Generalized Pattern Searches with Derivative Information

Mark A. Abramson
Air Force Institute of Technology
Department of Mathematics and Statistics
2950 Hobson Way, Building 640
Wright Patterson AFB, Ohio 45433-7765 USA
Mark.Abramson@afit.edu, http://www.caam.rice.edu/∼abramson

Charles Audet
École Polytechnique de Montréal and GERAD
Département de Mathématiques et de Génie Industriel
C.P. 6079, Succ. Centre-ville
Montréal (Québec), H3C 3A7 Canada
Charles.Audet@gerad.ca, http://www.gerad.ca/Charles.Audet

J.E. Dennis Jr.
Rice University
Department of Computational and Applied Mathematics
8419 42nd Ave SW
Seattle, Washington, 98136 USA
dennis@caam.rice.edu, http://www.caam.rice.edu/∼dennis

May 23, 2003

Acknowledgments: The authors were supported by AFOSR F49620-01-1-0013, The Boeing Company, ExxonMobil Upstream Research Co., the LANL Computer Science Institute (LACSI) contract 03891-99-23, and the Sandia Computer Science Research Institute (CSRI). The second author was also supported by FCAR grant NC72792 and NSERC grant 239436-01.

Abstract: A common question asked by users of direct search algorithms is how to use derivative information at iterates where it is available. This paper addresses that question with respect to Generalized Pattern Search (GPS) methods for unconstrained and linearly constrained optimization. Specifically this paper concentrates on the GPS POLL step. Polling is done to certify the need to refine the current mesh, and it requires $O(n)$ function evaluations in the worst case. We show that the use of derivative information significantly reduces the maximum number of function evaluations necessary for POLL steps, even to a worst case of a single function evaluation with certain algorithmic choices given here. Furthermore, we show that rather rough approximations to the gradient are sufficient to reduce the POLL step to a single function evaluation. We prove that using these less expensive POLL steps does not weaken the known convergence properties of the method, all of which depend only on the POLL step.

Key words: Pattern search algorithm, linearly constrained optimization, surrogate-based optimization, nonsmooth optimization, gradient-based optimization.

1 Introduction

We consider in this paper the class of Generalized Pattern Search (GPS) algorithms applied to the optimization problem

$$\min_{x \in X} \; f(x) \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and where $X \subseteq \mathbb{R}^n$ is defined, as in [24, 8], by a finite set of linear inequalities: $X = \{x \in \mathbb{R}^n : Ax \leq b\}$, where $A \in \mathbb{Q}^{m \times n}$ is a rational matrix and $b \in (\mathbb{R} \cup \{\infty\})^m$. The way of handling constraints here is the same as in [23, 24, 8]. Specifically, we apply the algorithm not really to f, but to the function $f_X = \psi_X + f$, where ψ_X is the indicator function for X. It is zero on X and ∞ elsewhere. Thus, when f_X is evaluated outside of the domain X, then the function f is not evaluated and f_X is set to ∞. Also, when a local exploration of the variable space is performed near the boundary of X, the directions that define the exploration are related to the local geometry of the boundary of the domain.

GPS algorithms can be applied to these problems, particularly when f is quite expensive to evaluate, and GPS can also be used in the surrogate management framework [10, 11]. We can not justify further global assumptions about the class of problems [10] we target, but fortunately, the Clarke [13] calculus enables a simple unified convergence analysis by allowing for f_X to be extended valued (see [8]). GPS algorithms are direct search algorithms: They rely only on function values. We like to view GPS algorithms as having each iteration divided in two steps (as presented in [10]): A global exploration of the variable space –to identify promising basins containing local optima– called the SEARCH step, and a local exploration –to refine the solution– called the POLL step. Both these steps are fully described in Sections 3.1 and 3.2, but this paper is concerned only with modifications to the POLL step. Our limited discussions of SEARCH steps are only in the context of being used with one of the modified POLL steps suggested here, and their purpose is to alleviate any concern a reader might feel about our POLL steps being too much like a crude gradient descent method.

Derivative-free GPS algorithms were defined by Torczon [29] for unconstrained optimization, and then extended by Lewis and Torczon to bound constrained optimization [23] and to problems with a finite number of linear constraints [24]. Lewis and Torczon [22] present a pattern search algorithm for general constraints through a sequence of augmented Lagrangian subproblems. Audet and Dennis extend GPS algorithms to general constraints [7] using a filter-based [17] approach, and they extend GPS to categorical optimization variables [6] as well. The latter are discrete variables that arise often in engineering design but cannot be treated by the common branch and bound techniques (*e.g.*, see [20, 1] where engineering applications with categoriacal variables are solved by a GPS algorithm).

In this paper, we add to the GPS algorithm class the possibility of using any derivative information for the objective function that might be available at an iterate. Our objective is to decrease the worst case cost of a POLL step at that iterate. To test our proposals, we give some numerical results with empty SEARCH steps to compare POLL steps with and without derivative information. These results should not be interpreted as saying anything about the efficiency or robustness of GPS algorithms in comparison with other direct search methods. We emphasize that their only purpose is to test our ideas for GPS POLL steps.

We will not discuss using derivative information in the SEARCH step, although that is an interesting direction for further research. The algorithm presented here reduces to the previous GPS versions when derivatives are not available. Abramson [3] synthesizes all this into a GPS algorithm for generally constrained mixed variable problems where derivative use is optional.

We are aware that users may choose direct search methods even for problems where $\nabla f(x)$, or a finite difference approximation, might be available and reliable. They are willing to accept more function evaluations in hopes of finding a better local minimum since gradient-based algorithms have a tendency to converge quickly to a local minimizer in the same basin as the starting point. In the tests given here for our gradient-informed POLL steps versus standard GPS POLL steps, the derivative and non-derivative versions using the same positive spanning set are equally likely to find a better local minimizer. We provide one example to illustrate the utility of a simple SEARCH in finding a better minimizer than would be found with no SEARCH. Our example supports the claim that locality is unlikely to be a problem in practice if a SEARCH procedure of the type suggested in Section 3.1 were used.

Lewis and Torczon [21] began the quest to reduce the worst case number of function calls per GPS iteration. They showed under global continuous differentiability of f that the set of directions used in the POLL step can be composed of as few as $n+1$ directions if their nonnegative linear combinations span the whole space, and still ensure subsequence convergence to a stationary point. In this paper, we show how to use gradient information to further reduce the worst case cost to one function evaluation per iteration in the unconstrained case, and to achieve a related reduction in the linearly constrained case without giving up perceived GPS advantages over gradient-based methods. Indeed, in the linearly constrained case, we may require no function evaluations and no additional gradient evaluations for some iterations. It is noted that if one computes gradient approximations by finite differences, then the linearly constrained case is the only one likely to see any savings from the use of such an expensive gradient. However, for many engineering problems, inexpensive, but inexact, gradient information can be available, and that is all we need for our modifications (see [26, 27]).

Our results do not assume that derivative information is used at every iterate, nor do we assume that derivative information in all directions is used. At an iterate where $\rho \leq n$ approximate partial derivatives are used, we can reduce the worst case cost of a POLL step from $n+1$ to $n+1-\rho$. Thus, if even a rough approximation to the full space gradient is available, the POLL step reduces to 1 evaluation of f_X. Of course, this may or may not be a worthwhile savings depending on the cost of computing the derivative information at an iterate. However, we suspect it will be especially worthwhile near convergence, when our POLL steps generate points close enough to the current iterate that they mimic gradient descent steps and are surely not expected to find a different basin.

Whether we have derivative information or not, polling directions considered by the new algorithm must still conform to the geometry of X, as proposed in [24], *i.e.*, they must contain the generators of the tangent cone at all nearby boundary points (a sufficient condition for this property is the rationality of the matrix A). If the directions do not conform to the boundary, then the iterates may stagnate at a non-optimal point even for a linear program. This leads to very satisfying optimality conditions for a finite number of linear constraints under local smoothness assumptions on f (see [8]). These conditions are unlikely to be realized for more general constraints, the key restriction being that there must be a single finite set containing generators for all the tangent cones to the boundary of the feasible region.

The remainder of the paper is as follows: In the next section, we give a brief collection of information about positive spanning sets. Next, we give a brief description of the GPS algorithm class and modify it to make use of available derivatives in the POLL step. Then, we propose in Section 4 a way to select polling directions that reduce as much as possible the number of function evaluations at each iteration. We also discuss the case of incomplete and inaccurate derivative information, such as when only a subset of the partial derivatives can be computed. In Section 5, we show that the key convergence result for the GPS algorithm still holds when it is modified to use any available derivatives. In Section 6, we use some unconstrained and bound constrained problems from the CUTE test set [9] to test reasonable variants of our approach in a GPS algorithm with an empty SEARCH step. To illustrate the value of even a simple SEARCH step, we give an academic two-dimensional example in which a rudimentary random SEARCH improves the quality of the solution. The paper concludes with a discussion of how derivative information could be used in a GPS algorithm that includes a SEARCH step.

Notation. $\mathbb{R}$, $\mathbb{Z}$, and $\mathbb{N}$ denote the set of real numbers, integers, and nonnegative integers, respectively, and N denotes the set $\{1, 2, \ldots, n\}$. For $i \in N$, e_i denotes the i^{th} coordinate vector in $\mathbb{R}^n$. For $x \in X \subset \mathbb{R}^n$, the *tangent cone* to X at x is given by $T_X(x) = \text{cl}\{\mu(w - x) : \mu \in \mathbb{R}_+, w \in X\}$ where *cl* denotes the closure, and the *normal cone* $N_X(x)$ to X at x is the polar of the tangent cone, namely, $N_X(x) = \{v \in \mathbb{R}^n : v^T w \leq 0 \;\forall\; w \in T_X(x)\}$. The normal cone is the nonnegative span of all outwardly pointing constraint normals at x.

2 Positive Spanning Sets

Since the GPS framework and theory relies on the use of positive spanning sets, it is useful to briefly define some terminology and cite some important results.

The following terminology and key theorem are due to Davis [15]. The theorem is the motivation behind the use of positive spanning sets in GPS algorithms [21].

Definition 2.1 *A* positive combination *of the set of vectors $D = \{d_i\}_{i=1}^r$ is a linear combination $\sum_{i=1}^{r} \alpha_i d_i$, where $\alpha_i \geq 0$, $i = 1, 2, \ldots, r$.*

Definition 2.2 *A finite set of vectors $D = \{d_i\}_{i=1}^r$ forms a* positive spanning set *for $\mathbb{R}^n$, if every $v \in \mathbb{R}^n$ can be expressed as a positive combination of vectors in D. The set of vectors D is said to* positively span *$\mathbb{R}^n$. The set D is said to be a* positive basis *for $\mathbb{R}^n$ if no proper subset of D positively spans $\mathbb{R}^n$.*

Davis [15] shows that the cardinality of any positive basis (*i.e.*, a minimal positive spanning set) in $\mathbb{R}^n$ is between $n + 1$ and $2n$. Common choices for positive bases include the columns of $[I, -I]$ and $[I, -e]$, where I is the identity matrix and e the vector of ones. In fact, for any basis $V \in R^{n\times n}$, the columns of $[V, -V]$ and $[V, -Ve]$ are easily shown to be positive bases.

Theorem 2.3 *A set D positively spans $\mathbb{R}^n$ if and only if, for all nonzero $v \in \mathbb{R}^n$, $v^T d > 0$ for some $d \in D$.*

Proof. (Davis [15]) Suppose D positively spans $\mathbb{R}^n$. Then for arbitrary nonzero $v \in \mathbb{R}^n$, $v = \sum_{i=1}^{|D|} \alpha_i d_i$ for some $\alpha_i \geq 0$, $d_i \in D$, $i = 1, 2, \ldots, |D|$. Then $0 < v^T v = \sum_{i=1}^{|D|} \alpha_i d_i^T v$, at least one term of which must be positive. Conversely, suppose D does not positively span $\mathbb{R}^n$; *i.e.*, suppose that the set D spans only a convex cone that is a proper subset of $\mathbb{R}^n$. Then there exists a nonzero $v \in \mathbb{R}^n$ such that $v^T d \leq 0$ for all $d \in D$. ∎

It is easy to see that for any positive spanning set D, if $\nabla f(x)$ exists at x and is nonzero, then by choosing $v = -\nabla f(x)$ in this theorem, there exists a $d \in D$ satisfying $\nabla f(x)^T d < 0$. Thus at least one $d \in D$ is a descent direction for f at x.

3 Description of pattern search algorithms

Pattern search algorithms are defined through a finite set of directions used at each iteration to create a conceptual mesh on which trial points are selected, evaluated and compared against the current best solution, called the *incumbent*. The basic ingredient in the definition of the mesh is a set of positive spanning directions D in $\mathbb{R}^n$ constructed by following rules: each direction $d_j \in D$ (for $j = 1, 2, \ldots, |D|$) is the product $G\bar{z}_j$ of the non-singular generating matrix $G \in \mathbb{R}^{n \times n}$ by an integer vector $\bar{z}_j \in \mathbb{Z}^n$. Note that the same generating matrix is used for all directions, and is often defined to be the identity matrix. These conditions are essential to the convergence theory (see [8], where the same notation is used). We conveniently view the set D as a real $n \times |D|$ matrix.

At iteration k, the mesh M_k is centered around the current iterate $x_k \in \mathbb{R}^n$, with its fineness described by the mesh size parameter $\Delta_k \in \mathbb{R}^n_+$ as

$$M_k \quad = \quad \{x_k + \Delta_k D z : z \in \mathbb{N}^{|D|}\}. \tag{2}$$

For any mesh M_k, the distance between any two distinct mesh points is bounded below by $\frac{\Delta_k}{\|G^{-1}\|}$ (see [8], Lemma 3.2).

3.1 The SEARCH step

At each iteration, the algorithm performs one or two steps, or only part of them. The objective function is evaluated at a finite set of mesh trial points with the goal of improving the incumbent value. If and when an improvement is found, that trial point becomes the new incumbent and the iteration is declared successful. The optional first step is called the SEARCH step. Any strategy of searching a finite set of mesh points can be used. It can be as simple as doing nothing (*i.e.*, the finite set of mesh points is empty), or it can make use of heuristics, such as genetic algorithms, or surrogate functions based on interpolation or on simplified models. See [10] and [11] to see how this can be done effectively.

At this point, we wish to make a distinction between two types of SEARCH strategies. In one case, the mesh size parameter Δ_k controls not only the resolution of the mesh, but the size of the step as well. Such is the case with the PDS or MDS method of Dennis and Torczon [16]. In the other case, the SEARCH routine is more or less independent of Δ_k (other than ensuring that trial point in fact lie on the mesh). We prefer the latter type because, at least near the beginning of the run, it allows for a more diversified sampling of the space. For example, a few points chosen by a classical space-filling sampling technique like the Latin hypercube [25, 28] has led to good local solutions in our experience with the NOMADm software [2] (*e.g.*, see the simple problem in Example 6.1).

Obviously, derivative information can be used in the SEARCH step; for example, one might use Hermite interpolants as surrogates, but that is outside the scope of this paper.

Our thesis here is that available derivative information can be used as detailed in the sequel to reduce the number of trial points considered in the POLL step.

3.2 The POLL step

If the SEARCH step is omitted or is unsuccessful in improving the incumbent value, a second step, named the POLL step, must be conducted before terminating the iteration. The POLL step consists of a local exploration of mesh points surrounding the incumbent solution. A global SEARCH step as described above is where the global quality of the final local solution is determined most effectively.

When the POLL step is invoked, the objective function must be tested at certain mesh points, called the poll set, near the current iterate $x_k \in \mathbb{R}^n$. At each iteration, a positive spanning matrix D_k composed of a subset of columns of D is used to construct the poll set. We write $D_k \subseteq D$ to signify that the columns of the matrix D_k are columns of D. The poll set P_k is composed of mesh points that lie a step Δ_k away from the current iterate x_k along the directions specified by the columns of D_k:

$$P_k \quad = \quad \{x_k + \Delta_k d : d \in D_k\}. \tag{3}$$

Rules for selecting D_k may depend on the iteration number and on the current iterate, *i.e.*, $D_k = D(k, x_k) \subseteq D$. We will show in Section 4 how to exploit this freedom when derivatives are available at iteration k.

In order to handle a finite number of linear constraints, we use the following definition from [8].

Definition 3.1 *A rule for selecting the positive spanning sets $D_k = D(k, x_k) \subseteq D$* conforms *to X for some $\epsilon > 0$, if at each iteration k and for each y in the boundary of X for which $\|y - x_k\| < \epsilon$, the nonnegative linear combinations of the columns of a subset $D_y(x_k)$ of D_k generate $T_X(y)$.*

This definition means that when x_k is near the boundary of the feasible region X, then D_k must contain the directions that span the tangent cones at all nearby boundary points (see [24]).

The rules for updating the mesh size parameter are as follows. Let $\tau > 1$ be a rational number and $w^- \leq -1$ and $w^+ \geq 0$ be integers constant over all iterations. At the end of iteration k, the mesh size parameter is set to

$$\Delta_{k+1} = \tau^{w_k} \Delta_k, \quad \text{where} \quad w_k \in \left\{ \begin{array}{ll} \{0, 1, \ldots, w^+\}, & \text{if the iteration is successful} \\ \{w^-, 1 + w^-, \ldots, -1\}, & \text{otherwise.} \end{array} \right. \tag{4}$$

By modifying the mesh size parameters this way, it follows that, for any $k \geq 0$, there exists an integer $r_k \in \mathbb{Z}$ such that $\Delta_k = \tau^{r_k} \Delta_0$, and the next iterate x_{k+1} can always be written as $x_0 + \sum_{i=1}^{k} \Delta_i D z_i$ for some set of vectors $\{z_i\}, i = 1, 2, \ldots, k$ in $\mathbb{N}^{|D|}$.

3.3 Generalized pattern search algorithm with a pruned poll set

The main difference between this paper and previous work on pattern search algorithms comes from the following. Whenever the derivative information for the objective function at the current iterate x_k is available, it can be used to prune (or eliminate from consideration) ascent directions from the poll set, thus reducing the number of function evaluations required by the POLL step. Because there is some ambiguity, at least in French and English, we remark here that when we use the term *pruned set of directions*, or *pruned set*, we mean what is left after the pruning operation removes some directions, not the directions that have been eliminated. The pruned set of polling directions, denoted $D_k^p \subseteq D_k$, is defined as follows.

Definition 3.2 *Given a positive spanning set D_k, the* pruned set of polling directions*, denoted by $D_k^p \subset D_k$, is given by*

$$D_k^p \quad = \quad \{d \in D_k : d^T \nabla f(x_k) \leq 0\}$$

when the gradient $\nabla f(x_k)$ is known, or by

$$D_k^p \quad = \quad D_k \setminus \{d \in V_k : f'(x_k; d) > 0\},$$

when incomplete derivative information is known, where $V_k \subset D_k$ is the set of directions in which the directional derivative is known. In the case where the gradient is known, $-\nabla f(x_k)$ is said to prune *D_k to D_k^p.*

Note that the pruning operation depends only on x_k, D_k, and the availability of the derivative information. It is independent of Δ_k. Note also that if no derivative information is available, then the pruned set of directions D_k^p equals D_k. We can now define the *pruned poll set* at iteration k as

$$P_k^p \quad = \quad \{x_k + \Delta_k d : d \in D_k^p\}. \tag{5}$$

The POLL step of the algorithm evaluates the constraints and objective at points in P_k^p looking for an improvement in f_X at a feasible point. If no improvement is found, then the incumbent solution x_k is said to be a *mesh local optimizer* with respect to the pruned poll set. Notice that if all the points y in the pruned poll set lie outside X, then $\psi_X(y) = \infty$ (and thus $f_X(y) = \infty$) for all $y \in P_k^p$ and f would not be evaluated at all in order to conclude that x_k is a mesh local optimizer. We have seen large savings from this fact in preliminary tests because in the standard POLL step, the only poll points at which one does not have to evaluate the function are those that lie outside X. The value of f is required for all poll points that lie in X, even they lie along ascent directions.

We now present the GPS algorithm that uses any available derivative information.

GPS Algorithm with derivative information

- INITIALIZATION:
  Let x_0 be such that $f_X(x_0)$ is finite, fix $\Delta_0 > 0$ and set the iteration counter k to 0.

- SEARCH AND POLL STEPS ON CURRENT MESH M_k (defined in equation (2)):
  Perform the SEARCH and possibly the POLL steps (or only part of the steps if a trial mesh point with a lower objective function value is found).

  - SEARCH: Evaluate f_X on a finite subset of trial points on the mesh M_k (the strategy that selects the points is provided by the user).
  - POLL: Evaluate f_X on the pruned poll set P_k^p (see equation (5)) around x_k defined by Δ_k and $D_k^p \subseteq D_k$ (D_k depends on k and x_k, and D_k^p on the availability of derivatives).

- PARAMETER UPDATE:
  If the SEARCH or POLL step produces an improved mesh point, *i.e.*, a feasible iterate $x_{k+1} \in M_k \cap X$ for which $f(x_{k+1}) < f(x_k)$, then update $\Delta_{k+1} \geq \Delta_k$ through rule (4).
  Otherwise, x_k is a mesh local optimizer. Set $x_{k+1} = x_k$, decrease $\Delta_{k+1} < \Delta_k$ through rule (4).
  Increase $k \leftarrow k + 1$ and go back to the SEARCH AND POLL step.

If the gradient is available and is nonzero, then its negative must prune at least one column of D_k since there must be at least one column for which $d^T \nabla f(x_k) > 0$ (see Theorem 2.3). Moreover, it could prune as many as $|D_k| - 1$ directions (*i.e.*, all but one), leading to considerable savings for an expensive function. In Section 4, we choose a particular D for which very rough derivative information allows us to prune the respective special choices we make for D_k to singletons. We think it is an interesting aside that the single poll directions left after pruning are generally the negatives of the ℓ_1 and ℓ_∞ gradients [12].

Of course, we expect that our pruning operation will yield more iterations in which the POLL step fails to find an improved mesh point, but this should happen less frequently as the mesh size parameter gets smaller. Furthermore, the role of the POLL step is mainly to determine whether the current iterate is a mesh local optimizer. This means that the string of successful searches on the current mesh is halted and searching can start afresh on a finer mesh. But for all sufficiently small values of the mesh size parameter, if our pruned POLL fails, then the "unpruned" POLL would also fail. This suggests that derivatives might be more useful when the mesh parameter is small and the POLL step would not be expected to get out of the current basin. Of course, the SEARCH can always move us into a deeper basin housing a better local optimum.

4 Using derivative information to prune well

Each iteration of a GPS algorithm is dependent upon the choice of a positive spanning set D_k, selected from the columns of the larger finite positive spanning set D. Here it is useful to think of D as preselected, but in fact, it can be modified finitely many times at the discretion of the user.

In this section, we first consider a simple measure of the richness of the set D. Theorem 2.3 says that a positive spanning set has at least one member in every half space. We are interested in a positive spanning set D so rich in directions that, for every half space, a positive spanning subset of vectors in D can be selected in such a way that a single element of the subset lies in the half space in question. We will show that no positive basis D can be so rich, but we will prove that the set $\mathbb{D} = \{-1, 0, 1\}^n$ has this property. We build up to this point by considering some useful measures of directional richness in D. The upshot is a major point of this paper: When gradients are not available, positive bases are useful in GPS because they give small poll sets [21]. However, when derivative information is available, positive bases may be a poor choice because a richer choice of directions makes it easier to isolate a descent direction and prune to a smaller poll set than would result from prunning a preselected positive basis.

The following definition is consistent with the earlier Definition 3.2, although it is more general in keeping with the applications to follow.

Definition 4.1 *(i) Given a positive spanning set $D \subset \mathbb{R}^n$ and a nonzero vector $v \in \mathbb{R}^n$, let $D(v) \subset D$ be a positive spanning set that minimizes the cardinality of the pruned set $D^p(v) = \{d \in D(v) : d^T v \geq 0\}$ (that cardinality is denoted $\rho(D, v)$). Then the vector v is said to* prune *D to $\rho(D, v)$ vectors.*

(ii) Let $\rho(D) = \max\{\rho(D, v) : v \in \mathbb{R}^n \setminus \{0\}\}$. Then the positive spanning set D is said to prune *to $\rho(D)$ vectors.*

If D is not only a positive spanning set, but also a positive basis, then one would expect to see less pruning in general since $D(v) = D$ for every choice of v. Indeed, it is not surprising for positive bases that $\rho(D)$ grows with the dimension n.

Proposition 4.2 *If D is a positive spanning set, then $1 \leq \rho(D, v) \leq \rho(D) \leq |D| - 1$ for all nonzero $v \in \mathbb{R}^n$. Moreover, if D is a positive basis, then $\frac{n+1}{2} \leq \frac{|D|}{2} \leq \rho(D) < |D| \leq 2n$.*

Proof. The first assertion is direct, because for any $v \neq 0$ and any choice of positive spanning set, there must be at least one member of the set that makes a positive inner product with v, and another that makes a negative one (see Theorem 2.3). Thus, every possible $D^p(v)$ contains at least one element and at most $|D| - 1$.

This argument also guarantees that $\rho(D) < |D|$. To derive a lower bound on $\rho(D)$, we simply consider v and $-v$. Since no proper subset of a positive basis is a positive spanning set, the only possible choice for $D(v)$ and for $D(-v)$ is D. Thus, $D^p(v) \cup D^p(-v) = D$, and so $2\rho(D) \geq \rho(D, v) + \rho(D, -v) \geq |D|$. The other inequalities follow from Theorem 2.3, where it is shown that $2n \geq |D| \geq n + 1$ for any positive basis. ∎

Taking v as the negative gradient $-\nabla f(x_k)$, Definition 4.1 can be applied to the poll set of a GPS algorithm, as introduced in Section 3.3.

Proposition 4.3 *If $\nabla f(x_k)$ is available at iteration k of the GPS algorithm, then the poll set at x_k has minimal cardinality $\rho(D, -\nabla f(x_k))$, which is an upper bound on the number of function evaluations required to execute the* POLL *step using a minimal poll set.*

Proof. From the discussion following Definition 3.2, the reader will see that $\rho(D, -\nabla f(x_k))$ represents the minimal number of polling directions that can be found by pruning choices of $D_k = D(-\nabla f(x_k)) \subset D$. Consequently, the cardinality of the pruned poll set P_k^p constructed from $D_k^p = D^p(-\nabla f(x_k)) \subset D_k \subset D$ is minimal. ∎

When P_k^p is constructed from $D_k^p = D^p(-\nabla f(x_k))$, we will say that P_k^p is a *minimal poll set*. The above results imply that the richer the directional choice in a positive spanning set D, the more sagacity can be employed to ensure that the poll set can be extensively pruned. In the next section, we propose a set of directions rich enough so that, when the gradient is available, the upper bound on the number of function evaluations in the POLL step (see Proposition 4.3) is one; thus P_k^p contains a single element.

4.1 Pruning with the gradient

We show here that the set $\mathbb{D} = \{-1, 0, 1\}^n$ prunes to a singleton, *i.e.*, for any nonzero vector $v \in \mathbb{R}^n$, there exists a positive spanning set $\mathbb{D}(v) \subset \mathbb{D}$ such that the subset of vectors of $\mathbb{D}(v)$ that makes a nonnegative inner product with v consists of a single element. Note that the set $\mathbb{D}$ is never actually constructed by the algorithm – it serves only as a mechanism to identify a pruned poll set consisting of a single direction. The key lies in the construction of $\mathbb{D}(v)$, which is done by taking the union of the ascent directions of $\mathbb{D}$ with an element $\hat{d}$ of $\mathbb{D}$ that satisfies the following properties:

$$\text{if } v_i = 0, \quad \text{then} \quad \hat{d}_i = 0, \tag{6}$$

$$\text{if } v_i \neq 0, \quad \text{then} \quad \hat{d}_i \in \{0, \text{sign}(v_i)\}, \tag{7}$$

$$\text{if } |v_i| = \|v\|_\infty, \quad \text{then} \quad \hat{d}_i = \text{sign}(v_i), \tag{8}$$

for every $i \in N = \{1, 2, \ldots, n\}$, with the convention that $\text{sign}(0) = 0$. Our construction will be such that $\hat{d}$ will be the *only* unpruned member of $\mathbb{D}(v)$.

We first show the existence of vectors in $\mathbb{D}$ satisfying properties (6)–(8). This means that there are various choices for $\mathbb{D}(v)$. Three different choices for the single unpruned direction are given, and we will discuss their significance below.

Proposition 4.4 *Let $\mathbb{D} = \{-1,0,1\}^n$. For any nonzero $v \in \mathbb{R}^n$, properties (6)–(8) are satisfied by each vector $d^{(1)}, d^{(2)}$, and $d^{(\infty)}$, where for each $i \in N$,*

$$d_i^{(1)} = \begin{cases} \operatorname{sign}(v_i) & \textit{if } |v_i| = \|v\|_\infty, \\ 0 & \textit{otherwise,} \end{cases} \tag{9}$$

$$d^{(2)} \in \arg\max_{d \in \mathbb{D}\setminus\{0\}} \frac{v^T d}{\|d\|_2}, \tag{10}$$

$$d_i^{(\infty)} = \operatorname{sign}(v_i). \tag{11}$$

Proof. Let v be a nonzero vector in $\mathbb{R}^n$. The proof is trivial for $d^{(1)}$ and for $d^{(\infty)}$. To show the result for $d^{(2)}$, first note that, since $d^{(2)} \in \mathbb{D} \setminus \{0\}$, $\|d^{(2)}\|_2^2 \in N$. For each $\delta \in N$, define the optimization problem

$$(P^\delta) \qquad \begin{array}{ll} \max\limits_{d\in\mathbb{D}} & v^t d \\ \text{s.t.} & \|d\|_2^2 = \delta. \end{array}$$

Let N^- and N^+ be a partition of the set of indices N such that $|N^+| = \delta$ and $|v_j| \geq |v_i|$ whenever $i \in N^-$ and $j \in N^+$. Define $d^\delta \in \mathbb{R}^n$ so that $d_i^\delta = 0$ for each $i \in N^-$ and $d_j^\delta = \operatorname{sign}(v_j)$ for each $j \in N^+$. It follows that d^δ is an optimal solution for (P^δ).

From (10), it follows that $d^{(2)} = d^\delta$ for some δ not less than the number of elements of v with magnitude $\|v\|_\infty$ and not greater than the number of nonzero elements of v. By construction, d^δ and thus $d^{(2)}$ satisfy properties (6)–(8). ■

The next theorem shows that for any nonzero vector $v \in \mathbb{R}^n$, any direction $\hat{d}$ satisfying properties (6)–(8) can be completed into a positive spanning set by adding the directions in

$$\mathbb{A}(v) = \left\{ d \in \mathbb{D} : v^T d < 0 \right\}, \tag{12}$$

and consequently, v prunes $\mathbb{D}$ to the single vector $\{\hat{d}\}$.

Theorem 4.5 *The set $\mathbb{D} = \{-1,0,1\}^n$ has $\rho(\mathbb{D}) = 1$.*

Proof. Let $v \in \mathbb{R}^n$ be nonzero, and let $\hat{d} \in \mathbb{D}$ be any vector satisfying the properties (6)–(8), and define $\mathbb{D}(v)$ to be the union of $\{\hat{d}\}$ with the set $\mathbb{A}(v)$ of directions in $\mathbb{D}$ that make negative inner products with $\hat{d}$.

Theorem 2.3 says that $\mathbb{D}(v) = \{\hat{d}\} \cup \mathbb{A}(v) \subset \mathbb{D}$ positively spans $\mathbb{R}^n$ if and only if, for any nonzero $b \in \mathbb{R}^n$ there exists a vector d in $\mathbb{D}(v)$ such that $b^T d > 0$. Let $b \in \mathbb{R}^n$ be a nonzero vector. If $b^T\hat{d} \neq 0$, then clearly either of $\pm\hat{d}$ makes a positive inner product with b, in which case, the proof is complete since both are in $\mathbb{D}(v)$. Thus, consider the situation where $b^T\hat{d} = 0$. The analysis is divided in two disjoint cases.

Case 1: $b_i v_i < 0$ for some index $i \in N$. Set $d_j = 0$ for $j \in N \setminus \{i\}$ and $d_i = \text{sign}(b_i)$. It follows that d belongs to $\mathbb{A}(v) \subset \mathbb{D}(v)$ and makes a positive inner product with b since

$$\begin{array}{rclclclcl} v^T d &=& v_j d_j &=& v_j \text{sign}(b_j) &=& -v_j \text{sign}(v_j) &=& -|v_j| \ < \ 0, \ \text{ and} \\ b^T d &=& b_j d_j &=& b_j \text{sign}(b_j) &=& |b_j| \ > \ 0. && \end{array}$$

Case 2: $b_i v_i \geq 0$ for all $i \in N$. From equations (6) and (7), we have $b_i \hat{d}_i \geq 0$ for all $i \in N$, and since $b^T \hat{d} = 0$, it follows that $b_i \hat{d}_i = 0$ for all $i \in N$. Let $i, j \in N$ be such that $|v_i| \geq |v_\ell|$ for all $\ell \in N$ and $b_j \neq 0$. Then $b_i = 0$, $\hat{d}_j = 0$, and by equation (8), $\hat{d}_i = \text{sign}(v_i) \neq 0$. Furthermore, since $\hat{d}_j = 0$, $|v_j| < |v_i|$. Set $d_\ell = 0$ for $\ell \in N \setminus \{i, j\}$ and $d_i = -\text{sign}(v_i)$ and $d_j = \text{sign}(b_j)$. It follows that d belongs to $\mathbb{A}(v) \subset \mathbb{D}(v)$ and makes a positive inner product with b since

$$\begin{array}{rclclcl} v^T d &=& v_i d_i + v_j d_j &=& -v_i \text{sign}(v_j) + v_j \text{sign}(b_j) \leq & -|v_i| + |v_j| \ < \ 0, \ \text{ and} \\ b^T d &=& b_i d_i + b_j d_j &=& b_j \text{sign}(b_j) \ = & |b_j| \ > \ 0. \end{array}$$

Thus $\mathbb{D}^p(v) = \{\hat{d}\}$, and the proof is complete since $v \neq 0$ was arbitrary. ∎

We now apply these results to the determination of the poll set in the GPS algorithm. Of course, for $v = -\nabla f(x_k)$, the set $\mathbb{A}(-\nabla f(x_k))$ would contain only ascent directions, which $-\nabla f(x_k)$ then prunes away. Therefore, the directions in $\mathbb{A}(-\nabla f(x_k))$ do not need to be explicitly constructed.

Theorem 4.6 *If $\nabla f(x_k)$ is available at iteration k of the GPS algorithm with positive spanning set $D = \mathbb{D}$ and if $D_k = \mathbb{D}(-\nabla f(x_k)) = \{\hat{d}\} \cup \mathbb{A}(-\nabla f(x_k))$ with $\hat{d}$ satisfying the properties (6)–(8), then a single function evaluation is required for the* POLL *step.*

Proof. Theorem 4.5 and Proposition 4.3 with $v = -\nabla f(x_k)$ guarantee the result. ∎

Thus, choosing the rich set of directions $\mathbb{D}$ and using the gradient information allows us to evaluate the barrier objective function f_X at a single poll point $x_k + \Delta_k \hat{d}$. This may not require *any* evaluations of the objective function f if the trial point lies outside of X, or obviously if $\nabla f(x_k) = 0$.

Again as an aside, we mention that our notation is meant to be consistent with [12] in the sense that for $v = -\nabla f(x_k)$, the vectors $d^{(1)}$ and $d^{(\infty)}$ are the negatives of the normalized ℓ_1 and ℓ_∞ gradients of f at x_k, respectively, while $d^{(2)}$ is the vector in $\mathbb{D}$ that makes the smallest angle with $-\nabla f(x_k)$.

4.2 Pruning with an approximation of the gradient

In many engineering applications, derivatives are approximated or inaccurately computed without much additional cost during the computation of the objective [26, 27]. Let us define a measure of the quality of an approximation of a vector.

Definition 4.7 *Let g be a nonzero vector in $\mathbb{R}^n$ and $\epsilon \geq 0$. Define $J^\epsilon(g) = \{i \in N : |g_i| + \epsilon \geq \|g\|_\infty\}$, and for every $i \in N$ set*

$$d_i^\epsilon(g) = \begin{cases} \operatorname{sign}(g_i) & \text{if } i \in J^\epsilon(g) \\ 0 & \text{otherwise.} \end{cases}$$

The vector g is said to be an $\epsilon-$approximation to the large components of a nonzero vector $v \in \mathbb{R}^n$ if $i \in J^\epsilon(g)$ whenever $|v_i| = \|v\|_\infty$ and if $\operatorname{sign}(g_i) = \operatorname{sign}(v_i)$ for every $i \in J^\epsilon(g)$.

Note that if $\epsilon = 0$, then $d^\epsilon(g)$ is identical to $d^{(1)}$ in equation (9), and if $\epsilon = \|g\|_\infty$, then then $d^\epsilon(g)$ is identical to $d^{(\infty)}$ in equation (11). The following result implicitly provides a sufficiency condition on the quality of the approximation.

Proposition 4.8 *Let $\epsilon \geq 0$ and g be an $\epsilon-$approximation to the large components of a nonzero vector $v \in \mathbb{R}^n$. Then $d^\epsilon(g)$ satisfies properties (6)–(8).*

Proof. Let $\epsilon \geq 0$ and $g \neq 0$ be an $\epsilon-$approximation to the large components of $v \neq 0 \in \mathbb{R}^n$. The set $J^\epsilon(g)$ is nonempty and contains every i for which $|v_i| = \|v\|_\infty$. If $i \in J^\epsilon(g)$ then $d_i^\epsilon(g) = \operatorname{sign}(g_i) = \operatorname{sign}(v_i)$, and therefore properties (6)–(8) are satisfied. If $i \notin J^\epsilon(g)$ then the properties are trivially satisfied. ∎

The next result establishes a mild accuracy requirement for an approximation of the negative gradient to prune $\mathbb{D}(-\nabla f(x_k))$ to a singleton.

Lemma 4.9 *Let $\epsilon \geq 0$ and g_k be an $\epsilon-$approximation to the large components of $\nabla f(x_k) \neq 0$. If there exists a vector $\hat{d}$ satisfying equations (6)–(8) for both $v = -\nabla f(x_k)$ and $v = -g_k$, then the set $\{\hat{d}\} \bigcup \mathbb{A}(-\nabla f(x_k))$ positively spans $\mathbb{R}^n$ and prunes to $\{\hat{d}\}$.*

Proof. This follows directly from the proof of Theorem 4.5. ∎

The significance of this result resides in the wide latitude allowed for the approximation. For example, if $d^{(\infty)}$ with $v = -g_k$ is used (see equation (11)) to obtain $\hat{d}$, then we only need the components of the approximation g_k to match signs with those of the true gradient $\nabla f(x_k)$ in order to prune to a singleton. If $d^{(1)}$ with $v = -g_k$ is used (see equation (9)) to obtain $\hat{d}$, then we only need the component of largest magnitude of g_k to have the same sign as that of $\nabla f(x_k)$.

The following result shows that if an approximation to the gradient matches signs with the true gradient for all components of sufficiently large magnitude, then the approximation prunes the set of poll directions to a singleton.

Theorem 4.10 *Let $\epsilon \geq 0$. At iteration k of the GPS algorithm with $D = \mathbb{D} = \{-1, 0, 1\}^n$, let $g_k \neq 0$ be an $\epsilon-$approximation of $\nabla f(x_k) \neq 0$. Then the set of directions $D_k = \{d^\epsilon(g)\} \bigcup \mathbb{A}(-\nabla f(x_k))$ positively spans $\mathbb{R}^n$. Thus, $D_k^p = \{d^\epsilon(g)\}$.*

Proof. This follows directly by combining the fact that $d^\epsilon(g)$ is the same if constructed from $v = \nabla f(x_k)$ or $v = g_k$ and by Lemma 4.9. ∎

Observe that when $\epsilon = 0$ or $\epsilon = \|g\|_\infty$, then Theorem 4.10 reduces to Theorem 4.6 with $d^\epsilon(g) = d^{(1)}$ or $d^\epsilon(g) = d^{(\infty)}$, respectively.

4.3 Pruning with incomplete derivative information

It is possible that in some instances and some iterations, the entire gradient is not available, but a few partial or directional derivatives might be known. This situation may occur for example when $f(x, y) = g(x, y) \times h(y)$ where $g(x, y)$ is analytically given, but the structure of $h(y)$ is unknown. In such a case the partial derivatives of f can be computed with respect to x but not with respect to y.

If at iteration k, the directional derivative $f'(x_k; d)$ exists, is available, and is nonnegative, then polling in the direction d from x_k will fail if the mesh size parameter is small enough. This leads to the following result.

Proposition 4.11 *Let $D_k \subseteq D$ be a positive spanning set and V_k be the subset of directions d in D_k for which the directional derivative $f'(x_k; d) > 0$ is known. Then the pruned set of directions is $D_k^p = D_k \setminus V_k$ and contains $|D_k| - |V_k|$ directions.*

Proof. The result follows from the fact that the directional derivative $f'(x_k; d)$ equals $d^T \nabla f(x_k)$. ∎

Note that when the gradient exists, if $f'(x_k; d)$ is nonpositive, then $f'(x_k; -d)$ is nonnegative. The most typical application of incomplete gradient information is when some, but not all, partial derivatives are known. The spanning set $\mathbb{D}$ can be used to reduce as much as possible the size of the pruned poll set. The approach for doing so can be outlined as follows:

Pruning Operation for Incomplete Gradients

For $x_k \in \mathbb{R}^n$ and a set of nonzero partial derivatives $\frac{\partial f}{\partial x_j}(x_k)$, $j \in J \subseteq N$,

- Set $V_k = \{s_j e_j : j \in J\} \subset \mathbb{D}$, where $s_j = \text{sign}\left(\frac{\partial f}{\partial x_j}(x_k)\right)$, $j \in J$, and e_i denotes the i^{th} coordinate vector, $i \in N$.
- Set $W_k = \{e_\ell : \ell \in N \setminus J\}$.
- Set $u = -\displaystyle\sum_{j \in J} s_j e_j - \sum_{\ell \in N \setminus J} e_\ell$.
- Set $D_k = V_k \cup W_k \cup \{u\}$, and prune D_k to $D_k^p = W_k \cup \{u\}$.

For this construction, we can prove the following result:

Corollary 4.12 *Let the partial derivatives at iteration k, $\frac{\partial f}{\partial x_j}(x_k)$ for $j \in J$ be available and all nonzero. If $\nabla f(x_k)$ exists, then the pruning operation yields a pruned set D_k^p with $n + 1 - |J|$ directions.*

Proof. By the construction above, it is clear that $B = V_k \cup W_k$ forms a basis for $\mathbb{R}^n$, and $u = -\sum_{i=1}^n b_i$, where $b_i \in B, i = 1, 2, \ldots, n$. Thus $D_k = B \cup \{u\}$ forms a positive basis for $\mathbb{R}^n$ with $|D_k| = n + 1$. Furthermore, since $\nabla f(x_k)$ exists, for any $v_j \in V_k$, we have $f'(x_k; v_j) = \nabla f(x_k)^T v_j = \nabla f(x_k)^T s_j e_j = |\frac{\partial f}{\partial x_j}(x_k)| > 0$. Since V_k has $|J|$ directions, the result follows from Proposition 4.11. ∎

The following example illustrates this result.

Example 4.13 *Consider $f : \mathbb{R}^3 \to \mathbb{R}$ with $\frac{\partial f}{\partial x_1}(x_k) > 0$ and $\frac{\partial f}{\partial x_2}(x_k) < 0$, where f is continuously differentiable at x_k. Then, with $\mathbb{D} = \{-1, 0, 1\}^n$, by defining D_k as*

$$D_k = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix},$$

the pruned set D_k^p can be constructed using only the two last columns of D_k.

The results derived in this section are compatible with the previous ones. Indeed, if the full gradient is available, and if all its components are nonzero, then using $\mathbb{D}$ the pruned set requires $n + 1 - n = 1$ direction, *i.e.,* $\rho(\mathbb{D}) = 1$ as in Theorem 4.5.

4.4 Pruning with linear constraints

A similar strategy can be used in the bound or linearly constrained case. Recall that $X = \{x \in \mathbb{R}^n : Ax \leq b\}$ defines the feasible region, where $A \in \mathcal{Q}^{m\times n}$ and $b \in \mathbb{R}^m$. Set $M = \{1, 2, \ldots, m\}$ and let a_j^T be the j-th row of A for $j \in M$.

For $\epsilon > 0$ and $x \in X$, define $\mathcal{A}_\epsilon(x) = \{j \in M : a_j^T x \geq b_j - \epsilon\}$, the set of ϵ-active constraints (as used in [24]). The set of positive spanning directions D is implicitly defined to contain all tangent cone generators of all points on all faces of the polytope X. Obviously, this set is never explicitly constructed. However, for any $x \in X$ and for a given $\epsilon > 0$, we will need to construct, as suggested in [24], the set $T(x) \subset D$ of tangent cone generators to all the ϵ-active constraints. With this construction, if $D_k \subset D$ is a positive spanning set containing $T(x_k)$, then D_k will conform to the boundary of X for $\epsilon > 0$.

If the gradient is known, then we can find a bound for the minimal cardinality of the pruned set through the following approach and subsequent proposition.

Pruning Operation under Linear Constraints

For $x_k \in \mathbb{R}^n$ with known gradient $\nabla f(x_k)$,

- Let $T(x_k)$ be the set of tangent cone generators to all ϵ-active constraints.
- Set $D' = T(x_k) \cup \{d \in \mathbb{D} : \nabla f(x_k)^T d > 0\}$.
- If D' positively spans $\mathbb{R}^n$, set $D_k = D'$; otherwise, set $D_k = D' \cup \{\hat{d}\}$, where $\hat{d}$ satisfies properties (6)–(8).
- Prune D_k to D_k^p.

Proposition 4.14 *Let $\epsilon > 0$. If $\nabla f(x_k)$ is known at iteration k, then the pruning operation yields a pruned set D_k^p containing at most $|T(x_k)| + 1$ directions.*

Proof. By construction, in the worst case, $D_k = T(x_k) \cup \{\hat{d}\} \cup \{d \in \mathbb{D} : \nabla f(x_k)^T d > 0\}$, and $D_k^p = T(x_k) \cup \{\hat{d}\}$, whose cardinality is $|T(x_k)| + 1$. ■

Again, this result is in agreement with previous ones: in the unconstrained case, or if there are no ϵ-active constraints, then $\mathcal{A}_\epsilon(x_k)$ is empty, reducing D_k^p to a singleton. Also, when $\{\hat{d}\}$ is used, then $\hat{d}$ points outside X for some boundary point within ϵ of x_k and if the trial point $x_k + \Delta_f \hat{d}$ does not belong to X, then f will not be evaluated at that trial point.

If only incomplete derivative information is available, the gradient cannot be used as strongly to prune the set of directions. Define $D' = T(x_k) \cup \{d \in V_k : f'(x_k; d) > 0\}$, then construct D_k by completing (if necessary) D' into a positive spanning set. As before, D_k^p is obtained by pruning ascent directions from D_k. By construction, all directions in $\{d \in V_k : f'(x_k; d) > 0\}$ will get pruned, and some directions in $T(x_k)$ might get pruned.

5 Convergence Results

The convergence results for the above GPS algorithm with gradients requires only the following assumptions.

A1: A function $f_X = f + \psi_X : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ is available.
A2: The constraint matrix A is rational.
A3: All iterates $\{x_k\}$ produced by the algorithm lie in a compact set.

Assumption A1 illustrates the lack of assumptions on f, which can be nondifferentiable or even discontinuous or unbounded for some x in X. The strength of our convergence result does depend on the local smoothness at the limit point. The algorithms presented above do not require the gradient to be known (or even to exist) at each iterate x_k. The algorithms use gradient information only when it is available.

Assumption A2 is necessary for the proof that the mesh size parameter gets infinitely fine since it is related to the conformity of the selection rule for D_k (see Definition 3.1).

Assumption A3 is a standard one for similar algorithms (see [6, 8, 14, 17, 18, 19]). It ensures that the algorithm produces a limit point. It is implied by more restrictive assumptions such as assuming that the level set $\{x \in X : f(x) \leq f(x_0)\}$ is compact.

For completeness, we state the following theorem from [8]. The easy proof and a discussion can be found there.

Theorem 5.1 *Under assumptions A1 and A3, there exists at least one limit point of the iteration sequence $\{x_k\}$. If f is lower semicontinuous at such a limit point $\bar{x}$, then $\lim_k f(x_k)$ exists and is greater than or equal to $f(\bar{x})$. If f is continuous at every limit point of $\{x_k\}$, then every limit point has the same function value.*

The following result was first shown by Torczon [29] for unconstrained optimization. The proof was adapted in [8] for our notation, and so it is not reproduced here since it is not affected by the contributions of the present paper.

Proposition 5.2 *The mesh size parameters satisfy* $\liminf\limits_{k\to+\infty} \Delta_k = 0$.

In [5], an example shows that this result cannot be strengthened to $\lim_k \Delta_k = 0$ without additional restrictions on the algorithm. Proposition 5.2 guarantees that there are infinitely many iterations for which the incumbent is a mesh local optimizer since the mesh size parameter shrinks only at such iterations. These iterations are the interesting ones as we will now show.

Definition 5.3 *A subsequence of the GPS iterates $\{x_k\}_{k\in K}$ (for some subset of indices K) consisting of mesh local optimizers is said to be a* refining subsequence *if $\{\Delta_k\}_{k\in K}$ converges to zero.*

The existence of convergent refining subsequences follows trivially from Proposition 5.2 and from Assumption A3. We now show results about their limit $\hat{x}$. Keep in mind that the objective function f is not evaluated at infeasible trial points. The proof is similar to that found in [8], but it is modified to take into account the iterations where the gradient is available.

Theorem 5.4 *Under assumptions A1–A3, if $\hat{x}$ is any limit of a refining subsequence, and d is any direction in D for which f at a poll step was evaluated or pruned by the gradient for infinitely many iterates in the subsequence, and if f is Lipschitz near $\hat{x}$, then the generalized directional derivative of f at $\hat{x}$ in the direction d is nonnegative, i.e., $f^\circ(\hat{x}; d) \geq 0$.*

Proof. Let $\{x_k\}_{k\in K}$ be a refining subsequence with limit point $\hat{x}$. Let $d \in D$ be obtained as in the statement of the Theorem (finiteness of D ensures the existence of d). The analysis is divided in two cases.

First, consider the case where the gradient is evaluated only a finite number of times in the subsequence $\{x_k\}_{k\in K}$. Then these finite number of iterates may be ignored, and therefore, for k sufficiently large all the poll steps in the direction d, $x_k + \Delta_k d$, are feasible. If they had not been, then f_X would have been infinite there and so f would not have been evaluated (recall that if $x \notin X$, then $f_X(x)$ is set at $+\infty$ and $f(x)$ is not evaluated). Note that since f is Lipschitz near $\hat{x}$, it must be finite near $\hat{x}$. Thus, we have that infinitely many of the right hand quotients of Clarke's generalized directional derivative definition [13]

$$f^\circ(\hat{x}; d) \;\equiv\; \limsup_{y\to\hat{x},\ t\downarrow 0} \frac{f(y+td)-f(y)}{t} \;\geq\; \limsup_{k\in K} \frac{f(x_k+\Delta_k d)-f(x_k)}{\Delta_k} . \tag{13}$$

are defined, and in fact they are the same as for f_X. This allows us to conclude that all of them must be nonnegative or else the corresponding poll step would have been successful in identifying an improved mesh point (recall that refining subsequences are constructed from mesh local optimizers).

Second, consider the case where the gradient is used in an infinite number of iterates in the subsequence. Then there is a subsequence that converges to $\hat{x}$ for which $d^T\nabla f(x_k) > 0$ and thus the right hand side of (13) is bounded below by zero. ■

The following corollary to this key result strengthens Torczon's unconstrained result. In this corollary, we will assume still that f is Lipschitz near $\hat{x}$, and in addition, we will assume that the generalized gradient of f at $\hat{x}$ is a singleton. This is equivalent to assuming that f is strictly differentiable at $\hat{x}$, *i.e.*, that $\nabla f(x)$ exists and $\lim\limits_{y\to x, t\downarrow 0} \frac{f(y+tw)-f(y)}{t} = w^T\nabla f(x)$ for all $w \in \mathbb{R}^n$ (see [13], Proposition 2.2.1 or Proposition 2.2.4). The proof is omitted since it is identical to that in [8].

Theorem 5.5 *Under assumptions A1 and A3, let $X = \mathbb{R}^n$ and $\hat{x}$ be any limit of a refining subsequence. If f is strictly differentiable at $\hat{x}$, then $\nabla f(\hat{x}) = 0$.*

For linearly constrained optimization we get the following result, whose proof can be found in [8].

Theorem 5.6 *Under assumptions A1-A3, if f is strictly differentiable at a limit point $\hat{x}$ of a refining subsequence, and if the rule for selecting the positive spanning sets $D_k = D(k, x_k) \subseteq D$ conforms to X for an $\epsilon > 0$, then $\nabla f(\hat{x})^T w \geq 0$ for all $w \in T_X(\hat{x})$, and $-\nabla f(\hat{x}) \in N_X(\hat{x})$. Thus, $\hat{x}$ is a KKT point.*

6 Numerical experiments

The algorithm described in Section 3 was programmed in Matlab [2] and applied to 20 problems from the CUTE [9] collection. Most of these problems have multiple first-order stationary points, which is important if we are to compare the solutions found by various strategies used by the algorithm. For each problem, performance of the algorithm is compared using seven different poll strategies. No SEARCH method is employed, which means that the algorithm does not use its full potential to identify deeper basins (this is why the values presented are sometimes higher than the best known values). All runs terminated either because $\Delta_k \leq 10^{-4}$ or 50,000 function evaluations have been performed. (The latter was the reasonable limit for the Matlab NOMADm code.) The initial mesh size parameter was set to $\Delta_0 = 1$, and was multiplied by 2 when an improved mesh point was identified, and divided by 2 at mesh local optimizers.

Table 1 shows objective function value attained, number of function evaluations, and number of gradient evaluations for each problem. An asterisk appearing after the problem name indicates that the problem has at least one bound constraint (none had more general linear constraints). The seven poll strategies identified in the column headings of Table 1 differ in their choice of directions and are described as follows:

- $\mathbf{Stand}_{2n}$**:** standard $2n$ directions, $D = [I, -I]$ with no gradient-pruning.
- $\mathbf{Stand}_{n+1}$**:** standard $n+1$ directions, $D = [I, -e]$ with no gradient-pruning, where e is the vector of ones.
- $\mathbf{Grad}_{2n}$**:** gradient-pruned subset of the standard $2n$ directions.
- $\mathbf{Grad}_{n+1}$**:** gradient-pruned subset of the standard $n+1$ directions.
- $\mathbf{Grad}_{3n}^{(1)}$**:** gradient-pruned subset of $\mathbb{D} = \{-1, 0, 1\}^n$, pruned by $d^{(1)}$.
- $\mathbf{Grad}_{3n}^{(2)}$**:** gradient-pruned subset of $\mathbb{D} = \{-1, 0, 1\}^n$, pruned by $d^{(2)}$.
- $\mathbf{Grad}_{3n}^{(\infty)}$**:** gradient-pruned subset of $\mathbb{D} = \{-1, 0, 1\}^n$, pruned by $d^{(\infty)}$.

- $\mathbf{Grad}^{(2)}_{3^n,2n}$**:** the union of the sets of $\text{Grad}^{(2)}_{3^n}$ and Grad_{2n}, explored in that order.

Although no SEARCH step was employed in our test runs (because we wanted to have a fair comparison of POLL steps), we include the following simple example to illustrate the value of even a very basic SEARCH step.

Example 6.1 *We ran* NOMADm *on the problem,*

$$\begin{array}{ll} \min\limits_{x,y} & f(x,y) = x^3 + y^3 - 10(x^2 + y^2) \\ s.t. & -5 \le x \le 10, \quad -5 \le y \le 10 \end{array}$$

with an initial point of $x_0 = (0.5, 0.5)$*. The results were:*

- $\mathbf{Stand}_{2n}$ *found* $\hat{x} = (6.666, -5)$ *with* $f(\hat{x}) = -523$ *in* 94 *evaluations of* f *and no evaluations of* ∇f*.*
- $\mathbf{Grad}^{(\infty)}_{3^n}$ *found* $\hat{x} = (6.666, 6.666)$ *with* $f(\hat{x}) = -296$ *in* 33 *evaluations of* f *and* 17 *evaluations of* ∇f*.*
- $\mathbf{Grad}^{(\infty)}_{3^n}$ *with only a two point feasible random* SEARCH *found* $\hat{x} = (-5, -5)$ *with* $f(\hat{x}) = -750$ *in* 85 *evaluations of* f *and* 5 *evaluations of* ∇f*.*

Thus, $\mathbf{Stand}_{2n}$ *finds a good local minimizer, and* $\mathbf{Grad}^{(\infty)}_{3^n}$ *without a search finds a higher local minimum, but requires fewer function evaluations to do so. But, when a* 2*-point random search is added to each iteration of* $\mathbf{Grad}^{(\infty)}_{3^n}$*, the global minimum is found, despite requiring fewer function evaluations than* $\mathbf{Stand}_{2n}$*.*

For the CUTE problem results given in Table 1, numbers appearing in parentheses indicate that a second set of runs was performed for that problem, but with a slightly perturbed initial point. In these cases, if a particular entry contains no number in parentheses, then that value remained unchanged in the second run. The initial point was perturbed whenever the number of required function evaluations for a run was very small. This occurs when the choices of initial point and poll directions quickly drive the algorithm to an *exact* stationary point (*i.e.*, $\nabla f(x_k) = 0$). Since this phenomenon is not common in practice, we didn't want to make a particular strategy to appear gratuitously advantageous over another.

It should be noted that three problems, ALLINIT, ALLINITU, and EXPFIT do not have starting points associated with them. Since ALLINIT is the constrained version of ALLINITU, we chose the same reasonable and feasible initial point for both problems, while the vector of ones was chosen for EXPFIT. Also, the problem OSLBQP has an infeasible starting point with respect to one of its lower bounds. Our NOMADm software automatically restores any such variable to its nearest bound before proceeding with the algorithm. (In the case of general linear constraints, our software shifts an infeasible starting point to the nearest feasible point with respect to the Euclidean norm.)

Table 1: Numerical results for selected CUTE test problems.

ALLINIT*, $n = 4$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	26.7643	26.7643	26.7643	26.7643	26.7643	26.7643	26.7643	26.7643
Function evaluations	85	85	47	47	47	56	47	56
Gradient evaluations	0	0	10	10	10	10	10	10

ALLINITU, $n = 4$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	6.9288	5.7444	6.9288	5.7661	5.8006	5.7812	9.2523	6.9293
Function evaluations	1628	425	846	133	30	65	25	650
Gradient evaluations	0	0	140	20	7	14	7	65

BARD, $n = 3$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	0.0082	0.0122	0.0082	0.0089	0.0089	0.0083	0.0091	0.0082
Function evaluations	11061	50000+	5963	50000+	2430	7799	1263	4871
Gradient evaluations	0	0	1640	16384	1018	2474	636	722

BOX2, $n = 3$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	0	0	0	0	0	0	0	0
Function evaluations	61	48	31	32	25	25	25	56
Gradient evaluations	0	0	2	2	8	8	8	4

BOX3, $n = 3$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	0	0	0	0	0.5656	0.2253	0.5656	0
Function evaluations	91	63	46	32	17	32	17	62
Gradient evaluations	0	0	2	2	2	2	2	2

DENSCHNA, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	0	0	0	0	0	0	0	0
Function evaluations	73(148)	47(156)	67(86)	47(92)	5(56)	2(23)	2(23)	2(83)
Gradient evaluations	0	0	4(22)	2(30)	3(20)	2(8)	2(8)	2(14)

DENSCHNB, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	0	0	0	0	0	0	0	0
Function evaluations	68(119)	130(95)	68(66)	87(53)	6(50)	4(28)	4(27)	4(71)
Gradient evaluations	0	0	3(15)	29(16)	3(16)	3(11)	3(10)	3(10)

DENSCHNC, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	0	0(0.0001)	0	0(0.001)	0	0	0	0
Function evaluations	75(119)	54(662)	67(68)	50(384)	7(87)	5(103)	4(62)	16(98)
Gradient evaluations	0	0	5(15)	4(168)	4(31)	3(33)	3(28)	5(16)

EXPFIT, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	0.2405	0.2406	0.2405	0.2406	0.2405	0.2405	0.2405	0.2405
Function evaluations	300	999	191	524	164	163	81	198
Gradient evaluations	0	0	66	251	66	69	37	47

MARATOSB, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Optimal f-value	-1	0	-0.0041	0	-1	-1	-1	-1
Function evaluations	66	50	42	34	17	17	17	17
Gradient evaluations	0	0	3	3	2	2	2	2

MDHOLE*, $n = 2$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	0	0	8130.9	0.0196	11127.6	8130.9	11127.6	8130.9
Function evaluations	37610	48	31	5043	15	31	15	46
Gradient evaluations	0	0	2	1516	1	2	1	2

MEXHAT, $n = 2$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	-0.0401	-0.0393	-0.0401	-0.0401	-0.0401	-0.0401	-0.0401	-0.0401
Function evaluations	350	69	203	31	32	35	20	285
Gradient evaluations	0	0	54	7	4	4	4	53

MEYER3, $n = 3$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	146888	374774	146888	146888	1947047	302439	1919139	152238
Function evaluations	15102	18926	8561	5207	2214	5252	2266	12596
Gradient evaluations	0	0	2206	1644	1099	1903	1128	2527

OSBORNEA, $n = 5$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	0.00106	0.00029	0.00106	0.00029	0.00185	0.00015	0.00089	0.00012
Function evaluations	2263	13300	1222	7783	496	1455	542	1770
Gradient evaluations	0	0	183	2042	286	528	314	237

OSBORNEB, $n = 11$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	0.04014	0.15192	0.04014	0.12229	0.04131	0.04088	0.0404	0.04014
Function evaluations	23191	50000+	11755	50000+	1606	3866	975	6574
Gradient evaluations	0	0	736	6479	777	1245	481	355

OSLBQP*, $n = 8$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25
Function evaluations	167	1355	115	472	5	5	5	8
Gradient evaluations	0	0	7	101	5	5	5	6

PALMER1*, $n = 3$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	11760	21166	11760	21166	43904	43904	43904	11760
Function evaluations	1498	438	730	208	31	30	31	1013
Gradient evaluations	0	0	160	64	11	11	13	161

PALMER1A*, $n = 4$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	116.6	7965.8	41.3	393.8	2262.7	4989.1	729.3	45.9
Function evaluations	50000+	50000+	50000+	50000+	50000+	50000+	50000+	50000+
Gradient evaluations	0	0	5636	9213	20128	17752	24999	4360

PALMER1B*, $n = 2$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	4.0137	9010.0	3.4682	6.3366	9080.6	13474.1	5135.9	3.4671
Function evaluations	50000+	50000+	43899	50000+	50000+	50000+	50000+	44127
Gradient evaluations	0	0	7684	11450	20441	25001	25001	5149

PALMER1C, $n = 8$	Stand$_{2n}$	Stand$_{n+1}$	Grad$_{2n}$	Grad$_{n+1}$	Grad$_{3^n}^{(1)}$	Grad$_{3^n}^{(2)}$	Grad$_{3^n}^{(\infty)}$	Grad$_{3^n,2n}^{(2)}$
Optimal f-value	16948398	30078	12514913	30078	39201	39859	41284	799100
Function evaluations	50000+	36077	50000+	5483	2752	3788	252	50000+
Gradient evaluations	0	0	5535	2491	1033	1140	120	4230

In general, the gradient-pruned GPS POLL-only methods converged in fewer function evaluations than the standard GPS POLL-only methods, but they sometimes terminated at a point with a higher function value, particularly if only one gradient-pruned descent direction was used in the POLL step. In fact, in no case did a standard method require fewer function evaluations than a "3^n" gradient method, and in only one case (MDHOLE, Stand_{n+1}) did a standard method require fewer function evaluations than its gradient-pruned counterpart with the same poll directions (*e.g.*, Stand_{2n} versus Grad_{2n}, Stand_{n+1} versus Grad_{n+1}). Also, $\text{Grad}^{(2)}_{3^n,2n}$ was always at least as fast as Stand_{2n}.

However, the problems ALLINITU, MARATOSB, MEXHAT, and OSBORNEA are examples of cases where a gradient-pruned method converged to a lower point than a standard poll. In fact, $\text{Grad}^{(2)}_{3^n}$ (with only one function evaluation per iteration) achieved a lower function value for OSBORNEA than both standard poll methods despite fewer function evaluations.

The problem PALMER1 is an interesting example with unflattering behavior. The three highly-pruned Grad_{3^n} strategies converge the quickest, but to the much poorer solution of 43904 than the other methods. The methods that use the most directions, Stand_{2n}, Grad_{2n}, and $\text{Grad}^{(2)}_{3^n,2n}$, converge to the best solution of 11760, with Grad_{2n} saving a considerable number of function evaluations over Stand_{2n}. The two $n+1$ direction methods converge to a solution of 21166, with gradient-pruning again saving a significant number of function evaluations. Finally, the $\text{Grad}^{(2)}_{3^n,2n}$ strategy achieves the best optimal value faster than Stand_{2n}, but it was not able to match Grad_{2n} in function evaluations, perhaps because it requires slightly more function evaluations per POLL step.

7 Discussion

We have presented a GPS algorithm that uses any available derivative information to reduce the size of the poll set. Numerical results with a GPS POLL-only method applied to problems from the CUTE test collection suggest that the use of gradient information in the POLL step often requires fewer function evaluations to terminate than a standard POLL-only GPS algorithm that does not uses gradient information. We emphasize that we envision this new POLL step being used with a SEARCH step with a global reach as explained in Section 3.1. The new approach does not necessarily save computational time if the time for computing a gradient is expensive, as for example, when it costs n times the cost to evaluate the objective function as in a finite difference gradient. Moreover, the two approaches often converge to different minimizers with different objective function values.

Acknowledgments: The authors would like to thank a pair of anonymous referees for their constructive reports, and in particular for the proof of Proposition 4.4 which is much shorter and simpler than the one we had in the initial version of the paper.

References

[1] Abramson, M.A. (2002): Mixed variable optimization of a load-bearing thermal insulation system, Technical Report *TR02-13*, Department of Computational and Applied Mathematics, Rice University, Houston Texas.

[2] Abramson, M.A. (2002): Nonlinear optimization with mixed variables and Derivatives–Matlab$^{©}$ (NOMADm). Software. Available for download at http://www.caam.rice.edu/∼abramson/NOMADm.html.

[3] Abramson, M.A. (2002): Pattern search algorithms for mixed variable general constrained optimization problems, PhD thesis, Rice University, Department of Computational and Applied Mathematics, Houston, Texas. Also appears as CAAM Technical Report TR-02-11.

[4] Alexandrov, N., Dennis, J.E. Jr, Lewis, R., Torczon, V. (1998): A trust region framework for managing the use of approximation models in optimization, *Struct. Optim.* **15**, 16–23.

[5] Audet C. (2002): Convergence results for pattern search algorithms are tight, Technical Report G-2002-56, Les Cahiers du GERAD, Montréal, Canada.

[6] Audet, C., Dennis, J.E. Jr (2000): Pattern search algorithms for mixed variable programming, *SIAM J. Optim.*, **11**, 573–594.

[7] Audet, C., Dennis, J.E. Jr (2000): A pattern search filter method for nonlinear programming without derivatives, Technical Report *TR00-09*, Department of Computational and Applied Mathematics, Rice University, Houston Texas.

[8] Audet, C., Dennis, J.E. Jr (2003): Analysis of generalized pattern searches, *SIAM J. Optim.*, **13**, 889–903.

[9] Bongartz, I., Conn, A.R., Gould, N., Toint, Ph.L. (1995): CUTE: Constrained and unconstrained testing environment, *ACM Trans. Math. Software*, **21**, 123–160.

[10] Booker, A.J., Dennis, J.E. Jr, Frank, P.D., Serafini, D.B., Torczon, V., Trosset M.W. (1999): A rigorous framework for optimization of expensive functions by surrogates, *Struct. Optim.*, **17**, 1–13.

[11] Booker, A.J., Dennis, J.E. Jr, Frank, P.D., Moore, D.W., Serafini, D.B. (1999): Managing surrogate objectives to optimize a helicopter rotor design – further experiments, AIAA Paper 98-4717, St. Louis, September 1998.

[12] Byrd, R.H., Tapia R.A. (1975): An extension of Curry's theorem to steepest descent in normed linear spaces, *Math. Programming*, **9**, 247–254.

[13] Clarke, F.H. (1990): Optimization and Nonsmooth Analysis, SIAM Classics in Applied Mathematics, Vol. 5, Philadelphia.

[14] Conn, A.R., Gould, N.I.M., Toint, Ph.L. (1991): A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds, *SIAM J. Numer. Anal.*, **28**, 545–572.

[15] Davis, C. (1954): Theory of positive linear dependence, *Amer. J. Math.*, **76**, 733–746.

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

[17] Fletcher, R., Leyffer, S. (2002): Nonlinear programming without a penalty function, *Math. Programming*, **91**, 239–269.

[18] Fletcher, R, Leyffer, S., Toint, Ph.L. (1998): On the global convergence of an SLP-filter algorithm, Report NA/183, Dundee University, Dept. of Mathematics.

[19] Fletcher, R, Gould, N.I.M., Leyffer, S., Toint, Ph.L. (1999): On the global convergence of trust-region SQP-filter algorithms for general nonlinear programming, Report 99/03, Department of Mathematics, FUNDP, Namur (B).

[20] Kokkolaras, M., Audet, C., Dennis, J.E. Jr (2001): Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system, *Optim. Engin.*, **2**, 5–29.

[21] Lewis, R.M., Torczon V. (1996): Rank ordering and positive basis in pattern search algorithms, Technical Report TR-96-71, ICASE NASA Langley Research Center.

[22] Lewis, R.M., Torczon, V. (2002): A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds, *SIAM J. Optim.*, **12**, 1075–1089.

[23] Lewis, R.M., Torczon, V. (1999): Pattern search algorithms for bound constrained minimization, *SIAM J. Optim.*, **9**, 1082–1099.

[24] Lewis, R.M., Torczon, V. (2000): Pattern search methods for linearly constrained minimization, *SIAM J. Optim.*, **10**, 917–941.

[25] McKay, M.D., Conover, W.J., Beckman, R.J. (1979): A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics*, **21**, 239–245.

[26] Mohammadi, B., Pironneau, O. (2001): Applied Shape Optimization for Fluids, Oxford University Press, Oxford.

[27] Soto, O., Löhner, R. (2001): CFD optimization using an incomplete–gradient adjoint formulation, *Int. J. for Num. Methods in Engin.*, **51**, 735–753.

[28] Stein, M (1987): Large sample properties of simulations using Latin hypercube sampling, *Technometrics*, **29**, 143–151.

[29] Torczon, V. (1997): On the convergence of pattern search algorithms, *SIAM J. Optim.*, **7**, 1–25.