ON THE CONVERGENCE OF THE MULTIDIRECTIONAL SEARCH ALGORITHM*

VIRGINIA TORCZON†

Abstract. This paper presents the convergence analysis for the multidirectional search algorithm, a direct search method for unconstrained minimization. The analysis follows the classic lines of proofs of convergence for gradient-related methods. The novelty of the argument lies in the fact that explicit calculation of the gradient is unnecessary, although it is assumed that the function is continuously differentiable over some subset of the domain. The proof can be extended to treat most nonsmooth cases of interest; the argument breaks down only at points where the derivative exists but is not continuous. Finally, it is shown how a general convergence theory can be developed for an entire class of direct search methods—which includes such methods as the factorial design algorithm and the pattern search algorithm—that share a key feature of the multidirectional search algorithm.

Key words. unconstrained optimization, convergence analysis, direct search methods, parallel optimization, multidirectional search, Nelder–Mead simplex algorithm

AMS(MOS) subject classifications. 49D30, 65K05

1. Introduction. In this paper we shall give the convergence analysis for the multidirectional search algorithm [13]. The multidirectional search algorithm is a direct search method designed to solve the unconstrained minimization problem:

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) \,,$$

where $f : \mathbb{R}^n \to \mathbb{R}$.

Direct search algorithms presume little of the function—typically only that the function is continuous—since they do not require, or even directly estimate, gradient information. It has long been recognized that these methods are gradient-related, but the convergence analysis for a large class of these algorithms has been incomplete, as we shall discuss in the next section.

Our proof of convergence for the multidirectional search algorithm will follow the classic lines of proofs for gradient-related methods. First we will show that the multidirectional search algorithm is a descent method. Then we will show that the search directions do not deteriorate. Finally we will show that the algorithm satisfies a notion of sufficient decrease in the value of the objective function for the size of the step taken. While we assume that the function is continuously differentiable over some subset of the domain, we never explicitly compute the gradient. The novelty in this argument lies in the fact that the algorithm provides enough structure to make explicit information about the gradient unnecessary. The multidirectional search algorithm will be introduced in §3. The convergence analysis for the differentiable case will be developed in §§4, 5, and 6.

The result for the smooth case can also be extended to handle most nonsmooth cases of interest. This extension will be given in §7. Finally, we will close by reviewing both what the convergence analysis tells us about how the multidirectional search

* Received by the editors March 19, 1990; accepted for publication (in revised form) June 12, 1990. This research was sponsored by SDIO/IST/ARO and the Air Force Office of Scientific Research grant 89–0363. A portion of this work is contained in the author's doctoral thesis under the supervision of J. E. Dennis, Jr. in the Department of Mathematical Sciences, Rice University, Houston, Texas 77251-1892.

† Department of Mathematical Sciences, Rice University, Houston, Texas 77251-1892.

algorithm works and what it also suggests about a more general convergence theory for an entire class of direct search algorithms, which is a subject of our current research.

2. Direct search methods. Direct search methods are characterized by the fact that they do not use derivatives. They forgo this very useful information for very practical reasons. Often, particularly in experimental settings, analytic derivatives are simply unavailable. Finite-difference approximations to the gradient could be used, but the cost of computing the function values on a sequential machine may make this option prohibitively expensive. Furthermore, in an experimental setting it is not at all unusual to have function values that can only be trusted to a few significant digits so that finite-difference approximations to the gradient may prove unreliable. Another possibility is that the experimental apparatus itself may make finite-difference approximations to the gradient difficult, if not impossible.

Having relinquished explicit derivative information, the direct search methods typically consider a natural alternative: at every iteration they explore each direction in a linearly independent set of n search directions. As we shall see, if this set of search directions has the right structure, it is possible to derive convergence results for these algorithms. Our interest in direct search methods arose from the observation that at every iteration the algorithms typically perform searches in each of n directions. Thus, it certainly seems reasonable to assume that in most, if not all, of these algorithms the n searches required for a single iteration can be conducted independently. This, in turn, suggests a natural way to develop new optimization algorithms for parallel machines [5], [13].

The direct search algorithms are distinguished both by the way in which the set of n search directions is chosen and maintained and by the way "exploratory" steps are taken in each of the n directions. The most important distinction, for theoretical purposes, is between those methods for which the set of search directions is modified at the end of each iteration and those methods for which the set of search directions remains fixed across all iterations. Examples of algorithms in the first class include Rosenbrock's method [11], Powell's method [10], and Zangwill's method [16]. In the second class of direct search methods one finds such examples as the factorial design algorithm of Box [2], the pattern search algorithm of Hooke and Jeeves [7], and the simplex algorithm of Spendley, Hext, and Himsworth [12] (the method upon which the simplex algorithm of Nelder and Mead is based).

The Nelder–Mead simplex algorithm [8], perhaps the most popular of the direct search methods, does not search in each of n linearly independent search directions at every iteration. It is worth noting that while we know of several attempts to prove convergence of the Nelder–Mead simplex algorithm, none of these has met with success. Furthermore, the experimental evidence gathered while testing and comparing the multidirectional search algorithm with the Nelder–Mead simplex algorithm suggests that a convergence result for the Nelder–Mead simplex algorithm may not be possible [13]. The fact that the Nelder–Mead simplex algorithm only searches in a single direction at each iteration removes it from the domain of this analysis—an observation which first suggested how the Nelder–Mead simplex algorithm might fail. However, as yet we can offer no satisfactory explanation of *why* this failure occurs. (For a further discussion of the experimental results, see [13].)

The methods which modify the set of search directions at the end of each iteration use the result of the exploratory searches along each of n linearly independent search directions to compute a new search direction—that defined by the point used to start the iteration and the point found at the conclusion of the n exploratory steps. The methods are distinguished by the way in which the new set of search directions is then determined. As Zangwill [16] points out, care must be taken to ensure that the set of search directions remains linearly independent. However, once proper precautions are observed, several nice results can be derived, as shown by both Powell [10] and Zangwill [16] in their original papers, as well as those found in later works. (See [1], [9].)

The direct search methods for which the search directions remain fixed across all iterations have been analyzed with varying degrees of success. One of the best known algorithms of this sort is the pattern search algorithm of Hooke and Jeeves. Céa [3] gives a concise proof of convergence that uses few of the features of the pattern search algorithm. His analysis does, however, rely on the fact that the algorithm requires the step sizes to be monotonically decreasing. The multidirectional search algorithm allows the steps to increase in size if there is a corresponding decrease in the value of the objective function. Thus, Céa's result cannot be extended to the multidirectional search algorithm. Until recently, this has been the only convergence result of which we were aware for this class of direct search methods. We have since discovered a more ambitious convergence analysis undertaken by Yu Wen-ci [14], [15]. He, too, noted that the pattern search algorithm of Hooke and Jeeves and the original simplex algorithm of Spendley, Hext, and Himsworth share the features of fixed search directions and fixed rescaling factors. Using these observations, and the notion of a positive basis, he derived a general convergence result for modified versions of both these algorithms.

Our analysis differs from that of Yu Wen-ci in two important respects. First, Yu Wen-ci, like Céa, must assume that the step sizes are monotonically decreasing. Again, this means that his analysis cannot be extended to cover the multidirectional search algorithm. Our analysis allows the step sizes to increase if decreases are seen in the corresponding objective function values.

Second, to show that these algorithms cannot take steps that are too long relative to the amount of decrease in the value of the objective function— in other words, that there is *sufficient* decrease in the value of the objective function for the size of the step taken—Yu Wen-ci introduces the notion of an "error-controlling" sequence. Unfortunately, it is not clear how such a sequence can be constructed and maintained in practice. In fact, one of the difficulties in applying the classical analysis for descent methods to these algorithms is that an explicit calculation of the initial descent slope is used to enforce the Armijo–Goldstein–Wolfe conditions for sufficient decrease (see [4], [9]). Direct search methods, by definition, do not compute such information. Our result places no such burden on the multidirectional search algorithm. We can guarantee that the algorithm cannot take steps that are too long, relative to the amount of decrease in the object function value, *without* enforcing the Armijo–Goldstein–Wolfe conditions or introducing any other such measure of sufficient decrease. The key feature of the multidirectional search algorithm which makes this analysis possible is that both the search directions and the scaling factors which determine the step sizes are fixed across all iterations of the algorithm. This feature can also be found in several of the traditional sequential direct search methods, so one of the goals of our current research is to extend the results presented here for the multidirectional search algorithm to these other direct search methods. The only limitation we must impose is that the scaling factors used to determine the size of the steps taken must be rational. This mild restriction is easily satisfied by all the algorithms in this class; the values typically recommended for the scaling factors are $\frac{1}{2}$ and 2.

We are now ready to introduce the multidirectional search algorithm.

3. Algorithm. A formal statement of the multidirectional search algorithm is given on the next page. We proceed with a brief description of the algorithm.

In order to understand the algorithm, consider the sequence of "best" vertices $\{\mathbf{v}_0^k\}$, where by "best" we mean that for all k,

$$f(\mathbf{v}_0^k) \le f(\mathbf{v}_i^k) \quad \text{for } i = 1, \cdots, n\,.$$

At each iteration of the inner **repeat** loop a search is conducted from the current best vertex $\mathbf{v}_0^k$ in each of the n directions determined by the n edges adjacent to $\mathbf{v}_0^k$. The goal of the search is to replace $\mathbf{v}_0^k$, i.e., to find a new vertex with a function value that is strictly less than the function value at $\mathbf{v}_0^k$.

There are three possible trial steps: the rotation[1] step, the expansion step, and the contraction step, which are illustrated in Fig. 1. Note that the algorithm always computes the rotation step and then tests to see if a new best vertex has been found. If a new best vertex has been identified, an expansion step is computed. Otherwise, the algorithm computes, and automatically accepts, the contraction step.

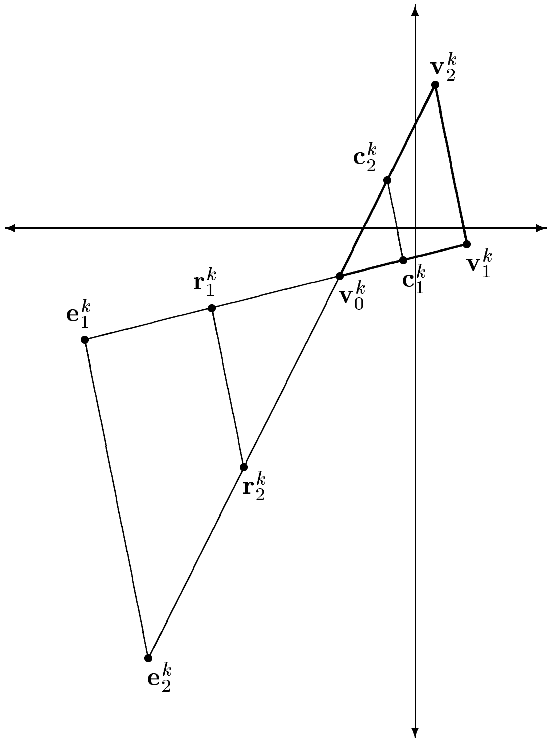

FIG. 1. *The three possible steps given the simplex S_k with vertices $\langle \mathbf{v}_0^k, \mathbf{v}_1^k, \mathbf{v}_2^k \rangle$.*

The algorithm can thus be viewed in terms of its two primary loops: the outer **while** loop, which determines a new set of search directions by considering a new

[1] The "rotation" step is analogous to the "reflection" step found in the simplex algorithm of Nelder and Mead [8]. We choose to depart from the terminology used by Nelder and Mead since "rotation" more accurately describes the effect of the step on the position of the simplex.

THE MULTIDIRECTIONAL SEARCH ALGORITHM

```
Given an initial simplex S_0 with vertices ⟨v_0^0, v_1^0, ···, v_n^0⟩, choose μ, θ ∈ Q such that
/* expansion factor */
μ ∈ (1, +∞), and
/* contraction factor */
θ ∈ (0, 1).

for i = 0, ···, n                                          /*** initialization loop ***/
    calculate f(v_i^k)
end
k ← 0
while (stopping criterion is not satisfied) do             /*** outer while loop ***/
    /* find a new best vertex */
    j ← arg min_i { f(v_i^k) : i = 0, ···, n }
    swap v_j^k and v_0^k
    repeat                                                  /* inner repeat loop */
        Check the stopping criterion.
        /* rotation step */
        for i = 1, ···, n
            r_i^k ← v_0^k − (v_i^k − v_0^k)
            calculate f(r_i^k)
        end
        replaced = (min { f(r_i^k) : i = 1, ···, n } < f(v_0^k))
        if replaced then
            /* expansion step */
            for i = 1, ···, n
                e_i^k ← v_0^k − μ (v_i^k − v_0^k)
                calculate f(e_i^k)
            end
            if (min { f(e_i^k) : i = 1, ···, n } < min { f(r_i^k) : i = 1, ···, n }) then
                /* accept expansion */
                v_i^k ← e_i^k for i = 1, ···, n
            else
                /* accept rotation */
                v_i^k ← r_i^k for i = 1, ···, n
            endif
        else
            /* contraction step */
            for i = 1, ···, n
                c_i^k ← v_0^k + θ (v_i^k − v_0^k)
                calculate f(c_i^k)
            end
            replaced = (min { f(c_i^k) : i = 1, ···, n } < f(v_0^k))
            /* accept contraction */
            v_i^k ← c_i^k for i = 1, ···, n
        endif
    until replaced                                          /* end repeat loop */
    k ← k + 1
end                                                         /**** end while loop ****/
```

best vertex, and the inner **repeat** loop, which determines the length of the steps to be taken.

The expansion factor μ and the contraction factor θ determine the lengths, relative to the original edges in the simplex, of the steps to be considered. In our implementation, μ is set equal to 2 and θ is set equal to $\frac{1}{2}$, as in the example shown in Fig. 1. Further discussion of the choice of scaling factors, as well as additional implementation details, can be found in [13].

4. Key lemmas. Two important results follow directly from the description of the multidirectional search algorithm. First, the multidirectional search algorithm is a *descent method* since

$$f(\mathbf{v}_0^{k+1}) < f(\mathbf{v}_0^k)\,.$$

For this to be true we need only show that if the best vertex $\mathbf{v}_0^k$ is at a point where the function is differentiable, and $\nabla f(\mathbf{v}_0^k)$ is not equal to zero, then the inner **repeat** loop terminates in a finite number of iterations. Second, the search directions determined by the multidirectional search algorithm are *uniformly linearly independent*. This means that there exists a constant $\gamma > 0$ such that for all $k \geq 0$ and $\mathbf{x} \neq 0$,

$$\max\left\{\frac{|\mathbf{x}^T(\mathbf{v}_i^k - \mathbf{v}_0^k)|}{\|\mathbf{x}\|\,\|\mathbf{v}_i^k - \mathbf{v}_0^k\|}, i = 1, \cdots, n\right\} \geq \gamma\,.$$

This follows from the observation that while the algorithm may translate and rotate the simplex, or change its scale, the angles in the original simplex S_0 are preserved across all iterations of the algorithm. Both results are given in the next two lemmas. We begin by introducing the following notation.

Let $\mathbf{v}_0^k$ be the best vertex of the simplex S_k used to start the kth iteration of the algorithm. Define the level set of f at $\mathbf{v}_0^k$ to be

$$L(\mathbf{v}_0^k) = \left\{\mathbf{x} : f(\mathbf{x}) \leq f(\mathbf{v}_0^k)\right\}.$$

Given $\mathbf{y} \in I\!R^n$, let the contour $C(\mathbf{y})$ be

$$C(\mathbf{y}) = \{\mathbf{x} : f(\mathbf{x}) = f(\mathbf{y})\}\,.$$

Let X_* be the set of stationary points of the function f in $L(\mathbf{v}_0^0)$.

We will assume, for now, that f is continuously differentiable on $L(\mathbf{v}_0^0)$; however, as we will discuss in §7, if we redefine X_*, this assumption can be weakened. We next show that the multidirectional search algorithm is a descent method.

LEMMA 4.1. *Suppose that f is continuously differentiable on $L(\mathbf{v}_0^0)$. If the multidirectional search algorithm finds an iterate $\mathbf{v}_0^{k+1}$, then*

$$f(\mathbf{v}_0^{k+1}) < f(\mathbf{v}_0^k)\,.$$

If $\nabla f(\mathbf{v}_0^k) \neq 0$, then the multidirectional search algorithm will find such a $\mathbf{v}_0^{k+1}$.

Proof. The multidirectional search algorithm will accept a new iterate $\mathbf{v}_0^{k+1}$ *only* if $f(\mathbf{v}_0^{k+1}) < f(\mathbf{v}_0^k)$, so the first conclusion in the lemma is tautologically true. The more interesting point is the second: if $\mathbf{v}_0^k$ is not a critical point—in particular, if $\nabla f(\mathbf{v}_0^k) \neq 0$—then the algorithm will find an acceptable $\mathbf{v}_0^{k+1}$ in a finite number of iterations of its inner **repeat** loop. This should not be too surprising since the algorithm is performing line searches along the n directions determined by the n edges adjacent to $\mathbf{v}_0^k$.

We will assume that $\mathbf{v}_0^k \notin X_*$ and show that the inner **repeat** loop terminates in a finite number of iterations.

The set of edges adjacent to any vertex in a simplex is linearly independent. Thus, the set of n edges adjacent to the current best vertex $\mathbf{v}_0^k$,

$$\left\{(\mathbf{v}_i^k - \mathbf{v}_0^k) : i = 1, \cdots, n\right\},$$

spans $I\!R^n$.

If $\mathbf{v}_0^k \notin X_*$, then $\nabla f(\mathbf{v}_0^k)$ is nonzero. Consequently, there exists at least one i, for $i = 1, \cdots, n$, such that

$$\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) \neq 0\,.$$

There are then two cases to consider.

Case 1. $\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) > 0$.

Note that since $(\mathbf{v}_i^k - \mathbf{v}_0^k) = -(\mathbf{r}_i^k - \mathbf{v}_0^k)$,

$$\nabla f(\mathbf{v}_0^k)^T(\mathbf{r}_i^k - \mathbf{v}_0^k) < 0\,.$$

Consider the right-hand directional derivative of f at $\mathbf{v}_0^k$ in the direction $(\mathbf{r}_i^k - \mathbf{v}_0^k)$:

$$f'(\mathbf{v}_0^k)^T(\mathbf{r}_i^k - \mathbf{v}_0^k) = \lim_{h\to 0^+} \frac{f\left(\mathbf{v}_0^k + h\,(\mathbf{r}_i^k - \mathbf{v}_0^k)\right) - f(\mathbf{v}_0^k)}{h} < 0\,.$$

Thus, there exists an $h_k > 0$ such that for $0 < t \leq h_k$

$$f\left(\mathbf{v}_0^k + t(\mathbf{r}_i^k - \mathbf{v}_0^k)\right) < f(\mathbf{v}_0^k)\,.$$

Case 2. $\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) < 0$.

Consider the right-hand directional derivative of f at $\mathbf{v}_0^k$ in the direction $(\mathbf{v}_i^k - \mathbf{v}_0^k)$:

$$f'(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) = \lim_{h\to 0^+} \frac{f\left(\mathbf{v}_0^k + h\,(\mathbf{v}_i^k - \mathbf{v}_0^k)\right) - f(\mathbf{v}_0^k)}{h} < 0\,.$$

Thus, there exists an $\bar{h}_k \in (0, h_k]$ such that for $0 < t \leq \bar{h}_k$

$$f\left(\mathbf{v}_0^k + t\,(\mathbf{v}_i^k - \mathbf{v}_0^k)\right) < f(\mathbf{v}_0^k)\,.$$

Conclusion. The vertices of the contracted simplex are defined to be

$$\mathbf{c}_i^k = \mathbf{v}_0^k + \theta\left(\mathbf{v}_i^k - \mathbf{v}_0^k\right),$$

for $i = 1, \cdots, n$, where $\theta \in (0, 1)$ is the fixed contraction factor. If the contracted step is accepted at the current iteration of the inner **repeat** loop, and a new best vertex has not been found, then at the next iteration of the inner loop, the rotated step can be defined in terms of the contracted vertices as

$$\begin{aligned}
\mathbf{r}_i^k &= \mathbf{v}_0^k - \left(\mathbf{c}_i^k - \mathbf{v}_0^k\right) \\
&= \mathbf{v}_0^k - \left(\left(\mathbf{v}_0^k + \theta\left(\mathbf{v}_i^k - \mathbf{v}_0^k\right)\right) - \mathbf{v}_0^k\right) \\
&= \mathbf{v}_0^k - \theta(\mathbf{v}_i^k - \mathbf{v}_0^k)\,.
\end{aligned}$$

Therefore, for any k, if $\mathbf{v}_0^k \notin X_*$, then there exists a positive integer p_k, such that $\theta^{p_k} < \bar{h}_k$ and so the inner **repeat** loop terminates in a finite number of iterations as required. □

LEMMA 4.2. *The search directions determined by the multidirectional search algorithm are uniformly linearly independent.*

Proof. Consider each of the three possible steps the multidirectional search algorithm may take.

1. The rotation step rotates the given initial simplex about the best vertex. This could be viewed as first translating the given initial simplex by $-\mathbf{v}_0^k$, applying the transformation $-I$, and then translating the simplex by $\mathbf{v}_0^k$. Since $-I$ is an orthogonal transformation, the Euclidean inner product is preserved. Thus, neither the translations nor the rotation affect the angles in the given initial simplex.
2. The expansion step rescales the rotated simplex by a factor of μ, which again leaves the angles of the given initial simplex unchanged.
3. The contraction simplex is formed by first translating the rotated simplex by $-\mathbf{v}_0^k$, applying the orthogonal transformation $-I$, and then translating the simplex by $\mathbf{v}_0^k$, which simply restores the initial simplex. The simplex is then rescaled by a factor of θ. Thus, the angles of the given initial simplex are unchanged.

Since none of the three possible steps affect the angles in the initial simplex, the angles of the simplex used to start the algorithm S_0 remain constant across all iterations of the multidirectional search algorithm. Therefore, there exists a $\gamma > 0$, which does not depend on k, such that

$$\max\left\{\frac{|\mathbf{x}^T(\mathbf{v}_i^k - \mathbf{v}_0^k)|}{\|\mathbf{x}\|\,\|\mathbf{v}_i^k - \mathbf{v}_0^k\|}, i = 1, \cdots, n\right\} \geq \gamma \tag{1}$$

for all $\mathbf{x} \in I\!R^n$, $\mathbf{x} \neq 0$, as required. □

5. The smooth case. We are now ready for a formal statement of the theorem.

THEOREM 5.1. *Assume that $L(\mathbf{v}_0^0)$ is compact and that f is continuously differentiable on $L(\mathbf{v}_0^0)$. Then some subsequence of $\{\mathbf{v}_0^k\}$ converges to a point $\mathbf{x}_* \in X_*$. Furthermore, $\{\mathbf{v}_0^k\}$ converges to $C_* = C(\mathbf{x}_*)$ in the sense that*

$$\lim_{k\to\infty}\left[\inf_{\mathbf{x}\in C_*}\|\mathbf{v}_0^k - \mathbf{x}\|\right] = 0\,.$$

COROLLARY 5.2. *Assume that f is continuously differentiable, $L(\mathbf{v}_0^0)$ is compact, and f is strictly convex on $L(\mathbf{v}_0^0)$. Then*

$$\lim_{k\to\infty}\mathbf{v}_0^k = \mathbf{x}_*\,,$$

where $\mathbf{x}_*$ is the unique minimizer of f in $L(\mathbf{v}_0^0)$.

We will show that if the conclusion of Theorem 5.1 were not true, then there would be a lower bound on the lengths of the edges of the simplices generated by the algorithm. This, in turn, would imply that the algorithm can generate at most a finite number of iterates, which would contradict the fact that away from X_* the algorithm produces a sequence of points with strictly monotonically decreasing function values, as we showed in Lemma 4.1.

The proof of Theorem 5.1 will be given in §6. The rest of this section is devoted to results needed for the proof.

5.1. Enforcing step length control. Together, Lemmas 4.1 and 4.2 guarantee that if f is continuously differentiable on $L(\mathbf{v}_0^0)$, then the multidirectional search algorithm generates at least one direction of descent that is uniformly bounded away from orthogonality with the gradient. However, we still need to ensure that the steps the algorithm takes are neither too long nor too short relative to the amount of decrease seen in the value of the objective function. In most other descent methods, step length control is guaranteed by enforcing the Armijo–Goldstein–Wolfe conditions. The multidirectional search algorithm cannot enforce the Armijo–Goldstein–Wolfe conditions because it does not have explicit knowledge of the gradient. There is enough structure in the algorithm, however, to enforce step length control without enforcing the Armijo–Goldstein–Wolfe conditions.

We begin by proving the existence of an upper bound on the lengths of the edges of the simplex.

PROPOSITION 5.3. *Assume that $L(\mathbf{v}_0^0)$ is compact and that f is continuously differentiable on $L(\mathbf{v}_0^0)$. Then there exists a constant $M > 0$ such that*

$$\left\| \mathbf{v}_i^k - \mathbf{v}_0^k \right\| \leq M \qquad \forall\, i, k\,.$$

Proof. Assume that $\mathbf{v}_0^0 \notin X_*$. If so, in the proof of Lemma 4.1 we showed that the multidirectional search algorithm will produce a new best vertex in a finite number of iterations of the inner **repeat** loop. Since $L(\mathbf{v}_0^0)$ is compact, for all $k \geq 1$ there exists at least one vertex $\mathbf{v}_i^k$ such that the vertices $\mathbf{v}_0^k, \mathbf{v}_i^k \in L(\mathbf{v}_0^0)$. Since $L(\mathbf{v}_0^0)$ is bounded, this implies a bound on the length of the edge $(\mathbf{v}_i^k - \mathbf{v}_0^k)$. Since the rescaling factors μ and θ are constant across all edges of the simplex for all iterations of the multidirectional search algorithm, the relative lengths of all edges in the simplex remain the same across all iterations of the algorithm. This implies the existence of $M > 0$ as required.

If $\mathbf{v}_0^0 \in X_*$, we can no longer argue that the inner **repeat** loop will produce a new best vertex in a finite number of iterations. There are then two possibilities. The first is that the multidirectional search algorithm produces a new best vertex. If so, the argument given above still holds. The other possibility is that the multidirectional search algorithm does not find a new best vertex, i.e., the inner **repeat** loop does not terminate. If so, then the contraction step has been accepted, because the rotation step and the expansion step are accepted only when they produce a new best vertex. Since the contraction factor θ is strictly less than one, this means that the length of every edge in the simplex is strictly monotonically decreasing. Thus, the maximum length across all edges in the initial simplex S_0 provides an upper bound on the length of all edges in all subsequent simplices, as required. □

The existence of an upper bound on the lengths of the edges of the simplex implies the existence of a compact set, which we shall call $\mathcal{M}$, that contains $L(\mathbf{v}_0^0)$ and all the simplices generated by the multidirectional search algorithm from a given initial simplex S_0.

We next posit the following *null* hypothesis: the sequence of best vertices $\mathbf{v}_0^k$ stays uniformly bounded away from X_*. We will show that under this hypothesis a lower bound on the lengths of the edges in the simplex exists and that once the edges in the simplex become small enough, the rotation step will always be acceptable. Note that this does not necessarily mean that the rotation step will be accepted. If the rotation step is acceptable, the multidirectional search algorithm automatically considers the expansion step; however, we are guaranteed that the contraction step will not be considered. Since the lengths of the edges in the simplex are reduced only when the simplex is contracted, this argument ensures no further reduction in the size of the simplex.

PROPOSITION 5.4. *Assume that $L(\mathbf{v}_0^0)$ is compact and f is continuously differentiable on $L(\mathbf{v}_0^0)$. Suppose that the best vertices stay uniformly bounded away from*

X_*; i.e., for a fixed $\epsilon > 0$, independent of k,

$$\left\|\mathbf{v}_0^k - \mathbf{x}_*\right\| \geq \epsilon \qquad \forall\, \mathbf{x}_* \in X_*\,, \qquad \forall\, k \geq 0\,.$$

Then there exists a constant $m > 0$ such that for all $k \geq 1$

$$m \leq \left\|\mathbf{v}_j^k - \mathbf{v}_l^k\right\| \qquad \forall\, 0 \leq j, l \leq n\,, \quad j \neq l\,.$$

Proof. Define the set $\mathcal{X}_*$ as follows:

$$\mathcal{X}_* = \left\{\mathbf{x} \in L(\mathbf{v}_0^0) : \|\mathbf{x} - \mathbf{x}_*\| < \frac{\epsilon}{2} \text{ for some } \mathbf{x}_* \in X_*\right\}.$$

Consider its complement Ω in $L(\mathbf{v}_0^0)$:

$$\Omega = L(\mathbf{v}_0^0) \setminus \mathcal{X}_*.$$

Note that Ω is compact. Furthermore, f is continuously differentiable on Ω, so

1. $\|\nabla f\|$ achieves a minimum $\sigma > 0$ on Ω, and
2. ∇f is uniformly continuous on Ω.

This leads to two observations. First, for all k,

$$\left\|\nabla f(\mathbf{v}_0^k)\right\| \geq \sigma > 0\,,$$

and σ does not depend on k. Second, the uniform linear independence of the set of search directions at each iteration gives us the constant $\gamma > 0$ seen in (1). From the uniform continuity of ∇f on Ω there exists a $\delta > 0$, depending only on σ, γ, and Ω, such that

$$\left\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{v}_0^k)\right\| < \sigma\gamma/2 \quad \text{whenever} \quad \left\|\mathbf{x} - \mathbf{v}_0^k\right\| < \delta \quad (\text{and } \mathbf{x} \in \Omega)\,.$$

Now we will show that once the edges of the simplex become "small enough," the rotation step is always acceptable so that the simplex will not contract; hence the simplex cannot become any smaller.

"Small enough" will mean that the simplex and its rotation are contained in Ω and $\|\mathbf{v}_j^k - \mathbf{v}_0^k\| < \delta$ for any choice of $j = 1, \cdots, n$. These conditions will determine our choice of m.

We need first some measure of the relative lengths of the edges in the simplex. We define η as follows:

$$\eta = \frac{e^k}{E^k}\,,$$

where

$$e^k = \min_{0 \leq j,l \leq n\,,\ j \neq l} \left\|\mathbf{v}_j^k - \mathbf{v}_l^k\right\|, \quad \text{and} \quad E^k = \max_{0 \leq j,l \leq n\,,\ j \neq l} \left\|\mathbf{v}_j^k - \mathbf{v}_l^k\right\|.$$

Thus, for any $1 \leq j, l \leq n$,

$$\left\|\mathbf{v}_j^k - \mathbf{v}_0^k\right\| \leq E^k = \frac{1}{\eta} e^k \leq \frac{1}{\eta}\left\|\mathbf{v}_l^k - \mathbf{v}_0^k\right\|.$$

Note that η is independent of k since, as we demonstrated in Lemma 4.2, the relative lengths of the edges in the simplex remain fixed across all iterations of the algorithm. Finally, observe that $0 < \eta \leq 1$.

We also define d to be the distance between the contour defined by $\mathbf{v}_0^0$ and the level set defined by $\mathbf{v}_0^1$:

$$d = \text{dist}\left(L(\mathbf{v}_0^1), C(\mathbf{v}_0^0)\right).$$

Note that Lemma 4.1, together with the assumption that the best vertices are uniformly bounded away from X_*, assures us that $d > 0$. Finally, we use S_k to denote the simplex defined by the vertices $\langle \mathbf{v}_0^k, \mathbf{v}_1^k, \cdots, \mathbf{v}_n^k \rangle$. We now make the following two claims.

CLAIM 1. *Suppose $k \geq 1$. If, for some $i = 1, \cdots, n$, $\|\mathbf{v}_i^k - \mathbf{v}_0^k\| \leq \eta d$, then $S_k \subset L(\mathbf{v}_0^0)$.*

Proof. The triangle inequality gives us

$$\begin{aligned} \text{dist}\left(\mathbf{v}_j^k, L(\mathbf{v}_0^1)\right) &\leq \text{dist}\left(\mathbf{v}_j^k, L(\mathbf{v}_0^k)\right) + \text{dist}\left(L(\mathbf{v}_0^k), L(\mathbf{v}_0^1)\right) \\ &= \text{dist}\left(\mathbf{v}_j^k, L(\mathbf{v}_0^k)\right) \\ &\leq \|\mathbf{v}_j^k - \mathbf{v}_0^k\| \\ &\leq \frac{1}{\eta}\|\mathbf{v}_i^k - \mathbf{v}_0^k\| \\ &\leq d, \end{aligned}$$

for *any* choice of $j = 1, \cdots, n$. Therefore, $S_k \in L(\mathbf{v}_0^0)$ as required. $\square$

CLAIM 2. *If, for some $i = 1, \cdots, n$, $\|\mathbf{v}_i^k - \mathbf{v}_0^k\| \leq \eta\frac{\epsilon}{3}$, then none of the vertices in the rotated simplex are contained in $\mathcal{X}_*$, i.e., $\mathbf{r}_j^k \notin \mathcal{X}_*$ for any $j = 1, \cdots, n$.*

Proof. Recall that $\|\mathbf{r}_j^k - \mathbf{v}_0^k\| = \|\mathbf{v}_j^k - \mathbf{v}_0^k\|$ for any $j = 1, \cdots, n$.

Let $\mathbf{x}_* \in X_*$. We appeal to the "reverse" form of the triangle inequality to obtain

$$\|\mathbf{r}_j^k - \mathbf{x}_*\| \geq \left|\, \underbrace{\|\mathbf{r}_j^k - \mathbf{v}_0^k\|}_{\leq \frac{\epsilon}{3}} - \underbrace{\|\mathbf{v}_0^k - \mathbf{x}_*\|}_{\geq \epsilon} \,\right| > \frac{\epsilon}{2}. \quad \square$$

We define a as follows:

$$a = \eta \min\left\{d, \frac{\epsilon}{3}, \delta\right\}.$$

We are now assured that if

$$\max_{j=1,\cdots,n} \|\mathbf{v}_j^k - \mathbf{v}_0^k\| \leq a, \tag{2}$$

then S_k and its rotation are contained in Ω and that $\|\nabla f(\mathbf{v}_j^k) - \nabla f(\mathbf{v}_0^k)\| < \sigma\gamma/2$ for all $j = 1, \cdots, n$. We are ready to argue that if, at any iteration k, (2) is satisfied, then the rotation step will always be acceptable. Again, we have two cases to consider.

Case 1. $\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) > 0$.

Choose a vertex $\mathbf{v}_i^k$, $i \neq 0$, which satisfies

$$\frac{\left|\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k)\right|}{\left\|\nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{v}_i^k - \mathbf{v}_0^k\right\|} \geq \gamma.$$

Lemma 4.2 guarantees the existence of at least one such vertex.

By definition,

$$\mathbf{r}_i^k = \mathbf{v}_0^k - (\mathbf{v}_i^k - \mathbf{v}_0^k) \quad \text{so} \quad -(\mathbf{r}_i^k - \mathbf{v}_0^k) = \mathbf{v}_i^k - \mathbf{v}_0^k\,.$$

Our choice of a guarantees that the edge $(\mathbf{r}_i^k - \mathbf{v}_0^k)$ is contained in Ω. We can then invoke the Mean Value Theorem to obtain

$$f(\mathbf{r}_i^k) - f(\mathbf{v}_0^k) = \nabla f(\xi)^T(\mathbf{r}_i^k - \mathbf{v}_0^k)$$

for some $\xi \in (\mathbf{r}_i^k - \mathbf{v}_0^k)$, whence

$$f(\mathbf{r}_i^k) - f(\mathbf{v}_0^k) = \nabla f(\mathbf{v}_0^k)^T(\mathbf{r}_i^k - \mathbf{v}_0^k) + \left(\nabla f(\xi) - \nabla f(\mathbf{v}_0^k)\right)^T(\mathbf{r}_i^k - \mathbf{v}_0^k). \tag{3}$$

Consider the first term on the right-hand side of (3), $\nabla f(\mathbf{v}_0^k)^T(\mathbf{r}_i^k - \mathbf{v}_0^k)$. Our choice of $\mathbf{v}_i^k$ gives us

$$\left|\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k)\right| \geq \gamma \left\|\nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{v}_i^k - \mathbf{v}_0^k\right\|.$$

Since $\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) > 0$ we have

$$\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) \geq \gamma \left\|\nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{v}_i^k - \mathbf{v}_0^k\right\|.$$

By construction, $\left\|\mathbf{r}_i^k - \mathbf{v}_0^k\right\| = \left\|\mathbf{v}_i^k - \mathbf{v}_0^k\right\|$. Thus,

$$\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) \geq \gamma \left\|\nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{r}_i^k - \mathbf{v}_0^k\right\|.$$

Since $\mathbf{v}_i^k - \mathbf{v}_0^k = -(\mathbf{r}_i^k - \mathbf{v}_0^k)$, we obtain

$$\nabla f(\mathbf{v}_0^k)^T(\mathbf{r}_i^k - \mathbf{v}_0^k) \leq -\gamma \left\|\nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{r}_i^k - \mathbf{v}_0^k\right\|. \tag{4}$$

Now consider the second term on the right-hand side of (3). The Cauchy–Schwarz inequality gives us

$$\left|\left(\nabla f(\xi) - \nabla f(\mathbf{v}_0^k)\right)^T\left(\mathbf{r}_i^k - \mathbf{v}_0^k\right)\right| \leq \left\|\nabla f(\xi) - \nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{r}_i^k - \mathbf{v}_0^k\right\|. \tag{5}$$

Combine (4) and (5) to rewrite (3) as

$$f(\mathbf{r}_i^k) - f(\mathbf{v}_0^k) \leq -\gamma \left\|\nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{r}_i^k - \mathbf{v}_0^k\right\| + \left\|\nabla f(\xi) - \nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{r}_i^k - \mathbf{v}_0^k\right\|.$$

Since ∇f is uniformly continuous on Ω and $\|\xi - \mathbf{v}_0^k\| \leq \|\mathbf{r}_i^k - \mathbf{v}_0^k\| = \|\mathbf{v}_i^k - \mathbf{v}_0^k\| \leq a$, the following holds:

$$f(\mathbf{r}_i^k) - f(\mathbf{v}_0^k) \leq \underbrace{\underbrace{\underbrace{\left(-\gamma \underbrace{\left\|\nabla f(\mathbf{v}_0^k)\right\|}_{\geq\sigma}\right.}_{\leq -\sigma\gamma} + \underbrace{\left\|\nabla f(\xi) - \nabla f(\mathbf{v}_0^k)\right\|}_{< \frac{\sigma\gamma}{2}}\left.\right)}_{< -\frac{\sigma\gamma}{2} < 0} \underbrace{\left\|\mathbf{r}_i^k - \mathbf{v}_0^k\right\|}_{>0}}_{<0}.$$

Thus,

$$f(\mathbf{r}_i^k) - f(\mathbf{v}_0^k) < 0 \Longrightarrow f(\mathbf{r}_i^k) < f(\mathbf{v}_0^k)\,,$$

whenever $\max_{j=1,\cdots,n} \|\mathbf{v}_j^k - \mathbf{v}_0^k\| \leq a$. We can then conclude that the vertex $\mathbf{r}_i^k$ can replace the best vertex $\mathbf{v}_0^k$. Therefore, the simplex will not contract.

Case 2. $\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) < 0$.

The proof for Case 2 follows that given for Case 1. The conclusion, however, is quite different. Since $\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) < 0$, we get, as in (4),

$$\nabla f(\mathbf{v}_0^k)^T(\mathbf{v}_i^k - \mathbf{v}_0^k) \leq -\gamma \left\|\nabla f(\mathbf{v}_0^k)\right\| \left\|\mathbf{v}_i^k - \mathbf{v}_0^k\right\|,$$

without substituting $-(\mathbf{r}_i^k - \mathbf{v}_0^k)$ for $\mathbf{v}_i^k - \mathbf{v}_0^k$. We continue the argument as in (5) but with $\mathbf{v}_i^k$ rather than with $\mathbf{r}_i^k$. Finishing the reductions, we are left with

$$f(\mathbf{v}_i^k) < f(\mathbf{v}_0^k)$$

whenever $\max_{j=1,\cdots,n} \|\mathbf{v}_j^k - \mathbf{v}_0^k\| \leq a$. But this leads to a contradiction! We have chosen $\mathbf{v}_i^k$ so that $i \neq 0$ and the multidirectional search algorithm requires that

$$f(\mathbf{v}_0^k) \leq f(\mathbf{v}_i^k) \qquad \forall\, i = 1, \cdots, n\,.$$

Therefore, Case 2 cannot happen.

Conclusion. We have now established that once every edge of the simplex has length less than a, then the rotation step will always be acceptable. Since the algorithm reduces the lengths of the edges of the simplex by at most θ at any iteration of the inner **repeat** loop, it follows that the length of every edge of the simplex will be between $\eta\theta a$ and a. Thus we choose $m = \eta\theta a$. $\square$

We emphasize that the existence of a lower bound for the lengths of the edges in the simplex is guaranteed *only* under the null hypothesis that the best vertices are uniformly bounded away from X_*. However, now that we have established both lower and upper bounds for the lengths of the edges in the simplex, we are ready to show that, given these lower and upper bounds, the multidirectional search algorithm can only visit a finite number of points. Again, let us stress that this is not what occurs, but is rather a consequence of the hypothesis that the best vertices are uniformly bounded away from X_*.

To see how the argument works, we begin by considering how the algorithm decides which points to visit. In Fig. 1 we are given an initial simplex with a designated "best" vertex. The algorithm automatically generates the rotated simplex. The result of the acceptance test then dictates whether the expanded or the contracted simplex will be constructed. In either event, given an initial simplex and the best vertex in that simplex, we can determine, in advance, all the simplices that can be generated during the first iteration of the inner **repeat** loop. This means we can enumerate, a priori, all the points that can possibly be visited during the first iteration through the **repeat** loop.

Now assume that we are given an initial simplex, but do not know any information about the function values at any of the vertices in the simplex. This means that we have no way of identifying the "best" vertex in the simplex. Even without this information, we can still determine all the simplices that might be generated during the first iteration of the inner **repeat** loop and we can still enumerate all the points that might be visited during this iteration. To do this, we simply allow each of the vertices in the original simplex to be "best" and consider all the possibilities, as shown in Fig. 2.

We now extend this speculation further. After one pass through the inner **repeat** loop there are two possibilities: either the best vertex has been replaced, and we go to the next iteration of the outer **while** loop, or the best vertex has not been replaced and

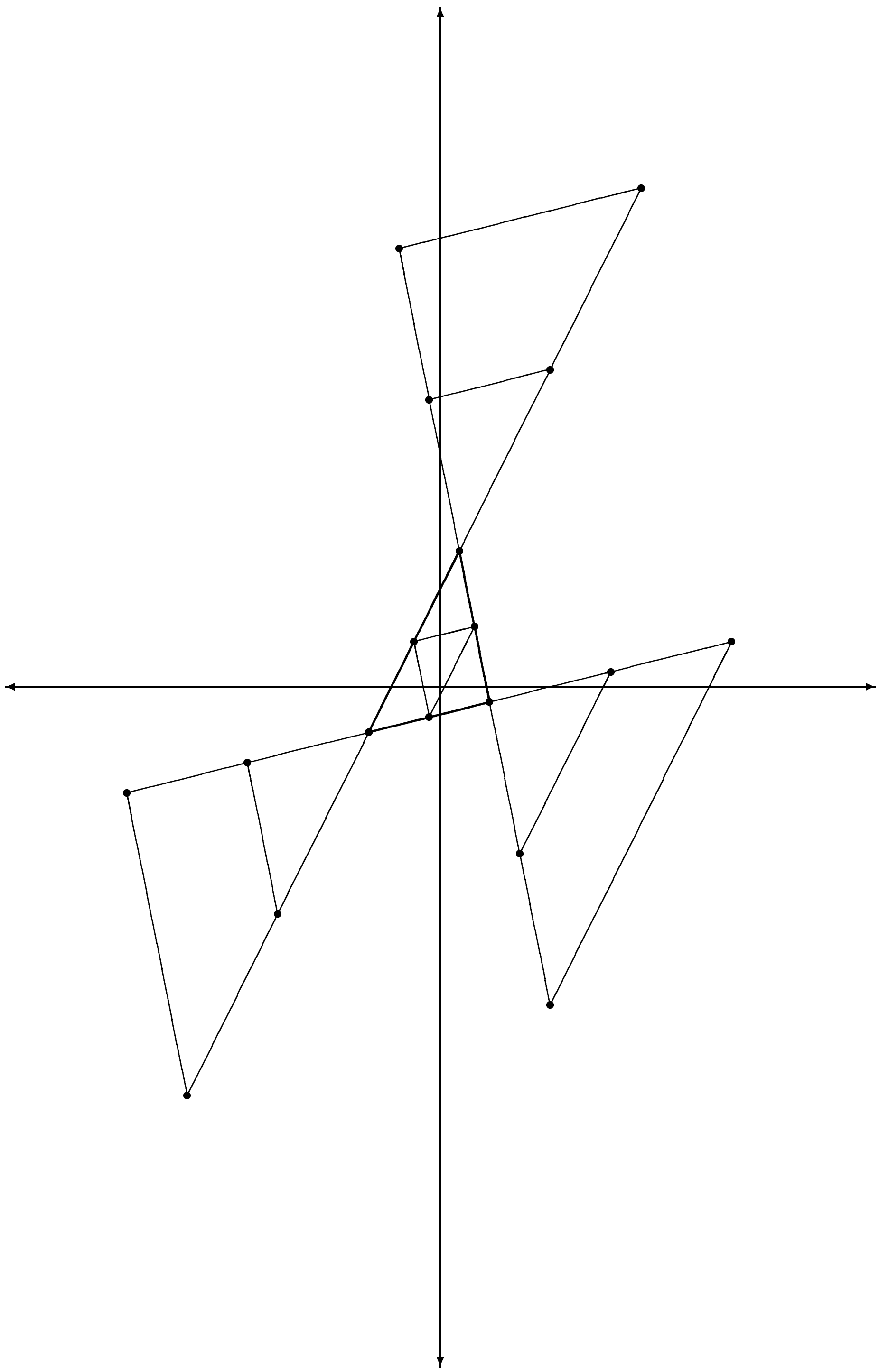

FIG. 2. *Enumerating the vertices when the best vertex is not known.*

we pass once more through the inner **repeat** loop. Again, we assume no knowledge of the function value at any of the vertices. We can still consider, a priori, all the possibilities by allowing each vertex, including the original best vertex, in each of the trial simplices generated at the previous iteration to be "best" and generate all possible new simplices. For our example, seen in Fig. 3, we begin to enforce a lower bound on the lengths of the edges in any of the simplices. Furthermore, we require our simplices to be contained in a compact set. Both of these restrictions are important because they eliminate several of the simplices that might otherwise have been considered.

Consider yet another iteration of either the outer **while** loop or the inner **repeat** loop. Again, we allow each vertex in each of the trial simplices generated at the previous iteration to be "best," but again we apply our restrictions to eliminate even more simplices. The result can be seen in Fig. 4.

Finally, if we remove all the edges, we see in Fig. 5 that the multidirectional search algorithm is, in fact, generating a grid. Since the grid must be contained in a compact set, and since its mesh size is fixed, this means that there are only a finite number of points that the algorithm can visit. We require the lower bound on the lengths of the edges in the simplex to fix the mesh size of the grid. We require the upper bound on the lengths of the edges in the simplex to give us a compact set over which to search. However, once we accept these two restrictions, this means that we can compute—without any knowledge of function information—all the points the algorithm can possibly visit from any given initial simplex.

Now we will prove that, given any initial simplex, if we assume that the best vertices are uniformly bounded away from the set X_*, then the multidirectional search algorithm can visit only a finite number of points.

PROPOSITION 5.5. *Assume that $L(\mathbf{v}_0^0)$ is compact and that f is continuously differentiable on $L(\mathbf{v}_0^0)$. Further, suppose that the best vertices stay uniformly bounded away from X_*, i.e., for a fixed $\epsilon > 0$, independent of k,*

$$\left\| \mathbf{v}_0^k - \mathbf{x}_* \right\| \geq \epsilon \qquad \forall\, \mathbf{x}_* \in X_*\,, \qquad \forall\, k \geq 0\,.$$

Then the multidirectional search algorithm can visit only a finite number of points.

Proof. The proof is by construction.

Rescale the initial simplex S_0, using the contraction factor θ, until every edge in the simplex satisfies the condition

$$\eta\theta a \leq \left\| \mathbf{v}_j^0 - \mathbf{v}_l^0 \right\| \leq a\,,$$

for all $0 \leq j, l \leq n$, $j \neq l$, where η and a are as defined in Proposition 5.4.

Find the least common denominator for the scale factors θ and μ. This makes sense since θ and μ are restricted to the set of rational numbers.

Divide the least common denominator by the reduction factor θ. Reduce the simplex one last time by this factor.

Designate any one of the vertices in the rescaled simplex as "best." (There is no need to consider function information.) Take the set of edges adjacent to the best vertex

$$\left\{ (\mathbf{v}_i^0 - \mathbf{v}_0^0) \ : \ i = 1, \cdots, n \right\}$$

as a basis for the grid. Now take all *integer* multiples of the basis that generate points inside the compact set $\mathcal{M}$. (Recall that Proposition 5.3 implies the existence of a compact set $\mathcal{M}$ which contains all the simplices generated from S_0 by the multidirectional search algorithm.)

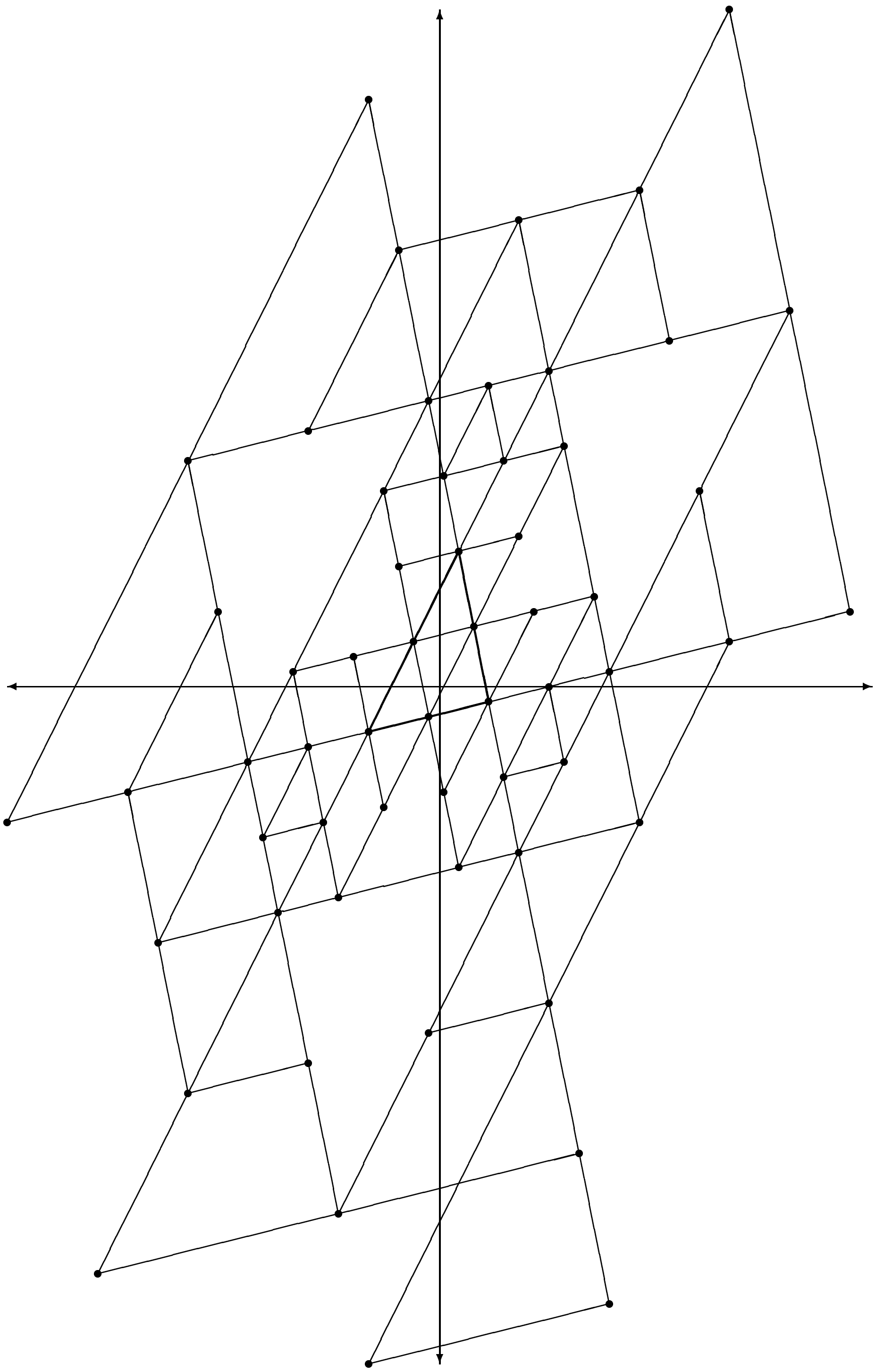

FIG. 3. *Enumerating the vertices after one additional iteration.*

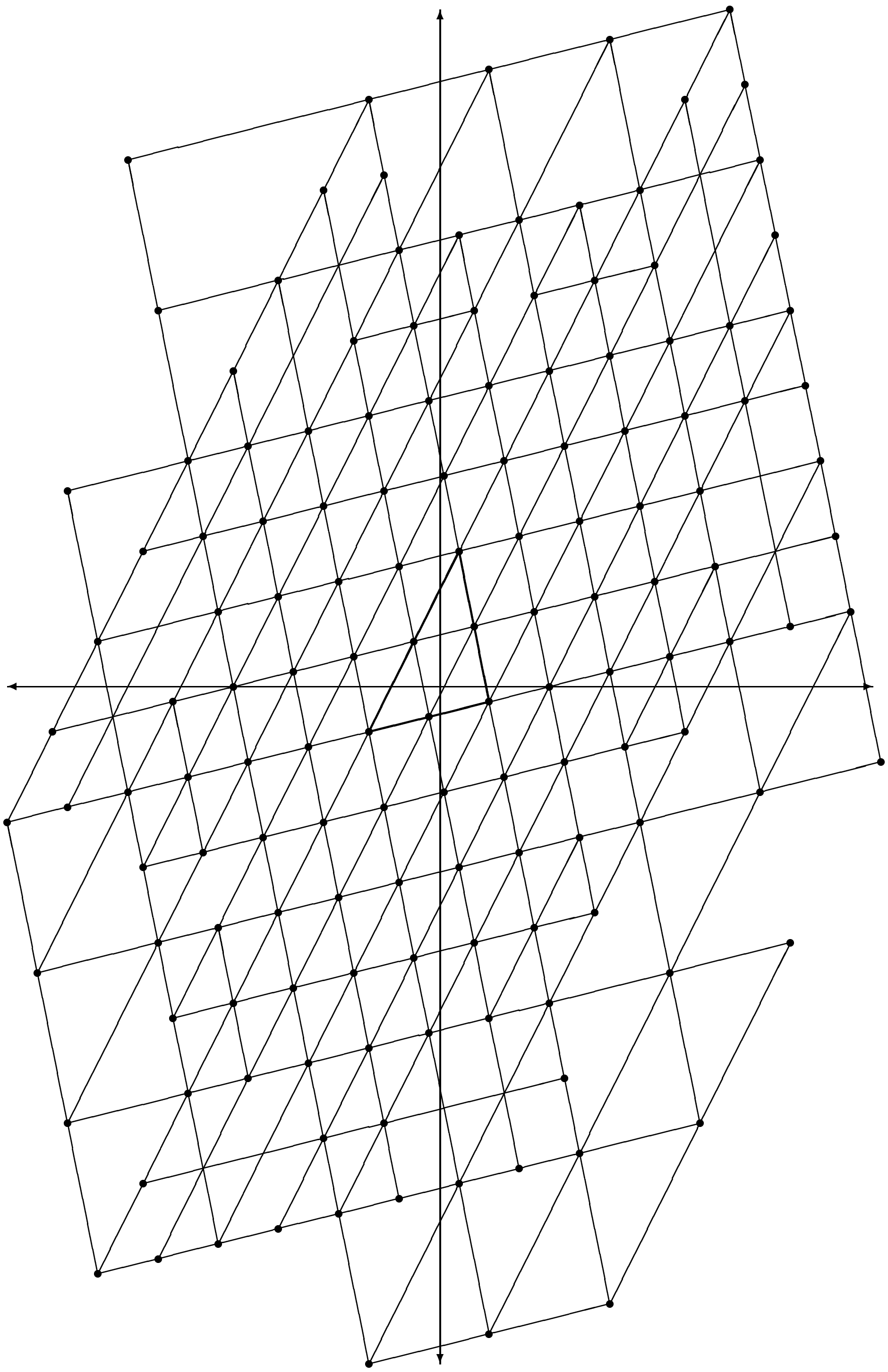

FIG. 4. *Enumerating the vertices after two additional iterations.*

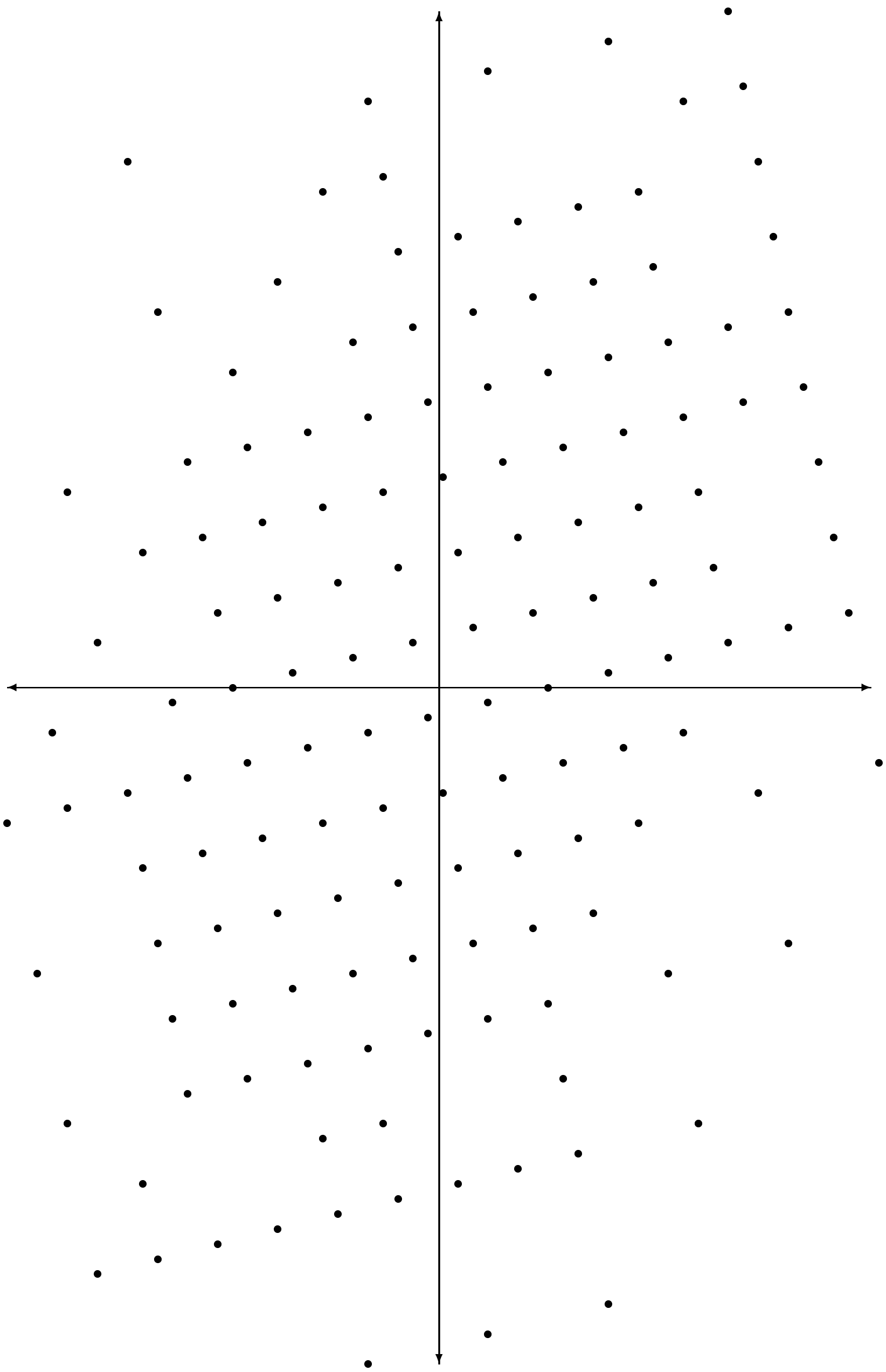

FIG. 5. *Enumerating the vertices after removing all the edges.*

This will give a grid with fixed mesh size inside a compact set. Thus the number of points in the grid will be finite. Furthermore, every possible simplex, given the initial simplex S_0, will be mapped onto the grid since all possible step sizes are integer multiples of the mesh size.

Therefore, the multidirectional search algorithm can visit only a finite number of points. □

5.2. Technical results. We now introduce one last technical result before giving the proof of Theorem 5.1.

Lemma 4.1 shows that the multidirectional search algorithm is a descent method with a strictly monotonically decreasing sequence of function values at the best vertices. The following proposition and its corollary demonstrate that this sequence of function values converges and that the sequence of best vertices converges to the corresponding level set.

PROPOSITION 5.6. *Assume that* $\{\mathbf{x}^k\}$ *is a sequence contained in a compact set* $\mathcal{S}$, *that* f *is continuous, and that* $\{f(\mathbf{x}^k)\}$ *is a monotone nonincreasing sequence. Then there is some* $\hat{f}$ *such that*

$$\lim_{k\to\infty} f(\mathbf{x}^k) = \hat{f}\,.$$

Furthermore, assume that $\hat{\mathbf{x}}$ *is any limit point of* $\{\mathbf{x}^k\}$. *Then* $f(\hat{\mathbf{x}}) = \hat{f}$ *and* $\{\mathbf{x}^k\}$ *converges to* $C(\hat{\mathbf{x}})$ *in the sense that*

$$\lim_{k\to\infty}\left[\inf_{\mathbf{y}\in C(\hat{\mathbf{x}})} \left\|\mathbf{x}^k - \mathbf{y}\right\|\right] = 0\,.$$

Proof. The proof of the first statement is immediate: $\hat{f}$ exists because $\{f(\mathbf{x}^k)\}$ is a monotone nonincreasing sequence that is bounded below.

To prove the second statement, consider the following: since $\mathcal{S}$ is compact, $\{\mathbf{x}^k\}$ contains a convergent subsequence. We will denote this convergent subsequence by $\{\mathbf{x}^{k_i}\}$ and say that $\mathbf{x}^{k_i} \to \hat{\mathbf{x}}$. By continuity, $f(\hat{\mathbf{x}}) = \lim_{i\to\infty} f(\mathbf{x}^{k_i}) = \hat{f}$.

Define

$$\delta_k = \inf_{\mathbf{y}\in C(\hat{\mathbf{x}})} \left\|\mathbf{x}^k - \mathbf{y}\right\|.$$

But $\lim_{i\to\infty} \delta_{k_i} = 0$, which implies that

$$\liminf_{k\to\infty} \delta_k = 0\,.$$

Next, suppose that $\{\delta_{k_i}\}$ is *any* convergent subsequence of $\{\delta_k\}$. There is a corresponding sequence of $\{\mathbf{x}^{k_i}\}$. This sequence has a convergent subsequence which we denote also by $\{\mathbf{x}^{k_i}\}$ and say that $\mathbf{x}^{k_i} \to \check{\mathbf{x}}$. Again, $f(\check{\mathbf{x}}) = \lim_{i\to\infty} f(\mathbf{x}^{k_i}) = \hat{f}$, so $\check{\mathbf{x}} \in C(\hat{\mathbf{x}})$. Thus, $\delta_{k_i} \to 0$. Hence, the only possible accumulation points of $\{\delta_k\}$ are 0 and ∞. Note that since $\mathcal{S}$ is compact, $\{\delta_k\}$ is bounded, i.e., there exists a $b \geq 0$ such that $0 \leq \delta_k \leq b$. Thus,

$$\limsup_{k\to\infty} \delta_k = 0\,.$$

Since $\liminf_{k\to\infty} \delta_k = \limsup_{k\to\infty} \delta_k = 0$,

$$\lim_{k\to\infty} \delta_k = 0$$

as required. □

COROLLARY 5.7. *Assume that $L(\mathbf{v}_0^0)$ is compact and that f is continuously differentiable on $L(\mathbf{v}_0^0)$. Then for the sequence of best vertices $\mathbf{v}_0^k$ generated by the multidirectional search algorithm there is some $\hat{f}$ such that*

$$\lim_{k \to \infty} f(\mathbf{v}_0^k) = \hat{f}\,.$$

Furthermore, assume that $\hat{\mathbf{x}}$ is any limit point of $\{\mathbf{v}_0^k\}$. Then $f(\hat{\mathbf{x}}) = \hat{f}$ and $\{\mathbf{v}_0^k\}$ converges to $C(\hat{\mathbf{x}})$ in the sense that

$$\lim_{k \to \infty} \left[\inf_{\mathbf{x} \in C(\hat{\mathbf{x}})} \left\| \mathbf{v}_0^k - \mathbf{x} \right\| \right] = 0\,.$$

Proof. Lemma 4.1 established that $\{f(\mathbf{v}_0^k)\}$ is a monotone decreasing sequence. We appeal to Proposition 5.6 to complete the proof. □

6. Proof of convergence. We have established that the multidirectional search algorithm is a descent method. In addition, we can guarantee first, that the search directions will not deteriorate, and second, that the steps taken by the algorithm cannot become too long or too short. Finally, we have shown that the sequence of function values at the best vertices converges and the sequence of best vertices converges to the corresponding level set. Thus, we are now ready to prove Theorem 5.1.

Proof. The proof is by contradiction.

Suppose that for all but finitely many k there exists a fixed constant $\epsilon > 0$, which does not depend on k, such that

$$\left\| \mathbf{v}_0^k - \mathbf{x}_* \right\| \geq \epsilon \qquad \forall\ \mathbf{x}_* \in X_*\,.$$

Then, taken together, the upper and lower bounds on the lengths of the edges in the simplex (Propositions 5.3 and 5.4) imply that the algorithm can visit only a finite number of points (Proposition 5.5). But this contradicts Lemma 4.1, which guarantees *strict* decrease in the function values at the best vertex in a finite number of iterations. Thus, the hypothesis cannot hold, which means that

$$\liminf_{k \to \infty} \left\| \mathbf{v}_0^k - \mathbf{x}_* \right\| = 0\,.$$

Then there exists some subsequence of the best vertices $\{\mathbf{v}_0^k\}$ that converges to a point $\mathbf{x}_* \in X_*$.

We invoke Corollary 5.7 to complete the proof. □

7. The nonsmooth case. To extend Theorem 5.1 to handle most cases when the function f is nondifferentiable, we need only modify the set X_*.

Let X_* include the set of stationary points of the function f in $L(\mathbf{v}_0^0)$, the set of all points in $L(\mathbf{v}_0^0)$ where f is nondifferentiable, and the set of all points in $L(\mathbf{v}_0^0)$ where the derivative of f exists but is not continuous. Our construction of the set Ω in the proof of Proposition 5.4 then guarantees that f is continuously differentiable on Ω. We can now replace the assumption that f is continuously differentiable on $L(\mathbf{v}_0^0)$ with the assumption that f is continuous on $L(\mathbf{v}_0^0)$. Since the proofs given in the previous sections were constructed to require only that ∇f be continuous on Ω, the results can be extended, without modification, to cover this new characterization of X_*.

We can state the result as follows.

THEOREM 7.1. *Assume that $L(\mathbf{v}_0^0)$ is compact and that f is continuous on $L(\mathbf{v}_0^0)$. Then some subsequence of $\{\mathbf{v}_0^k\}$ converges to a point $\mathbf{x}_* \in X_*$. Furthermore, $\{\mathbf{v}_0^k\}$ converges to $C_* = C(\mathbf{x}_*)$ in the sense that*

$$\lim_{k\to\infty}\left[\inf_{\mathbf{x}\in C_*}\left\|\mathbf{v}_0^k - \mathbf{x}\right\|\right] = 0 .$$

Proof. The proof follows directly from the proof of Theorem 5.1. □

This is not a general result for the nonsmooth case; rather, it is an extension of the result for the smooth case. The proof of Proposition 5.4 requires that the derivative be continuous wherever it exists to give us the uniform continuity of ∇f on the set Ω. The constant we derive from the uniform continuity of ∇f on Ω plays a key part in deriving a lower bound on the lengths of the edges in the simplex. Thus X_* must include the set of all points in $L(\mathbf{v}_0^0)$ where the derivative of f exists but is not continuous.

While this restriction is unfortunate, it does not prevent us from considering most nonsmooth cases of practical interest. For instance, it means that our theory can handle the following example constructed by Dennis and Woods [6]. Consider the strictly convex function $f(\mathbf{x}) = \frac{1}{2}\max\left\{\|\mathbf{x}-c_1\|^2, \|\mathbf{x}-c_2\|^2\right\}$, whose level sets are shown in Fig. 6, where $c_1 = (0,32)^T$ and $c_2 = (0,-32)^T$.

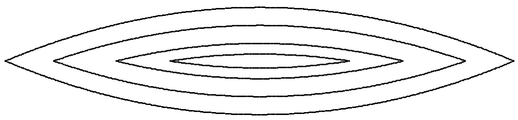

FIG. 6. *Level sets for the Dennis–Woods function.*

Our numerical experience has shown that, given any initial simplex, the multidirectional search algorithm converges unfailingly to a point of the form $(\alpha,0)^T$ but not necessarily to the minimizer $(0,0)^T$. Theorem 7.1 confirms that convergence to a critical point (i.e., either a point where the function is nondifferentiable or where the gradient is equal to zero) is the best that can be expected for this example.

8. Conclusions. The convergence analysis for the multidirectional search algorithm actually explains how—and why—the algorithm works. As we have seen, the fixed search directions and fixed rescaling factors used to determine the step sizes allow us to construct a grid underlying the progress of the algorithm. We should note, however, that once we remove the null hypothesis, which required the sequence of best vertices to be uniformly bounded away from the set X_*, we no longer have a lower bound on the lengths of the edges in the simplex. Thus, the algorithm allows for further and further grid refinement until a satisfactory solution is obtained.

Furthermore, once we add function information, we eliminate many of the points the algorithm might otherwise visit. This means that while there is an implicit grid structure underlying the search, the algorithm by no means visits every point on the grid. Rather, it uses function information to prune the number of grid points that are considered—and to dictate when the grid must be refined further.

The simplex that is used to start the multidirectional search algorithm determines the shape of the grid underlying the search. However, we could just as easily have used another *fixed* "shape," or "pattern," to determine the shape of the underlying grid. For example, the factorial design algorithm of Box [2], in its simplest form, constructs a hypercube centered at the current iterate and then computes the function values at the vertices in an effort to find a new iterate with a function value that is strictly less than the function value at the current iterate. If a new best point is found, the hypercube is then centered on the new best iterate and the search is restarted. If not, the size of the hypercube is reduced by halving the lengths of the edges. Again we have a pattern for the grid underlying the search. We also have all the necessary ingredients to argue for convergence of the method: fixed search directions, a fixed rescaling factor, and strict decrease in the function value at the sequence of best points.

We should be able to use the arguments presented here to analyze any algorithm that maintains this sort of underlying grid structure. This is the subject of our current research. There is, however, one additional caveat that should be added to the above discussion. The number of points required by the simplex "pattern" of the multidirectional search algorithm is $O(n)$. For the algorithms that do not use a simplex pattern, the number of points required to guarantee convergence, though not necessarily to execute the algorithm on a sequential machine, is exponential in n. (For example, the hypercube constructed in the factorial design algorithm of Box requires $O(2^n)$ new points at each iteration.) All of these algorithms share the same convergence analysis and yet the number of points required by the multidirectional search algorithm is linear, rather than exponential, in the size of the problem—which is why we believe the multidirectional search algorithm to be the most practical of these algorithms to implement on a parallel machine.

REFERENCES

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

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

[3] J. CÉA, *Optimisation : théorie et algorithmes*, Dunod, Paris, 1971.

[4] J. E. DENNIS, JR. AND R. B. SCHNABEL, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[5] J. E. DENNIS, JR. AND V. TORCZON, *Direct search methods on parallel machines*, Tech. Report 90-19, Department of Mathematical Sciences, Rice University, Houston, TX 77251–1892, 1990.

[6] J. E. DENNIS, JR. AND D. J. WOODS, *Optimization on microcomputers: The Nelder–Mead simplex algorithm*, in New Computing Environments: Microcomputers in Large-Scale Computing, A. Wouk, ed., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1987, pp. 116–122.

[7] R. HOOKE AND T. A. JEEVES, *"Direct search" solution of numerical and statistical problems*, J. Assoc. Comput. Mach., 8 (1961), pp. 212–229.

[8] J. A. NELDER AND R. MEAD, *A simplex method for function minimization*, Comput. J., 7 (1965), pp. 308–313.

[9] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[10] M. J. D. POWELL, *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Comput. J., 7 (1964), pp. 155–162.

[11] H. H. ROSENBROCK, *An automatic method for finding the greatest or least value of a function*, Comput. J., 3 (1960), pp. 175–184.

[12] W. SPENDLEY, G. R. HEXT, AND F. R. HIMSWORTH, *Sequential application of simplex designs in optimisation and evolutionary operation*, Technometrics, 4 (1962), pp. 441–461.

[13] V. TORCZON, *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*,

Ph.D. thesis, Department of Mathematical Sciences, Rice University, Houston, TX, 1989; also available as Tech. Report 90-7, Department of Mathematical Sciences, Rice University, Houston, TX 77251-1892.

[14] Y. WEN-CI, *The convergence property of the simplex evolutionary techniques*, Scientia Sinica, Special Issue of Mathematics, 1 (1979), pp. 68–77.

[15] ———, *Positive basis and a class of direct search techniques*, Scientia Sinica, Special Issue of Mathematics, 1 (1979), pp. 53–67.

[16] W. I. ZANGWILL, *Minimizing a function without calculating derivatives*, Comput. J., 10 (1967), pp. 293–296.