From Evolutionary Operation to Parallel Direct Search: Pattern Search Algorithms for Numerical Optimization

Virginia Torczon

Department of Computer Science
College of William & Mary
Williamsburg, Virginia 23187–8795

Michael W. Trosset

Department of Computational &
Applied Mathematics—MS 134
Rice University
Houston, Texas 77005–1892

Abstract

G.E.P. Box's seminal suggestions for Evolutionary Operation led other statisticians to propose algorithms for numerical optimization that rely exclusively on the direct comparison of function values. These contributions culminated in the development of the widely used simplex algorithm of Nelder and Mead.

Recent examination of these popular methods by the numerical optimization community has produced new insights. Numerical experiments and carefully constructed examples have revealed that the Nelder-Mead algorithm may be unreliable even in fairly simple situations. In contrast, many of the original methods, which we collectively describe as pattern searches, are guaranteed to converge to a stationary point of the objective function under conventional nonlinear programming assumptions. In addition, the structure of these algorithms is such that they are easily parallelized.

We will briefly survey the history of pattern search methods and explicate their common structure, pointing out the key features that the Nelder-Mead simplex algorithm lacks. We will close with some practical suggestions for using pattern searches in serial and in distributed computing environments.

1 Introduction

We consider the problem of minimizing $f : S \subseteq \mathbb{R}^p \rightarrow \mathbb{R}$. For simplicity we restrict our discussion to the unconstrained case of $S \equiv \mathbb{R}^p$, though analysis exists for the linearly constrained case [8, 10]. We assume that f is a continuously differentiable function and that evaluation of $f(x)$ is deterministic, i.e., if we repeatedly evaluate f at x, then we always obtain the same value of $f(x)$. However, some of the algorithms to which we refer were originally intended for use when evaluation of $f(x)$ is stochastic, i.e., when $f(x)$ is a random variable.

Pattern search methods evolved directly from the statistical literature on response surface methodology (RSM). The foundations of RSM were established by Box and Wilson [2], who were concerned with minimizing an unknown quadratic objective function in the case that observed function values represent the unknown quadratic plus normal homoscedastic error. In this case, gradient-based methods for optimization are problematic because analytic derivatives are not available and random errors render finite-difference approximations to the gradient unreliable. Hence, Box and Wilson proposed constructing local models of the objective function by performing linear or quadratic regression experiments in a neighborhood of the current iterate. The designs employed in these experiments would become the patterns in pattern search methods.

Classical RSM is a sequential, but finite, procedure. Because the objective function is assumed to be quadratic, a typical application might involve two linear regression experiments followed by a final quadratic regression experiment. In 1955, however, Box [1] proposed an ongoing procedure for improving industrial efficiency. Evolutionary Operation (EVOP) was conceived, not as a substitute for sophisticated statistical methods like RSM, but as a simple procedure that could be used by plant personnel. Accordingly, the estimation of regression models was replaced by the direct inspection of data and the experimental designs of RSM became the "patterns of variants" of EVOP. Rules for determining a "cycle of variants" were intentionally vague, left to the discretion of the plant manager and an EVOP committee.

Subsequently, Spendley, Hext, and Himsworth [16] proposed an automatic EVOP procedure. In so doing, they made two crucial innovations. First, Box [1] had recommended 2-level factorial designs plus center

point(s). Spendley et al. substituted simplex designs, arguing that it was desirable to complete a new design as rapidly as possible and noting that simplex designs are the first-order designs with the minimal number of points. (They also noted that simplex designs are efficient, rotatable, and optimal for estimating slope in the presence of error.) Second, Spendley et al. observed that an automatic procedure for EVOP also might be used for numerical optimization.

The well-known simplex method of Nelder and Mead [12] was conceived as a modification of the simplex method of Spendley, Hext, and Himsworth. Although Nelder and Mead acknowledged the legacy of RSM and EVOP, they considered their method to be an algorithm for numerical optimization and published it in *The Computer Journal*. Perhaps not surprisingly, theirs is the only method of this lineage that is well known in the numerical optimization community.

Direct search methods have remained popular with users for many reasons. They are simple to program, easy to use, and widely applicable. Furthermore, many of these methods work well in practice on a wide variety of problems. In this paper, we distinguish a particular subset of direct search methods, the *pattern search* methods, that have strong ties to experimental design, employ heuristics that identify a direction of "steep" descent, and can be demonstrated to be as robust in theory as they are in practice.

2 Convergence Analysis

Direct search methods are easy to describe; however, their very simplicity complicates analysis of their convergence properties. Pattern search methods possess enough structure that these complications can be overcome to produce surprisingly strong convergence results. In this section we explain why this is the case. The reader interested in further details and formal proofs is referred to other sources [8, 9, 10, 21].

Until recently, direct search methods were often derided by members of the numerical optimization community. Extensive convergence analyses of derivative-based optimization methods have been produced by this community (for a good introduction, see [5]), but no corresponding analysis of direct search methods was widely known to exist (though some results had been derived as early as 1971; see, for example, [4] and [14]). The specific concern was that there was no guarantee of global convergence, by which is meant convergence to a stationary point or local minimizer of f from an arbitrary starting point x_0. There are ways to ensure global convergence, which is relatively easy to achieve in practice, *if* one has an explicit representation (or approximation) to the directional derivative. Direct search methods, however, are characterized by the absence of such representation.

The "classic" approach to global convergence analysis (see, for instance, [13]) requires that the method under consideration possess three properties: it must be a *descent* method, it must be a *gradient-related* method, and there must be sufficient step length control to prevent premature convergence to a nonstationary point of the function. If one has an explicit representation (or approximation) to the directional derivative at the current iterate $\nabla f(x_k)$, then it is straightforward to construct methods for which the classic analysis holds: first one chooses a direction that is "related" to $-\nabla f(x_k)$, then one forces the step from x_k to some trial iterate x_+ to satisfy a "sufficient" amount of decrease predicted by both $\|\nabla f(x_k)\|$ and $\|s_k\|$, where $s_k = x_+ - x_k$.

Convergence analyses of direct search methods are considerably complicated by the fact that these methods do not have an explicit representation of ∇f: by design, they only use values of f. Furthermore, direct search methods accept the step s_k and set $x_{k+1} = x_k + s_k$ if $f(x_{k+1}) < f(x_k)$. This criterion only requires the user to compare function values, which is a profound advantage if one only has access to ordinal information or if one does not trust the evaluation enough (due to noise, truncation error, measurement error, etc.) to give full credence to numeric values. The important implication for the analysis is that direct search methods use a *simple* decrease criterion as opposed to the *sufficient* decrease criterion used by gradient-based methods.

Pattern search methods are a subset of direct search methods. Their formal characterization is described in [9, 21], but critical to the success of the early algorithms (coordinate search with fixed step length [14], evolutionary operation with two-level factorial designs [1, 3, 17], and the original pattern search method of Hooke and Jeeves [7]) is their reliance on classical experimental designs. Furthermore, each of these methods employs designs that include at least $p+1$ points and "safeguards" the designs from degeneracy, thereby ensuring adequate information about the entire p-dimensional domain in the neighborhood of the current iterate x_k. Under very mild assumptions on f, these simple heuristics provide enough structure to guarantee global convergence, i.e., one can prove that

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

or, if additional assumptions can be made about the method, that

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

As an important aside we note that the popular Nelder-Mead simplex method is *not* a pattern search method. There has long been anecdotal evidence that this algorithm can fail in practice because "the simplex collapses." Numerical experiments [18] demonstrated that the single search direction constructed by Nelder-Mead can become arbitrarily bad in practice, resulting in convergence to a nonstationary point of the function. Recent analysis [11] shows that convergence to a nonstationary point can occur for a family of two-dimensional, smooth, convex functions. In all cases, the difficulty is that a sequence of simplices produced by the Nelder-Mead simplex method can come arbitrarily close to degeneracy.

3 Pattern Search Methods

Generalized pattern search is outlined below. The method starts with an initial guess at a solution (x_0, provided by the user), an initial pattern (the matrix P_0, usually determined by the particular method of pattern search), and a scalar parameter (Δ_0, which plays the role of a step length control parameter). Note that Δ_k provides a natural stopping criterion for the search: one stops when the steps are deemed "small enough." This is accomplished by setting the tolerance at a level that reflects the known accuracy of the computation or the required accuracy of the result. (It should be appreciated, however, that inaccuracies in the computation of either the steps s_k^i or the objective f cannot be magically overcome by choosing an arbitrarily small value for the stopping tolerance.)

Generalized Pattern Search:

Given $x_0 \in I\!R^n$, $f(x_0)$, $P_0 \in I\!R^{n\times p}$, and $\Delta_0 > 0$, for $k = 0, 1, \ldots$ until convergence do

1. Find a step s_k using `Exploratory Moves`(Δ_k,P_k).
2. If $f(x_k+s_k) < f(x_k)$, then $x_{k+1} = x_k+s_k$. Otherwise, $x_{k+1} = x_k$.
3. `Update`(Δ_k,P_k)

Individual pattern search methods are further characterized by the `Exploratory Moves`, the choice of which is critical, and the way in which updates are performed. Fortunately, only a few simple rules must be enforced:

(1) *The trial steps s_k^i must be defined by the pattern.* In other words, $s_k^i \in \Delta_k P_k$. Arbitrary steps of arbitrary length are not allowed.

(2) Any *step $s_k^i \in \Delta_k P_k$ for which $f(x_k + s_k^i) < f(x_k)$ is acceptable.* This allows for a wealth of strategies. At one extreme, one could sample the function at every trial step defined by the current pattern $\Delta_k P_k$. At the other extreme, one could rely on chance, divine intervention, or prior knowledge about the behavior of f to choose a single trial step $s_k^i \in \Delta_k P_k$ for which one believes that the condition $f(x_k + s_k^i) < f(x_k)$ will be satisfied.

(3) *An exploratory moves algorithm must have a "fall back" strategy.* This strategy must contain a minimum of $p + 1$ points so as to guarantees a direction of descent *in the limit.* It is this guarantee of at least one descent direction—and that a step of sufficient length will be taken along this direction—that prevents premature convergence to a nonstationary point of f.

(4) *One may keep x_k—and reduce Δ_k—only if* no *step s_k^i defined by the fall back strategy satisfies the simple decrease condition.* Thus, if one fails to identify a step that satisfies $f(x_k + s_k^i) < f(x_k)$, then one may have to evaluate f at as many as $p+1$ points in order to proceed. This is the price that one pays for the guarantee of global convergence.

4 Examples

We begin by illustrating a simple example of a pattern search method, designed to examine one step at a time, applied to a convex quadratic function (the concentric ellipses in Figures 1–4 denote the level sets of the function), and started from three different points. Note that for this problem the unique global minimizer lies at the common center of the ellipses.

In Figure 1, the exploratory moves algorithm tries the first step s_k^1 and evaluates f at $x_k^1 = x_k + s_k^1$. Since $f(x_k^1) < f(x_k)$, the step s_k^1 is accepted and $x_{k+1} = x_k^1$.

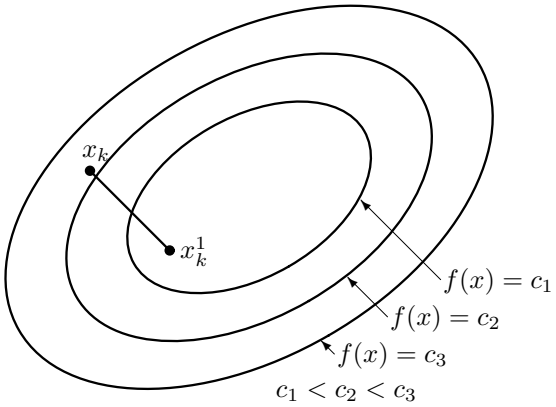

Figure 1: Scenario 1: Accept x_k^1.

In Figure 2, the exploratory moves algorithm again tries the step s_k^1 first, but $f(x_k^1) > f(x_k)$, so the step s_k^2

is tried next and f is evaluated at $x_k^2 = x_k + s_k^2$. Since $f(x_k^2) < f(x_k)$, the step s_k^2 is accepted and $x_{k+1} = x_k^2$.

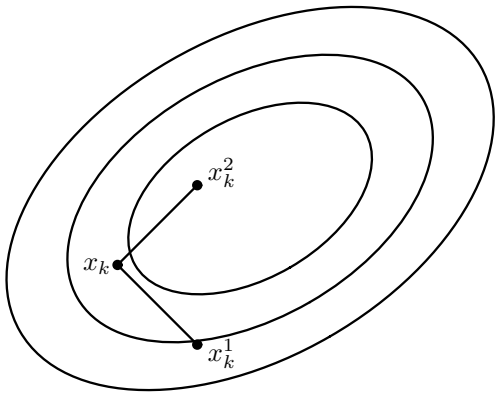

Figure 2: Scenario 2: Accept x_k^2.

In Figure 3, the exploratory moves algorithm evaluates the function at the three trial iterates $x_k^1 = x_k + s_k^1$, $x_k^2 = x_k + s_k^2$, and $x_k^3 = x_k + s_k^3$ and none of the three trial iterates produce a function value that is better than the value at $f(x_k)$. At this point, we have satisfied the conditions for a successful fall back strategy: we have evaluated the function at a "sufficient" set of $p + 1$ trial iterates. This means that we may choose to keep x_k and reduce Δ_k to continue the search. It is clear from Figure 3 that this is a reasonable strategy: the direction from x_k to x_k^1 is a descent direction; the problem is that s_k^1 is too long. Note also that the trial iterates x_k^1, x_k^2, and x_k^3 form a simplex, an experimental design with the minimal number of points that span $\mathbb{R}^2$.

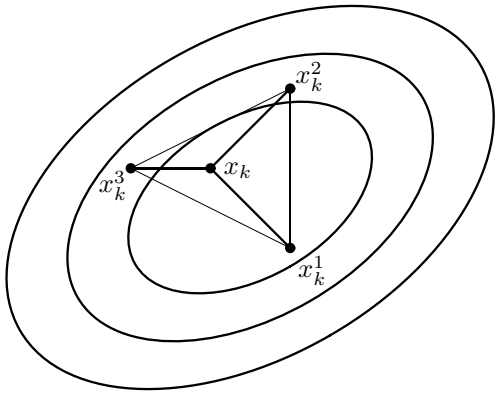

Figure 3: Scenario 3: Reject all 3 steps and reduce Δ_k.

In Figure 4 we see the result of continuing the search started in Figure 3, but with a smaller simplex. Now when the exploratory moves algorithm tries $x_{k+1}^1 = x_{k+1} + s_{k+1}^1$, $f(x_{k+1}^1) < f(x_{k+1})$ and the situation is similar to that in Figure 1.

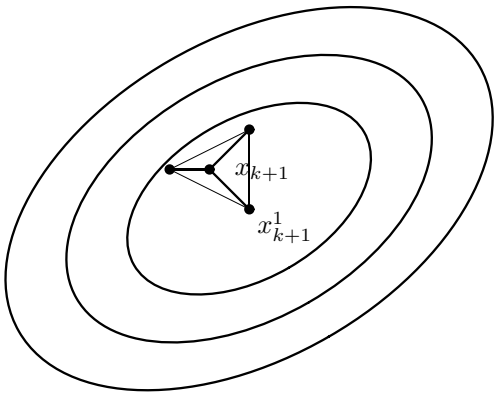

Figure 4: Scenario 3, continued: Repeat the search with $x_{k+1} = x_k$, $P_{k+1} = P_k$, and $\Delta_{k+1} = \frac{1}{2}\Delta_k$. Accept x_{k+1}^1.

The pattern matrix P_k used for this simple pattern search method is the same for all four examples (and remains unchanged across iterations):

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

Each column of the pattern corresponds to a trial step, with Δ_k serving as a scaling factor to determine the final length of the step. Thus, for Figures 1–3, $\Delta_k P_k = \begin{bmatrix} s_k^1 & s_k^2 & s_k^3 & 0 \end{bmatrix}$, with the last column corresponding to the zero step, i.e., setting x_{k+1} equal to x_k. In Figure 4, we see the effect on the length of the trial steps s_{k+1}^1, s_{k+1}^2, and s_{k+1}^3 when $\Delta_{k+1} = \frac{1}{2}\Delta_k$.

We have mentioned that pattern search methods admit a wide range of heuristics for the exploratory moves. They also admit a wide choice of pattern matrices P_k and update rules for P_k and Δ_k. For the convergence analysis to apply, it suffices that certain mild algebraic and geometric considerations are respected. Four possible patterns are displayed in Figure 5. Here we see the patterns associated with several different pattern search methods. In the upper left-hand corner is a composite design that might be used for evolutionary operation [1, 3, 17]. In the upper right-hand corner we see a simplex and its reflection, which define the pattern for the multidirectional search algorithm [19]. In the lower right-hand corner, we see the pattern that is associated with coordinate search with fixed step size [14]. In the lower left-hand corner are the two simplices (for both Δ_k and Δ_{k+1}) associated with the simple algorithm illustrated in Figures 1–4.

The convergence analysis for pattern search methods is possible because the iterates of a pattern search method lie on a scaled, translated rational lattice. This allows us to relax the classical requirements on the acceptance of the step and still guarantee global convergence.

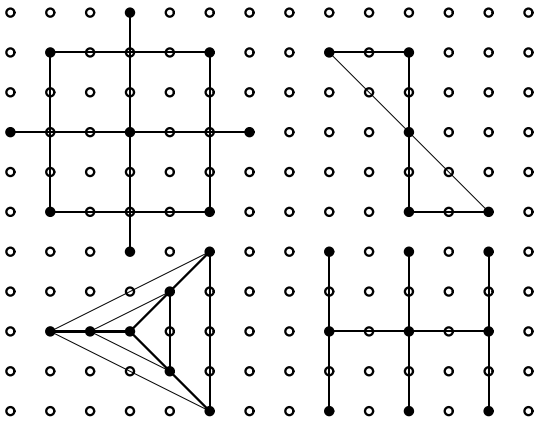

Figure 5: A selection of admissible patterns and their mapping onto a rational lattice.

5 Computational Features

Pattern search methods admit a wide range of heuristics. Coordinate search cautiously varies one coordinate at a time, whereas evolutionary operation using two-level factorial designs or composite designs may evaluate f at as many as 3^p points before choosing the next step. More recent examples of pattern search methods exhibit a similar diversity of strategies.

For example, parallel direct search (PDS) [6] was designed to exploit the strengths of multiprocessor computing environments. By simultaneously evaluating f at multiple points on multiple processors, additional evaluations can be obtained "for free," often accelerating the search by reducing the total number of iterations required. Opportunities for computational parallelism abound and a parallel implementation of a particular class of pattern search methods has been developed [20].

In contrast, model-assisted grid search [22] exploits methods developed for the design and analysis of computer experiments [15]. The idea is to construct approximations to f that can be used to predict a single trial step that is likely to realize simple decrease on the current value of $f(x_k)$. Thus, pattern search methods can also be quite frugal.

6 Recommendations

Experience suggests that pattern search methods are often useful in the following situations:

(1) *Evaluation of f is inaccurate.* Even when f is deterministic, computationally induced errors can occur, e.g., when evaluating f involves iterating to an approximate answer. One common difficulty occurs when these inaccuracies result in high-frequency oscillations in the returned function values. In such situations, derivative-based methods are likely to become trapped in the oscillations and thus fail to identify larger basins of smaller function values.

(2) *The derivatives of f are either not available or not reliable.* In particular, inaccuracies in the evaluation of f often render finite-difference approximations to the gradient unreliable.

(3) *The function f is not smooth.* Although the existing convergence analysis for pattern search methods assumes that f is continuously differentiable, we have found that pattern search methods are often effective in practice when the partial derivatives of f are not continuous and even when f is nondifferentiable. Our experience is consistent with the folklore that surrounds these methods: they are often suggested as the methods of choice for general, nonsmooth, nonlinear problems.

(4) *Only ordinal information about function values is available.* Because pattern search methods rely on simple (rather than sufficient) decrease, they do not require numerical function values.

We close with some general guidelines to keep in mind when using pattern search methods to solve nonlinear optimization problems.

(1) *Pattern search provides a useful exploratory tool.* Pattern search methods are widely applicable and easy to use. They provide a simple way to find a "quick and dirty" solution to a problem. For many applications, this may be sufficient. If not, they may provide a good starting guess to a more sophisticated method.

(2) *Pattern search methods have good global behavior.* Pattern search methods are good at locating the general region of a stationary point from any given starting point x_0. This is not surprising: they are gradient-related methods and thus share this property with the method of steepest descent.

(3) *Pattern search methods typically exhibit slow local (asymptotic) convergence rates.* Because pattern search methods do not use second-order (curvature) information, one cannot expect the fast (quadratic or superlinear) local convergence rates of Newton or quasi-Newton methods. However, in the low accuracy situations for which pattern search methods are most often recommended, the importance of fast local convergence to a highly accurate solution is questionable.

(4) *Pattern search methods may require relatively large numbers of function evaluations; hence they tend to be effective for problems of relatively small dimension.* Because pattern search methods acquire information by sampling the domain, they are plagued by the curse of dimensionality. Success, however, depends on the features of the particular problem. Even problems with

only $p = 2, 3$ variables may bedevil pattern search methods if there is a high degree of nonlinearity. On the other hand, pattern search methods have been applied successfully to problems with as many as 256 variables.

References

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

[2] 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.

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

[4] J. Céa. *Optimisation : Théorie et Algorithmes*. Dunod, Paris, 1971.

[5] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, 1996.

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

[7] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery (ACM)*, 8(2):212–229, 1961.

[8] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. Technical Report 96-20, ICASE, Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681-0001, 1996. To appear in *SIAM Journal on Optimization*.

[9] R. M. Lewis and V. Torczon. Rank ordering and positive bases in pattern search algorithms. Technical Report 96-71, ICASE, Mail Stop 403, NASA Langley Research Center, Hampton, Virginia 23681-0001, 1996. Submitted for publication.

[10] R. M. Lewis and V. Torczon. Pattern search algorithms for linearly constrained minimization. In preparation, 1997.

[11] K. I. M. McKinnon. Convergence of the Nelder–Mead simplex method to a non-stationary point. Technical Report MS 96-006, Department of Mathematics and Statistics, University of Edinburgh, 1996. To appear in *SIAM Journal on Optimization*.

[12] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[13] J. Nocedal. Theory of algorithms for unconstrained optimization. *Acta Numerica*, 1:199–242, 1991.

[14] E. Polak. *Computational Methods in Optimization: A Unified Approach*. Academic Press, New York, 1977.

[15] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4(4):409–423, 1989.

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

[17] W. H. Swann. Direct search methods. In Walter Murray, editor, *Numerical Methods for Unconstrained Optimization*, pages 13–28. Academic Press, London and New York, 1972.

[18] V. Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*. PhD thesis, Department of Mathematical Sciences, Rice University, Houston, TX, 1989.

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

[20] V. Torczon. PDS: Direct search methods for unconstrained optimization on either sequential or parallel machines. Technical Report TR92-09, Rice University, Department of Computational and Applied Mathematics, Houston, TX 77005–1892, 1992.

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

[22] M. W. Trosset and V. Torczon. Numerical optimization using computer experiments. Technical Report TR97-02, Department of Computational and Applied Mathematics, Rice University, 1997. Submitted for publication.