SIAM J. OPTIM.
Vol. 13, No. 1, pp. 79–96

NEW SEQUENTIAL AND PARALLEL DERIVATIVE-FREE ALGORITHMS FOR UNCONSTRAINED MINIMIZATION*

U. M. GARCÍA-PALOMARES† AND J. F. RODRÍGUEZ‡

Abstract. This paper presents sequential and parallel derivative-free algorithms for finding a local minimum of smooth and nonsmooth functions of practical interest. It is proved that, under mild assumptions, a sufficient decrease condition holds for a nonsmooth function. Based on this property, the algorithms explore a set of search directions and move to a point with a sufficiently lower functional value. If the function is strictly differentiable at its limit points, a (sub)sequence of points generated by the algorithm converges to a first-order stationary point ($\nabla f(x) = 0$). If the function is convex around its limit points, convergence (of a subsequence) to a point with nonnegative directional derivatives on a set of search directions is ensured. Preliminary numerical results on sequential algorithms show that they compare favorably with the recently introduced pattern search methods.

Key words. nonsmooth function, unconstrained minimization, derivative-free algorithm, parallel algorithms, necessary and sufficient conditions

AMS subject classifications. 49D30, 65K05

PII. S1052623400370606

1. Introduction.

We are concerned with the problem of obtaining an unconstrained *local* minimizer and a *local* minimum of a nonsmooth functional $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$. More specifically, we look for the values of the variables $x^1, \ldots, x^n$ in the whole Euclidean space $\mathbb{R}^n$, where $f(x^1, \ldots, x^n)$ attains a *local* minimum value. We recall that penalization and Lagrange techniques are usually applied to transform a constrained minimization problem into an unconstrained and/or box constrained problem. Hence, the efficient solution of unconstrained problems is of broad interest.

The algorithms proposed here use only function values, but we are aware that, when first and/or second derivatives are available, Newton-related methods are highly efficient. Nonetheless, real world applications in many cases preclude the use of derivatives, because the functional values may arise either from a complex simulation package or from inaccurate sample values. Furthermore, a numerical approximation of the derivatives is not always a reliable approach. Therefore, practitioners require efficient derivative-free methods. Old algorithms, developed mainly in the late '60s and early '70s, had a strong intuitive approach and often lacked a convergence theory. (The interested reader can examine these methods in many optimization books, such as [4, 17, 44].) Other recent approaches are reported in [8, 11, 43]. The simplex method *(a simplex in $\mathbb{R}^n$ is the convex hull of $n+1$ points $x_0, \ldots, x_n$)* [27] has become, due perhaps to its simplicity and success in the solution of practical problems with a small number of variables, the most widely used and cited in the literature of unconstrained minimization. Nonetheless, it can fail on small problems, and convergence to a nonstationary point may occur [25, 39]. We are just starting to understand the properties of the simplex method [20], which has triggered active research on derivative-free meth-

*Received by the editors October 27, 2000; accepted for publication (in revised form) November 30, 2001; published electronically June 5, 2002. This research was partially supported by our respective Departments and the *Decanato de Investigación y Desarrollo* of the Universidad Simón Bolívar.
http://www.siam.org/journals/siopt/13-1/37060.html

†Universidad Simón Bolívar, Departamento Procesos y Sistemas, Apdo 89000, Caracas 1080-A, Venezuela (garciap@usb.ve).

‡Departamento Mecánica, Apdo 89000, Caracas 1080-A, Venezuela (jrodri@usb.ve).

ods in the last decade. Related *simplicial* methods with a formal convergence theory have appeared in [9, 18, 36, 39, 40, 42]. Convergence of well-known derivative-free methods and *pattern search methods* have been analyzed in [3, 21, 41]. Essentially, a pattern search method examines the function values on a *nondegenerate simplex*. An iteration starts with a new simplex which satisfies a form of descent condition for the function values. Under standard assumptions, convergence (of a subsequence) to a point satisfying the first-order necessary condition ($\nabla f(x) = 0$) of general smooth functions is ensured. A possible drawback of these methods is that function values must be obtained infinitely often at all vertices of a simplex before a new iteration starts, which implies at least n function evaluations per iteration. To circumvent this difficulty, it is desirable to establish at each iteration a decrease of the function value sufficient to guarantee convergence. An old effort in this direction is found in [12], and more recent attempts in [18, 24, 42]. Additional material can be found in [19, Chapters 6,7] and references therein.

While this paper was under review, the referees brought [3] to our attention, in which the convergence of *generalized pattern search* methods is ensured without assuming global continuity. Convergence relies on the differentiability properties of limit points. All other works on derivative-free methods cited above assume $f(\cdot) \in C^1$ and often ask for Lipschitz continuity of the gradients to ensure convergence. Therefore, the convergence theory cannot be applied in certain cases of practical interest:

(i) Some industrial problems often require the minimization of functions which arise from a complex simulation process or from sample values. Smoothness of the function cannot be guaranteed.

(ii) Common functions, like the norm function $f(x) = \|f_1(x)\|$ and the Max function $f(x) = \text{Max}(f_1(x), \ldots, f_m(x))$, may not be everywhere differentiable, even in the convex case.

(iii) Most exact penalty functions are not everywhere differentiable.

This paper has a twofold objective: (a) to define a practical necessary condition for a class of nonsmooth functions, which should be valid as well for smooth functions and readily allow (b) the implementation of converging algorithms. The rest of the paper is organized as follows. The next section states the assumptions C1–C3, needed to ensure that the algorithm is well defined, and the nonsmooth necessary condition (NSNC) (2.4). Section 3 introduces the sufficient decrease criterion (3.1)–(3.2), proposes sequential and parallel algorithms, and develops the convergence theory. It is shown that the algorithms are well defined under conditions C1–C3. Furthermore, convergence to a point x satisfying NSNC is shown if $f(\cdot)$ is strictly differentiable at x or convex in a neighborhood of x. Section 4 presents extensions to smooth functions and the box constrained problem. It also analyzes useful features that improve the algorithm notably. Section 5 shows preliminary numerical results with examples from the CUTE collection and the Rosenbrock function with two, three, five, or ten variables. The number of function evaluations of our sequential algorithms are in general lower than those needed by the pattern search methods (PSM) with the same termination criteria. Finally, we state our conclusions and final remarks in section 6. It is pointed out that no PSM ensures convergence to a minimum of a convex nonsmooth function.

We end this introduction with a note on notation: A sequence is denoted by $\{x_i\}_1^\infty$, and a subsequence by $\{x_{i_k}\}_{k=1}^\infty$. Sometimes we just denote a sequence by $\{x_i\}$ and use $y \to x$ to denote that $\{y_i\} \to x$. $\mathbb{R}^n$ is the Euclidean n-dimensional space.

Greek lowercase letters are scalars. Latin lowercase letters $i, \ldots, q$ denote integers; f is reserved to denote the functional $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$; and $o(\cdot) : \mathbb{R}_+ \to \mathbb{R}$ is a scalar function such that $\lim_{\eta \downarrow 0} \frac{o(\eta)}{\eta} = 0$. All other Latin lowercase letters represent vectors (points) in $\mathbb{R}^n$. Subindices represent different entities, and superindices components; for instance, y_i^k is the kth component of the vector y_i. The standard inner product in $\mathbb{R}^n$ is denoted by $x^T y \doteq \sum_{k=1}^n x^k y^k$, and xy^T is an $n \times n$ matrix with elements $x^i y^j$. The rest of the notation is standard.

2. Necessary condition for nonsmooth functions. B-differentiable functions, which were introduced in [34], have a directional derivative (B-derivative) $f'(\cdot, \cdot) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ that satisfies

$$[\eta > 0] \Rightarrow [f'(x, \eta d) = \eta f'(x, d)], \tag{2.1}$$

$$f(x + d) - f(x) - f'(x, d) = o(||d||). \tag{2.2}$$

In general, these functions are not Fréchet differentiable but appear naturally in many optimization problems. Some difficult smooth problems can be reformulated as nonsmooth problems with a simpler structure, which can be efficiently solved by suitably adapting Newton-related methods [29, 30, 33]. Moreover, necessary and sufficient conditions for nonlinear programming have been established for this kind of function [14, 16, 46]. Nonetheless, to the authors' knowledge there exists no direct search algorithm with guaranteed convergence to a local minimum of a B-differentiable function. We partially answer this question in this paper. We propose an algorithm that generates a subsequence $\{x_{i_k}\}_{k=1}^{\infty}$ that converges to a point x satisfying the NSNC given below, but if $f(\cdot)$ happens to be differentiable, then $\nabla f(x) = 0$.

In what follows we will assume the following conditions:

C1. $f(\cdot)$ is bounded below.

C2. For any $x, d \in \mathbb{R}^n$ there exists $f'(x, d) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ such that

$$[\eta > 0] \Rightarrow \left[\begin{array}{l} f'(x, \eta d) = \eta f'(x, d), \\ f(x + \eta d) - f(x) - \eta f'(x, d) = o(\eta) \end{array} \right]. \tag{2.3}$$

Note that $f'(x, d) = \lim_{\eta \downarrow 0} (f(x + \eta d) - f(x))/\eta$.

C3. The sequence $\{x_i\}_1^{\infty}$ remains in a compact set.

Condition C3 will be needed to merely ensure the existence of accumulation points of $\{x_i\}_1^{\infty}$. Conditions that imply C3 are, for instance,

(i) $f(\cdot)$ is coercive, i.e., $[\{\|x_i\|\} \to \infty] \Rightarrow [\{f(x_i)\} \to \infty]$, or

(ii) the lower level set $\{x \in \mathbb{R}^n : f(x) \leq f(x_1)\}$ is compact.

The next lemma follows a standard proof. It shows that C2 implies a well-known property of a local minimizer.

LEMMA 2.1. *Let* C2 *hold. If x is a local minimizer of $f(\cdot)$, then $f'(x, d) \geq 0$ for all $d \in \mathbb{R}^n$.*

Proof. If $f'(x, d) < 0$ for some $d \in \mathbb{R}^n$, there exists $\bar{\eta} > 0$ such that $o(\eta)/\eta \leq -f'(x, d)/2$ for all $0 < \eta \leq \bar{\eta}$. We now obtain from C2 that

$$f(x + \eta d) - f(x) = \eta(f'(x, d) + o(\eta)/\eta) \leq \eta \left(f'(x, d) - f'(x, d)/2\right) = \eta f'(x, d)/2 < 0,$$

and x is not a minimizer. ∎

The conclusion of the previous lemma is in general hard to verify unless $f(\cdot)$ is (sub)differentiable. We now state a practical NSNC, which will be helpful as well in constrained minimization on subspaces.

Nonsmooth necessary condition (NSNC). Let $x \in \mathbb{R}^n$ be a local minimizer of $f(\cdot)$ on a subspace S, and let $\mathcal{D} = \{d_1, \ldots, d_m\}$ be a set of m bounded nonzero directions in $\mathbb{R}^n$ that spans S. If C2 holds, then x satisfies the NSNC

$$[d \in \mathcal{D}] \Rightarrow [f'(x,d) \geq 0, f'(x,-d) \geq 0]. \tag{2.4}$$

We point out that (2.4) is adequate for differentiable functions, for if $\mathcal{S} = \mathbb{R}^n$, if $f'(x,d) \stackrel{\text{def}}{=} \nabla f(x)^T d$, and if x satisfies (2.4), then $\nabla f(x) = 0$. (The proof is a simpler version of Theorem 3.4.) To end this section we recall a definition that we will use frequently in this paper: $f(\cdot)$ *is strictly differentiable at* x *if* $\nabla f(x)$ *exists and* $\lim_{y \to x, \eta \downarrow 0} \frac{f(y+\eta d)-f(y)}{\eta} = \nabla f(x)^T d$ *for all* $d \in \mathbb{R}^n$ (see [7] for further details).

3. Sequential and parallel algorithms.

3.1. Sequential algorithms. This subsection studies a prototype algorithm amenable to both single and multiprocessor environments. It will be shown that, under C1–C3, the algorithm generates a subsequence $\{x_{i_k}\}_{k=1}^{\infty}$ that converges to a point satisfying the NSNC (2.4).

We now describe the algorithm identified below as Prototype Algorithm 3.1. Given an estimate x_i, a bounded stepsize $h_i > 0$, and a *finite* set of search directions $\mathcal{D} = \{d_1, \ldots, d_m\}$, the algorithm explores the function values at the points $x_i + h_i^j d_j$, $x_i - h_i^j d_j$, $j = 1, \ldots, m$. If the function sufficiently decreases along a search direction, i.e., for some $d_j \in \mathcal{D}$,

$$\text{either} \quad f(x_i + h_i^j d_j) - f(x_i) \leq -|o^j(h_i^j)|, \tag{3.1}$$

$$\text{or} \quad f(x_i - h_i^j d_j) - f(x_i) \leq -|o^j(h_i^j)|, \tag{3.2}$$

a new estimate x_{i+1} is generated and the associate stepsize component h_i^j may be expanded as long as $h_{i+1}^j \leq \lambda_t \tau_i$, with $\lambda_t > 1$, and $\{\tau_i\} \to 0$. On the other hand, if neither (3.1) nor (3.2) holds for any $d_j \in \mathcal{D}$, we declare the point x_i to be *blocked* by the stepsize vector h_i. The algorithm reduces the upper bound τ_i ($\tau_{i+1} < \tau_i$) as an attempt to unblock x_i. In the prototype algorithms, τ_i represents an upper bound of the ∞-norm of the stepsize vector h at blocked points.

PROTOTYPE ALGORITHM 3.1 ($f'(x,d)$ exists).

Data: $0 < \mu < 1,\ 0 < \lambda_s < 1 < \lambda_t,$ with $\mu\lambda_t < 1,$
$\mathcal{D} := \{d_1, \ldots, d_m\},\ x_i,\ 0 < h_i,\ ||h_i||_\infty \leq \tau_i,\ o^j(h_i^j),\ j = 1, \ldots, m.$

1. Define the index set $\mathcal{J}_i$ of unblocked directions as

$$\mathcal{J}_i \doteq \{1 \leq j \leq m : \quad f(x_i + \beta h_i^j d_j) - f(x_i) \leq -|o^j(h_i^j)| \text{ for some } \beta \in \{-1, 1\}\}. \tag{3.3}$$

2. **If** $\mathcal{J}_i \neq \emptyset$, let $\tau_{i+1} = \tau_i$ and choose $j \in \mathcal{J}_i$, x_{i+1}, h_{i+1}^j such that

$$x_{i+1} \in \{x \in \mathbb{R}^n : f(x) \leq f(x_i + \beta h_i^j d_j)\}, \quad \lambda_s \tau_i \leq h_{i+1}^j \leq \lambda_t \tau_i; \tag{3.4}$$

else ($\mathcal{J}_i = \emptyset$)

$$\text{let } x_{i+1} = x_i,\ \tau_{i+1} = \mu||h_i||_\infty, \text{ and choose } h_{i+1}^j \text{ such that } \lambda_s \tau_{i+1} \leq h_{i+1}^j \leq \tau_{i+1}, \quad j = 1, \ldots, m. \tag{3.5}$$

end if

Repeat 1–2 **while** τ_i is not small enough.

When gradients are available, a sufficient decrease condition has been formally established [1, 45], and a descent direction $d \in \mathbb{R}^n$ at x is easily characterized, namely, $d^T \nabla f(x) < 0$. Convergence of the search methods is based on the fact that at least one of the directions of search satisfies this descent condition. Our proof of convergence departs from this idea because our algorithms are mainly addressed to nonsmooth functions. In order to ensure convergence of the algorithm, we introduce the sufficient decrease condition (3.1)–(3.2) for nonsmooth functions. A similar condition was first discussed in [12] and later analyzed in [24] for continuously differentiable functions. A related concept, denoted as the fortified descent condition, is given in [42].

The following lemma is useful because it ensures that the algorithm is well-defined, in the sense that there always exists an h_i^j such that (3.1) or (3.2) holds whenever $f'(x_i, d_j) < 0$ or $f'(x_i, -d_j) < 0$.

LEMMA 3.1. *Let $x, d \in \mathbb{R}^n$ be, respectively, a given point and a bounded direction of search. Let $f'(x, d) < 0$. There exists $\eta > 0$ such that $f(x + \eta d) - f(x) \leq -|o(\eta)|$.*

Proof. Assume that no such η exists; then $[\eta > 0] \Rightarrow [f(x + \eta d) - f(x) > -|o(\eta)|]$. Hence, $\frac{f(x+\eta d)-f(x)}{\eta} > -\frac{|o(\eta)|}{\eta}$, and in the limit we obtain $f'(x, d) \geq 0$, which contradicts the assumption. ∎

We now proceed with the theoretical justification of the algorithm. We assume $\mu\lambda_t < 1$, C1–C3, and that given any $\epsilon > 0$ there exists $\delta(\epsilon) > 0$ such that $[\{h_{i_k}^j\}_{k=1}^{\infty} \geq \epsilon] \Rightarrow [\{o^j(h_{i_k}^j)\}_{k=1}^{\infty} \geq \delta(\epsilon)]$. (See the remark after Corollary 3.6.) Theorem 3.2 ensures that $\{h_i\}_1^{\infty} \to 0$. Theorems 3.4, 3.5 and Corollary 3.6 state that the sequence of blocked points converges to a point satisfying (2.4).

THEOREM 3.2. $\{h_i\}_1^{\infty} \to 0$.

Proof. By construction, $\lambda_s \tau_i \leq h_i^j \leq \lambda_t \tau_i$, $j = 1, \ldots, m$, and $\{\tau_i\}$ is a nonincreasing sequence that reduces its values only at blocked points. Indeed, by (3.4) and (3.5) we have $\tau_{i+1} = \mu ||h_i||_{\infty} \leq \mu\lambda_t \tau_i < \tau_i$. Hence, if blocked points occur infinitely often, the proof follows trivially for $\{\tau_i\} \to 0$. We now assume that (3.5) occurs a finite number of times and will reach a contradiction.

Let $\tau_i = \tau_k > 0$ for all $i \geq k$, and let $\epsilon = \lambda_s \tau_k$. We assert that $h_i^j \geq \epsilon$ for any $j \in \mathcal{J}_i, i \geq k$. Therefore for $i \geq k$ we obtain that

$$f(x_{i+1}) \leq f(x_i) - |o^j(h_i^j)| \leq f(x_i) - \delta(\epsilon),$$

and $\{f(x_i)\}$ decreases without bound, contradicting C1. ∎

COROLLARY 3.3. *There is an infinite number of blocked points.*

THEOREM 3.4. *Let $Span(\mathcal{D}) = \mathbb{R}^n$. Let x be a limit point of a sequence of blocked points, and let $f(\cdot)$ be strictly differentiable at x. Under these assumptions $\nabla f(x) = 0$.*

Proof. With no loss of generality, let us assume that $\{x_i\}_1^{\infty}$ is the subsequence of blocked points, and $\{x_i\}_1^{\infty} \to x$. For any $d_j \in \mathcal{D}$ we have

$$f(x_i + h_i^j d_j) - f(x_i) > -|o^j(h_i^j)|;$$

hence,

$$\nabla f(x)^T d_j = \lim_{\{x_i\} \to x, \{h_i^j\} \downarrow 0} \frac{f(x_i + h_i^j d_j) - f(x_i)}{h_i^j} \geq \lim_{\{h_i^j\} \downarrow 0} \frac{-|o^j(h_i^j)|}{h_i^j} = 0.$$

Similarly, $\nabla f(x)^T (-d_j) \geq 0$. Therefore $\nabla f(x)^T d_j = 0$. Since this equation is valid for all $d_j \in \mathcal{D}$ and $Span(\mathcal{D}) = \mathbb{R}^n$, we conclude that $\nabla f(x) = 0$. ∎

The last theorem is useful when strict differentiability holds at limit points. Obviously, (NSNC) holds. Although $f(\cdot) \in C^1$ is not required everywhere, C2 plus strict differentiability implies that $f(\cdot)$ must be smooth in a neighborhood of the limit point. We now turn our attention to convergence conditions that ensure (NSNC) without assuming strict differentiability. It is straightforward to show that for $d \in \mathcal{D}$, $\limsup_{y \to x, \eta \downarrow 0} (f(y + \eta d) - f(y))/\eta \geq 0$ at limit points of blocked point sequences. However, this result is in general not useful. There are examples that show that negative directional derivatives may appear at x and along some directions $d \in \mathcal{D}$ [2]. Generally, local convexity must be assumed in smooth problems to make sure x is a local minimum. Theorem 3.5 below proves that (NSNC) holds with a local convexity assumption. However, the Dennis–Woods function (see section 6) reveals that thus far no known search method ensures convergence to a minimum of a nonsmooth convex function.

Let us recall that when $f(\cdot)$ is convex, the function $\varphi(\eta) \doteq (f(x + \eta d) - f(x))/\eta$ is a nondecreasing function of $\eta > 0$ for fixed $x, d \in \mathbb{R}^n$. Indeed $\varphi'(\eta) = \frac{1}{\eta^2} \left[f(x) - f(x + \eta d) + \eta d^T \nabla f(x + \eta d) \right] > 0$. A general result for nondifferentiable convex functions appears in [35, Theorem 23.1].

THEOREM 3.5. *Let* C1–C3 *hold. Let* $\{x_i\} \to x$ *be a (sub)sequence of blocked points generated by the Prototype Algorithm* 3.1*, and let* $f(\cdot)$ *be convex in a neighborhood of* x*. Under these assumptions* (NSNC) *holds at* x*.*

Proof. By assumption we have for any j that

$$\begin{gathered} \{h_i^j\} \to 0 \text{ and} \\ (f(x_i + h_i^j d_j) - f(x_i))/h_i^j > -|o^j(h_i^j)|/h_i^j. \end{gathered} \tag{3.6}$$

We will prove that $f'(x, d_j) \geq 0$. Assume on the contrary that $f'(x, d_j) = -\alpha < 0$. If so, there exists $\bar{\eta} > 0$ such that $(f(x + \bar{\eta} d_j) - f(x))/\bar{\eta} \leq -\alpha/2$. Hence, for any sequence $\{x_i\} \to x$ and large enough i, we have that $(f(x_i + \bar{\eta} d_j) - f(x_i))/\bar{\eta} \leq -\alpha/4$. By convexity, $(f(x_i + \eta d_j) - f(x_i))/\eta \leq -\alpha/4$ for all $0 < \eta \leq \bar{\eta}$, which contradicts (3.6). We prove similarly that $f'(x, -d_j) \geq 0$. Since j was arbitrary, we conclude that (NSNC) holds. ∎

COROLLARY 3.6. *If the number of points that satisfy* (2.4) *is finite, the sequence of blocked points converges.*

Proof. The proof is trivial. See [28, Theorem 14.1.5]. ∎

Remark. For a practical implementation, the sufficient decrease condition may be written as $f(x_i \pm h_i^j d_j) - f(x_i) \leq -(h_i^j)^2$. Note that $h_i^j > \epsilon > 0 \Rightarrow o^j(h_i^j) = (h_i^j)^2 > \epsilon^2 = \delta(\epsilon) > 0$.

Remark. Once you have chosen an index, $j \in \mathcal{J}_i, x_{i+1}$ can be obtained by any heuristic or by any *finite* procedure that fulfills (3.4).

3.2. Parallel algorithms. We assume that we have p processors that share x_i, the best estimate, and can compute function values. We associate processor k with the index set $\mathcal{K}_k$, and $\bigcup_{k=1}^{p} \mathcal{K}_k = \{1, \ldots, m\}$. We define $\mathcal{D}_k \doteq \{d_j \in \mathcal{D} : j \in \mathcal{K}_k\}$.

Table 3.1 presents two direct translations of the prototype algorithm to parallel implementations with a balanced load among processors. Version 1 assumes that a function evaluation is costly and time consuming, whereas version 2 assumes that the communication and synchronization load can override the computational work.

In the parallel version 1, all processors simultaneously perform one function evaluation, say $f(x + \beta h^j d_j)$, for some $\beta \in \{-1, 1\}$, $j \in \mathcal{K}_k$, $k = 1, \ldots, p$, and a global reduction is used to determine the minimum function value among all these values. The minimizer, its function value, and the new stepsize vector are broadcast to all

TABLE 3.1
Iteration of derivative-free algorithms.

Input: $x \in \mathbb{R}^n, \varphi = f(x), \mathcal{D} = \{d_1, \ldots, d_m\}, h \in \mathbb{R}^m, block, \mathcal{K}_1, \ldots, \mathcal{K}_p$		
Sequential	Parallel (version 1)	Parallel (version 2)
		do in parallel
		$k =$ processor_id
if $block = 2$, **then**	**if** $block = 2$, **then**	**if** $block = 2$, **then**
$block = 0, \tau = 0.2\|\|h\|\|_\infty$	$block = 0, \tau = 0.2\|\|h\|\|_\infty$	$block = 0, \tau = 0.2\|\|h\|\|_\infty$
endif	**endif**	**endif**
$block = block + 1$	$block = block + 1$	$y_k = x, g^k = \varphi, b^k =$ **true**
for j = 1, . . . ,m	**for** i = 1, . . . ,m/p	**for** $j \in \mathcal{K}_k$
	do in parallel	
	$k =$ processor_id	
	$j_k = i$th index of $\mathcal{K}_k$	
$z = x + h^j d_j$	$z_k = x + h^{j_k} d_{j_k}$	$z = y_k + h^j d_j$
$\theta = f(z)$	$\theta_k = f(z_k)$	$\theta = f(z)$
if $\theta - \varphi \le -(h^j)^2$, **then**	$g^k = \theta_k - \varphi + (h^j)^2$	**if** $\theta - g^k \le -(h^j)^2$, **then**
$h^j = \min(1.4h^j, 4.9\tau)$		$h^j = \min(1.4h^j, 4.9\tau)$
$x = z, \varphi = \theta, block = 0$		$y_k = z, g^k = \theta, b^k =$ **false**
else	**if** $g^k > 0$ **then**	**else**
$h^j = -h^j$	$h^{j_k} = -h^{j_k}$	$h^j = -h^j$
endif	**endif**	**endif**
end for		**end for**
	end do in parallel	**end do** in parallel
		if and$(b^1, \ldots, b^p)$, **then**
		$block = block + 1$
		else
	$k = \arg\min_{1 \le q \le p}(g^q)$	$j = \arg\min_{1 \le k \le p}(g^k)$
	if $g^k \le 0$, **then**	
	$h^{j_k} = \min(1.4h^{j_k}, 4.9\tau)$	
	$x = z_k, \varphi = \theta_k, block = 0$	$x = y_j, \varphi = g^j, block = 0$
	endif	**endif**
	end for	

processors. In the parallel version 2, all processors carry out several function evaluations on a subset of the directions of search and broadcast the best point found and its function value. The iteration is completed as in the previous version. Practical implementations of both versions are given in Table 3.1. Note that if $\mathcal{K}_k = \{k\}$, both versions generate the same sequence $\{x_i\}_1^\infty$.

Both implementations have a serious drawback. Function evaluations may stem from simulations of complex systems with indefinite response time, which renders useless any effort to balance the load among processors. Consequently, both parallel versions in Table 3.1 may become highly inefficient. Fortunately, our prototype algorithm can be naturally adapted to the asynchronous parallel implementation proposed in [15] and overcome this difficulty. A processor, say processor k, works with its associate index set $\mathcal{K}_k$ and broadcasts an estimate x_{i+1} that satisfies (3.4) with $j \in \mathcal{K}_k$. The asynchronous algorithm is basically as follows.

ASYNCHRONOUS ALGORITHM (kth processor).

1. Get $x_i, f(x_i), h_i$, the successful triplet broadcast by the other processor.
2. Perform the (appropriate) function evaluation required by Algorithm 3.1 ($\mathcal{D}$ is replaced by $\mathcal{D}_k \doteq \{d_j : j \in \mathcal{K}_k\}$).
3. **If** there is a new better broadcast point, go to step 1.
4. **If** a better point is found along a direction d_j, then set $h_{i+1}^j = 1.4h_i^j$ and

TABLE 3.2
Example of a fault tolerance for a parallel algorithm.

Direction	Processor		
	1	2	3
d_1	X		X
d_2	X	X	
d_3		X	X

broadcast a successful triplet $x_{i+1}, f(x_{i+1}), h_{i+1}^j$;
else reduce $h^j, j \in \mathcal{K}_k$, by (5.1).
5. **If** $||h_i|| > \epsilon$, go to step 2; **else** STOP.
end of algorithm

Convergence theory of the asynchronous algorithm along with numerical results for all parallel implementations will be given in a forthcoming paper.

Before we end this section we would like to point out that a certain degree of fault tolerance in any parallel version can be included from the onset. We simply force every index to appear in at least two index subsets. This in turn forces any direction to be searched by at least two processors. If a processor goes down, it can pass unnoticed until it again goes up. Let us illustrate this idea with a trivial example. Let $\mathcal{D} = \{d_1, d_2, d_3\}$, $p = 3$, and $\mathcal{D}_1 = \{d_1, d_2\}$, $\mathcal{D}_2 = \{d_2, d_3\}$, and $\mathcal{D}_3 = \{d_1, d_3\}$, as shown in Table 3.2. If any one processor goes down, the others still search the whole set $\mathcal{D} = \{d_1, d_2, d_3\}$.

4. Extensions and future research.

4.1. The searching set. We can use any static set of linearly independent *unit* directions: the coordinate axis, random generated directions, conjugate directions [31, 32], and so on. It is commonly accepted that occasional but judicious adjustments to the search set might improve the convergence of direct search methods. For instance, the *rank ordered* pattern search method suggested in [21] includes the direction of best decrease on the simplex vertices; the *implicit filtering algorithm* searches on the *simplex gradient* (see subsection 4.2), and in [13] a quasi-Newton direction is included at blocked points. There is as well a convergence proof for dynamic sets in the algorithm proposed in [24].

For the sake of completeness we now sketch the convergence for dynamic sets. We denote by $\mathcal{D}_i = \{d_{i1}, \ldots, d_{im}\}$ a set of m unit directions at the ith iteration.

THEOREM 4.1. *Assume that* $\{d_{ij}\}_{i=1}^{\infty} \to d_j, j = 1, \ldots, m$. *If* C2 *holds and* $f(\cdot)$ *is Lipschitzian near any limit point* x *of the sequence of blocked points* $\{x_i\}_1^{\infty}$, *then* $f'(x, d_j) \geq 0$.

Proof. Let κ be the Lipschitz constant. With no loss of generality, assume that the sequence of blocked points $\{x_i\}_1^{\infty}$ converges to x. For any $d_{ij} \in \mathcal{D}_i$ we have

$$\begin{aligned} & f(x_i + h_i^j d_j) - f(x_i) \\ & \quad = f(x_i + h_i^j d_{ij}) - f(x_i) + f(x_i + h_i^j d_j) - f(x_i + h_i^j d_{ij}) \\ & \quad > -|o^j(h_i^j)| + f(x_i + h_i^j d_j) - f(x_i + h_i^j d_{ij}) \\ & \quad > -|o^j(h_i^j)| - |f(x_i + h_i^j d_j) - f(x_i + h_i^j d_{ij})|; \end{aligned}$$

therefore,

$$\frac{f(x_i + h_i^j d_j) - f(x_i)}{h_i^j} > -\frac{|o^j(h_i^j)|}{h_i^j} - \kappa ||d_j - d_{ij}||.$$

By taking limits, we obtain that $f'(x, d_j) \geq 0$. ∎

We note that if $f(\cdot)$ is convex and bounded above near the limit point x, it fulfills the conditions of the previous theorem [7, Proposition 2.2.6]. Furthermore, strict differentiable functions at x also satisfy the conditions of the theorem [7, Proposition 2.2.1]. In this case, we obtain that $\nabla f(x)^T d_j \geq 0$. Therefore, it is trivial to conclude that if $Span(d_1, \ldots, d_m) = \mathbb{R}^n$, if x is a limit point of a sequence of blocked points, and if $f(\cdot)$ is strictly differentiable at x, then $\nabla f(x) = 0$.

Since the sequence of blocked points converges to a point that satisfies (2.4), it seems worthwhile to explore the direction determined by the last two blocked points. The set $\mathcal{D}_3$ given below (see also [13]) includes this direction. Furthermore, it has desirable characteristics: (i) it is completely determined by the vector u, which significantly reduces the communication load in a multiprocessing environment, and (ii) its associated $n \times n$ matrix $D_3 \doteq [d_1, \ldots, d_n]$ is easily invertible, which is a nice feature to be discussed below.

This paper investigates the performance of the algorithm on the searching sets $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$ given next:

$\mathcal{D}_1$: $d_j = e_j, j = 1, \ldots, n$, the coordinate axis.

$\mathcal{D}_2$: See [17, p. 80]; also suggested in [9].

$$d_j^k = \begin{cases} \alpha & \text{if } k \neq j, \\ \beta & \text{if } k = j, \end{cases} \qquad \text{where } \alpha = \frac{\sqrt{n+1} - 1}{\sqrt{2}n},\ \beta = \alpha + \frac{1}{\sqrt{2}},\ j = 1, \ldots, n.$$

$\mathcal{D}_3$: Let x_{q+1}, x_q be two consecutive blocked points, and let

$$s = \frac{x_{q+1} - x_q}{||x_{q+1} - x_q||}, \quad j = \arg\max_k(|s^k|),$$

$$u^j = +\sqrt{\frac{(1 + |s^j|)}{2}}, \quad u^k = sign(s^j)\ s^k / 2u^j \text{ for } k \neq j.$$

Choose $d_k = (I - 2uu^T)(e_j + e_k), k = 1, \ldots, n$.

The columns of D_3^{-T} are

$$D_3^{-T} e_j = \frac{1}{2}(I - 2uu^T)\left(e_j - \sum_{k \neq j} e_k\right),$$

$$D_3^{-T} e_k = (I - 2uu^T)e_k, \quad k \neq j.$$

We end this subsection with a remark on dynamic sets $\mathcal{D}_i = \{d_{i1}, \ldots, d_{im}\}$. Let $\{d_{ij}\}_{i=1}^{\infty} \to d_j$, $j = 1, \ldots, m$. It is important that d_j, $j = 1, \ldots, m$, be linearly independent. Consider, for example, the problem $\min(x^2 + y)$ and $\mathcal{D}_i = \{(1, 0), (1, h_i)\}$. Starting at $(x_0, y_0) = (0, 0)$, the algorithm stalls at $(0, 0)$, which is not a stationary point. Indeed, $f(0, 0) = 0$, and

$$f(h_i(\pm 1, 0)) = h_i^2 > 0, \quad f(h_i(1, h_i)) = 2h_i^2 > 0, \quad \text{and } f(h_i(-1, -h_i)) = 0.$$

4.2. Smooth functions. If we assume that $f(\cdot)$ is strictly differentiable at any limit point, Theorem 3.4 shows that the sequence of blocked points generated by Algorithm 3.1 (and its parallel counterparts) converges to a first-order stationary point. We show below that, under this differentiability condition, a blocked point can be detected with fewer function values per iteration, which seems to imply that an algorithm with this property should be more efficient.

It is known that a basis of $n+1$ *positively independent* directions that positively span $\mathbb{R}^n$ suffices to prove convergence in direct search methods [3, 21, 24]. We recall that a basis $\{d_1, \ldots, d_{n+1}\}$ positively spans $\mathbb{R}^n$ if

$$\forall (x \in \mathbb{R}^n)\ \exists (\nu_1 \geq 0, \ldots, \nu_{n+1} \geq 0) : x = \sum_{k=1}^{n+1} \nu_k d_k.$$

We remark that the set $\mathcal{D}$ can easily be constructed. Let *Span* $\{d_1, \ldots, d_n\} = \mathbb{R}^n$, and let $d_{n+1} = -\sum_{k=1}^{n} \alpha_k d_k, \alpha_k > 0$.

Let us establish the counterpart of Lemma 3.1 for differentiable functions and a direction set that positively spans $\mathbb{R}^n$.

LEMMA 4.2. *Let $\nabla f(x)$ exist, and let $f'(x,d) \stackrel{\text{def}}{=} \nabla f(x)^T d$ for all $d \in \mathbb{R}^n$. Let $\mathcal{D}$ be a basis set of $n+1$ search directions that positively spans $\mathbb{R}^n$. If $\nabla f(x) \neq 0$, then there exist $\eta > 0, d_j \in \mathcal{D}$ such that $f(x + \eta d_j) - f(x) \leq -|o(\eta)|$.*

Proof. If no such η, d_j exist, then $f(x+\eta d) - f(x) > -|o(\eta)|$ for all $\eta > 0, d \in \mathcal{D}$. In the limit we obtain that $\nabla f(x)^T d \geq 0$ for all $d \in \mathcal{D}$. But by assumption $-\nabla f(x) = \sum_{k=1}^{n+1} \nu_k d_k$ for some $\nu_k \geq 0$; therefore $-\nabla f(x)^T \nabla f(x) = \sum_{k=1}^{n+1} \nu_k \nabla f(x)^T d_k \geq 0$, which only holds for $\nabla f(x) = 0$. ∎

Based on Lemma 4.2, the following algorithm seems appropriate for differentiable functions.

PROTOTYPE ALGORITHM 4.1 ($f(\cdot) \in C^1$).

Data: $0 < \mu < 1,\ 0 < \lambda_s < 1 < \lambda_t,$ with $\mu\lambda_t < 1,$
$\mathcal{D} := \{d_1, \ldots, d_{n+1}\},\ x_i,\ 0 < h_i,\ ||h_i||_\infty \leq \tau_i, o^j(h_i^j),\ j = 1, \ldots, n+1.$

1. Define the index set $\mathcal{J}_i$ of unblocked directions as

$$\mathcal{J}_i \doteq \{1 \leq j \leq n+1 : f(x_i + h_i^j d_j) - f(x_i) \leq -|o^j(h_i^j)|\}.$$

2. **If** $\mathcal{J}_i \neq \emptyset$, let $\tau_{i+1} = \tau_i$ and choose $j \in \mathcal{J}_i$, x_{i+1}, h_{i+1}^j such that

$$x_{i+1} \in \{x \in \mathbb{R}^n : f(x) \leq f(x_i + h_i^j d_j)\}, \quad \lambda_s \tau_i \leq h_{i+1}^j \leq \lambda_t \tau_i;$$

else ($\mathcal{J}_i = \emptyset$) let $x_{i+1} = x_i$, $\tau_{i+1} = \mu||h_i||_\infty$, and choose h_{i+1}^j such that

$$\lambda_s \tau_{i+1} \leq h_{i+1}^j \leq \tau_{i+1}, \quad j = 1, \ldots, n+1.$$

end if

Repeat 1–2 **while** τ_i is not small enough.

Theorem 3.4 and Lemma 4.2 lead to the following result: If $f(\cdot)$ is everywhere differentiable and strict differentiable at limit points, Algorithm 4.1 generates a (sub)sequence $\{x_i\}$ that converges to a point that satisfies the first-order necessary condition. If local convexity is assumed in place of strict differentiability, Theorem 3.5 can be used to prove that $f'(x,d) \geq 0$ for all $d \in \mathcal{D}$, but this does not seem to be a useful nonsmooth necessary condition. Section 5 reports some numerical results on differentiable functions with Algorithms 3.1 and 4.1.

Now, let us extract first-order information. It is well known that the vector $r = D^{-T}c$, where $d_k, k = 1, \ldots, n$, is the kth column of the matrix D, and $c^k = (f(x+\eta d_k) - f(x))/\eta$, $k = 1, \ldots, n$, is a good approximation to $\nabla f(x)$ for η small enough. The vector r computed for a given simplex was denoted as the *simplex gradient* in [5] and used as a possible direction of descent in the implicit filtering algorithm [19, Chapter 7]. First-order information is quite helpful for sufficiently smooth functions

because it allows quasi-Newton directions *(superlinear rate of convergence)* along the lines suggested in [6, 26, 37] and more recently in [38].

If we are certain that $f(\cdot) \in C^2$, the above approach is practical; otherwise, a lot of effort is being wasted. In [13] the gradient approximation r, with $c^j \doteq (f(x_i + h_i^j d_j) - f(x_i - h_i^j d_j))/2h_i^j$, is only computed at blocked points. The direction $d_{m+1} = -Hr$, where H is a variable metric, can be used to obtain x_{i+1} by (3.4).

4.3. Box constraints. There is a trivial way to adapt Algorithm 3.1 to the box constrained minimization problem $\min f(x)$, for $x \in \mathcal{B} := \{x \in \mathbb{R}^n : s \le x \le t\}$, where s, t are vectors in $\mathbb{R}^n$ and $s \le t$. We merely use the coordinate axes as the directions of search, i.e., $\mathcal{D} = \{e_1, \ldots, e_n\}$, and define a function $F(\cdot)$ as

$$F(x) \doteq \begin{cases} f(x), & x \in \mathcal{B}, \\ \max\{f(x), f(x_B)\}, & \text{otherwise}, \end{cases}$$

where $x_B^k = \mathit{median}\,(s^k, x^k, t^k)$ is the kth component of the projection of x onto the set $\mathcal{B}$. ($F(x) = \infty$, $x \notin B$, was suggested in [22, 23].) Obviously, $\min_{x\in\mathcal{B}} f(x)$ and $\min_{x\in\mathbb{R}^n} F(x)$ are equivalent minimization problems. It is as well immediate to observe that, starting at any $x_0 \in \mathcal{B}$, convergence is preserved when Algorithm 3.1 is used for solving the latter minimization problem. We remark that in a practical implementation no evaluation of $f(x)$ should be performed for $x \notin \mathcal{B}$.

More efficient algorithms can be suggested. This is the subject of a forthcoming paper that will be coupled with the more general linearly constrained optimization problem.

5. Numerical experiments. We implemented Algorithms 3.1 and 4.1, with $\tau_i = ||h_i||_\infty$ at blocked points, $\lambda_s = 0.01/n$, and $\lambda_t = 0.98/\mu$. Algorithm 3.1 detects a blocked point when $2n$ consecutive function evaluations fail to satisfy the sufficiency decrease condition (both side evaluations on n independent directions fail). This algorithm will be denoted as the nonsmooth directional search algorithm (NSDSA) because it is especially suited for nonsmooth functions. Algorithm 4.1 detects a blocked point when $n + 1$ consecutive function evaluations fail to satisfy the sufficiency decrease condition. This implementation is called the smooth directional search algorithm (SDSA) because it does not seem adequate for nonsmooth functions.

Implemented NSDSA Algorithm.

Input: An estimate x, its function value $\varphi = f(x)$, stepsizes $h^j = 1, j = 1, \ldots, n$,
descent index $j = 1$, index search $k = 1$, # of failures $fail = 0$,
direction generator u (or the set $\mathcal{D} : Span(\mathcal{D}) = \Re^n$),
$\tau = 1$, convergence precision $\epsilon = 10^{-6}$.

repeat
Generate $d_k = (I - 2uu^T)(e_j + e_k)$ (or obtain d_k from the set $\mathcal{D}$),
$z = x + h^k d_k, \theta = f(z)$.
if $(\theta - \varphi \le -(h^k)^2)$, **then**
$h^k = \min(\gamma h^k, (0.98/\mu)\tau), k = j$,
$x = z, \varphi = \theta, fail = 0$;
else
$h^k = -h^k, fail = fail + 1$,
$k = (k \bmod n) + 1$,
if $(fail = 2n)$, **then**
reduce $||h||_\infty$ by (5.1), $\tau = ||h||_\infty, fail = 0$.
Update the direction generator u and the indicator j,

$k = j$.
end if
end if
until ($||h||_\infty < \epsilon$) (or similar function values)

IMPLEMENTED SDSA ALGORITHM.

Input: An estimate x, its function value $\varphi = f(x)$, stepsizes $h^j = 1, j = 1, \ldots, n+1$,
descent index $j = 1$, index search $k = 1$, # failures $fail = 0$,
direction generator u (or the set $\mathcal{D}$ with $n+1$ positive basis),
$\tau = 1$, convergence precision $\epsilon = 10^{-6}$.
repeat
Generate $d_k = (I - 2uu^T)(e_j + e_k)$ (or obtain d_k from the set $\mathcal{D}$),
$z = x + h^k d_k$, $\theta = f(z)$.
if ($\theta - \varphi \leq -(h^k)^2$), **then**
$h^k = \min(\gamma h^k, (0.98/\mu)\tau), k = j$,
$x = z, \varphi = \theta, fail = 0$;
else
$fail = fail + 1$,
$k = (k \bmod (n+1)) + 1$,
if ($fail = n + 1$), **then**
reduce $||h||_\infty$ by (5.1), $\tau = ||h||_\infty, fail = 0$.
Update the direction generator u and the indicator j,
$k = j$.
end if
end if
until ($||h||_\infty < \epsilon$) (or similar function values)

The direction indicator given by j means that d_j is the descent direction determined by the last two blocked points. As described above, d_j is explored after any successful iteration, and it is also the first direction in the set $\mathcal{D}$ that the algorithm explores. The numerical results reported below with the adaptive direction $\mathcal{D}_3$ also include a heuristic that improved the performance of the algorithm notably: d_k was explored before d_p only if $h_i^k \geq h_i^p$.

To get an initial insight into the performance of the sequential algorithms, we implemented both versions in C. The results obtained were compared with those from the rank ordered pattern search (ROPS) algorithm described in [21] ($n+1$ directions) and from the multidirectional search (MDS) algorithm from [40] ($2n$ directions). We report the number of function evaluations needed to obtain a solution.

In most direct search methods, the choice of parameters seems to be crucial for the quality of convergence of the algorithm. We tried different choices of γ and μ. Intuitively, the jth stepsize component in (3.4) should not increase significantly, for convergence is determined when $||h_i|| < \epsilon$ for a small positive ϵ. We report results with $h_{i+1}^j = (1 + \frac{1}{q})h_i^j$, where q is the number of contractions. In some experiments not reported here we observed that a uniform reduction in all components of h_i could now and then cause unnecessary small steps in later iterations. The stepsize vector in the SDSA and NSDSA was reduced according to the following rule:

$$h_{i+1}^j := \begin{cases} \mu h_i^j & \text{if } h_i^j > 0.01||h_i||_\infty/n, \\ 0.01||h_i||_\infty/n & \text{otherwise.} \end{cases} \tag{5.1}$$

The initial stepsize was $h = 1$, and the stopping criteria were $||h||_\infty \leq 10^{-6}$ or $\max(|f(x \pm h^j d_j) - f(x)|) \leq 10^{-6}(|f(x)| + 1)$ at blocked points. The latter criterion

TABLE 5.1
Number of function evaluations for the Rosenbrock function ($\gamma = 1.4, \mu = 0.6$).

$x_o = 3$								
n	MDS	ROPS	SDSA			NSDSA		
			$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
2	3563	14261	90071	1320	473	14105	3298	466
3	21196	19530	F	143	1011	23914	10883	1054
5	26366	26030	719820	203	1689	53347	37912	1874
10	126051	82337	F	860	5018	144749	185823	5705
x_o standard$(-1.2, 1, \ldots)$								
n	MDS	ROPS	SDSA			NSDSA		
			$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
2	6287	8810	7246	F	247	4674	4666	406
3	16666	15578	10340	F	772	13618	6416	878
5	27426	24158	39637	1211	759	22413	21390	1437
10	89051	57158	12076	120918	3821	42050	68420	3287

attempts to terminate the algorithm when it detects that no significant improvement of the functional values will take place, or when the function value decreases too slowly. This forced premature termination in some problems, probably due to very shallow function level sets. We should also point out that when the function values are imprecise, ϵ need not be too small. Its value can be determined from the engineering process.

For the MDS and ROPS algorithms, the initial polytope from [17, p. 80] was taken, as suggested in [9]. (See searching set $\mathcal{D}_2$ in section 3 above.) For the SDSA algorithm, the searching sets $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ were augmented with the unit direction along $-\sum_{k=1}^{n} d_k$.

A generalized Rosenbrock function (5.2) of n variables ($n = 2, 3, 5, 10$) was used to study the influence of the parameters γ and μ and searching sets $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ on the performance of the algorithm. Note that this function possesses multiple stationary points for $n > 2$.

$$f(x) = \sum_{k=1}^{n-1} \left[(x^k - 1)^2 + 100 \left(x^{k+1} - (x^k)^2 \right)^2 \right]. \tag{5.2}$$

Tables 5.1–5.3 show the results for the ROPS, SDSA, MDS, and NSDSA algorithms on two starting points $x = 3$ and $x = (-1.2, 1, -1.2, 1, \ldots)$. In these tables, F stands for a solution which differs by more than 20% from the optimum value. This situation always occurred when the algorithm stopped due to a small relative change in function value (i.e, $< 10^{-6}$). Had the algorithm continued, it would have taken a significant number of function evaluations to generate the solution. Also, in Table 5.3, γ was taken as $\gamma = 1 + 1/q$, with q being the number of contractions performed by the algorithm. In this way, the expansion parameter of the stepsize gets smaller when the algorithm is converging to the solution.

MDS and ROPS always called for more function evaluations on fixed direction sets and sometimes failed for $\gamma = 1 + 1/q$, $\mu = 0.2$, while NSDSA always found the optimal solution for the given termination criteria. For a specific combination of the parameters γ and μ, fewer than 1000 function evaluations were needed by NSDSA to obtain a stationary point of the Rosenbrock function of 10 variables.

TABLE 5.2
Number of function evaluations for the Rosenbrock function ($\gamma = 1.4, \mu = 0.2$).

$x_o = 3$								
n	MDS	ROPS	SDSA			NSDSA		
			$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
2	4887	19427	F	1418	482	24094	4205	495
3	21988	17398	F	84	793	33953	19119	830
5	106726	40226	F	84	1523	72636	54570	1694
10	113811	88585	F	1375	4365	201939	186207	4134
x_o standard$(-1.2, 1, \ldots)$								
n	MDS	ROPS	SDSA			NSDSA		
			$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
2	6255	8246	5783	F	350	7780	6373	346
3	15376	14410	14756	F	918	14916	8311	758
5	7126	14000	71017	9536	1180	6527	23684	822
10	101911	83074	1656	72541	1452	50293	107927	909

TABLE 5.3
Number of function evaluations for the Rosenbrock function ($\gamma = 1 + 1/q, \mu = 0.2$).

$x_o = 3$								
n	MDS	ROPS	SDSA			NSDSA		
			$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
2	F	22838	F	65	1026	23842	4315	572
3	F	33162	F	100	1809	36731	18409	1376
5	48696	73892	552616	F	2977	62096	55127	2327
10	197251	F	F	2648	7611	125061	143467	5716
x_o standard$(-1.2, 1, \ldots)$								
n	MDS	ROPS	SDSA			NSDSA		
			$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$	$\mathcal{D}_1$	$\mathcal{D}_2$	$\mathcal{D}_3$
2	F	20078	7771	F	659	6389	6699	528
3	16060	36834	24026	F	2058	14310	6814	1389
5	F	70778	83179	1261	1850	9824	15882	880
10	190811	F	33310	F	8277	38844	121122	3864

The results for the adaptive searching set $\mathcal{D}_3$ are certainly remarkable for both SDSA and NSDSA, but the performance of SDSA may be quite sensitive to the set of positive bases used. Table 5.4 shows the results for $\gamma = 1.4$, $\mu = 0.2$, and $\mathcal{D} = \{-e_1, \ldots, -e_n, \frac{1}{\sqrt{n}} \sum_{j=1}^{n} e_j\}$ (the negative of $\mathcal{D}_1$). These results seem to indicate that the adaptive searching set not only contributes to improving the efficiency of the algorithm but also makes it more robust and reliable. We might also conjecture that the extra computation of function values needed by NSDSA to detect a blocked point provides it with additional information that improves its overall performance.

To close this section and in order to get a better picture of the performance of the algorithms, we solved some test problems from the CUTE collection. For the SDSA and NSDSA algorithms, the adaptive searching set $\mathcal{D}_3$ was used. All algorithms were run with $\gamma = 1.4$, $\mu = 0.2$, and the termination criteria indicated above. Table 5.5 reports the number of function evaluations needed by the algorithms. Along with

TABLE 5.4
Number of function evaluations for the SDSA algorithm for the Rosenbrock function ($\gamma = 1.4, \mu = 0.2$) for the negative of the searching set $\mathcal{D}_1$.

n	$x_o = 3$	x_o standard$(-1.2, 1, \ldots)$
2	6956	3281
3	19271	9098
5	23196	36408
10	36746	185451

TABLE 5.5
Number of function evaluations (function value) for different algorithms in some problems from the CUTE collection.

Problem	ROPS	SDSA	MDS	NSDSA
HATFLDD, n=3 (6.6E-8)	2606 (3.8E-3)	177 (7.9E-4)	3652 (1.0E-3)	114 (2.9E-5)
MOREBV, n=10 (0.0)	1916 (5.1E-4)	2546 (5.3E-4)	3591 (3.2E-4)	3476 (5.7E-6)
FMINSURF, n=16 (1.0)	2467 (1.0)	11881 (1.0)	3697 (1.0)	17135 (1.0)
DIXMAANK, n=15 (1.0)	2882 (1.0)	251 (1.0)	2836 (1.0)	8601 (1.0)
EDENSCH, n=36 (219.3)	9844 (219.3)	20469 (219.3)	15733 (219.3)	5622 (219.3)
CRAGGLVY, n=50 (15.4)	94454 (F) (22.9)	38442 (15.4)	86351 (F) (21.5)	38993 (17.6)
ERRINROS, n=50 (39.9)	59570 (F) (40.7)	36810 (F) (45.3)	88851 (F) (40.7)	223668 (39.9)
CHAINWOO, n=100 (1.0)	>1E6 (F) (3.38E2)	258677 (1.0)	>1E6 (F) (10.06)	84061 (1.0)

the number of function evaluations, the value of the objective function attained by the algorithm at termination is given in parenthesis. An F indicates a solution which differs by more than 20% from the minimum function value or a final solution which is far away from the minimizer.

For all the test problems, SDSA and NSDSA were robust and always found optimum or near-optimum solutions. On the other hand, ROPS and MSD failed on the largest problems, although for two problems ROPS gave a solution with the lowest number of function evaluations. These results are particularly appealing because they show NSDSA to be competitive with derivative-free algorithms designed for smooth functions, which do not share the convergence property to a point satisfying (2.4) for a class of nonsmooth functions. Finally, we conjecture that an adaptive polytope would be an asset for any pattern search algorithm.

6. Conclusion and final remarks. This paper introduces the NSNC (2.4) and a sufficient decrease condition for nonsmooth functions. It also presents a detailed implementation of practical algorithms which, under mild conditions, converge to a stationary point of smooth and nonsmooth functions of practical interest. We visualize our algorithms as new direct search algorithms with the additional feature of allowing a sufficient decrease of function values that still ensure convergence. This is achieved

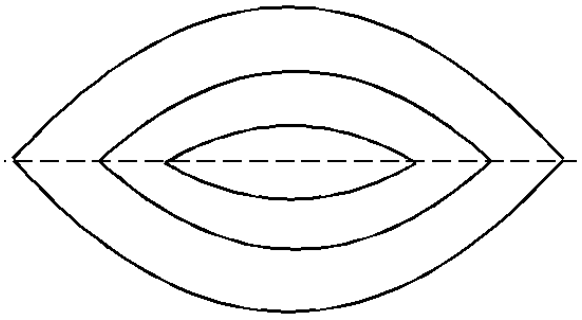

FIG. 6.1. *The Dennis–Woods function.*

by assuming that condition C2 holds globally. Our implementations can be thought of as a simplicial search, with "edges" defined by the directions of the searching set $\mathcal{D}$. This *simplex* is translated to the next iterate. The numerical results reported for sequential algorithms compare favorably with modern derivative-free algorithms recently introduced in the literature.

This paper complements recent work on generalized pattern search methods, while imposing a weaker set of conditions on the trial steps:

- Pattern search methods require a simple decrease. If $f(\cdot) \in C^1$, function values must be computed at all simplex vertices to ensure $\{\nabla f(x_i)\} \to 0$. Our algorithm can go from one "vertex" to the next as soon as it fulfills a sufficient decrease condition.
- Pattern search methods enforce a constant shrinkage/expansion factor for all edges, while ours allows independent shrinkage/expansion factors along the search directions.

Numerical results on the Rosenbrock function and some problems from the CUTE collection seem to indicate that adapting the searching set $\mathcal{D}$ to the direction of movement may have a remarkable effect on the quality of convergence. Other additional features of practical interest in actual implementations are (i) different stepsizes on the directions of search and (ii) the possibility of extracting first-order information that can be used to formulate variable metric algorithms [13].

Algorithms suitable to a multiprocessing environment were also suggested, and their computational performance will be reported in a forthcoming paper.

Either strict differentiability at limit points or convexity in a neighborhood of limit points is required for convergence to an NSNC point. Convergence to a first-order stationary point is ensured in [3] when the function $f(\cdot)$ is strictly differentiable at limit points, or differentiable, Lipschitzian, and regular near limit points. Previous works assumed $f(\cdot) \in C^1$ [24, 41]. Theorem 3.5 is a novel theoretical contribution; it requires local convexity to ensure convergence to a point satisfying (2.4). Actually, there still exists an intriguing gap in theory. No known pattern search method ensures convergence to the minimum of a nonsmooth convex function. We illustrate this with an analysis of the algorithms' behavior on the nonsmooth convex 2-variable Dennis–Woods function [10] (see Figure 6.1):

$$f(x) = \frac{1}{2}\max\left\{||x - c_1||^2, ||x - c_2||^2\right\}, \quad c_1 = (0, 32), \quad c_2 = (0, -32).$$

The origin is the minimizer of this function. If the searching set is defined as

$\mathcal{D} \doteq \{(1,1),(1,-1)\}$, the nonsmooth necessary condition (2.4) is satisfied as well for all points in the set $\mathcal{S} \doteq \{x \in \mathbb{R}^2 : x = (\alpha, 0)\}$ and any α value. Regardless of the initial point, our algorithm always converges to some point in S, which is theoretically what we can hope for. To circumvent this "convergence failure" away from the minimizer, we can randomly generate a new set $\mathcal{D}$ of search directions (or a new polytope) at unspecified blocked points, or we can work with a searching set $\mathcal{D}$ with more than n directions. Extension of multidirectional search to nonsmooth functions was considered in [40, Theorem 7.1], and we might as well expect convergence to (2.4). Indeed, the MDS algorithm always converges to a point in S [40].

Acknowledgments. This paper has been improved with the help of many colleagues and the decisive referees' contribution. Dr. J. Judice, Dr. M. Solodov, Dr. T. Kolda, and one anonymous referee called the authors' attention to recent reports that had not appeared in the open literature while this paper was under review. We are indebted to Dr. C. Audet who meticulously read the paper; he pointed out an error in a previous version and provided us with the excellent example cited in [2]. Dr. E. Hernández helped with the description of the parallel algorithm.

REFERENCES

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

[2] C. Audet, *A Counter-Example for the Derivative-Free Algorithm of García-Palomares and Rodríguez*, personal communication, École Polytechnique de Montréal and GERAD, Département de Mathématiques et de Génie Industriel, Montreal, 2000.

[3] C. Audet and J.E. Dennis, *Analysis of Generalized Pattern Searches*, Technical report TR00-07, Department of Computational and Applied Mathematics, Rice University, Houston, TX, 2000.

[4] M. Avriel, *Nonlinear Programming: Analysis and Methods*, Prentice–Hall, Englewood Cliffs, NJ, 1976.

[5] D.M. Bortz and C.T. Kelley, *The simplex gradient and noisy optimization problems*, in Computational Methods in Optimal Design and Control, J.T. Borggaard et al., eds., Birkhaüser Boston, Cambridge, MA, 1998, pp. 77–90.

[6] J. Brauninger, *A variable metric algorithm for unconstrained minimization without evaluation of derivatives*, Numer. Math., 36 (1981), pp. 359–373.

[7] F.H. Clarke, *Optimization and Nonsmooth Analysis*, Classics Appl. Math. 5, SIAM, Philadelphia, 1990.

[8] A.R. Conn, K. Scheinberg, and Ph.L. Toint, *Recent progress in unconstrained nonlinear optimization without derivatives*, Math. Programming, 79 (1997), pp. 397–414.

[9] J.E. Dennis, Jr., and V. Torczon, *Direct search methods on parallel machines*, SIAM J. Optim., 1 (1991), pp. 448–474.

[10] J.E. Dennis and D.J. Woods, *Optimization on microcomputers: The Nelder-Mead simplex algorithm*, in New Computing Environments: Microcomputers in Large-Scale Computing, A. Wouk, ed., SIAM, Philadelphia, 1987, pp. 116–122.

[11] L.C.W. Dixon, *Neural networks and unconstrained optimization*, in Algorithms for Continuous Optimization: The State of the Art, E. Spedicato, ed., Kluwer Academic Publishers, Norwell, MA, 1994, pp. 513–530.

[12] U.M. García-Palomares, *Análisis y Teorema de Convergencia de un Algoritmo de Minimización sin el Cálculo de Derivadas*, Acta Cient. Venezolana, 27 (1976), pp. 187–189.

[13] U.M. García-Palomares and J.F. Rodríguez, *Second-order Information in the Adaptive Search Exploration Algorithm*, presented at the 8th AIAA/ USAF/NASA/ISSMO/ Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, 2000, paper AIAA-2000-4765.

[14] F. Gianessi, *General optimality conditions via a separation scheme*, in Algorithms for Continuous Optimization: The State of the Art, E. Spedicato, ed., Kluwer Academic Publishers, Norwell, MA, 1994, pp. 1–23.

[15] P.D. Hough, T.G. Kolda, and V.J. Torczon, *Asynchronous parallel pattern search for nonlinear optimization*, SIAM J. Sci. Comput., 23 (2001), pp. 134–156.

[16] L.R. Huang and K.F. Ng, *Second-order necessary and sufficient conditions in nonsmooth*

optimization, Math. Programming, 66 (1994), pp. 379–402.
[17] S.L.S. Jacoby, J.S. Kowalik, and J.T. Pizzo, *Iterative Methods for Nonlinear Optimization Problems*, Prentice–Hall, Englewood Cliffs, NJ, 1972.
[18] C.T. Kelley, *Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition*, SIAM J. Optim., 10 (1999), pp. 43–55.
[19] C.T. Kelley, *Iterative Methods for Optimization*, Frontiers Appl. Math., SIAM, Philadelphia, 1999.
[20] J.C. Lagarias, J.A. Reeds, M.H. Wright, and P.E. Wright, *Convergence properties of the Nelder–Mead simplex method in low dimensions*, SIAM J. Optim., 9 (1998), pp. 112–147.
[21] R.M. Lewis and V. Torczon, *Rank Ordering and Positive Basis in Pattern Search Algorithms*, Technical report TR96-71, ICASE, Langley Research Center, Hampton, VA, 1996.
[22] R.M. Lewis and V. Torczon, *A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds*, SIAM J. Optim., 12 (2002), pp. 1075–1089.
[23] R.M. Lewis and V. Torczon, *Pattern search algorithms for bound constrained minimization*, SIAM J. Optim., 9 (1999), pp. 1082–1099.
[24] S. Lucidi and M. Sciandrone, *On the Global Convergence of Derivative Free Methods for Unconstrained Optimization*, Technical report, Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica, Rome, 1997.
[25] K.I.M. McKinnon, *Convergence of the Nelder–Mead simplex method to a nonstationary point*, SIAM J. Optim., 9 (1998), pp. 148–158.
[26] R. Mifflin, *A superlinearly convergent algorithm for minimization without evaluating derivatives*, Math. Programming, 9 (1975), pp. 100–117.
[27] J.A. Nelder and R. Mead, *A simplex method for function minimization*, The Computer Journal, 7 (1965), pp. 308–313.
[28] J. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.
[29] J.-S. Pang, *Newton's methods for B-differentiable equations*, Math. Oper. Res., 15 (1990), pp. 311–341.
[30] J.-S. Pang, S.-P. Han, and N. Rangaraj, *Minimization of locally Lipschitzian functions*, SIAM J. Optim., 1 (1991), pp. 57–82.
[31] M.J.D. Powell, *An efficient method of finding the minimum of a function of several variables without calculating derivatives*, The Computer Journal, 7 (1964), pp. 155–162.
[32] B.N. Psenichny, *A method of minimizing functions without computing derivatives*, Dokl. Akad. SSSR, 235 (1977), pp. 1097–1100.
[33] L. Qi, A. Ruszczyński, and R. Womersley, eds., *Computational Nonsmooth Optimization*, Mathematical Programming Series B 3-1, Elsevier Science, New York, 1997.
[34] S.M. Robinson, *Local structure of feasible sets in nonlinear programming, Part* III*: Stability and sensitivity*, Mathematical Programming Study, 30 (1987), pp. 45–66.
[35] R.T. Rockafellar, *Convex Analysis*, Princeton Math. Ser. 28, 2nd ed., Princeton University Press, Princeton, NJ, 1972.
[36] A.S. Rykov, *Simplex algorithms for unconstrained optimization*, Probl. Control Inform. Theory, 12 (1983), pp. 195–208.
[37] G.W. Stewart, *A modification of Davidon's method to accept difference approximations of derivatives*, J. ACM, 14 (1967), pp. 72–83.
[38] D. Stoneking, G. Bilbro, R. Trew, P. Gilmore, and C.T. Kelley, *Yield optimization using a GaAs process simulator coupled to a physical device model*, IEEE Trans. Microwave Theory and Techniques, 40 (1992), pp. 1353–1363.
[39] V. Torczon, *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines*, Ph.D. thesis, Department of Mathematical Sciences, Rice University, Houston, TX, 1989.
[40] V. Torczon, *On the convergence of the multidirectional search algorithm*, SIAM J. Optim., 1 (1991), pp. 123–145.
[41] V. Torczon, *On the convergence of pattern search algorithms*, SIAM J. Optim., 7 (1997), pp. 1–25.
[42] P. Tseng, *Fortified-descent simplicial search method: A general approach*, SIAM J. Optim., 10 (1999), pp. 269–288.
[43] M.N. Vrahatis, G.S. Androulakis, and G.E. Manoussakis, *A new unconstrained optimization method for imprecise function and gradient values*, J. Math. Anal. Appl., 197 (1996), pp. 586–607.
[44] G.R. Walsh, *Methods of Optimization*, Wiley and Sons, New York, 1975.
[45] P. Wolfe, *On the convergence of gradient methods under descent*, IBM J. Research and Development, 16 (1972), pp. 407–411.
[46] B. Xiao and P.T. Harker, *A nonsmooth Newton method for variational inequalities,* I*: Theory*, Math. Programming, 65 (1994), pp. 151–194.