

Pattern search in the presence of degeneracy

Mark A. Abramson ^{*} Olga A. Brezhneva [†] J. E. Dennis, Jr. [‡] and Rachael L. Pingel [§]

July 7, 2007

Abstract

This paper deals with generalized pattern search (GPS) algorithms for linearly constrained optimization. At each iteration, the GPS algorithm generates a set of directions that conforms to the geometry of any nearby linear constraints. This set is then used to construct trial points to be evaluated during the iteration. In previous work, Lewis and Torczon developed a scheme for computing the conforming directions; however, the issue of degeneracy merits further investigation. The contribution of this paper is to provide a detailed algorithm for constructing the set of directions whether or not the constraints are degenerate. One difficulty in the degenerate case is in classifying constraints as redundant or nonredundant. We give a short survey of the main definitions and methods for treating redundancy and propose an approach to identify nonredundant ε -active constraints, which may be useful for other active set algorithms. We also introduce a new approach for handling nonredundant linearly dependent constraints, which maintains GPS convergence properties without significantly increasing computational cost. Some simple numerical tests illustrate the effectiveness of the algorithm. We conclude by briefly considering the extension of our ideas to nonlinear constrained optimization in which constraint gradients are linearly dependent.

Keywords: Pattern search, linearly constrained optimization, derivative-free optimization, degeneracy, redundancy, constraint classification

AMS 90C56, 90C30, 65K05, 49M30

1 Introduction

This paper continues the development of generalized pattern search (GPS) algorithms [1, 2] for linearly constrained optimization problems

$$\min_{x \in \Omega} f(x), \tag{1}$$

^{*}Department of Mathematics and Statistics, Air Force Institute of Technology, Building 641, AFIT/ENC, 2950 Hobson Way, Wright-Patterson AFB, Ohio 45433 (mark.abramson@afit.edu <http://www.afit.edu/en/enc/Faculty/MAbramson/abramson.html>)

[†]Department of Mathematics and Statistics, Miami University, 123 Bachelor Hall, Oxford, Ohio 45056 (brezhnoa@muohio.edu).

[‡]Computational and Applied Mathematics Department, Rice University, MS 134, 6100 Main Street, Houston, Texas, 77005-1892 (dennis@rice.edu, <http://www.caam.rice.edu/~dennis>).

[§]Department of Mathematics, Brigham Young University, 292 TMCB, Provo, Utah 84602 (rachaelp@byu.net)

where $f : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{\infty\}$ may be discontinuous, and the feasible region is given by

$$\Omega = \{x \in \mathbb{R}^n : a_i^T x \leq b_i, i \in I\} = \{x \in \mathbb{R}^n : A^T x \leq b\}, \quad (2)$$

where, for $i \in I = \{1, 2, \dots, |I|\}$, $a_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, and $A \in \mathbb{Q}^{n \times |I|}$ is a rational matrix. Though not specifically included here, equality constraints can be treated by the traditional approach of representing each one by two inequalities.

We target the case when the function $f(x)$ may be an expensive “black box”, provide few correct digits, or may fail to return a value even for feasible points $x \in \Omega$. In this situation, the accurate approximation of derivatives is not likely to be practical.

Lewis and Torczon [2] introduced and analyzed the generalized pattern search for linearly constrained minimization problems. They proved that if the objective function is continuously differentiable and if the set of directions that defines a local search is chosen properly with respect to the geometry of the boundary of the feasible region, then GPS has at least one limit point that is a Karush-Kuhn-Tucker point. By applying the Clarke nonsmooth calculus [3], Audet and Dennis [1] simplified the analysis in [2] and introduced a new hierarchy of convergence results for problems with varying degrees of nonsmoothness. Second-order behavior of GPS is studied in [4].

Generalized pattern search algorithms generate a sequence of iterates $\{x_k\}$ in \mathbb{R}^n with nonincreasing objective function values. At each iteration, a set of positive spanning directions is used to generate trial points, and, in the case of linearly constrained problems, these directions must conform to the geometry of any nearby constraint boundaries. The key idea, which was first suggested by May in [5] and applied to GPS in [2], is to use as search directions the generators of cones polar to those generated by the normals of faces near the current iterate.

Lewis and Torczon [2] presented an algorithm for constructing the set of generators in the nondegenerate case, and left the degenerate case for future work. In more recent work, Kolda *et al.* [6] note that the problem with degenerate constraints has been well studied in computational geometry, and that the solution to the problem exists in [7, 8] and is incorporated into pattern search methods in [9]. However, in some cases, the method proposed in [7, 8] requires full enumeration, which can be cost-prohibitive. Thus, the issue of degeneracy merits further investigation.

Price and Coope [10] gave as an aside a result that can be used for constructing a set of generators in the degenerate case, but their work did not include details of the implementation in the degenerate case. It follows from their result that, in order to construct a set of generators, it is sufficient to consider maximal linearly independent subsets of the active constraints. However, this approach also implies enumeration of all possible linearly independent subsets of maximal rank and does not take into account properties of the problem that can help to reduce this enumeration.

The purpose of this paper is to give detailed consideration to GPS in the degenerate case in a way that is complementary to [1] and [2]. Our main result is a detailed algorithm for constructing the set of generators at a current GPS iterate in both the degenerate and nondegenerate cases. To construct the set of generators in the degenerate case, we identify the redundant and nonredundant active constraints and then use either QR decomposition or a construction proposed in [2].

Classification of constraints as redundant or nonredundant is one of the main issues here because it is sufficient to construct the set of generators only for nonredundant constraints. Several methods

for classifying constraints exist, including deterministic algorithms [11, 12], probabilistic hit-and-run methods [13], and a probabilistic method based on an equivalence between the constraint classification problem and the problem of finding a feasible solution to a set covering problem [14]. A survey and comparison of strategies for classifying constraints are given in [14, 12]. Any of these approaches can be applied in the GPS framework to identify redundant and nonredundant constraints. However, in this paper, we propose a new projection approach to identify nonredundant constraints that is more suitable for GPS methods.

The projection method is similar to the hit-and-run algorithm [13], in which nonredundant constraints are searched for along random direction vectors from each point in a sequence of random interior points, but differs in its use of a deterministic direction. The major advantage of the projection method for our application is that the number of direction vectors (in the terminology of the hit-and-run algorithm) is equal to the number of constraints that have to be identified. For us, this is generally a small number. In the hit-and-run algorithm, this number is determined by a stop criterion and can be large if many of the randomly generated directions do not detect a nonredundant constraint. Moreover, the formulas used in the projection method are simpler than those used for computing the intersection points of a direction vector with the hyperplanes in the hit-and-run algorithm. We should note also that the goal of hit-and-run is to detect all nonredundant constraints in a full system of linear inequalities. We use the projection method to detect the nonredundant constraints among only active constraints in the case when they are linearly dependent.

To classify constraints not detected by the projection method, we use another approach outlined in [11]. As a result, we ensure that every active constraint is detected as either redundant or nonredundant. In the worst case, we may have linearly dependent, nonredundant constraints. We propose a general approach for handling this case with an accompanying convergence theorem, along with two specific instances that can be used effectively in practice.

In the end, we briefly discuss the extension of our ideas to optimization problems with general nonlinear constraints that are linearly dependent at a solution. We do so by applying the projection method to a linearization of the constraints, and we argue that it is less costly than applying the approach of [11].

The organization of the paper is as follows. In the next section, we give a brief description of GPS and its main convergence result for linearly constrained minimization. Section 3 is devoted to the topic of redundancy. We first introduce a definition of the ε -active constraints, briefly discuss scaling issues, and review essential definitions and results on redundancy [11, 12, 15, 16] that are required for our analysis. We then introduce the projection method for determining nonredundant constraints, followed by a brief description of a more expensive follow-up approach to be applied if some constraints are not identified by the projection method. In Section 4, we give an algorithm for constructing the set of generators and discuss implementation details, including a new approach for handling nonredundant linearly dependent constraints in a rigorous way without significantly increasing computational cost. In Section 5, we consider the extension of our ideas to nonlinearly constrained problems. Section 6 is devoted to some concluding remarks.

Notation. \mathbb{R} , \mathbb{Z} , and \mathbb{N} denote the set of real numbers, integers, and nonnegative integers, respectively. For any finite set S , we may refer to the matrix S as the one whose columns are the elements of S . Similarly, for any matrix A , the notation $a \in A$ means that a is a column of A .

2 Generalized pattern search algorithms

In this section, we briefly describe the class of GPS algorithms for linearly constrained minimization, along with the main convergence result. We follow papers by Audet and Dennis [1] and by Lewis and Torczon [2], and we refer the reader there for details of managing the mesh size Δ_k . Throughout, we will always use the ℓ_2 norm.

GPS algorithms can be applied either to the objective function f or to the barrier function $f_\Omega = f + \psi_\Omega : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{+\infty\}$, where ψ_Ω is the indicator function for Ω , which is zero on Ω and ∞ elsewhere. The value of f_Ω is $+\infty$ on all points that are either infeasible or at which f is declared to be $+\infty$. This barrier approach is probably as old as direct search methods themselves.

A GPS algorithm for linearly constrained optimization generates a sequence of iterates $\{x_k\}$ in Ω . The *current iterate* $x_k \in \mathbb{R}^n$ is chosen from a finite number of points on a *mesh*, which is a discrete subset of \mathbb{R}^n . At iteration k , the mesh is centered around the *current mesh point* (*current iterate*) x_k and its fineness is parameterized through the *mesh size parameter* $\Delta_k > 0$ as

$$M_k = \{x_k + \Delta_k D z : z \in \mathbb{N}^{n_D}\}, \quad (3)$$

where D is a finite matrix whose columns form a *set of positive spanning directions* in \mathbb{R}^n and n_D is the number of columns of the matrix D . At each iteration, some positive spanning matrix D_k composed of columns of D is used to construct the *poll set*,

$$P_k = \{x_k + \Delta_k d : d \in D_k\}. \quad (4)$$

A two-dimensional mesh and poll set are illustrated in Figure 1.

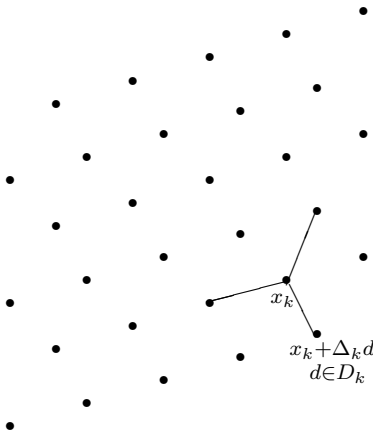


Figure 1: A mesh and poll set in \mathbb{R}^2 .

If $x_k \in \Omega$ is not near the boundary, then D_k is a positive spanning set for \mathbb{R}^n [2]. If $x_k \in \Omega$ is near the boundary, the matrix D_k is constructed so its columns d_j also span the cone of feasible directions at x_k and conform to the geometry of the boundary of Ω . Hence, the set D must be rich enough to contain generators for the tangent cone $T_\Omega(x) = \text{cl}\{\mu(\omega - x) : \mu \geq 0, \omega \in \Omega\}$ for every $x \in \Omega$. More formally, the sets D_k must satisfy the following definition.

Definition 2.1 A rule for selecting the positive spanning sets $D_k \subseteq D$ conforms to Ω for some $\varepsilon > 0$, if at each iteration k and for every y in the boundary of Ω for which $\|y - x_k\| < \varepsilon$, the tangent cone $T_\Omega(y)$ is generated by a nonnegative linear combination of columns of D_k .

Each GPS iteration is divided into two phases: an optional search and a local poll. In each step, the barrier objective function is evaluated at a finite number of mesh points in an attempt to find one that yields a lower objective function value than the incumbent (no sufficient decrease is needed). We refer to such a point as an *improved mesh point*. If an improved mesh point is found, it becomes the incumbent, so that $f(x_{k+1}) < f(x_k)$. The mesh size parameter is then either held constant or increased.

In the SEARCH step, there is complete flexibility. Any strategy may be used (including none), and the user's knowledge of the domain may be incorporated. If the SEARCH step fails to yield an improved mesh point, the POLL step is invoked. In this second step, the barrier objective function is evaluated at points in the poll set P_k (*i.e.*, neighboring mesh points) until an improved mesh point is found or until all the points in P_k have been evaluated. If both the SEARCH and POLL steps fail to find an improved mesh point, then the incumbent is declared to be a *mesh local optimizer* and is retained as the incumbent, so that $x_{k+1} = x_k$. The mesh size parameter is then decreased. Figure 2 gives a description of a basic GPS algorithm.

We remind the reader that the normal cone $N_\Omega(x)$ to Ω at x is the nonnegative span of all the outwardly pointing constraint normals at x and can be written as the polar of the tangent cone: $N_\Omega(x) = \{v \in \mathbb{R}^n : \forall \omega \in T_\Omega(x), v^T \omega \leq 0\}$.

Assumptions. We make the following standard assumptions [1]:

A1 A function f_Ω and $x_0 \in \mathbb{R}^n$ (with $f_\Omega(x_0) < \infty$) are available.

A2 The constraint matrix A is rational.

A3 All iterates $\{x_k\}$ produced by the GPS algorithm lie in a compact set.

Under these assumptions, Torczon [17] showed that $\liminf \Delta_k = 0$, and Audet and Dennis [1] identified the following subsequences, for which the limit of Δ_k is zero.

Definition 2.2 A subsequence of mesh local optimizers $\{x_k\}_{k \in K}$ (for some subset of indices K) is said to be a refining subsequence if $\{\Delta_k\}_{k \in K}$ converges to zero.

Audet and Dennis [1] proved the following convergence results for GPS in the linearly constrained case using only these assumptions.

Lemma 2.3 Under assumptions A1–A3, if \hat{x} is any limit of a refining subsequence, if d is any direction in D for which f at a POLL step was evaluated 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.*, $f^\circ(\hat{x}; d) \geq 0$.

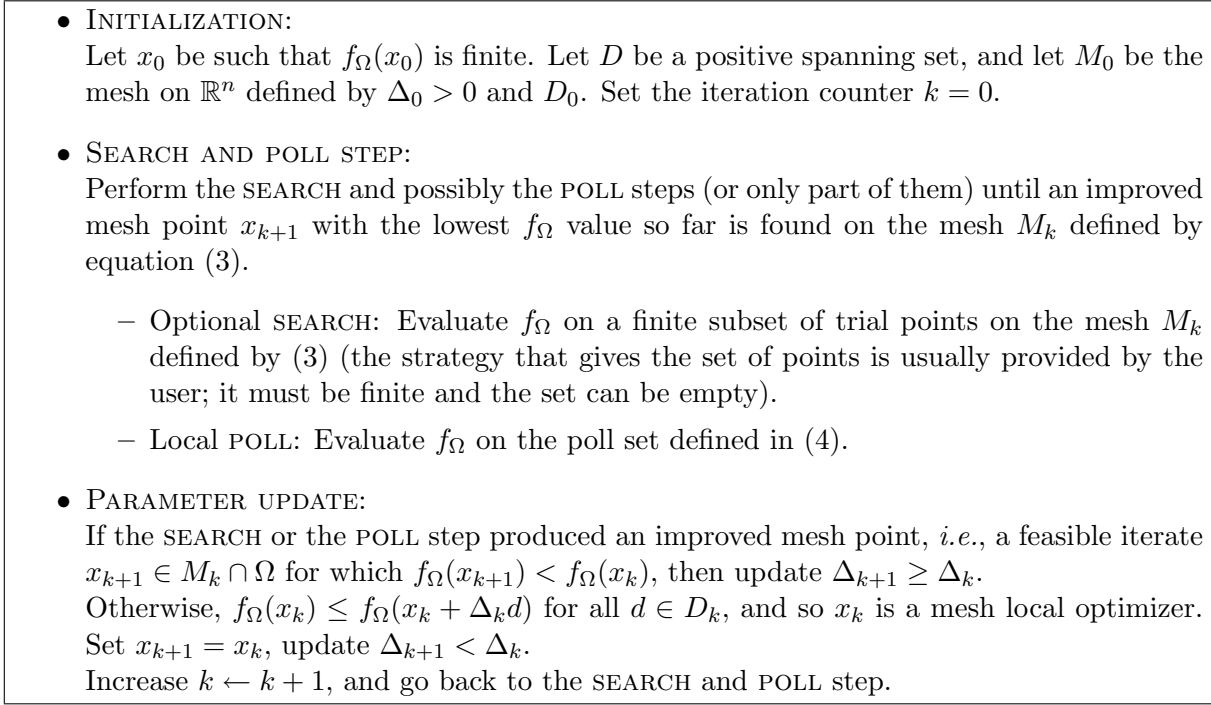


Figure 2: A simple GPS algorithm

Theorem 2.4 (Convergence to a Karush-Kuhn-Tucker point) *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 positive spanning sets $D_k \subseteq D$ conforms to Ω for some $\varepsilon > 0$, then $\nabla f(\hat{x})^T \omega \geq 0$ for all $\omega \in T_\Omega(\hat{x})$, and so $-\nabla f(\hat{x}) \in N_\Omega(\hat{x})$. Thus, \hat{x} is a Karush-Kuhn-Tucker point.*

Note that a KKT point always exists for linearly constrained problems (provided that the feasible region is not empty) [18].

The purpose of this paper is to provide an algorithm for constructing sets D_k that conform to the boundary of Ω . If the active constraints are linearly dependent, we apply strategies for the identification of redundant and nonredundant constraints, which are described in the next section, and then construct sets D_k taking into account only nonredundant constraints. We now pause to outline the main results concerning redundancy from mathematical programming, and then in Section 4, we continue consideration of GPS and strategies for constructing the sets D_k .

3 Redundancy

We now present some essential definitions and results concerning redundancy [13, 11, 12, 15, 16] that are required for our analysis. Then we propose our approach, the projection method, to determining the nonredundant constraints and briefly describe another approach that is applied if some constraints are not identified by the projection method.

We consider the feasible region Ω defined by (2), and refer to the inequality $a_j^T x \leq b_j$ as the j -th constraint. The region represented by all but the j th constraint is given by

$$\Omega_j = \{x \in \mathbb{R}^n : a_i^T x \leq b_i, i \in I \setminus \{j\}\},$$

where $I \setminus \{j\}$ is the set I with the element j removed.

The following definition is consistent with those given in [11, 12] and is illustrated in Figure 3.

Definition 3.1 *The j th constraint $a_j^T x \leq b_j$ is said to be redundant in the description of Ω if $\Omega = \Omega_j$, and otherwise is said to be nonredundant.*

3.1 ε -active constraints

We next compare two definitions of ε -active constraints and discuss some associated scaling issues. They are replicated from [10] and [2], respectively.

Definition 3.2 (e.g., [10]). *Let some scalar $\varepsilon > 0$ be given and $x_k \in \Omega$. The j th constraint is ε -active at x_k if*

$$0 \leq b_j - a_j^T x_k \leq \varepsilon. \quad (5)$$

Definition 3.3 (e.g., [2]). *Let some scalar $\varepsilon > 0$ be given and $x_k \in \Omega$. The j th constraint is ε -active at x_k if*

$$\text{dist}(x_k, H_j) \leq \varepsilon, \quad (6)$$

where $H_j = \{x \in \mathbb{R}^n : a_j^T x = b_j\}$, and $\text{dist}(x_k, H_j) = \min_{y \in H_j} \|y - x_k\|$ is the distance from x_k to the hyperplane H_j .

Clearly, the j th constraint can be made ε -active at x_k in the sense of Definition 3.2 by multiplying the inequality $b_j - a_j^T x_k \geq 0$ by a sufficiently small number. On the other hand, this multiplication does not change the distance between the point x_k and any H_j defined in Definition 3.3. In the paper, we prefer to use Definition 3.2, since it is easier to check than Definition 3.3. However, Definition 3.2 is proper, if we assume preliminary scaling of the constraints so that the following lemma applies.

Lemma 3.4 *Let some scalar $\varepsilon > 0$ be given, $x_k \in \Omega$, and $\|a_j\| = 1$ for all $j \in I$ in (2). Then, for any $j \in I$, Definition 3.2 of the ε -active constraint is equivalent to Definition 3.3, and the projection $P_j(x_k)$ of the point x_k onto the hyperplane $H_j = \{x \in \mathbb{R}^n : a_j^T x = b_j\}$ is defined by*

$$P_j(x_k) = x_k + a_j(b_j - a_j^T x_k). \quad (7)$$

Proof. For any $j \in I$, the distance from x_k to the hyperplane H_j is given by

$$\text{dist}(x_k, H_j) = \frac{|b_j - a_j^T x_k|}{\|a_j\|}. \quad (8)$$

Hence, if $\|a_j\| = 1$ and $x_k \in \Omega$, (5) is equivalent to (6).

By definition of the projection of x_k onto H_j ,

$$\|P_j(x_k) - x_k\| = \text{dist}(x_k, H_j).$$

Since $x_k \in \Omega$ and $\|a_j\| = 1$, it follows from (8) that $\text{dist}(x_k, H_j) = b_j - a_j^T x_k$ and

$$P_j(x_k) = x_k + a_j \text{dist}(x_k, H_j) = x_k + a_j(b_j - a_j^T x_k).$$

Hence, (7) holds. ■

To satisfy the conditions of Lemma 3.4, we introduce the matrix \bar{A} and vector \bar{b} that are additional scaled copies of A and b , respectively from (2), such that

$$\bar{a}_i = \frac{a_i}{\|a_i\|}, \quad \bar{b}_i = \frac{b_i}{\|a_i\|}, \quad i \in I. \quad (9)$$

Consequently, $\|\bar{a}_i\| = 1$ for all $i \in I$ and $\Omega = \{x \in \mathbb{R}^n : A^T x \leq b\} = \{x \in \mathbb{R}^n : \bar{A}^T x \leq \bar{b}\} = \{x \in \mathbb{R}^n : \bar{a}_i^T x \leq \bar{b}_i, i \in I\}$.

We then use \bar{A} and \bar{b} to define the set of indices of the ε -active constraints as

$$I(x_k, \varepsilon) = \{i \in I : 0 \leq \bar{b}_i - \bar{a}_i^T x_k \leq \varepsilon\}, \quad (10)$$

and we apply the projection method for detection of the nonredundant constraints (see Section 3.3.1 for more details.) We refer to the set $I(x_k, \varepsilon)$ as the *working index set* at the current iterate x_k .

This paper also makes use of the regions given by

$$\Omega(x_k, \varepsilon) = \{x \in \mathbb{R}^n : a_i^T x \leq b_i, i \in I(x_k, \varepsilon)\}, \quad (11)$$

and

$$\Omega_j(x_k, \varepsilon) = \{x \in \mathbb{R}^n : a_i^T x \leq b_i, i \in I(x_k, \varepsilon) \setminus \{j\}\}, \quad j \in I(x_k, \varepsilon).$$

Clearly, $\Omega \subseteq \Omega(x_k, \varepsilon) \subseteq \Omega_j(x_k, \varepsilon)$. Furthermore, since $\Omega \subseteq \Omega(x_k, \varepsilon)$, if the j th constraint is redundant in the description of $\Omega(x_k, \varepsilon)$, it is also redundant in the description of Ω .

3.2 Redundancy in mathematical programming

We now give definitions and theorems consistent with the mathematical programming literature [13, 11, 12, 15, 16]. We begin with the following definitions, which can be found in [15, 16]. In the discussion that follows, we use notation consistent with that of Section 1 (see (2) and the discussion that follows it).

Definition 3.5 *A subset of \mathbb{R}^n described by a finite set of linear constraints $P = \{x \in \mathbb{R}^n : A^T x \leq b\}$ is a polyhedron.*

Obviously, Ω given by (2) and $\Omega(x_k, \varepsilon)$ given by (11) are polyhedra.

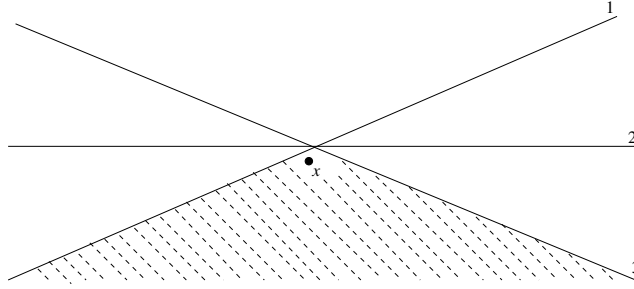


Figure 3: An illustration of ε -active and redundant constraints. Constraints 1–3 are ε -active at the current iterate x , and constraint 2 is redundant.

Definition 3.6 *The points $z^1, \dots, z^p \in \mathbb{R}^n$ are affinely independent if the $p - 1$ directions $z^2 - z^1, \dots, z^p - z^1$ are linearly independent, or alternatively, the p vectors $(z^1, 1), \dots, (z^p, 1) \in \mathbb{R}^{n+1}$ are linearly independent.*

We will assume that Ω is full-dimensional, as defined below.

Definition 3.7 *The dimension $\dim(P)$ of a polyhedron P is one less than the maximum number of affinely independent points in P . Then $P \subseteq \mathbb{R}^n$ is full-dimensional if and only if $\dim(P) = n$.*

Note that, if Ω were not full-dimensional, then a barrier GPS approach would not be a reasonable way to handle linear constraints because it would be difficult to find any trial point in Ω . Since we assume Ω is full-dimensional, this implies that its supersets $\Omega(x_k, \varepsilon)$ and $\Omega_j(x_k, \varepsilon)$ are full-dimensional.

Definition 3.8 *An inequality $a_j^T x \leq b_j$ is a valid inequality for $P \subseteq \mathbb{R}^n$ if $a_j^T x \leq b_j$ for all $x \in P$.*

Definition 3.9 (i) *F defines a face of the polyhedron P if $F = \{x \in P : a_j^T x = b_j\}$ for some valid inequality $a_j^T x \leq b_j$ of P . $F \neq \emptyset$ is said to be a proper face of P if $F \neq P$.*

(ii) *F is a facet of P if F is a face of P and $\dim(F) = \dim(P) - 1$.*

Definition 3.10 *A point $x \in P$ is called an interior point of P if $A^T x < b$.*

We also need the following results from integer programming [16, pp. 142–144] and [15, pp. 85–92].

Proposition 3.11 [15, Corollary 2.5] *A polyhedron is full-dimensional if and only if it has an interior point.*

Theorem 3.12 [16, Theorem 9.1] *If P is a full-dimensional polyhedron, it has a unique minimal description*

$$P = \{x \in \mathbb{R}^n : a_i^T x \leq b_i, \quad i = 1, 2, \dots, m\},$$

where each inequality is unique to within a positive multiple.

Corollary 3.13 [16, Proposition 9.2] *If P is full-dimensional, a valid inequality $a_j^T x \leq b_j$ is necessary in the description of P if and only if it defines a facet of P .*

Corollary 3.13 means that the following concepts are equivalent for $\Omega(x_k, \varepsilon)$ defined in (11).

- The j th inequality $a_j^T x \leq b_j$ defines a facet of $\Omega(x_k, \varepsilon)$.
- The j th inequality $a_j^T x \leq b_j$ is necessary (nonredundant) in description of $\Omega(x_k, \varepsilon)$, or in other words,

$$\Omega(x_k, \varepsilon) \subsetneq \Omega_j(x_k, \varepsilon). \quad (12)$$

Our approach for identifying nonredundant constraints is based primarily on the following proposition.

Proposition 3.14 *Let a working index set $I(x_k, \varepsilon)$ be given. An inequality $a_j^T x \leq b_j$, $j \in I(x_k, \varepsilon)$, is nonredundant in the description of $\Omega(x_k, \varepsilon)$ if and only if either $I(x_k, \varepsilon) = \{j\}$ or there exists $\bar{x} \in \mathbb{R}^n$ such that $a_j^T \bar{x} = b_j$ and $a_i^T \bar{x} < b_i$ for all $i \in I(x_k, \varepsilon) \setminus \{j\}$.*

Proof. Since the case $I(x_k, \varepsilon) = \{j\}$ is trivial, we give the proof for the case when $I(x_k, \varepsilon) \setminus \{j\} \neq \emptyset$.

Necessity. Since the inequality $a_j^T x \leq b_j$ is nonredundant, then, by (12), there exists $x^* \in \mathbb{R}^n$ such that $a_i^T x^* \leq b_i$ for all $i \in I(x_k, \varepsilon) \setminus \{j\}$, and $a_j^T x^* > b_j$. By Proposition 3.11, there exists an interior point $\hat{x} \in \Omega(x_k, \varepsilon)$ such that $a_i^T \hat{x} < b_i$ for all $i \in I(x_k, \varepsilon)$. Thus on the line between x^* and \hat{x} there is a point $\bar{x} \in \mathbb{R}^n$ satisfying $a_j^T \bar{x} = b_j$ and $a_i^T \bar{x} < b_i$ for all $i \in I(x_k, \varepsilon) \setminus \{j\}$.

Sufficiency. Let $\hat{x} \in \Omega(x_k, \varepsilon)$ be an interior point, i.e., $a_i^T \hat{x} < b_i$ for all $i \in I(x_k, \varepsilon)$. Since there exists $\bar{x} \in \mathbb{R}^n$ such that $a_j^T \bar{x} = b_j$ and $a_i^T \bar{x} < b_i$ for all $i \in I(x_k, \varepsilon) \setminus \{j\}$, then there exists $\delta > 0$ such that $\tilde{x} = \bar{x} + \delta(\bar{x} - \hat{x})$ satisfies $a_j^T \tilde{x} > b_j$ and $a_i^T \tilde{x} \leq b_i$, $i \in I(x_k, \varepsilon) \setminus \{j\}$. Therefore, (12) holds, and by Definition 3.1, the j th constraint is nonredundant. ■

Proposition 3.14 means that if the j th constraint, $j \in I(x_k, \varepsilon)$, is nonredundant, then there exists a feasible point $\bar{x} \in \Omega(x_k, \varepsilon)$ such that only this constraint holds with equality at \bar{x} .

Our approach for identifying redundant constraints is based primarily on the following theorem [11].

Theorem 3.15 *The j th constraint is redundant in system (2) if and only if the linear program,*

$$\text{maximize } a_j^T x, \quad \text{subject to } x \in \Omega_j, \quad (13)$$

has an optimal solution x^ such that $a_j^T x^* \leq b_j$.*

3.3 Approaches for identifying redundant and nonredundant constraints

We now outline two approaches for identifying redundancy in the constraint set: a projection method for identifying nonredundant constraints and a linear programming (LP) approach for identifying redundant ones. The LP approach, which is based on Theorem 3.15, is described in [11]. In Section 4, we will explain in more detail how these ideas are implemented in the class of GPS algorithm for linearly constrained problems, even in the presence of degeneracy.

3.3.1 A projection method

The main idea of the projection method we propose is the construction, if possible, of a point \bar{x} such that $a_j^T \bar{x} = b_j$ and $a_i^T \bar{x} < b_i$ for all $i \in I(x_k, \varepsilon) \setminus \{j\}$. If such a point \bar{x} exists, then by Proposition 3.14, the j th constraint is nonredundant.

Recall that we defined in (9) a scaled copy \bar{A} of the matrix A and a scaled vector \bar{b} . We denote by $P_j(x_k)$, the projection of $x_k \in \mathbb{R}^n$ onto the hyperplane $H_j = \{x \in \mathbb{R}^n : \bar{a}_j^T x = \bar{b}_j\}$. Assume that $x_k \in \Omega$. Then by (7) and by $\|\bar{a}_j\| = 1$,

$$P_j(x_k) = x_k + \bar{a}_j(\bar{b}_j - \bar{a}_j^T x_k). \quad (14)$$

The following proposition is the main one for the projection method.

Proposition 3.16 *Let $x_k \in \Omega$ and let a working index set $I(x_k, \varepsilon)$ be given. An inequality $a_j^T x \leq b_j$, $j \in I(x_k, \varepsilon)$, is nonredundant in the description of $\Omega(x_k, \varepsilon)$ if*

$$\bar{a}_i^T P_j(x_k) < \bar{b}_i \quad \text{for all } i \in I(x_k, \varepsilon) \setminus \{j\}, \quad (15)$$

where $P_j(x_k)$ is the projection of x_k onto H_j .

Proof. The proof follows from Proposition 3.14. ■

Proposition 3.16 allows us to very quickly classify the j th constraint as nonredundant if (15) holds for all $i \in I(x_k, \varepsilon) \setminus \{j\}$, where $P_j(x_k)$ in (15) is obtained from (14). The only drawback is that it identifies nonredundant constraints and not redundant ones.

3.3.2 The linear programming approach

If some constraints have not been identified by the projection method, we can apply another approach based on Theorem 3.15 to identify redundant and nonredundant constraints. It follows from Theorem 3.15 that all redundant and nonredundant constraints could be conclusively identified by solving n LP problems of the form given in (13). While doing so is clearly more expensive than the projection method given in Section 3.3.1, it could be accomplished during the initialization step of GPS (*i.e.*, before the GPS iteration sequence begins), at a cost of solving n LP problems. This is possible because redundancy of linear constraints is independent of the location of the current iterate. However, the projection method could be advantageous when many linear constraints are present (which is often the case with redundant constraints), or when dealing with linear constraints formed by linearizing nonlinear ones. In the latter case, redundancy would depend upon location in the domain, since the linear constraints would change based on location.

Different methods in the context of the LP approach are described in [12]. They include some very special propositions involving slack variables that simplify and reduce the computational cost of the numerical solution of the LP problem (13). We refer the reader to [12] for a more detailed discussion of these issues.

4 Construction of the set of generators

The purpose of this section is to provide a detailed algorithm for constructing the set of directions D_k introduced in Section 2, even in the presence of degenerate constraints.

Let some scalar $\varepsilon > 0$ be given, and let \bar{a}_i^T be the i th row of the matrix \bar{A}^T in (9). At the current iterate x_k , we construct the working index set $I(x_k, \varepsilon)$ such that

$$0 \leq \bar{b}_i - \bar{a}_i^T x_k \leq \varepsilon \iff i \in I(x_k, \varepsilon).$$

The last inequality means that every constraint that is active at x_k or at some point near x_k appears in $I(x_k, \varepsilon)$. In [1], the authors suggest not setting ε so small that Δ_k is made small by approaching the boundary too closely before including conforming directions that allow the iterates to move along the boundary of Ω . A good discussion of how to choose ε can be found in [6].

Without loss of generality, we assume that $I(x_k, \varepsilon) = \{1, 2, \dots, m\}$ for $m \geq 2$. This avoids more cumbersome notation, like $I(x_k, \varepsilon) = \{i_1(x_k, \varepsilon), \dots, i_m(x_k, \varepsilon)\}$. Furthermore, we denote by B_k , the matrix whose columns are the columns of A corresponding to the indices $I(x_k, \varepsilon) = \{1, \dots, m\}$; *i.e.*,

$$B_k = [a_1, \dots, a_m]. \tag{16}$$

4.1 Classification of degeneracy at the current iterate

Let the matrix B_k be defined by (16). At the current iterate x_k , the matrix B_k satisfies one of the following conditions:

- *nondegenerate case*: B_k has full rank;
- *degenerate redundant case*: B_k does not have full rank, and the nonredundant constraints are linearly independent;
- *degenerate nonredundant case*: B_k does not have full rank, and the nonredundant constraints are linearly dependent.

The last condition is illustrated by the following example provided by Charles Audet.

Example 4.1 *Suppose that the feasible region Ω (see (2)), shown in Figure 4, is defined by the following system of inequalities:*

$$\begin{aligned} x_1 - 2x_2 - 2x_3 &\leq 0 \\ -2x_1 + x_2 - 2x_3 &\leq 0 \\ -2x_1 - 2x_2 + x_3 &\leq 0 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \end{aligned} \tag{17}$$

If $x_k \in \mathbb{R}^3$ is near the origin, all six constraints are active, linearly dependent, and nonredundant. The matrix B_k is given as

$$B_k = \begin{pmatrix} 1 & -2 & -2 & -1 & 0 & 0 \\ -2 & 1 & -2 & 0 & -1 & 0 \\ -2 & -2 & 1 & 0 & 0 & -1 \end{pmatrix}.$$

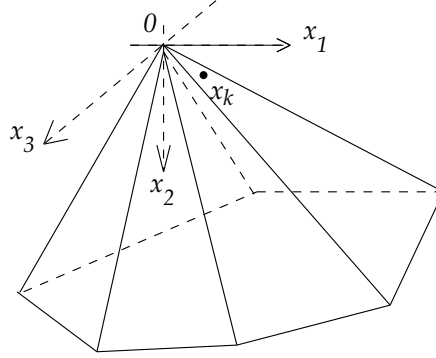


Figure 4: An illustration of the degenerate nonredundant case shown in Example 4.1.

4.2 Set of generators

Following [2], we define the cone $K(x_k, \varepsilon)$ as the cone generated by the normals to the ε -active constraints, and $K^\circ(x_k, \varepsilon)$ as its polar:

$$K^\circ(x_k, \varepsilon) = \{w \in \mathbb{R}^n : a_i^T w \leq 0 \quad \forall i \in I(x_k, \varepsilon)\}. \quad (18)$$

This cone can also be expressed as a finitely generated cone [19]. To see this, first consider the following definition.

Definition 4.2 A set $V = \{v_1, v_2, \dots, v_r\}$ is called a set of generators of the cone K defined by (18) if the following conditions hold:

1. Every vector $v \in K$ can be expressed as a nonnegative linear combination of vectors in V .
2. No proper subset of V satisfies 1.

Thus, given Definition 4.2, we can express $K^\circ(x_k, \varepsilon)$ as

$$K^\circ(x_k, \varepsilon) = \{w \in \mathbb{R}^n : w = \sum_{j=1}^r \lambda_j v_j, \lambda_j \geq 0, v_j \in \mathbb{R}^n, j = 1, \dots, r\}, \quad (19)$$

where $V = \{v_1, v_2, \dots, v_r\}$ is the set of generators for $K^\circ(x_k, \varepsilon)$.

The key idea, which was first suggested by May in [5] and applied to GPS in [2], is to include in D_k the generators of the cone $K^\circ(x_k, \varepsilon)$. Hence, the problem of construction of the set D_k reduces to the problem of constructing generators $\{v_1, \dots, v_r\}$ of the cone $K^\circ(x_k, \varepsilon)$ and then completing them to a positive spanning set for \mathbb{R}^n .

The following proposition means that it is sufficient to construct the set of generators only for nonredundant constraints.

Proposition 4.3 *Let $I(x_k, \varepsilon)$ be the set of indices of the ε -active constraints at $x_k \in \mathbb{R}^n$. Let $I_N(x_k, \varepsilon) \subseteq I(x_k, \varepsilon)$ be the subset of indices of the nonredundant constraints that define $\Omega(x_k, \varepsilon)$. Let the cone $K^\circ(x_k, \varepsilon)$ be defined by (18), and let the cone $K_N^\circ(x_k, \varepsilon)$ be given by*

$$K_N^\circ(x_k, \varepsilon) = \{w \in \mathbb{R}^n : a_i^T w \leq 0 \quad \forall i \in I_N(x_k, \varepsilon)\}.$$

If $\{v_1, \dots, v_p\}$ is a set of generators for $K_N^\circ(x_k, \varepsilon)$, then it is also a set of generators for $K^\circ(x_k, \varepsilon)$.

Proof. The proof of this proposition follows from Corollary 3.13. ■

Pattern search requires that iterates lie on a rational lattice [2]. To ensure this, Lewis and Torczon [2] require that the constraint matrix A^T in (2) have rational entries, in which case, they prove the existence of rational generators for the cones $K^\circ(x_k, \varepsilon)$, which, with the rational mesh size parameter Δ_k , ensures that GPS iterates lie on a rational lattice.

Moreover, for the case of linearly independent active constraints, Lewis and Torczon [2] proposed constructing the set of generators for all the cones $K^\circ(x_k, \varepsilon)$, $0 \leq \varepsilon \leq \delta$, as follows:

Theorem 4.4 *Suppose that for some δ , $K(x, \delta)$ has a linearly independent set of rational generators V . Let N be a rational positive basis for the null space of V^T . Then, for any ε , $0 \leq \varepsilon \leq \delta$, a set of rational generators for $K^\circ(x, \varepsilon)$ can be found among the columns of N , $V(V^T V)^{-1}$, and $-V(V^T V)^{-1}$.*

The matrix N can be constructed by taking columns of the matrices $\pm(I - V(V^T V)^{-1}V^T)$ [2].

Recall that we use the scaled matrix \bar{A} defined in (9) to determine ε -active, redundant, and nonredundant constraints. Then we use the result stated in Theorem 4.4 together with rational columns of A , which correspond to the nonredundant and ε -active constraints, to obtain a set of rational generators.

A set of generators, which may be irrational in exact arithmetic, can also be found by using the QR factorization of the matrix V . The following corollary shows how to use the QR factorization of V to construct the generators for all the cones $K^\circ(x_k, \varepsilon)$, $0 \leq \varepsilon \leq \delta$. Recall that the full QR factorization of V can be represented as

$$V = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 & R_2 \\ 0 & 0 \end{bmatrix}, \quad (20)$$

where R_1 is upper triangular, $\text{rank}(R_1) = \text{rank}(V)$, and the columns of Q_1 form an orthonormal basis for the space spanned by the columns of V , while the columns of Q_2 constitute an orthonormal basis for null space of V^T .

Corollary 4.5 *Suppose that for some δ , $K(x, \delta)$ has a linearly independent set of rational generators V . Then, for any ε , $0 \leq \varepsilon \leq \delta$, a set of generators for $K^\circ(x, \varepsilon)$ can be found among the columns of Q_2 , $Q_1 R_1 (R_1^T R_1)^{-1}$, and $-Q_1 R_1 (R_1^T R_1)^{-1}$.*

Proof. By substituting $V = QR$ and using the properties of the matrices in the QR factorization, we obtain

$$V(V^T V)^{-1} = QR((QR)^T(QR))^{-1} = QR(R^T Q^T QR)^{-1} = QR(R^T R)^{-1}. \quad (21)$$

By applying Theorem 4.4 and by taking into account that columns of Q_2 span the null space of V^T , we obtain the statement of the corollary. \blacksquare

From the theoretical point of view, a set of generators obtained by using Corollary 4.5 may be irrational since an implementation of the QR decomposition involves calculation of square roots. This would violate theoretical assumptions required for convergence of pattern search. However, since V is rational, both sides of (21) must also be rational. Therefore, by Corollary 4.5, any generators with irrational elements would be found in the matrix Q_2 . But in the degenerate case, Q_2 will often be empty, since it represents a positive spanning set for the null space of V^T , and most examples of degeneracy occur when the number of ε -active constraints exceeds the number of variables. Furthermore, since we use floating point arithmetic in practice, irrational generators would be represented as rational approximations. This has the effect of generating a slightly different cone. Thus, it would be enough to ensure convergence, but to a stationary point of a slightly different problem. However, the error experienced in representing an irrational number as rational is probably smaller than the typical roundoff error associated with LU factorization.

4.3 The nonredundant degenerate case

Perhaps the most difficult case to handle is the one in which the ε -active constraints at x_k are nonredundant, but linearly dependent. This can happen, in particular, when there are more ε -active constraints than variables, as is the case in Example 4.1. The difficulty of this case lies in the fact that the number of directions required to generate the tangent cone can become large.

Let $S_k = \{a_1, a_2, \dots, a_{p_k}\}$ denote the set of vectors corresponding to the ε -active nonredundant constraints at x_k . Price and Coope [10] showed that, in order to construct D_k , it is sufficient to identify the tangent cone generators of all maximally linearly independent subsets of S_k . For S_k with $r_k = \text{rank}(S_k)$, we can estimate the number s_k of these subsets, by

$$s_k = \frac{p_k!}{r_k!(p_k - r_k)!}. \quad (22)$$

Thus, in order to identify the entire set of tangent cone generators, we would have to consider s_k different sets of positive spanning directions, where s_k could become quite large. While some efficient vertex enumeration techniques (e.g., [7, 20, 21]) have been employed in pattern search algorithms [6, 9], we now present a potentially less expensive alternative approach – first in general, and then followed by some specific instances that can be implemented in practice.

4.3.1 Partially conforming generator sets

In our approach, we choose a subset of r_k linearly independent elements of S_k and store them as columns of B_k . Based on the methods described in Section 3.3.1, we can construct a set of generators for the cone defined only by a subset of the constraints represented in S_k . Furthermore, we require B_k to change at each *unsuccessful* iteration so that, in the limit, each constraint that is active at the limit point \hat{x} has been used infinitely often in constructing directions. Since the set of tangent cone generators is finite, the ordering scheme for ensuring this is straightforward.

This approach is essentially equivalent to using all the tangent cone generators, except that it is spread out over more than one iteration. The advantage is that it keeps the size of the poll set no larger than it would be in the nondegenerate case. However, the drawback is that we no longer have a full set of directions that conform to the geometry of Ω , which is an important hypothesis in the statement of Theorem 2.4.

The proof of Theorem 2.4, given in [1], relies on two crucial ideas; namely, Lemma 2.3 and the use of conforming directions. Under the proposed method of handling degenerate nonredundant constraints, Lemma 2.3 still applies, but Theorem 2.4 cannot be applied, since not all the tangent cone generators are used at each iteration. We introduce the following theorem, which establishes the same result as Theorem 2.4, but with a different hypothesis and a proof that is essentially identical (see [1]).

Theorem 4.6 *Let $\hat{x} \in \Omega$ be the limit point of a refining subsequence $\{x_k\}_{k \in K}$. Under Assumptions A1–A3, if f is strictly differentiable at \hat{x} and all generators of the tangent cone $T_\Omega(\hat{x})$ are used infinitely often in K , then $\nabla f(\hat{x})^T \omega \geq 0$ for all $\omega \in T_\Omega(\hat{x})$, and so $-\nabla f(\hat{x}) \in N_\Omega(\hat{x})$. Thus, \hat{x} is a Karush-Kuhn-Tucker point.*

Proof. Lemma 2.3 and the strict differentiability of f at \hat{x} ensure that $\nabla f(\hat{x})^T d \geq 0$ for all $d \in D \cap T_\Omega(\hat{x})$. Since D includes all the tangent cone generators, and each is used infinitely often in K , it follows that every $\omega \in T_\Omega(\hat{x})$ can be represented as a nonnegative linear combination of $D \cap T_\Omega(\hat{x})$; thus, $\nabla f(\hat{x})^T \omega \geq 0$ for all $\omega \in T_\Omega(\hat{x})$. To complete the proof, we multiply both sides by -1 and conclude that $-\nabla f(\hat{x}) \in N_\Omega(\hat{x})$. ■

4.3.2 Sequential and Random Selection

While the new hypothesis of Theorem 4.6 is weaker and makes the result more general than Theorem 2.4, its enforcement requires modification of the algorithm. The enumeration scheme mentioned above will not only ensure that the tangent cone generators get used infinitely often, but also that they get used infinitely often *in the refining subsequence*. Before specifying the enumeration schemes, we introduce the following lemma to establish an important connection between the constraints and tangent cone generators.

Lemma 4.7 *Let \hat{x} be the limit of a subsequence of GPS iterates, and let \hat{S} and \hat{D} be the sets of active constraints and tangent cone generators, respectively, at \hat{x} . If every constraint in \hat{S} is used to form tangent cone generators infinitely often in the subsequence, then every tangent cone generator in \hat{D} is also used infinitely often in the same subsequence.*

Proof. Let $\hat{S}_j, j = 1, \dots, s$ be maximally linearly independent subsets of \hat{S} , such that $\hat{S} = \bigcup_{j=1}^s \hat{S}_j$.

Furthermore, let $D(\hat{S}_j)$ denote the set of tangent cone generators produced by only the constraints in \hat{S}_j . Price and Coope [10] show that $\hat{D} \subset \bigcup_{j=1}^s D(\hat{S}_j)$. Thus, if \hat{S}_j is used infinitely often, then $D(\hat{S}_j)$ is used infinitely often, and if every $\hat{S}_j, j = 1, \dots, s$, is used infinitely often, then every direction in \hat{D} is used infinitely often. ■

We now give two examples of approaches that generate directions satisfying the hypotheses of Theorem 4.6, followed by convergence theorems for each.

Sequential Selection: At each iteration k , order the s_k subsets of r_k linearly independent elements of S_k as $S_k^i, i = 1, \dots, s_k$, and use subset $S_k^j, j = 1 + k \bmod s_k$, at iteration numbers m_{k+1}, \dots, m_{k+1} , where $K = \{m_k\}_{k=1}^\infty$ denotes the indices of the unsuccessful iterations and $m_0 = 0$. Anytime that S_k changes, restart the ordering process, noting that $S_k = \hat{S}$ for all sufficiently large k .

Random Selection: At each iteration k , randomly select (with uniform probability) r_k linearly independent ε -active constraints to form tangent cone generators.

Theorem 4.8 *Let \hat{x} be the limit of a refining subsequence $\{x_k\}_{k \in K}$ in which the set of nonredundant binding constraints at \hat{x} is linearly dependent. If search directions are obtained by Sequential Selection whenever the elements of S_k are linearly dependent, then all tangent cone generators at \hat{x} will be used infinitely often in K .*

Proof. Without loss of generality, assume that k is sufficiently large so that $S_k = \hat{S}$ is fixed, where \hat{S} is the set of active constraints at \hat{x} . Since subset $S_k^j, j = 1 + k \bmod s_k$, is used at iteration $m_{k+1} \in K$ (an infinite sequence), each $S_k^j \subset \hat{S}$ is used infinitely often in K . The result follows from Lemma 4.7. ■

Theorem 4.9 *Let \hat{x} be the limit of a refining subsequence $\{x_k\}_{k \in K}$ in which the set of nonredundant binding constraints at \hat{x} is linearly dependent. If search directions are obtained by Random Selection whenever the elements of S_k are linearly dependent, then with probability 1, all tangent cone generators at \hat{x} will be used infinitely often in K .*

Proof. For any nonredundant active constraint at \hat{x} , let P_k denote the probability that the constraint is randomly selected at iteration k . Then for sufficiently large k , the set S_k is fixed with p_k elements (corresponding to the active constraints at \hat{x}), and $P_k = \frac{r_k}{p_k}$. Then the probability that the constraint is selected infinitely often in any infinite subsequence M of iterates (with sufficiently large k) is equal to $1 - \prod_{k \in M} (1 - P_k) = 1 - \prod_{k \in M} \frac{p_k - r_k}{p_k} = 1$. The result then follows from Lemma 4.7. ■

Furthermore, this by no means exhausts the possibilities for choosing tangent cone generators when nonredundant constraints are linearly dependent. Considering that the projection method measures distance to each constraint boundary, one promising alternative is to select the closest $n-1$

constraints (with ties broken arbitrarily), plus one more constraint obtained by either sequential or random selection. The latter constraint allows the theory in the previous two theorems to hold, while offering an intelligent heuristic in selecting those constraints that are closer to the current iterate. Choosing the closest constraints is equivalent to reducing ε at each iteration so that fewer constraints are flagged as ε -active.

We recognize that the use of partially conforming sets will generate some infeasible directions. In fact, in highly degenerate cases where the number of ε -active constraints greatly exceeds the number of variables, the percentage of directions that are infeasible may be very high. This partly explains why Lewis, Shepherd, and Torczon [9] observe a large discrepancy between the number of possible generators and the number actually computed by computational geometry methods. The cost of including infeasible directions is negligible, since infeasible points are not evaluated, but premature termination can occur in practice if the mesh size shrinks too much before the right directions are selected.

On the other hand, if the incumbent is a mesh local optimizer (which would not be known beforehand) and a degenerate condition exists there, then a computational geometry approach, such as [20] or [21], would evaluate points in all the generating directions, perhaps at great expense, while the smaller sets of directions generated by sequential or random selection would result in fewer function evaluations. This is an important consideration because the class of problems we target includes those with expensive function evaluations. In fact, even if a computational geometry approach is used to efficiently identify all of the tangent cone generators, the expense of actually evaluating the objective function at the resulting POLL points may still be considerable. In these cases, we can apply sequential or random selection to the set generated by the computational geometry approach to possibly save function evaluations without having to worry too much about premature termination. An example of this phenomenon is shown in Section 4.4.3.

4.4 An algorithm for constructing the set of generators

In this section, we present an algorithm for constructing a set of generators for the cone $K^\circ(x_k, \varepsilon)$ at the current iterate x_k for a given parameter ε .

4.4.1 Comments on the algorithm

The algorithm consists of two main parts. In the first part, we determine the set of indices of the nonredundant ε -active constraints, $I_N(x_k, \varepsilon) \subseteq I(x_k, \varepsilon)$, and form the matrix B_N whose columns are the columns of A corresponding to the indices in $I_N(x_k, \varepsilon)$. We use information about the set $I_N(x_k, \varepsilon)$ from the previous iterations of the GPS algorithm. Namely, we put into the set $I_N(x_k, \varepsilon)$ all indices that correspond to the ε -active constraints at the current iterate and that were detected as indices of the nonredundant constraints at the previous iterations of the algorithm. In the second part of the algorithm, we construct the set of generators D_k required by GPS and by Theorem 2.4.

First, we try to identify the nonredundant active constraints. If the matrix B_k defined by (16) has full rank, then all ε -active constraints are nonredundant, $I_N(x_k, \varepsilon) = I(x_k, \varepsilon)$, and $B_N = B_k$. If the matrix B_k does not have full rank and we have indices that have not been classified at the previous iterations of the algorithm, we propose using two steps in succession.

The first strategy is intended to determine nonredundant constraints cheaply by applying the projection method described in section 3.3.1. By Proposition 3.16, if the projection $P_j(x_k)$ of the current iterate x_k onto the hyperplane $H_j = \{x \in \mathbb{R}^n : \bar{a}_j^T x = \bar{b}_j\}$ is feasible and only the j th constraint holds with equality at $P_j(x_k)$, then the j th constraint is nonredundant, and we can put index j into the set $I_N(x_k, \varepsilon)$. If some constraints have not been identified by the projection method, we can either apply the projection method with some other point $\tilde{x} \neq x_k$ or apply the second strategy.

The second strategy is intended to classify redundant and nonredundant constraints among those that have not already been determined as nonredundant by the projection method. To identify each constraint, the approach outlined in [11] and in Section 3.15 is applied. If the number of constraints to be identified is too large, we can skip an application of this strategy and construct a set of generators using the set $I_N(x_k, \varepsilon)$ obtained from the first strategy. Then, while performing the POLL step, if we find some point $\bar{x} = x_k + \Delta \bar{d}$, where \bar{d} is some column of D_k , such that $a_j^T \bar{x} > b_j$ and $a_i^T \bar{x} \leq b_i$ for all $i \in I(x_k, \varepsilon) \setminus \{j\}$, we can conclude that $\Omega(x_k, \varepsilon) \subsetneq \Omega_j(x_k, \varepsilon)$. Hence, by Corollary 3.13, the j th constraint is nonredundant, and we add j to the set $I_N(x_k, \varepsilon)$.

Once we have specified all redundant and nonredundant constraints, we compose the matrix B_N of those columns of A that correspond to nonredundant constraints. The rank of B_N can be determined by QR factorization. If B_N has full rank, then we construct the set of generators using QR or LU factorization. If B_N does not have full rank, we construct the set of generators from a set of linearly independent columns of B_N , and as the iteration sequence progresses, we invoke one of the methods described in Section 4.3 to ensure that all maximally linearly independent subsets get used infinitely often.

4.4.2 Algorithm

We denote the set of indices of the nonredundant ε -active constraints at x_k by $I_N(x_k, \varepsilon)$. Thus, for $j \in I(x_k, \varepsilon)$,

1. if $j \in I_N(x_k, \varepsilon)$, then the inequality $a_j^T x \leq b_j$ is nonredundant; and
2. if $j \in I(x_k, \varepsilon) \setminus I_N(x_k, \varepsilon)$, then the inequality $a_j^T x \leq b_j$ is redundant.

We use $I_N \subseteq I$ to denote the set of indices that are detected as nonredundant at some iteration of the algorithm. Thus, $I_N = \emptyset$ at the beginning of the algorithm.

We denote the rational matrix in (2) by A^T and the scaled matrix defined in (9) by \bar{A}^T . The matrix B_k is defined by (16) and is composed of columns a_j of A , where $j \in I(x_k, \varepsilon)$, while the matrix B_N is composed of those columns of A whose indices are in the set $I_N(x_k, \varepsilon)$. Thus, the columns of B_N are those vectors normal to the nonredundant constraints.

Algorithm for constructing the set of generators D_k .

Let the current iterate $x_k \in \mathbb{R}^n$ and a parameter $\varepsilon > 0$ be given.

% Part I: Constructing the set $I_N(x_k, \varepsilon)$

% Construct the working index set $I(x_k, \varepsilon)$

```

for  $i = 1$  to  $|I|$ 
  if  $0 \leq \bar{b}_i - \bar{a}_i^T x_k \leq \varepsilon$ 
     $I(x_k, \varepsilon) \leftarrow I(x_k, \varepsilon) \cup \{i\}$ 
     $B_k \leftarrow [B_k, a_i]$ 
  endif
endfor

if  $\text{rank}(B_k) = |I(x_k, \varepsilon)|$       % if all constraints are nonredundant
   $I_N(x_k, \varepsilon) \leftarrow I(x_k, \varepsilon)$ 
   $B_N \leftarrow B_k$ 

else
  % using information from previous iterations
  for each  $j \in \{I(x_k, \varepsilon) \cap I_N\}$ 
     $I_N(x_k, \varepsilon) \leftarrow I_N(x_k, \varepsilon) \cup \{j\}$ 
     $B_N \leftarrow [B_N, a_j]$ 
  endfor

  % Identification of the nonredundant and redundant constraints
  for each  $j \in \{I(x_k, \varepsilon) \setminus I_N(x_k, \varepsilon)\}$ 
    % the first strategy
     $P_j(x_k) = x_k + \bar{a}_j(\bar{b}_j - \bar{a}_j^T x_k)$       % see Lemma 3.4
    if  $\bar{a}_i^T P_j(x_k) < \bar{b}_i$  for all  $i \in I \setminus \{j\}$ 
       $I_N(x_k, \varepsilon) \leftarrow I_N(x_k, \varepsilon) \cup \{j\}$ 
       $B_N \leftarrow [B_N, a_j]$ 
       $I_N \leftarrow I_N \cup \{j\}$ 
    else
      % the second strategy
      solve LP problem (3.15) for  $x^*$ 
      if  $a_j^T x^* \leq b_j$       % the  $j$ th constraint is redundant
        remove  $a_j x \leq b_j$  from  $\Omega$ 
         $I \leftarrow I \setminus \{j\}$ 
         $I(x_k, \varepsilon) \leftarrow I(x_k, \varepsilon) \setminus \{j\}$ 
      else
        % the  $j$ th constraint is nonredundant
         $I_N(x_k, \varepsilon) \leftarrow I_N(x_k, \varepsilon) \cup \{j\}$ 
         $B_N \leftarrow [B_N, a_j]$ 
         $I_N \leftarrow I_N \cup \{j\}$ 
      endif
    endif
  endfor
endif

```

% Part II: Constructing the set of generators D_k

```

r = rank(B_N)
if r ≠ |I_N(x_k, ε)|      % degenerate case
    B_N ← [r linearly independent columns of B_N]  % see Section 4.3
endif
V = B_N
D_1 ← V(V^T V)^-1
D_2 ← I - V(V^T V)^-1 V^T
D = [D_1, D_2, -D_1, -D_2]

```

As discussed in Section 4.2, the construction of the directions in D , in practice, can be done making use of either LU decomposition, as suggested by Lewis and Torczon [2], or by the more efficient QR factorization approach presented in Section 4.2. In the latter case, D_1 and D_2 are computed according to Corollary 4.5.

We should point out that, in practice, the choice of ε can have a significant effect on numerical performance. If the value is set too low, then the mesh size may become very small before appropriate conforming directions are generated. If this happens, the algorithm may then progress along a new conforming direction, but with the significantly reduced mesh size, resulting in a larger number of function evaluations. On the other hand, too large a value may mark too many constraints as active. This could result in otherwise good directions being replaced by worse ones, and even a false detection of degeneracy, costing additional unnecessary function evaluations. Lewis, Shepherd, and Torczon [9] suggest tying the value of ε directly to that of Δ_k .

4.4.3 Numerical Illustrations

To illustrate the algorithm, we first formed five test problems with varying numbers of variables and redundant linear constraints to test the ability of our approach to accurately construct the set $I_N(x_k, \varepsilon)$ of nonredundant constraints. In doing so, we chose a trial point x_k close to several of the constraints and tested the ability of our algorithm to identify the nonredundant ones. The test problems are described as follows:

Problem 1 Same as the problem given in (17), but with the current iterate at $(0.1, 0.1, 0.1)^T$.

Problem 2 The following problem with the current iterate at $(0.01, -0.01, -0.01, -0.00001, 0.01)^T$:

$$\begin{aligned}
 -x_1 + x_2 &\leq 0 \\
 x_1 + x_2 &\leq 1 \\
 x_2 + x_3 + x_4 &\leq 0 \\
 -x_2 + x_5 &\leq 5 \\
 -x_1 + x_2 &\leq 0 \\
 x_3 &\leq 0 \\
 -0.8x_1 + x_2 + x_3 &\leq 0.
 \end{aligned}$$

Problem 3 The following problem with the current iterate at $(0.01, 0.01, 0.01, 0.01, 0.01)^T$:

$$\begin{aligned}
 x_1 - 2x_2 - 2x_3 + x_4 &\leq 0 \\
 -2x_1 + x_2 - 2x_3 &\leq 0 \\
 -2x_1 - 2x_2 + x_3 &\leq 0 \\
 -x_1 &\leq 0 \\
 -x_2 &\leq 0 \\
 -x_3 - x_5 &\leq 0 \\
 -x_4 - 0.1x_5 &\leq 0.
 \end{aligned}$$

Problem 4 Same as Problem 3, but with the current iterate at $(0.000001, 0.000001, 0.000001, 0.000001, 0.1)^T$.

Problem 5 Same as Problem 3, but with the current iterate at $(0.001, 0.001, 0.001, 0.001, 0.001)^T$.

We report results in Table 1, where each row corresponds to one of the five test problems (in the order presented), and where the number of variables is given in the first column. Columns 2 and 3 show the number of nonredundant and redundant constraints, respectively, with their sum representing the total number of constraints for each problem. The last two columns indicate how many of the constraints were identified as nonredundant, first by the projection method, and then by the LP approach if projection failed to identify all the nonredundant ones. As is shown in the table, the projection method identifies most of the nonredundant constraints, and with the LP method as a backup, all the constraints are correctly identified.

Table 1: Constructing the set $I_N(x_k, \varepsilon)$ at the current iterate x_k

Variables	Constraints		Detected as nonredundant	
	Nonredundant	Redundant	by Projection	by LP approach
3	6	0	6	
5	6	1	5	1
5	7	0	6	1
5	7	0	5	2
5	7	0	6	1

With this approach in place, the number of GPS iterations required for a problem with no redundant constraints will be no different than for a modified version of the same problem, in which any number of additional redundant constraints are added, since the algorithm detects and removes the redundant constraints at each iteration.

Finally, we coded up the random selection and sequential selection approaches for handling linearly dependent nonredundant ε -active constraints, as well as a naive full enumeration scheme, to numerically test the ideas posed in Section 4.3. We added this code to the NOMADm software [22] and tested the approaches on two problems, parameterized by the number of variables n and having $2n$ linear constraints that are all active at the origin. In both problems, the set of linear constraints

is constructed to be a generalization of Example 4.1. The only differences between the two problems are in the objective function and the initial point. The problems are described as follows:

Problem 6 The following problem with the initial point at $(0, 0, \dots, 0)^T$:

$$\begin{aligned} & \min_x \sum_{i=1}^n (x_i - 1)^2 \\ \text{s. t.} & \\ & x_1 - 2x_2 - 2x_3 - \dots - 2x_{n-1} - 2x_n \geq 0 \\ & -2x_1 + x_2 - 2x_3 - \dots - 2x_{n-1} - 2x_n \geq 0 \\ & \quad \vdots \\ & -2x_1 - 2x_2 - 2x_3 - \dots - 2x_{n-1} + x_n \geq 0 \\ & x_i \geq 0, \quad i = 1, 2, \dots, n. \end{aligned}$$

Problem 7 The following problem with the initial point at $(3, 3, \dots, 3)^T$:

$$\begin{aligned} & \min_x \sum_{i=1}^n x_i^2 \\ \text{s. t.} & \\ & x_1 - 2x_2 - 2x_3 - \dots - 2x_{n-1} - 2x_n \geq 0 \\ & -2x_1 + x_2 - 2x_3 - \dots - 2x_{n-1} - 2x_n \geq 0 \\ & \quad \vdots \\ & -2x_1 - 2x_2 - 2x_3 - \dots - 2x_{n-1} + x_n \geq 0 \\ & x_i \geq 0, \quad i = 1, 2, \dots, n. \end{aligned}$$

In Problem 6, the initial point was chosen to be the origin in order to study the performance of each approach in moving off a non-optimal degenerate point toward the minimizer at $(1, 1, \dots, 1)^T$. In Problem 7, the initial point of $(3, 3, \dots, 3)^T$ was chosen so that convergence to the degenerate optimal point (the origin) could be studied (and so that no iterate lands exactly at the minimizer).

The scenarios we studied were for $n = 6, 7, 8$ with and without mesh coarsening (i.e., $\Delta_{k+1} = 2\Delta_k$ or $\Delta_{k+1} = \Delta_k$, respectively, when an improved mesh point is found). The initial mesh size was chosen to be $\Delta_0 = 16$, and the GPS algorithm was run until the termination criterion, $\Delta_k < 10^{-4}$, was satisfied. An empty SEARCH step was used, and the POLL step employed the set of standard $2n$ directions, $D = D_k = \{\pm e_i\}$, whenever the incumbent was not sufficiently close to a constraint boundary.

Results for Problems 6 and 7 are given in Tables 2 and 3, respectively. Since the algorithm converged to the solution in all cases, we recorded the number of function evaluations performed as a measure of comparison. Sequential selection, random selection, and full enumeration are as described in Section 4.3. The numbers recorded for random selection are averages over 10 replications. To mitigate concerns that the performance of sequential selection might be too closely tied to the order in which constraints are given to the NOMADm software, we randomly reordered the constraints 20 times, and recorded the averages under the label of ‘‘Sequential Selection 2’’.

Table 2: Problem 6: Number of function evaluations needed for each method.

Method	No mesh coarsening			with mesh coarsening		
	$n = 6$	$n = 7$	$n = 8$	$n = 6$	$n = 7$	$n = 8$
Sequential Selection	208	253	300	221	281	332
Sequential Selection 2	213	269	296	241	331	347
Random Selection	1825	3323	1769	460	552	647
Full Enumeration	449	589	748	466	612	778

Table 3: Problem 7: Number of function evaluations needed for each method.

Method	No mesh coarsening			with mesh coarsening		
	$n = 6$	$n = 7$	$n = 8$	$n = 6$	$n = 7$	$n = 8$
Sequential Selection	154	273	253	218	281	340
Sequential Selection 2	171	243	256	208	258	278
Random Selection	139	227	239	191	244	305
Full Enumeration	800	1963	1365	2269	2838	3850
Dynamic Enumeration	135	157	206	139	191	239

The results for Problems 6 and 7 show that sequential selection performed significantly better than full enumeration in solving this set of problems. In Problem 6, the computational cost of full enumeration was higher because many unsuccessful iterations had to be performed before GPS could find an improved mesh point. In Problem 7, the cost was even higher for full enumeration because many trial points in feasible but poor directions had to be evaluated at each iteration as the iterates approached the solution. In this case, we made an extra set of runs for this problem, in which we changed the order in which poll points were evaluated so that a successful direction in one iteration was evaluated first in the next iteration. The results for this case are listed in Table 3 under the label of “Dynamic Enumeration”. Clearly, this strategy dramatically improved the performance of the full enumeration scheme, causing it to outperform the other strategies. However, in practice, the existence of a degenerate condition may not be known beforehand, and the use of dynamic poll ordering may not be the ideal strategy to use for all problems.

The main point that the results for Problems 6 and 7 make is that, even if a computational geometry approach is used to quickly and efficiently identify all of the tangent cone generators, it may still be more costly to evaluate the resulting trial points than to use Sequential (or even Random) Selection.

5 Nonlinearly constrained minimization

The goal of this section is to illustrate how the projection approach proposed in this paper can also be effective for handling degeneracy in nonlinearly constrained optimization problems. In doing so,

we should point out that our approach is different than that of [23] and [24] (and others cited in these papers). Both approaches use local information about the (twice continuously differentiable) objective and constraint functions to identify active constraints. Moreover, the focus in [24] is on distinguishing between strongly and weakly active constraints – the latter having Lagrange multiplier values of zero. In our case, we do not have multiplier values available, and even if we did, most direct search methods we might consider using, such as [25] and [26], can handle weakly active constraints transparently if constraint gradients are linearly independent.

We consider the nonlinearly constrained optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{subject to} \quad x \in \Omega = \{x \in \mathbb{R}^n : c_i(x) \leq 0, i = 1, \dots, q\}. \quad (23)$$

All constraint functions $c_i, i = 1, 2, \dots, q$, are assumed to be continuously differentiable, but their gradients may not be available. The algorithm in [26] uses constraint gradients, while the one in [25] uses only approximations. Our intent is to be as general as possible, so that the ideas presented here might be extendable to both algorithms, as well as other direct search methods.

Similar to Section 3, the region defined by all but the j -th constraint is given by

$$\Omega_j = \{x \in \mathbb{R}^n : c_i(x) \leq 0, i \in I \setminus \{j\}\},$$

where $I = \{1, 2, \dots, q\}$. Additionally, for $\delta > 0$, we define $U_\delta(x) = \{y \in \mathbb{R}^n : \|y - x\| \leq \delta\}$, and offer a definition of local redundancy (nonredundancy), in the sense that the constraints are locally nonredundant if they define the shape of the feasible region in some neighborhood of a point $x \in \mathbb{R}^n$. This is illustrated in Figure 5.

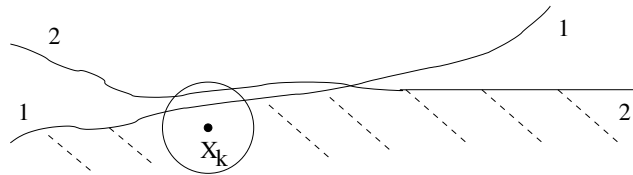


Figure 5: An illustration of a locally redundant constraint. Constraint 2 is locally redundant at x_k .

Definition 5.1 *The j th constraint $c_j(x) \leq 0$ is locally redundant at x in the description of Ω if, for some $\delta > 0$, $\Omega \cap U_\delta(x) = \Omega_j \cap U_\delta(x)$, and is locally nonredundant otherwise.*

Our main interest is in the problem of constructing search directions that conform to the boundary of Ω . First, we define constraint j as ε -active if $-\varepsilon \leq c_j(x) \leq 0$. For the iterate x_k at iteration k , we denote by $I(x_k, \varepsilon)$ the set of indices of the ε -active constraints at x_k ; namely,

$$I(x_k, \varepsilon) = \{j = 1, 2, \dots, q : -\varepsilon \leq c_j(x_k) \leq 0\},$$

and extend the following from similar definitions given in Section 3:

$$\begin{aligned} \Omega(x_k, \varepsilon) &= \{x \in \mathbb{R}^n : c_i(x) \leq 0, i \in I(x_k, \varepsilon)\}, \\ \Omega_j(x_k, \varepsilon) &= \{x \in \mathbb{R}^n : c_i(x) \leq 0, i \in I(x_k, \varepsilon) \setminus \{j\}\}, \quad j \in I(x_k, \varepsilon). \end{aligned}$$

If x_k is close to the boundary of Ω , then the set of directions should contain generators for the tangent cone $T_\Omega(x_k)$ for boundary points near x_k .

We assume that estimates $a_i^{(k)}$ of the gradients $\nabla c_i(x_k)$, $i = 1, 2, \dots, q$, are available. Thus, an estimate $C^{(k)}(I(x_k, \varepsilon))$ of the tangent cone $T_\Omega(x_k)$ is given by

$$C^{(k)}(I(x_k, \varepsilon)) = \{v \in \mathbb{R}^n : v^T a_i^{(k)} \leq 0 \quad \forall i \in I(x_k, \varepsilon)\}. \quad (24)$$

By Definition 4.2, each of these cones can be expressed as the set of nonnegative linear combinations of a finite number of generators, $\{v_j\}_{j=1}^p \subset \mathbb{R}^n$.

One of the main assumptions in [25] is that at each point x on the boundary of Ω , the gradients of the constraints active at x are linearly independent. By extending our ideas from previous subsections to the nonlinear case, this assumption can be relaxed.

The next proposition is simply an application of Proposition 4.3 to the cone defined by the linearized constraints, except that the index sets apply to nonlinear constraints. As a consequence, it is sufficient to construct the set of generators for only the locally nonredundant constraints.

Proposition 5.2 *Let $I_N(x_k, \varepsilon) \subseteq I(x_k, \varepsilon)$ be the subset of indices of the locally nonredundant constraints that define $\Omega(x_k, \varepsilon)$. Let the cone $C^{(k)}(I(x_k, \varepsilon))$ be defined by (24), and let the cone $C_N^{(k)}$ be given by $C_N^{(k)} = \{v \in \mathbb{R}^n : v^T a_i^{(k)} \leq 0 \quad \forall i \in I_N(x_k, \varepsilon)\}$. If $\{v_1, \dots, v_p\}$ is a set of generators for $C_N^{(k)}$, then it is also a set of generators for $C^{(k)}(I(x_k, \varepsilon))$.*

Proof. This follows directly from Corollary 3.13. ■

To extend the projection approach described in Section 3.3.1 for detecting locally nonredundant nonlinear constraints, we simply project onto a linearization of the constraint boundary, based on approximations to the constraint gradients at the current iterate; *i.e.*, we project onto the hyperplane $H_j = \{v \in \mathbb{R}^n : v^T a_j^{(k)} = 0\}$. Scaling the constraints similar to (9) and applying Lemma 3.4 yields a projection equation similar to (14); namely,

$$P_j(x_k) = x_k + \bar{a}_j^{(k)} c_j(x_k), \quad \bar{a}_j^{(k)} = \frac{a_j^{(k)}}{\|a_j^{(k)}\|}, \quad j = 1, 2, \dots, q. \quad (25)$$

If the generators of $C_N^{(k)}$ at iteration k are linearly independent, then they would all be included in the set of search directions for that iteration. Otherwise, the set of search directions would include a maximal linearly independent subset of the generators, selected in exactly the same manner as discussed in Section 4.3.

We omit a formal discussion of convergence, since any results would be dependent on the algorithm being used and on the details of its implementation. However, it appears safe to assume that any convergence results will require a certain degree of accuracy by the vectors $a_j^{(k)}$ as approximations to the constraint gradients $\nabla c_j(x_k)$.

We view these ideas as a natural extension of those of Section 3.3.1 that can achieve a significant cost savings over the LP approach. Recall from Section 3.3.2 that the expense of the LP

approach for linear constrained problems can be circumvented by performing it before the algorithm commences, since the redundancy of each constraint is independent of the location of the current iterate. However, this is not true for nonlinear constraints, in which case, the LP approach would have to be performed at every iteration, which is considerably more expensive than projection.

6 Concluding remarks

This paper fills an important gap in the pattern search literature, complementing the previous work of Lewis and Torczon [2] by rigorously treating the case of degenerate linear constraints. We have introduced an inexpensive projection method for identifying nonredundant constraints, which, when used in conjunction with a linear programming approach as a backup, can cheaply assess the redundancy of each constraint, and thus aid pattern search in computing directions that conform to the boundary of the feasible region. We believe that this approach has potential for being applied to other algorithms, such as those that make use of active sets.

For the case in which nonredundant ε -active constraints are linearly dependent, we have introduced an approach in which complete enumeration of tangent cone generators at each iteration is avoided by including only a subset of them, and then changing them at each unsuccessful iteration. We have proved that, in this case, all generators are used infinitely often in any refining subsequence, and that first-order convergence properties still hold. Numerical testing has shown that this approach can be less expensive than fully enumerating the tangent cone generators at each iteration. Finally, we have shown how our ideas can be extended to nonlinearly constrained optimization problems under similar degenerate conditions.

Acknowledgements

The research of the third author was supported in part by AFOSR F49620-01-1-0013, the Boeing Company, Sandia CSRI, ExxonMobil, the LANL Computer Science (LACSI) contract 03891-99-23, by the Institute for Mathematics and its Applications with funds provided by the National Science Foundation, and by funds from the Ordway Endowment at the University of Minnesota. This work was begun at the IMA, where Olga Brezhneva was a postdoctoral fellow, John Dennis was a long-term visitor, and Mark Abramson was a short-term visitor. We thank the IMA for providing such a fine atmosphere for collaboration. We also thank Charles Audet for many useful discussions.

The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, United States Government, or research sponsors.

References

- [1] Audet, C., and Dennis, Jr., J. E., 2003, Analysis of generalized pattern searches. *SIAM J. Optim.*, **13**(3), 889–903.

- [2] Lewis, R. M., and Torczon, V., 2000, Pattern search methods for linearly constrained minimization. *SIAM J. Optim.*, **10**(3), 917–941.
- [3] Clarke, F. H., 1990, *Optimization and Nonsmooth Analysis*. SIAM Classics in Applied Mathematics (Vol. 5) (Philadelphia: SIAM Publications).
- [4] Abramson, M. A., 2005, Second-order behavior of pattern search. *SIAM J. Optim.*, **16**(2), 515–530.
- [5] May, J. H., 1974, Linearly Constrained Nonlinear Programming: A Solution Method That Does Not Require Analytic Derivatives. PhD thesis, Yale University.
- [6] Kolda, T. G., Lewis, R. M., and Torczon, V., 2003, Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Rev.*, **45**(3), 385–482.
- [7] Avis, D. M., and Fukuda, K., 1992, A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comp. Geom.*, **8**(3), 295–313.
- [8] Avis, D. M., and Fukuda, K., 1996, Reverse search for enumeration. *Discrete Appl. Math.*, **6**(1), 21–46.
- [9] Lewis, R. M., Shepherd, A., and Torczon, V., 2005, Implementing generating set search methods for linearly constrained minimization. Technical report WM-CS-2005-01. College of William and Mary, Department of Computer Science (Williamsburg, VA).
- [10] Price, C. J., and Coope, I. D., 2003, Frames and grids in unconstrained and linearly constrained optimization: A nonsmooth approach. *SIAM J. Optim.*, **14**(2), 415–438.
- [11] Caron, R. J., McDonald, J. F., and Ponic, C. M., 1989, A degenerate extreme point strategy for the classification of linear constraints as redundant or necessary. *J. Optim. Theory Appl.*, **62**(2), 225–237.
- [12] Karwan, M. H., Lotfi, V., Telgen, J., and Zionts, S., 1983. *Redundancy in Mathematical Programming* (Berlin: Springer-Verlag).
- [13] Berbee, H. C. P., Boender, C. G. E., Kan, A. H. G. R., Scheffer, C. L., Smith, R. L., and Telgen, J., 1987, Hit-and-run algorithms for the identification of nonredundant linear inequalities. *Math. Program.*, **37**(2), 184–207.
- [14] Boneh, A., Boneh, S., and Caron, R. J., 1993, Constraint classification in mathematical programming. *Math. Program.*, **61**(1), 61–73.
- [15] Nemhauser, G. L., and Wolsey, L. A., 1988. *Integer and Combinatorial Optimization* (New York: John Wiley & Sons).
- [16] Wolsey, L. A., 1998. *Integer Programming* (New York: John Wiley & Sons).
- [17] Torczon, V., 1997, On the convergence of pattern search algorithms. *SIAM J. Optim.*, **7**(1), 1–25.

- [18] Bertsekas, D., 1999. *Nonlinear Programming*, 2nd ed. (Athena Scientific).
- [19] Van Tiel, J., 1984. *Convex Analysis* (New York: John Wiley & Sons).
- [20] Fukuda, K., and Prodon, A., 1997, *Double description method revisited. Lecture Notes in Computer Science* (Vol. 1120) (Springer-Verlag), pp. 91-111.
- [21] Motzkin, T. S., Raiffa, H., Thompson, G., and Thrall, R. M., 1953, The double description method. In: *Contributions to the Theory of Games*, (Vol. 2) (Princeton University Press).
- [22] Abramson, M. A., 2006, NOMADm optimization software. Available online at: www.afit.edu/en/-ENC/-Faculty/-MAbramson/-NOMADm.html (accessed 22 June 2006).
- [23] Oberlin, C., and Wright, S. J., 2005, Active constraint identification in nonlinear programming. Technical Report. Computer Sciences Department, University of Wisconsin-Madison.
- [24] Wright, S. J., 2003, Constraint identification and algorithm stabilization for degenerate nonlinear programs. *Math. Program., Ser. B*, **95**(1), 137–160.
- [25] Coope, I. D., Dennis, Jr., J. E., and Price, C. J., 2004, Direct search methods for nonlinearly constrained optimization using filters and frames. *Optim. Engng.*, **5**(2), 123–144.
- [26] Lucidi, S., Sciandrone, M., and Tseng, P., 2002, Objective-derivative-free methods for constrained optimization. *Math. Program.*, **92**(1), 37–59.