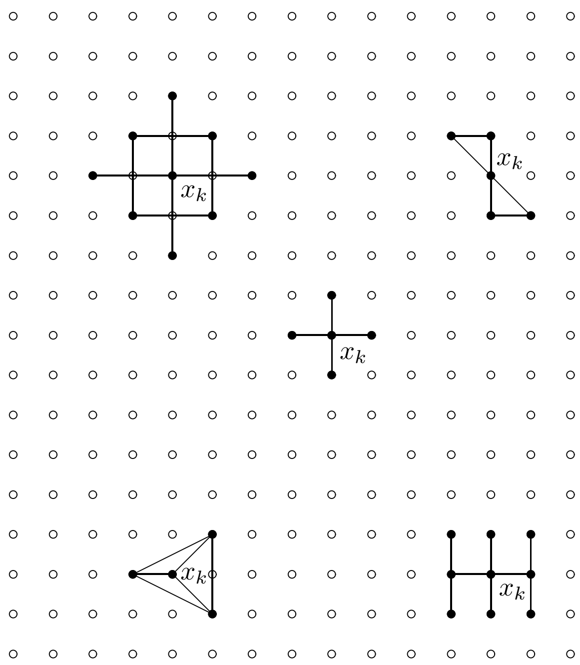

Figure 7: The same patterns on a refinement of the grid

1. The step returned must be an element of $\Delta_k D_k$.

2. The step s_k must satisfy either $f(x_k + s_k) < f(x_k)$ or $s_k = 0$.

   Furthermore, s_k may be 0 only if none of the steps in $\Delta_k \Pi_k$ yielded decrease on $f(x_k)$.

The first condition prevents arbitrary steps along arbitrary directions; the second condition is a back-tracking control mechanism that prevents us from taking shorter steps unless it is truly necessary.

As for the procedure `Update()` for D_k and Δ_k, we are free to make modifications to D_k before the next iteration. Classical pattern search methods typically specify a single $D = D_k$ for all k. Others make substantive changes in response to the outcome of the exploratory moves. This is just one of many options to consider when designing a pattern search method, and it leads to a great deal of flexibility in this class of algorithms. There are conditions that must be satisfied to preserve the lattice structure, but these are straightforward to satisfy in practice. The interested reader is referred to [7], for a complete discussion of the technical conditions, and to [16], for a description of some traditional choices.

The rules for updating Δ_k are also restricted by the need to preserve the algebraic structure of the possible iterates. Historically, the popular choices have been to halve Δ_k at unsuccessful iterations, and to either leave Δ_k alone at successful iterations or possibly double it. The convergence analysis leads to other possibilities: we can rescale Δ_k by $\theta \in \{\tau^{\omega_0}, \ldots, \tau^{\omega_L}\}$, where τ is a rational number, $\{\omega_0, \ldots, \omega_L\}$ are integers, $L \geq 2$, $\omega_0 < 0$ and $\omega_L \geq 0$. This provides at least one option for reducing Δ_k when back-tracking is called for, and at least one option that does not reduce Δ_k.

The proof of convergence now goes like this. Suppose x_k is not a stationary point of $f(x)$. Because at least one of the directions d_k in Π_k is necessarily a descent direction, we can always find an acceptable step $\Delta_k d_k$ once we reduce Δ_k sufficiently. Thus, we can always find x_{k+1} with $f(x_{k+1}) < f(x_k)$ for k in some subsequence K.

Now, if $\liminf_{k\to\infty} \|\nabla f(x_k)\| \neq 0$, then for some $\varepsilon > 0$, $\|\nabla f(x_k)\| > \varepsilon$ for all k. Under this assumption we can show that once Δ_k is sufficiently small relative to ε, it will no longer be reduced. This is so because one of the directions d_k in Π_k is sufficiently close to $-\nabla f(x_k)$ to be a uniformly good descent direction, and $\|\nabla f(x_k)\|$ is uniformly not too small, so we will have $f(x_k + \Delta_k d_k) < f(x_k)$ without having to drive Δ_k to zero.

However, if $\liminf_{k\to\infty} \Delta_k = \Delta* > 0$, then due to the lattice structure of the iterates, there can be only finitely many possible x_k, contradicting the fact that we have an infinite subsequence K with $f(x_{k+1}) < f(x_k)$ for all $k \in K$ (assuming $\{ \ x \ \mid \ f(x) \leq f(x_0) \ \}$ is bounded). Hence $\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0$.

The correlation between the fineness of the grid of possible iterates and the size of Δ_k also explains why long steps are not a problem. We have argued above that if $\liminf_{k\to\infty} \Delta_k = 0$, then $\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0$. Now, unless $\liminf_{k\to\infty} \Delta_k = 0$, there can be only a finite number of distinct iterates, and hence only a finite number of long steps (or any type of step, for that matter). Thus even if an infinite number of "bad" long steps are taken (i.e., steps that decrease $f(x)$ but that violate (1)), the mere fact that there are infinitely many distinct iterates means that $\liminf_{k\to\infty} \Delta_k = 0$, and hence $\liminf_{k\to\infty} \|\nabla f(x_k)\| = 0$.

5.4 Observations

This analysis might suggest an interpretation of pattern search as a search over successively finer finite grids. If the finite set of candidates is exhausted without finding a point that improves $f(x)$, then the grid is refined by reducing Δ_k and the process is repeated.

However, this interpretation is misleading insofar as it suggests that pattern search algorithms are exceedingly inefficient. In practice, pattern search algorithms do not resort to searching over all the points in increasingly fine grids but instead behave more like a steepest descent method. In this sense, the analysis does not reflect the actual behavior of the algorithm. This should not be entirely surprising since, unlike gradient-based methods, the specification of pattern search algorithms does not obviously contain a mechanism designed to guarantee convergence.

The situation is analogous to that of the simplex method in linear programming. Once one establishes that the simplex method cannot cycle, the convergence of the algorithm follows from the fact that there is only a finite number of vertices that the simplex method can visit in its search for a solution. This means that the simplex method could and does have a theoretical worst-case complexity that is exponential, but in practice the simplex method has proven much more efficient than that.

Moreover, the actual behavior of pattern search in any single iteration can be very different than the proof of convergence might be thought to suggest. The search can accept as the next iterate *any* point in P_k that satisfies the simple decrease condition $f(x_{k+1}) < f(x_k)$. In particular, the algorithm does not necessarily need to examine every point in $\Delta_k \Pi_k$; it need only do so before deciding to reduce Δ_k, which is the worst case.

In the best case, we may need only a single evaluation of $f(x)$ to find an acceptable step. In contrast, in a forward-difference gradient-based method one needs at least $n+1$ evaluations of $f(x)$ (in addition to $f(x_k)$) to find a new iterate; n additional values of $f(x)$ to approximate $\nabla f(x_k)$ and at least one more evaluation of $f(x)$ to decide whether or not to accept a new iterate.

In order to make progress, pattern search requires the eventual reduction of Δ_k. The cost of discovering the necessity of this step is one evaluation of $f(x)$ for each direction defined by the positive basis Π_k. For a minimal positive basis of $n+1$ elements, this cost is the same as the cost of an unsuccessful quasi-Newton step using a forward-difference approximation of the gradient; n evaluations of $f(x)$ to form the finite-difference approximation to $\nabla f(x_k)$, and the evaluation of $f(x)$ at the rejected x_+. On the other hand, following an unsuccessful step in the latter algorithm, one gets to reuse the gradient approximation; it is not clear how best to reuse information from unsuccessful iterations of pattern search in subsequent iterations.

5.5 The Resulting Convergence Results

Let $\Omega = \{ \ x \ \mid \ f(x) \le f(x_0) \ \}$, and suppose f is C^1 on a neighborhood of Ω.

Theorem If Ω is bounded, then the iterates produced by a pattern search algorithm satisfy

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

If, in addition, $\lim_{k\to\infty} \Delta_k = 0$ and we require $f(x_{k+1}) < f(x_k + s_k)$ for all $s_k \in \Delta_k \Pi_k$, the steps associated with the positive basis, and the columns of D_k are bounded in norm uniformly in k, then we have

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

By way of comparison, we obtain the result $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$ for line-search methods without the assumption that Ω is bounded [12]. However, we must also require sufficient decrease between iterates according to (1)–(2), rather than just simple decrease, that is, $f(x_{k+1}) < f(x_k)$.

For trust-region methods, with the assumption that $\nabla f(x)$ is uniformly continuous (but again, without the assumption that Ω is bounded), requiring only simple decrease $f(x_{k+1}) < f(x_k)$ suffices to prove that $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$, provided the approximation of the Hessian does not grow too rapidly in norm [15]. With a sufficient decrease condition, one obtains the stronger result [11], $\lim_{k\to\infty} \|\nabla f(x_k)\| = 0$. However, for either result $\nabla f(x)$ is used in both the fraction of Cauchy decrease condition on the step and the update of the trust radius.

Thus, under the hypothesis that Ω is bounded, the global convergence results for pattern search algorithms are as strong as those for gradient-based methods. This might seem surprising, but it simply reflects just how little one needs to establish global convergence. Pattern search is sufficiently like steepest descent that it works.

This leads to one caveat for users: like steepest descent, pattern search methods are good at improving an initial guess and finding a neighborhood of a local solution, but fast local convergence should not be expected. In general, one can expect only a linear rate of local convergence.

6 Concluding Remarks

We have tried to explain how and why pattern search works while refraining from a detailed description of the convergence analysis. Once one understands the essential ideas, the proof of global convergence is reasonably straightforward, if sometimes tedious. Precisely because pattern search methods have so little analytical information explicitly built into them, it takes some effort to extract an assurance that they actually do work. However, as we have tried to indicate, many of the ideas are familiar from standard analysis of nonlinear programming algorithms. The novelty lies in the restriction of the iterates to a lattice, which allows us to relax the conditions on accepting steps.

The ideas discussed here also appear in the analysis of pattern search methods for constrained minimization [6, 9, 8]. For readers who would like to explore the connections between pattern search methods and gradient-based algorithms in greater detail, we particularly recommend [10].

References

[1] L. Armijo, *Minimization of functions having Lipschitz continuous first partial derivatives*, Pacific Journal of Mathematics, 16 (1966), pp. 1–3.

[2] G. E. P. Box, *Evolutionary operation: A method for increasing industrial productivity*, Applied Statistics, 6 (1957), pp. 81–101.

[3] C. Davis, *Theory of positive linear dependence*, American Journal of Mathematics, 76 (1954), pp. 733–746.

[4] A. A. Goldstein, *Constructive Real Analysis*, Harper & Row, New York, NY, 1967.

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

[6] R. M. Lewis and V. J. Torczon, *Pattern search algorithms for bound constrained minimization*, Tech. Rep. 96–20, Institute for Computer Applications in Science and Engineering, Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681–2199, March 1996. To appear in SIAM Journal on Optimization.

[7] ———, *Rank ordering and positive bases in pattern search algorithms*, Tech. Rep. 96–71, Institute for Computer Applications in Science and Engineering, Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681–2199, 1996. In revision for Mathematical Programming.

[8] ———, *A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds*, Tech. Rep. 98–31, Institute for Computer Applications in Science and Engineering, Mail Stop 403, NASA Langley Research Center, Hampton, VA 23681–2199, July 1998. Submitted to SIAM Journal on Optimization.

[9] ———, *Pattern search methods for linearly constrained minimization*, Tech. Rep. 98–3, Institute for Computer Applications in Science and Engineering, Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681–2199, January 1998. To appear in SIAM Journal on Optimization.

[10] S. Lucidi and M. Sciandrone, *On the global convergence of derivative free methods for unconstrained optimization*, Tech. Rep. 18-96, DIS, Universitá di Roma "La Sapienza", 1996. Submitted to SIAM Journal on Optimization.

[11] J. J. Moré, *The Levenberg-Marquardt algorithm: implementation and theory*, in Numerical Analysis: Proceedings of the biennial conference, Dundee 1977, Lecture Notes in Mathematics no. 630, G. A. Watson, ed., Springer-Verlag, Berlin, 1978, pp. 105–116.

[12] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw–Hill, New York, 1996.

[13] E. Polak, *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, 1971.

[14] M. J. D. Powell, *A new algorithm for unconstrained optimization*, in Nonlinear Programming, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., Academic Press, New York, NY, 1970, pp. 31–65.

[15] ———, *Convergence properties of a class of minimization algorithms*, in Nonlinear Programming 2, O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds., Academic Press, New York, NY, 1975, pp. 1–27.

[16] V. Torczon, *On the convergence of pattern search algorithms*, SIAM Journal on Optimization, 7 (1997), pp. 1–25.

[17] V. Torczon and M. W. Trosset, *From evolutionary operation to parallel direct search: Pattern search algorithms for numerical optimization*, Computing Science and Statistics, 29 (1998), pp. 396–401.

[18] Y. Wen-ci, *Positive basis and a class of direct search techniques*, Scientia Sinica, Special Issue of Mathematics, 1 (1979), pp. 53–67.

[19] P. Wolfe, *Convergence conditions for ascent methods*, SIAM Review, 11 (1969), pp. 226–235.