ACTIVE SET IDENTIFICATION FOR LINEARLY CONSTRAINED MINIMIZATION WITHOUT EXPLICIT DERIVATIVES

ROBERT MICHAEL LEWIS* AND VIRGINIA TORCZON†

Abstract. We consider active set identification for linearly constrained optimization problems in the absence of explicit information about the derivative of the objective function. We begin by presenting some general results on active set identification that are not tied to any particular algorithm. These general results are sufficiently strong that, given a sequence of iterates converging to a Karush–Kuhn–Tucker point, it is possible to identify binding constraints for which there are nonzero multipliers. We then focus on generating set search methods, a class of derivative-free direct search methods. We discuss why these general results, which are posed in terms of the direction of steepest descent, apply to generating set search, even though these methods do not have explicit recourse to derivatives. Nevertheless, there is a clearly identifiable subsequence of iterations at which we can reliably estimate the set of constraints that are binding at a solution. We discuss how active set estimation can be used to accelerate generating set search methods and illustrate the appreciable improvement that can result using several examples from the CUTEr test suite. We also introduce two algorithmic refinements for generating set search methods. The first expands the subsequence of iterations at which we can make inferences about stationarity. The second is a more flexible step acceptance criterion.

Key words. active sets, constrained optimization, linear constraints, direct search, generating set search, generalized pattern search, derivative-free methods.

AMS subject classifications. 90C56, 90C30, 65K05

1. Introduction.

We consider active constraint identification for generating set search (GSS) algorithms for linearly constrained minimization. For our discussion the optimization problem of interest will be posed as

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax \leq b, \end{array} \tag{1.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $A \in \mathbb{R}^{m \times n}$.

GSS algorithms are a class of direct search methods and as such do not make explicit use of derivatives of the objective f [13]. Results on active constraint identification for smooth nonlinear programming, on the other hand, do rely explicitly on the gradient $\nabla f(x)$ of f (e.g., [5, 3, 2, 4, 7, 22]). Nevertheless, GSS algorithms compute information in the course of their operation that can be used to identify the set of constraints active at a Karush–Kuhn–Tucker (KKT) point of (1.1). Thus, while we assume for our theoretical results that f is continuously differentiable, for the application to GSS methods we do *not* assume that $\nabla f(x)$ (or an approximation thereto) is computationally available.

We first prove some general active set identification results. While these results reveal the active set identification properties of GSS methods, they are general and not tied to any particular class of algorithms. The results in Theorem 3.4 establish the active set identification properties of a sequence of points in terms of the projection of the direction of steepest descent onto a cone associated with constraints near the points in the sequence. Theorem 3.4 allows the sequence to be entirely interior to the

*Department of Mathematics, College of William & Mary, P.O. Box 8795, Williamsburg, Virginia, 23187-8795; buckaroo@math.wm.edu. This work was supported by the National Science Foundation under Grant No. DMS-0713812.

†Department of Computer Science, College of William & Mary, P.O. Box 8795, Williamsburg, Virginia, 23187-8795; va@cs.wm.edu.

feasible region in (1.1), and this limits slightly the active set identification properties of the sequence. However, there is an easily computed auxiliary sequence of points (that can include points on the boundary) that possesses even stronger active set identification properties. In Theorem 3.7 we show that for this auxiliary sequence the projection of the direction of steepest descent onto the tangent cone tends to zero. It follows then from results of Burke and Moré [3, 4] that this auxiliary sequence allows us to identify the active constraints for which there are nonzero Lagrange multipliers (unique or not) without any assumption of linear independence of the active constraints. Moreover, under mild assumptions this auxiliary sequence will find the face determined by the active constraints in a finite number of iterations.

Any algorithm for which these general results hold will have active set identification properties at least as strong as the gradient projection method [5, 3, 2, 4]. It is therefore perhaps surprising that these results apply to GSS methods, which do not have direct access to the gradient. However, even though GSS methods do not have recourse to $\nabla f(x)$, there does exist an implicit relationship between an explicitly known algorithmic parameter used to control step lengths and an appropriately chosen measure of stationarity for (1.1) [15, Section 1.2]. As the step-length control parameter tends to zero, so does this measure of stationarity. This correlation means that GSS methods generate at least one subsequence of iterates that satisfies the hypotheses of our general active set identification results, and so GSS methods are able to identify active sets of constraints, even without direct access to the derivative of the objective.

We illustrate the practical benefits of incorporating active set estimation into GSS algorithms. We propose several ways in which one can use estimates of the active set to accelerate the search for a solution of (1.1) and present numerical tests that show these strategies can appreciably accelerate progress towards a solution. These tests show that generating set search can be competitive with a derivative-based quasi-Newton algorithm. As part of these algorithmic developments we also introduce two refinements for GSS methods. The first expands the subsequence of iterations at which we can make inferences about stationarity by introducing the notion of *tangentially unsuccessful* iterations, as discussed further in Section 4.4. The second is a more flexible step acceptance criterion that scales automatically with the magnitude of the objective, as discussed further in Section 4.3.

The general results on active set identification are given in Section 3. The relevant features of GSS methods are reviewed and expanded in Section 4. Section 5 discusses the active set identification properties of GSS methods and proposes acceleration schemes based on them. Section 6 contains some numerical tests of these ideas.

2. Notation. Let Ω denote the feasible region: $\Omega = \{\, x \mid Ax \le b \,\}$. Let a_i^T be the ith row of A. The set of constraints active at a point $x \in \Omega$ is defined to be

$$\mathcal{A}(x) = \left\{\, i \mid a_i^T x = b_i \,\right\}.$$

Let $\mathcal{C}_i = \left\{\, y \mid a_i^T y = b_i \,\right\}$, the affine subspace associated with the ith inequality constraint. Given $x \in \Omega$ and $\varepsilon \ge 0$, define

$$I(x,\varepsilon) = \left\{\, i \mid \mathrm{dist}(x, \mathcal{C}_i) = \left| \, b_i - a_i^T x \, \right| / \, \| \, a_i \, \| \le \varepsilon \,\right\}, \tag{2.1}$$

where $\| \cdot \|$ is the Euclidean norm. The vectors a_i for $i \in I(x,\varepsilon)$ are outward-pointing normals to the constraints defining the boundary of Ω within distance ε of x. We call the set of such vectors the *working set* of constraints associated with x and ε. The

ε-*normal* cone $N(x,\varepsilon)$ and its polar the ε-*tangent* cone $T(x,\varepsilon)$ are defined to be

$$N(x,\varepsilon) = \left\{ v \mid v = \sum_{i \in I(x,\varepsilon)} c_i a_i,\ c_i \geq 0 \right\} \cup \{\mathbf{0}\} \quad \text{and} \quad T(x,\varepsilon) = N^\circ(x,\varepsilon).$$

If $\varepsilon = 0$ these cones are the usual normal cone $N(x)$ and tangent cone $T(x)$ of Ω at x. If $K \subseteq \mathbb{R}^n$ is a convex cone and $v \in \mathbb{R}^n$, then we denote the Euclidean norm projection of v onto K by $[v]_K$.

Let $\mathbb{R}_+$ be the set of nonnegative real numbers. If $\{a_k\}$ is a sequence, and $\mathcal{K} \subset \mathbb{N}$ are the indices of an infinite subsequence, then we define $\{a_k\}_\mathcal{K} = \{\, a_k \mid k \in \mathcal{K} \,\}$, and use the notation $a_k \xrightarrow{\mathcal{K}} a_*$ to mean that $\{a_k\}_\mathcal{K}$ converges to a_* as $k \to \infty$.

3. Results concerning active set identification. Suppose $x_\star$ is a KKT point of (1.1), and consider an algorithm that generates a sequence of feasible points $\{x_k\}_\mathcal{K} \subset \Omega$ such that $x_k \xrightarrow{\mathcal{K}} x_\star$. At $x_\star$ there exist Lagrange multipliers λ_i, $i \in \mathcal{A}(x_\star)$, possibly nonunique, such that

$$-\nabla f(x_\star) = \sum_{i \in \mathcal{A}(x_\star)} \lambda_i a_i, \quad \lambda_i \geq 0 \text{ for all } i \in \mathcal{A}(x_\star). \tag{3.1}$$

We wish to construct a sequence $\{\hat{x}_k\}_\mathcal{K}$ such that:

1. For all k sufficiently large, we either identify the active set at $x_\star$ or at least those constraints in $\mathcal{A}(x_\star)$ with nonzero multipliers.
2. If $x_\star$ lies in a face of Ω, then the iterates land on the face that contains $x_\star$ in a finite number of iterations. That is, $\mathcal{A}(\hat{x}_k) = \mathcal{A}(x_\star)$ for all k sufficiently large. In particular, if $x_\star$ is at a vertex of Ω, then the iterates land on $x_\star$ in a finite number of iterations: $\hat{x}_k = x_\star$ for all k sufficiently large.

Moreover, we want these results to hold under assumptions on $\nabla f(x_\star)$ and $\mathcal{A}(x_\star)$ that are as mild as possible. Because the gradient projection algorithm possesses these properties [5, 3, 2, 4], we view these requirements as reasonable expectations for any combination of optimization algorithm and active set identification technique applied to (1.1) when f is differentiable.

The following serves as our set of assumptions throughout this section.

Assumption 3.1. *Suppose f is continuously differentiable on Ω, and suppose $x_\star \in \Omega$, $\mathcal{K} \subset \mathbb{N}$, $\{x_k\}_\mathcal{K} \subset \Omega$, and $\{\varepsilon_k\}_\mathcal{K}$, $\{\eta_k\}_\mathcal{K} \subset \mathbb{R}_+$ are such that $x_k \xrightarrow{\mathcal{K}} x_\star$, $\varepsilon_k, \eta_k \xrightarrow{\mathcal{K}} 0$, and $\| \, [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \, \| \leq \eta_k$ for all $k \in \mathcal{K}$.*

The use of a standard set of assumptions simplifies the exposition, though some of the results (e.g., Proposition 3.2) do not require the full strength of Assumption 3.1.

We begin by recalling the following result [19, Proposition 1].

Proposition 3.2. *Let $x_\star$ and the sequences $\mathcal{K}$, $\{x_k\}_\mathcal{K}$, $\{\varepsilon_k\}_\mathcal{K}$, and $\{\eta_k\}_\mathcal{K}$ be as in Assumption 3.1. Then $I(x_k,\varepsilon_k) \subseteq \mathcal{A}(x_\star)$ for all $k \in \mathcal{K}$ sufficiently large.*

We use this result to show that under Assumption 3.1, $x_\star$ is a KKT point.

Theorem 3.3. *Let f, $x_\star$ and the sequences $\mathcal{K}$, $\{x_k\}_\mathcal{K}$, $\{\varepsilon_k\}_\mathcal{K}$, and $\{\eta_k\}_\mathcal{K}$ be as in Assumption 3.1. Then $x_\star$ is a KKT point of (1.1).*

Proof. From Proposition 3.2 we know that $I(x_k,\varepsilon_k) \subseteq \mathcal{A}(x_\star)$ for all $k \in \mathcal{K}$ sufficiently large. For such k it follows that $N(x_k,\varepsilon_k) \subseteq N(x_\star)$. Therefore $T(x_\star) \subseteq T(x_k,\varepsilon_k)$ and so

$$\| \, [-\nabla f(x_k)]_{T(x_\star)} \, \| \leq \| \, [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \, \| \leq \eta_k.$$

Taking the limit as $k \to \infty$ yields $[-\nabla f(x_\star)]_{T(x_\star)} = 0$, and the result follows. ∎

The next result, Theorem 3.4, summarizes what we can infer about the active set $\mathcal{A}(x_\star)$ from the sets $I(x_k, \varepsilon_k)$. If $\mathcal{I} \subset \mathcal{A}(x_\star)$ is a subset of constraints active at $x_\star$, let $N(\mathcal{A}(x_\star) \setminus \mathcal{I})$ be the cone generated by the set $\{ a_j \mid j \in \mathcal{A}(x_\star) \setminus \mathcal{I} \}$, and let $T(\mathcal{A}(x_\star) \setminus \mathcal{I})$ be its polar. If $i \in \mathcal{A}(x_\star)$, let Π_i denote the projection onto the orthogonal complement of the linear span of the other active constraints; i.e., Π_i is the projection onto $(\operatorname{span}\{ a_j \mid j \in \mathcal{A}(x_\star) \setminus \{i\} \})^\perp$.

Also, problem (1.1) is posed in terms of inequality constraints, so any equality constraint is ostensibly expressed as a pair of opposing inequalities: $a_i^T x \le b_i$ and $-a_i^T x \le -b_i$. Define $\mathcal{E} = \left\{ i \mid a_i^T x = b_i \text{ for all } x \in \Omega \right\}$ to be the set of equality constraints; then the set of "true" inequalities active at $x_\star$ is $\mathcal{A}(x_\star) \setminus \mathcal{E}$.

THEOREM 3.4. *Let f, $x_\star$ and the sequences $\mathcal{K}$, $\{x_k\}_\mathcal{K}$, $\{\varepsilon_k\}_\mathcal{K}$, and $\{\eta_k\}_\mathcal{K}$ be as in Assumption 3.1, and λ_i, $i \in \mathcal{A}(x_\star)$, be as in (3.1).*

Then for all $k \in \mathcal{K}$ sufficiently large we have

1. *$-\nabla f(x_\star) \in N(x_k, \varepsilon_k)$;*
2. *if $\mathcal{I} \subset \mathcal{A}(x_\star)$ but $\mathcal{I} \cap I(x_k, \varepsilon_k) = \emptyset$, then $[-\nabla f(x_\star)]_{T(\mathcal{A}(x_\star)\setminus\mathcal{I})} = 0$;*
3. *if $i \in \mathcal{A}(x_\star)$ but $i \notin I(x_k, \varepsilon_k)$, then $\lambda_i \Pi_i(a_i) = 0$.*

In addition, suppose that for all $i \in \mathcal{A}(x_\star) \setminus \mathcal{E}$ strict complementarity holds at $x_\star$ and $\Pi_i(a_i) \neq 0$. Then $I(x_k, \varepsilon_k) = \mathcal{A}(x_\star)$ for all $k \in \mathcal{K}$ sufficiently large.

Proof. We first prove Part 1. Since there is only a finite number of distinct working sets $I(x_k, \varepsilon_k)$, we know that there exists $\bar{k} \in \mathcal{K}$ such that if $k \ge \bar{k}$ then the associated working set $I(x_k, \varepsilon_k)$ appears infinitely often among the working sets $\left\{ I(x_\ell, \varepsilon_\ell) \mid \ell \ge \bar{k} \right\}$. Suppose $k \in \mathcal{K}$ and $k \ge \bar{k}$. Then

$$\lim_{\substack{\ell \to \infty \\ I(x_\ell, \varepsilon_\ell) = I(x_k, \varepsilon_k)}} \left\| \, [-\nabla f(x_\ell)]_{T(x_k, \varepsilon_k)} \, \right\| \quad \le \quad \lim_{\substack{\ell \to \infty \\ I(x_\ell, \varepsilon_\ell) = I(x_k, \varepsilon_k)}} \eta_\ell,$$

whence $[-\nabla f(x_\star)]_{T(x_k, \varepsilon_k)} = 0$. It follows that $-\nabla f(x_\star) \in N(x_k, \varepsilon_k)$, which is Part 1.

Next, it follows from Proposition 3.2 and Part 1 that there exists $\hat{k} \in \mathcal{K}$ such that $I(x_k, \varepsilon_k) \subseteq \mathcal{A}(x_\star)$ and $-\nabla f(x_\star) \in N(x_k, \varepsilon_k)$ for all $k \in \mathcal{K}$ with $k \ge \hat{k}$. Suppose that for some such k we have $\mathcal{I} \subset \mathcal{A}(x_\star)$ but $\mathcal{I} \cap I(x_k, \varepsilon_k) = \emptyset$. Then $I(x_k, \varepsilon_k) \subseteq \mathcal{A}(x_\star) \setminus \mathcal{I}$, so $N(x_k, \varepsilon_k) \subseteq N(\mathcal{A}(x_\star) \setminus \mathcal{I})$ and $T(\mathcal{A}(x_\star) \setminus \mathcal{I}) \subseteq T(x_k, \varepsilon_k)$. Thus,

$$\left\| \, [-\nabla f(x_\star)]_{T(\mathcal{A}(x_\star)\setminus\mathcal{I})} \, \right\| \le \left\| \, [-\nabla f(x_\star)]_{T(x_k, \varepsilon_k)} \, \right\| = 0,$$

which yields Part 2. In addition, since $-\nabla f(x_\star) \in N(x_k, \varepsilon_k)$ there exist nonnegative scalars $\lambda_{k,j}$, $j \in I(x_k, \varepsilon_k)$, such that

$$-\nabla f(x_\star) = \sum_{j \in I(x_k, \varepsilon_k)} \lambda_{k,j} \, a_j.$$

If we combine this with (3.1) then we see that for any $i \in \mathcal{A}(x_\star)$ we have

$$\lambda_i a_i = \sum_{j \in I(x_k, \varepsilon_k)} \lambda_{k,j} \, a_j - \sum_{j \in \mathcal{A}(x_\star)\setminus\{i\}} \lambda_j \, a_j.$$

Thus, if $i \in \mathcal{A}(x_\star)$ but $i \notin I(x_k, \varepsilon_k)$, then $I(x_k, \varepsilon_k) \subseteq \mathcal{A}(x_\star) \setminus \{i\}$, whence $\lambda_i a_i \in \operatorname{span}\{ a_j \mid j \in \mathcal{A}(x_\star) \setminus \{i\} \}$. Therefore, $0 = \Pi_i(\lambda_i a_i) = \lambda_i \Pi_i(a_i)$, which is Part 3. (We are indebted to Jim Burke for this proof of Part 3.)

Finally, if strict complementarity holds for the constraints in $\mathcal{A}(x_\star) \setminus \mathcal{E}$, then $\mathcal{A}(x_\star) = \mathcal{E} \cup \{ j \in \mathcal{A}(x_\star) \setminus \mathcal{E} \mid \lambda_j > 0 \}$. By assumption $x_k \in \Omega$ for all $k \in \mathcal{K}$, so all the equality constraints are satisfied at every iteration and thus $\mathcal{E} \subseteq I(x_k, \varepsilon_k)$ for all $k \in \mathcal{K}$. If $\Pi_i(a_i) \neq 0$ for all $i \in \mathcal{A}(x_\star) \setminus \mathcal{E}$, then by Part 3 we know that $\mathcal{A}(x_\star) \setminus \mathcal{E} \subseteq I(x_k, \varepsilon_k)$. Thus, $\mathcal{A}(x_\star) \subseteq I(x_k, \varepsilon_k)$. However, for $k \in \mathcal{K}$ sufficiently large we know that $I(x_k, \varepsilon_k) \subseteq \mathcal{A}(x_\star)$, so $I(x_k, \varepsilon_k) = \mathcal{A}(x_\star)$. □

Roughly speaking, if $I(x_k, \varepsilon_k)$ is used as an estimate of $\mathcal{A}(x_\star)$, then the only constraints in $\mathcal{A}(x_\star)$ we might fail to identify are constraints that are not essential to make $x_\star$ a KKT point. For instance, Part 2 says that if a constraint in $\mathcal{A}(x_\star)$ is missing from $I(x_k, \varepsilon_k)$, then $x_\star$ would remain a KKT point if the constraint were deleted (though, of course, the feasible region Ω would probably change). Part 3 says that if a constraint has an associated positive multiplier, then it is missing from $I(x_k, \varepsilon_k)$ only if it is a linear combination of the other active constraints.

The following two examples help to illustrate Theorem 3.4. In the first, $I(x_k, \varepsilon_k)$ fails to identify an active constraint with a unique associated multiplier of 0:

$$\begin{array}{ll} \text{minimize} & f(x_1, x_2) = (x_1 - 1)^2 - x_2 \\ \text{subject to} & x_1, x_2 \leq 1 \end{array} \tag{3.2}$$

The unique KKT point is $x_\star = (1, 1)^T$, and both constraints are active at $x_\star$. Let $a_1 = (1, 0)^T$ and $a_2 = (0, 1)^T$. Choose a sequence $\varepsilon_k \downarrow 0$ and define $x_k = (1 - 2\varepsilon_k, 1)^T$. For any k, $I(x_k, \varepsilon_k)$ contains only the constraint corresponding to a_2. We have $\| \, [-\nabla f(x_k)]_{T(x_k, \varepsilon_k)} \, \| = 2\varepsilon_k \to 0$, but $I(x_k, \varepsilon_k)$ is a proper subset of $\mathcal{A}(x_\star)$ for all k. At $x_\star$ we have $-\nabla f(x_\star) = a_2$ and $-\nabla f(x_\star) \perp a_1$, so there is a zero multiplier associated with a_1; $x_\star$ remains a KKT point if we drop the constraint $x_1 \leq 1$.

In the next example $x_\star$ lies at the apex of a pyramidal feasible region. Let

$$A = \begin{pmatrix} 1 & 1 & 1 \\ -1/2 & 1/2 & 1 \\ -1 & -1 & 1 \\ 1/2 & -1/2 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix},$$

and consider the problem

$$\begin{array}{ll} \text{minimize} & f(x_1, x_2, x_3) = -x_3 \\ \text{subject to} & Ax \leq b. \end{array} \tag{3.3}$$

The unique KKT point is $x_\star = (0, 0, 1)^T$. All four constraints are active at $x_\star$. Choose a sequence $\varepsilon_k \downarrow 0$ and define $x_k = (0, 0, 1 - \sqrt{3}\varepsilon_k)^T$. For any k, $I(x_k, \varepsilon_k)$ contains only the constraints corresponding to a_1 and a_3. Since $-\nabla f(x_k) = (a_1 + a_3)/2$, we have $[-\nabla f(x_k)]_{T(x_k, \varepsilon_k)} = 0$ for all k; however, $I(x_k, \varepsilon_k)$ is a proper subset of $\mathcal{A}(x_\star)$. In this example, nonzero multipliers exist for a_2 and a_4 since $-\nabla f(x_\star) = (a_2 + a_4)/2$ and yet $x_\star$ remains a KKT point if we drop the corresponding constraints.

Theorem 3.4 says that using $I(x_k, \varepsilon_k)$ to estimate $\mathcal{A}(x_\star)$ mostly achieves the first of the goals outlined at the start of this section. To realize our first goal fully, as well as to achieve our second goal—that $\mathcal{A}(x_k) = \mathcal{A}(x_\star)$ after a finite number of iterations—in general we will need iterates that lie on the boundary of Ω (in the example (3.3), for instance, $\{x_k\}$ remains strictly interior to Ω). To address this issue we define an auxiliary sequence $\{\hat{x}_k\}$ of points that can lie on the boundary of Ω.

To motivate our particular choice of $\hat{x}_k$, we recall some results of Burke and Moré [3, 4], stated here in terms of the linearly constrained problem (1.1). Theorem 3.5 gives a necessary and sufficient condition for identifying the active set $\mathcal{A}(x_\star)$ from $\mathcal{A}(\hat{x}_k)$ in a finite number of iterations, and a condition under which $x_\star$ will be attained in a finite number of iterations. Theorem 3.6 gives a necessary and sufficient condition for the identification of all constraints in $\mathcal{A}(x_\star)$ for which there is a strictly positive Lagrange multiplier.

THEOREM 3.5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on Ω, and assume that $\{\hat{x}_k\} \subset \Omega$ converges to a KKT point $x_\star$ for which $-\nabla f(x_\star)$ is in the relative interior of $N(x_\star)$. Then*

1. *[3, Corollary 3.6] $\mathcal{A}(\hat{x}_k) = \mathcal{A}(x_\star)$ for all k sufficiently large if and only if $\{[-\nabla f(\hat{x}_k)]_{T(\hat{x}_k)}\}$ converges to zero; and*
2. *[3, Corollary 3.5] if $N(x_\star)$ has a nonempty interior and $\{[-\nabla f(\hat{x}_k)]_{T(\hat{x}_k)}\}$ converges to zero, then $\hat{x}_k = x_\star$ for all k sufficiently large.*

The example in (3.2) illustrates the need for the assumption that $-\nabla f(x_\star)$ lies in the relative interior of $N(x_\star)$. This assumption is equivalent to

$$-\nabla f(x_\star) = \sum_{j \in \mathcal{A}(x_\star)} \lambda_j a_j, \quad \lambda_j > 0 \text{ for all } j \in \mathcal{A}(x_\star)$$

(see [3, Lemma 3.2]). The example in (3.3) illustrates the need for $\{[-\nabla f(\hat{x}_k)]_{T(\hat{x}_k)}\}$ to converge to zero; since $\{x_k\}$ remains strictly interior to Ω, $T(x_k) = \mathbb{R}^n$ for all k.

THEOREM 3.6. *[4, Theorem 4.5] Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuously differentiable on Ω, and assume that $\{\hat{x}_k\} \subset \Omega$ converges to a KKT point $x_\star$ at which we have*

$$-\nabla f(x_\star) = \sum_{i \in \mathcal{A}(x_\star)} \lambda_i a_i, \quad \lambda_i \geq 0.$$

Then $\lim_{k\to\infty}[-\nabla f(\hat{x}_k)]_{T(\hat{x}_k)} = 0$ if and only if $\{\, i \in \mathcal{A}(x_\star) \mid \lambda_i > 0 \,\} \subseteq \mathcal{A}(\hat{x}_k)$ for all k sufficiently large.

Under Assumption 3.1 we have $[-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \xrightarrow{\mathcal{K}} 0$. Theorems 3.5 and 3.6 say that we would obtain desirable active set identification properties if we were to find a sequence $\hat{x}_k \xrightarrow{\mathcal{K}} x_\star$ for which $[-\nabla f(\hat{x}_k)]_{T(\hat{x}_k)} \xrightarrow{\mathcal{K}} 0$. This suggests choosing $\hat{x}_k$ so that $T(\hat{x}_k) \subseteq T(x_k, \varepsilon_k)$, or, equivalently, that $I(x_k, \varepsilon_k) \subseteq \mathcal{A}(\hat{x}_k)$. To this end, given $k \in \mathcal{K}$, if the set $\left\{\, x \mid x \in \Omega,\ a_i^T x = b_i \text{ for } i \in I(x_k, \varepsilon_k) \,\right\}$ is nonempty, then define

$$\hat{x}_k \in \underset{x}{\operatorname{argmin}} \left\{\, \|\, x - x_k \,\| \mid x \in \Omega,\ a_i^T x = b_i \text{ for } i \in I(x_k, \varepsilon_k) \,\right\}; \tag{3.4}$$

otherwise, $\hat{x}_k = x_k$. In (3.4), $\hat{x}_k$ is the projection of x_k onto a convex set (provided the set is nonempty), so the argmin is a single point. If $\hat{x}_k$ is defined by (3.4) then $I(x_k, \varepsilon_k) \subseteq \mathcal{A}(\hat{x}_k)$. In the example (3.2), $\hat{x}_k = x_k$ for all k, while in the example (3.3), $\hat{x}_k = x_\star$ for all k.

THEOREM 3.7. *Let f, $x_\star$ and the sequences $\mathcal{K}$, $\{x_k\}_{\mathcal{K}}$, $\{\varepsilon_k\}_{\mathcal{K}}$, and $\{\eta_k\}_{\mathcal{K}}$ be as in Assumption 3.1. Then*

1. *$\|\, \hat{x}_k - x_k \,\| \xrightarrow{\mathcal{K}} 0$;*
2. *$[-\nabla f(\hat{x}_k)]_{T(\hat{x}_k)} \xrightarrow{\mathcal{K}} 0$.*

Proof. Suppose $k \in \mathcal{K}$ is sufficiently large that $I(x_k, \varepsilon_k) \subseteq \mathcal{A}(x_\star)$. Then

$$x_\star \in \left\{\, x \mid Ax \leq b \text{ and } a_i^T x = b_i \text{ for all } i \in I(x_k, \varepsilon_k) \,\right\}.$$

Since $x_\star$ is feasible for the optimization problem in (3.4), it follows that $\hat{x}_k$ is defined via (3.4) and consequently $\|\, \hat{x}_k - x_k \,\| \leq \|\, x_\star - x_k \,\|$. Since $x_k \xrightarrow{\mathcal{K}} x_\star$ we conclude that $\|\, \hat{x}_k - x_k \,\| \xrightarrow{\mathcal{K}} 0$, which is Part 1.

Moreover, because $\hat{x}_k$ is defined via (3.4) we know that $I(x_k, \varepsilon_k) \subseteq \mathcal{A}(\hat{x}_k)$. Thus $N(x_k, \varepsilon_k) \subseteq N(\hat{x}_k)$ and $T(\hat{x}_k) \subseteq T(x_k, \varepsilon_k)$, whence

$$\begin{aligned}
&\| \, [-\nabla f(\hat{x}_k)]_{T(\hat{x}_k)} \, \| \le \| \, [-\nabla f(\hat{x}_k)]_{T(x_k,\varepsilon_k)} \, \| \\
&\quad \le \| \, [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \, \| + \| \, [-\nabla f(\hat{x}_k)]_{T(x_k,\varepsilon_k)} - [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \, \| \\
&\quad \le \eta_k + \| \, \nabla f(\hat{x}_k) - \nabla f(x_k) \, \| .
\end{aligned}$$

Part 2 then follows, since from Part 1 we know that $\| \, \hat{x}_k - x_k \, \| \xrightarrow{\mathcal{K}} 0$. ∎

We now can conclude that $\{\hat{x}_k\}$ has even stronger active set identification properties than $\{x_k\}$. From Theorems 3.5 and 3.6 and Part 2 of Theorem 3.7 we obtain Theorem 3.8, which achieves the two goals concerning active set identification laid out at the beginning of this section. Theorem 3.4 says that $I(x_k, \varepsilon_k)$ serves as a useful estimate of $\mathcal{A}(x_\star)$; Theorem 3.8 says that $\mathcal{A}(\hat{x}_k)$ is a sharper estimate.

THEOREM 3.8. *Let f, $x_\star$ and the sequences $\mathcal{K}$, $\{x_k\}_\mathcal{K}$, $\{\varepsilon_k\}_\mathcal{K}$, and $\{\eta_k\}_\mathcal{K}$ be as in Assumption 3.1. Then we have the following:*

1. *If $-\nabla f(x_\star)$ lies in the relative interior of $N(x_\star)$, then $\mathcal{A}(\hat{x}_k) = \mathcal{A}(x_\star)$ for all k sufficiently large. If, in addition, $N(x_\star)$ has a nonempty interior, then $\hat{x}_k = x_\star$ for all k sufficiently large.*
2. *Suppose*

$$-\nabla f(x_\star) = \sum_{i \in \mathcal{A}(x_\star)} \lambda_i a_i, \quad \lambda_i \ge 0.$$

Then $\{ \, i \in \mathcal{A}(x_\star) \mid \lambda_i > 0 \, \} \subseteq \mathcal{A}(\hat{x}_k)$ for all k sufficiently large.

4. Generating set search for linearly constrained minimization. We next review the features of GSS that are relevant to active set identification for (1.1) (see [15, 16] for further discussion). GSS algorithms for linearly constrained problems are feasible iterate methods. Associated with each x_k is a value $\varepsilon_k \to 0$, defined shortly, which is used to compute a working set of nearby constraints $I(x_k, \varepsilon_k)$ as defined by (2.1). The connection with the active set identification results of the previous section comes through the stationarity properties of GSS discussed in Section 4.7.

4.1. Partitioning the search directions. At the center of the convergence properties of GSS methods is the notion of a *core set* of search directions, denoted $\mathcal{G}_k$, which is constructed at each iteration so as to ensure the inclusion of at least one direction of descent along which a feasible step of sufficient length can be taken.

In the linearly constrained case, $\mathcal{G}_k$ must comprise a set of generators for the ε-tangent cone $T(x_k, \varepsilon_k)$. A set of vectors $\mathcal{G}$ *generates* a cone K if K is the set of all nonnegative linear combinations of elements of $\mathcal{G}$. The cone K is *finitely generated* if it can be generated by a finite set of vectors. If the finite set $\mathcal{G}_k$ is chosen so as to ensure that it generates the cone $T(x_k, \varepsilon_k)$, then convergence results for GSS methods for linearly constrained problems can be obtained under straightforward conditions detailed in [15, Section 6]. To simplify the discussion, we assume that all of the directions in $\mathcal{G}_k$ are normalized. For details on other options, see [15, section 2.3]. As a set of generators for $T(x_k, \varepsilon_k)$, the *core search directions* (the elements of $\mathcal{G}_k$) conform to the boundary of Ω near x_k, where "near" is determined by the value of ε_k. It is this property of $\mathcal{G}_k$ that ensures a direction of descent along which a feasible step of length at least ε_k can be taken (if x_k is not a KKT point).

If the working set of constraints is linearly independent, then it is straightforward to calculate a set of generators for $T(x_k, \varepsilon_k)$ as described in [20, 17, 16], though some care is needed to handle the interactions between equality and inequality constraints correctly, as detailed in [16, Section 4]. If the working set is linearly dependent, then it is possible to calculate a set of generators as described in [16, Section 5.4.2] using a state-of-the-art implementation of the double description algorithm [8, 9].

As a practical matter it is useful to allow additional search directions. For instance, in Section 5 we introduce additional search directions to effect the acceleration strategies based on active set estimation. The set of additional search directions is denoted by $\mathcal{H}_k$. The total set of search directions $\mathcal{D}_k$ is thus $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$.

In the event that $T(x_k, \varepsilon_k)$ contains a nontrivial lineality space consisting of the set $T(x_k, \varepsilon_k) \cap (-T(x_k, \varepsilon_k))$, as in Figure 4.1, we have some freedom in our choice of generators. In order to preserve the convergence properties of the algorithm, we place Condition 4.1 on the choice of $\mathcal{G}_k$. Condition 4.1 makes use of the quantity $\kappa(\mathcal{G})$, which appears in [15, (2.1)] and is a generalization of that given in [13, (3.10)]. For any finite set of vectors $\mathcal{G}$ define

$$\kappa(\mathcal{G}) = \inf_{\substack{v \in \mathbb{R}^n \\ [v]_K \neq 0}} \max_{d \in \mathcal{G}} \frac{v^T d}{\| \, [v]_K \, \| \; \| \, d \, \|}, \quad \text{where } K \text{ is the cone generated by } \mathcal{G}. \tag{4.1}$$

We place the following condition on the set of search directions $\mathcal{G}_k$.

CONDITION 4.1. *There exists a constant $\kappa_{\min} > 0$, independent of k, such that the following holds. For every k, $\mathcal{G}_k$ generates $T(x_k, \varepsilon_k)$ and if $\mathcal{G}_k \neq \{\mathbf{0}\}$, then $\kappa(\mathcal{G}_k) \geq \kappa_{\min}$.*

The lower bound $\kappa_{\min}$ from Condition 4.1 precludes a sequence of $\mathcal{G}_k$'s for which $\kappa(\mathcal{G}_k) \to 0$. Consider such a situation, borrowed from [15, Section 2.3]. Let

$$\mathcal{G} = \left\{ \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ -\nu \end{pmatrix} \right\}.$$

In Figure 4.1 we illustrate $\mathcal{G}$ with three choices of $\nu > 0$. Given $-\nabla f(x) = (0, -1)^T$, neither of the first two elements of $\mathcal{G}$ are descent directions while the remaining element in $\mathcal{G}$ will be an increasingly poor descent direction as $\nu \to 0$.

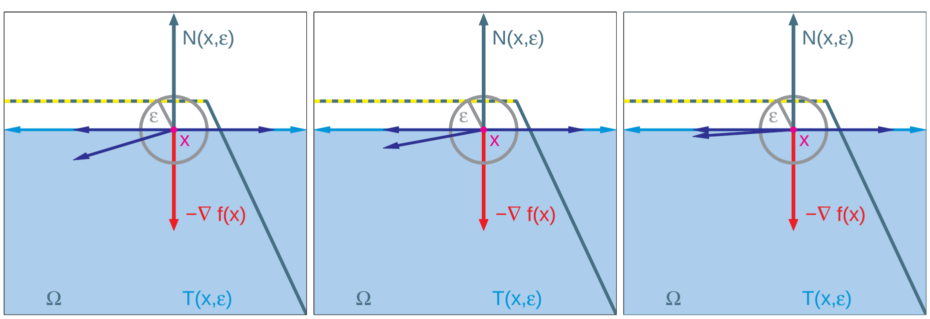


FIG. 4.1. *Condition 4.1 is needed to avoid a sequence of $\mathcal{G}_k$'s for which $\kappa(\mathcal{G}_k) \to 0$.*

As first noted in [15, Section 2.3], there is a simple technique to ensure that Condition 4.1 is satisfied. Start with Proposition 2.3 in [15], which states that for all $x \in \Omega$ and $\varepsilon > 0$, there are at most 2^m distinct working sets defined by $I(x, \varepsilon)$. Thus, there are at most 2^m distinct ε-normal cones and at most 2^m distinct ε-tangent cones. Let $k_2 > k_1$. If $I(x_{k_2}, \varepsilon_{k_2}) = I(x_{k_1}, \varepsilon_{k_1})$, then use the same set of generators for $T(x_{k_2}, \varepsilon_{k_2})$ as were used for $T(x_{k_1}, \varepsilon_{k_1})$. It then follows that there are at most 2^m distinct sets $\mathcal{G}$. Set $\kappa_{\min} = \min \{ \, \kappa(\mathcal{G}_k) \mid \mathcal{G}_k \neq \{\mathbf{0}\} \, \}$ and Condition 4.1 is satisfied.

4.2. Determining the lengths of steps. The *step-length control parameter* Δ_k regulates the steps tried along the search directions in $\mathcal{D}_k$ and prevents them from becoming either too long or too short. The trial point associated with any $d \in \mathcal{D}_k$ is

$$x_k + \tilde{\Delta}(d)d, \quad \text{where} \quad \tilde{\Delta}(d) = \max\left\{\, \Delta \in [0, \Delta_k] \mid x_k + \Delta d \in \Omega \,\right\}. \tag{4.2}$$

This construction ensures that the full step (with respect to Δ_k) is taken if the resulting trial point is feasible. Otherwise, the trial point is found by taking the longest feasible step possible from x_k along d.

With the directions in $\mathcal{G}_k$ normalized, the value of ε_k used to determine the current working set is derived from the current value of Δ_k according to $\varepsilon_k = \min\{\varepsilon_{\max}, \Delta_k\}$ for some $\varepsilon_{\max} > 0$. Prudent choices of $\varepsilon_{\max}$ and Δ_0 mean that in practice it is typically the case that $\varepsilon_k = \Delta_k$.

4.3. Ascertaining success. A trial point is considered acceptable only if it satisfies the sufficient decrease condition

$$f(x_k + \tilde{\Delta}(d)d) < f(x_k) - \rho_k(\Delta_k). \tag{4.3}$$

An appropriate sufficient decrease condition allows the possibility of taking exact steps to the boundary of the feasible region [19, 15, 16]. Here we use

$$\rho_k(\Delta) = \alpha_k \Delta^2. \tag{4.4}$$

We require the sequence $\{\alpha_k\}$ be bounded away from zero; i.e., there exists $\alpha_{\min} > 0$ such that $\alpha_{\min} \le \alpha_k$ for all k. The choice of α_k in the work reported here is

$$\alpha_k = \alpha \max\left\{ |f_{\text{typ}}|,\ |f(x_k)| \right\}, \tag{4.5}$$

where $\alpha > 0$ is fixed and $f_{\text{typ}} \neq 0$ is some fixed value that reflects the typical magnitude of the objective, given feasible inputs.

In earlier work [19, 13, 15, 16], the decrease condition was independent of k; here it is not. We introduce $\rho_k(\Delta)$, as in (4.4), so the step acceptance rule (4.3) scales automatically with the magnitude of the objective. In Section 4.7 we prove that this change does not vitiate the key stationarity results from [15, Section 6].

To ensure asymptotic success, if there is no direction $d \in \mathcal{D}_k$ for which (4.3) is satisfied, then $\Delta_{k+1} \leftarrow \theta_k \Delta_k$ for some choice of θ_k satisfying $0 < \theta_k \le \theta_{\max} < 1$. Otherwise, $\Delta_{k+1} \leftarrow \phi_k \Delta_k$ for some choice of ϕ_k satisfying $1 \le \phi_k \le \phi_{\max} < \infty$. To keep things simple, in both the examples shown here and in our implementation, we use $\theta_k = \frac{1}{2}$ and $\phi_k = 1$ for all k.

4.4. Tangentially unsuccessful iterations. We distinguish three types of iterations. This is a departure from previous work [17, 15, 16] in which the iterations are designated as either *successful* when a trial point that yields acceptable decrease on the value of the objective is found, or *unsuccessful*, when no acceptable decrease is found. Let $\mathcal{S}$ and $\mathcal{U}$ denote the sets of successful and unsuccessful iterations, respectively; then the set of all iterations is $\mathcal{S} \cup \mathcal{U}$ with $\mathcal{S} \cap \mathcal{U} = \emptyset$.

We want to track more closely a lack of decrease along the core search directions contained in the set $\mathcal{G}_k$, so we introduce a new set $\mathcal{U}_T$, for *tangentially unsuccessful* iterations. A tangentially unsuccessful iteration is one at which we know that (4.3) is not satisfied for any $d \in \mathcal{G}_k$. We call the iteration tangentially unsuccessful since no decrease is realized by any of the trial steps defined by the generators of the ε-tangent cone. Thus, any unsuccessful step is also tangentially unsuccessful, since at an unsuccessful step no direction in $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$ is found to satisfy (4.3). However, a tangentially unsuccessful step is not necessarily unsuccessful, since steps along the directions in $\mathcal{G}_k$ may be unsuccessful but a step along one of the extra search directions in $\mathcal{H}_k$ is successful. To summarize, we have $\mathcal{U} \subseteq \mathcal{U}_T$ but, possibly, $\mathcal{S} \cap \mathcal{U}_T \neq \emptyset$.

4.5. Illustration. In Figure 4.3 we illustrate several iterations of one variant of linearly constrained GSS applied to a simple example. This example will be used later to give some insight into how GSS can be accelerated by active set identification. The objective in Figure 4.3 is a modification of the two-dimensional Broyden tridiagonal function [1, 21]. Its level sets are indicated in the background in Figure 4.3. We have introduced two linear inequality constraints, both of which are active at the KKT point, which is indicated by the star in Figure 4.3(a). Figure 4.2 provides a legend.

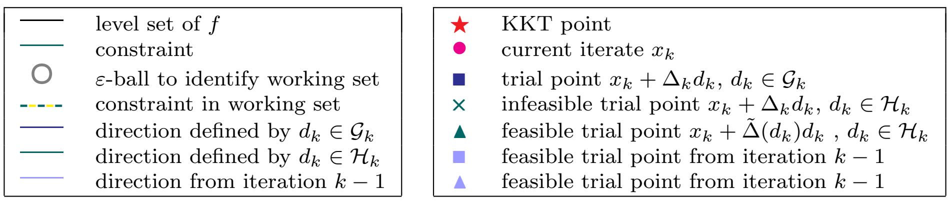
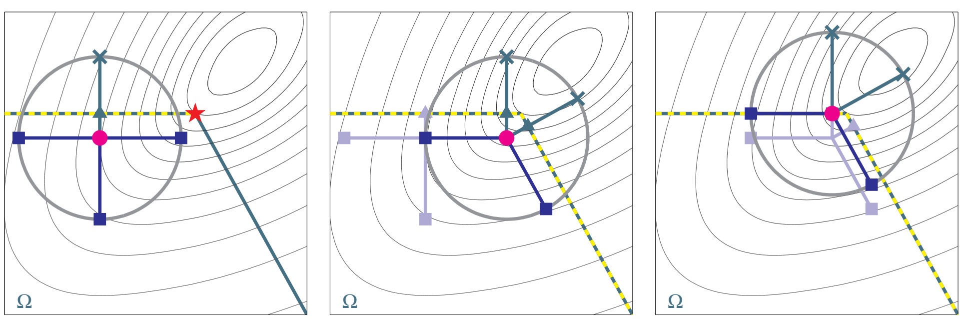


FIG. 4.2. *Legend for Figures 4.3–5.4.*



(a) $k = 0$. The working set contains one constraint. Move East; $k \in \mathcal{S}$.

(b) $k = 1$. New working set. Change set of search directions. Move North; $k \in \mathcal{S}$ and $k \in \mathcal{U}_T$.

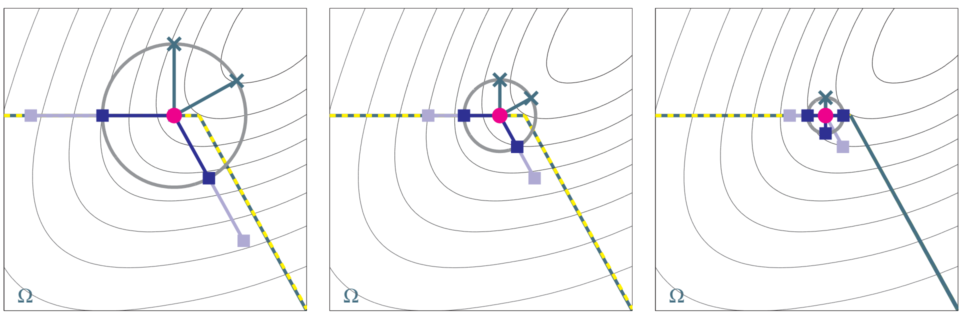
(c) $k = 2$. Same working set. Keep set of search directions. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$.

FIG. 4.3. *A version of linearly constrained GSS applied to the modified Broyden tridiagonal function augmented with two linear inequality constraints.*

At the first iteration, illustrated in Figure 4.3(a), we choose $\mathcal{G}_0 = \{e_1, -e_1, -e_2\}$, where e_i, $i \in \{1, 2\}$ is a unit coordinate vector. The choice of e_1 and $-e_1$ is dictated by the single constraint in the working set. Since $T(x_0, \varepsilon_0)$ is a nontrivial lineality space—the same one encountered in Figure 4.1—we require a third vector to ensure that $\mathcal{G}_0$ generates $T(x_0, \varepsilon_0)$; $-e_2$ is a sensible choice as it yields the best possible value of $\kappa(\mathcal{G}_0)$ while using the smallest possible number of generators for $T(x_0, \varepsilon_0)$. We also elect to include the vector e_2, which generates $N(x_0, \varepsilon_0)$, since it allows the search to move toward the constraint contained in the working set. Thus, $\mathcal{H}_0 = \{e_2\}$. A full step of length Δ_0 along e_2 is not feasible, so we reduce the length of the step so we stop at the boundary, leading to a step of the form $\tilde{\Delta}(e_2)e_2$. The step to the East satisfies (4.3), so this trial point becomes the next iterate and we assign $k = 0$ to $\mathcal{S}$.

At the next iteration, illustrated in Figure 4.3(b), the second constraint enters the working set of constraints. This gives us a new core set $\mathcal{G}_1$ containing only two vectors. However, none of the trial steps defined by the vectors in $\mathcal{G}_1$ improve upon the value of the objective at the current iterate. Again, $\mathcal{H}_1$ comprises a set of generators for the ε-normal cone. By considering the feasible steps along the directions in $\mathcal{H}_1$ we see the most improvement by taking the step to the North, so that is the step we accept. We assign $k = 1$ to both $\mathcal{S}$ and $\mathcal{U}_T$ since, while the iteration was a success, no improvement was found along the directions defined by the vectors in $\mathcal{G}_1$.

At the next iteration, illustrated in Figure 4.3(c), the working set remains unchanged, so we do not change the set of search directions: $\mathcal{G}_2 \leftarrow \mathcal{G}_1$ and $\mathcal{H}_2 \leftarrow \mathcal{H}_1$. Once again, the full steps of length Δ_2 along the vectors contained in the core set $\mathcal{G}_2$ are feasible but do not yield improvement. Moreover, there are no feasible steps along the two vectors in $\mathcal{H}_2$. Since no feasible decrease has been found, the iteration is *unsuccessful*; the current iterate is unchanged, $k = 2$ is assigned to the set $\mathcal{U}$ as well as to the set $\mathcal{U}_T$, and $\Delta_3 \leftarrow \frac{1}{2}\Delta_2$. The same is true in the next two iterations, illustrated (at a finer scale) in Figures 4.4(a)–4.4(b).



(a) $k = 3$. Same working set. Keep set of search directions. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$.

(b) $k = 4$. Same working set. Keep set of search directions. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$.

(c) $k = 5$. New working set (with respect to $k = 4$). Change set of search directions. Move East; $k \in \mathcal{S}$.

FIG. 4.4. *A continuation (at a finer scale) of the example started in Figure 4.3.*

At iteration $k = 5$, illustrated in Figure 4.4(c), the reductions in Δ_k finally pay off. Now $\varepsilon_5 = \Delta_5$ is sufficiently small that the second constraint is dropped from the working set. Then it is possible to take a feasible step of length Δ_5 along the vector $e_1 \in \mathcal{G}_5$. The iteration is successful and $k = 5$ is assigned to the set $\mathcal{S}$.

4.6. The algorithm. Algorithm 4.1 restates Algorithm 5.1 from [15]. We assume that the search directions in $\mathcal{G}_k$ have been normalized, though this simplification is not essential to the results here. We use (4.3) as our new sufficient decrease criterion. We also now keep track of tangentially unsuccessful iterations.

4.7. Stationarity results. Next we examine the relevant convergence properties of Algorithm 4.1. The results in this section and their proofs are extensions of results in [17, 13, 15]. They are key to active set identification for GSS.

The following theorem, similar to [15, Theorem 6.3] shows that for the subsequence of iterations $k \in \mathcal{U}_T$, Algorithm 4.1 bounds the size of the projection of $-\nabla f(x_k)$ onto $T(x_k, \varepsilon_k)$ as a function of the step-length control parameter Δ_k. The two modifications introduced in Algorithm 4.1 mean that Theorem 4.2 differs from

Step 0. Initialization. Let $x_0 \in \Omega$ be the initial iterate. Let $\Delta_{\text{tol}} > 0$ be the tolerance used to test for convergence. Let $\Delta_0 > \Delta_{\text{tol}}$ be the initial value of the step-length control parameter. Let $\Delta_0 \le \Delta_{\max} < \infty$ be an upper bound on the step-length control parameter. Let $\varepsilon_{\max} > \Delta_{\text{tol}}$ be the maximum distance used to identify nearby constraints ($\varepsilon_{\max} = +\infty$ is permissible). Let $\alpha > 0$. Let $\rho_k(\Delta_k) = \alpha \max\left\{|f_{\text{typ}}|,\, |f(x_k)|\right\} \Delta_k^2$, where $f_{\text{typ}} \neq 0$ is some value that reflects the typical magnitude of the objective for $x \in \Omega$ (use $f_{\text{typ}} = 1$ as a default). Let $1 \le \phi_{\max} < \infty$. Let $0 < \theta_{\max} < 1$.

Step 1. Choose search directions. Let $\varepsilon_k = \min\{\varepsilon_{\max}, \Delta_k\}$. Choose a set of search directions $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$ satisfying Condition 4.1. Normalize the core search directions in $\mathcal{G}_k$ so that $\| d \| = 1$ for all $d \in \mathcal{G}_k$.

Step 2. Look for decrease. Consider trial steps of the form $x_k + \tilde{\Delta}(d)d$ for $d \in \mathcal{D}_k$, where $\tilde{\Delta}(d)$ is as defined in (4.2), until either finding a $d_k \in \mathcal{D}_k$ that satisfies (4.3) (a successful iteration) or determining that (4.3) is not satisfied for any $d \in \mathcal{G}_k$ (an unsuccessful iteration).

Step 3. Successful Iteration. If there exists $d_k \in \mathcal{D}_k$ such that (4.3) holds, then:

– Set $x_{k+1} = x_k + \tilde{\Delta}(d_k)d_k$.

– Set
$$\Delta_{k+1} = \min\{\Delta_{\max}, \phi_k \Delta_k\} \text{ with } 1 \le \phi_k \le \phi_{\max}. \tag{4.6}$$

– Set $\mathcal{S} = \mathcal{S} \cup \{k\}$.

– If during Step 2 it was determined that (4.3) is not satisfied for any $d \in \mathcal{G}_k$, set $\mathcal{U}_T = \mathcal{U}_T \cup \{k\}$.

Step 4. Unsuccessful Iteration. Otherwise,

– Set $x_{k+1} = x_k$.

– Set
$$\Delta_{k+1} = \theta_k \Delta_k \text{ with } 0 < \theta_k \le \theta_{\max}. \tag{4.7}$$

– Set $\mathcal{U} = \mathcal{U} \cup \{k\}$.

– Set $\mathcal{U}_T = \mathcal{U}_T \cup \{k\}$.

If $\Delta_{k+1} < \Delta_{\text{tol}}$, then terminate.

Step 5. Advance. Increment k by one and go to Step 1.

Algorithm 4.1: A linearly constrained GSS algorithm

[15, Theorem 6.3] in two respects. First, we have replaced $\rho(\Delta_k)$ with a $\rho_k(\Delta_k)$ that does not satisfy the requirements on $\rho(\cdot)$ found in [15, Condition 4]. Second, the bound (4.8) is shown to hold for the subsequence $\mathcal{U}_T$, not just the subsequence $\mathcal{U}$ (recall that $\mathcal{U} \subseteq \mathcal{U}_T$).

THEOREM 4.2. *Suppose that ∇f is Lipschitz continuous with constant M on Ω. Consider the iterates produced by Algorithm 4.1. If $k \in \mathcal{U}_T$ and ε_k satisfies $\varepsilon_k = \Delta_k$,* *then*

$$\| [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \| \leq \frac{1}{\kappa_{\min}} (M + \alpha_k) \Delta_k, \tag{4.8}$$

where $\kappa_{\min}$ *is from* Condition *4.1* *and* α_k *is from* *(4.5)*.

Proof. Clearly, we need only consider the case when $[-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \neq 0$. Condition 4.1 guarantees a set $\mathcal{G}_k$ that generates $T(x_k, \varepsilon_k)$. Recall that Algorithm 4.1 requires $\| d \| = 1$ for all $d \in \mathcal{G}_k$. Thus, by (4.1) with $K = T(x_k, \varepsilon_k)$ and $v = -\nabla f(x_k)$, there exists some $\hat{d} \in \mathcal{G}_k$ such that

$$\kappa_{\min} \| [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \| \leq -\nabla f(x_k)^T \hat{d}. \tag{4.9}$$

The fact that $k \in \mathcal{U}_T$ ensures that $f(x_k + \tilde{\Delta}(d)d) \geq f(x_k) - \rho_k(\Delta_k)$ for all $d \in \mathcal{G}_k$. By assumption $\varepsilon_k = \Delta_k$ and Algorithm 4.1 requires $\| d \| = 1$ for all $d \in \mathcal{G}_k$, so $\| \Delta_k d \| = \varepsilon_k$ for all $d \in \mathcal{G}_k$. From [15, Proposition 2.2] we know that if $x \in \Omega$ and $v \in T(x, \varepsilon)$ satisfies $\| v \| \leq \varepsilon$, then $x + v \in \Omega$. Therefore $x_k + \Delta_k d \in \Omega$ for all $d \in \mathcal{G}_k$. This, together with (4.2), ensures

$$f(x_k + \Delta_k d) - f(x_k) + \rho_k(\Delta_k) \geq 0 \quad \text{for all } d \in \mathcal{G}_k. \tag{4.10}$$

Meanwhile, the mean value theorem says that for each $d \in \mathcal{G}_k$ there is some $\sigma_k(d) \in (0, 1)$ for which

$$f(x_k + \Delta_k d) - f(x_k) = \Delta_k \nabla f(x_k + \sigma_k(d)\Delta_k d)^T d \text{ for all } d \in \mathcal{G}_k.$$

Combining this with (4.10) yields

$$0 \leq \Delta_k \nabla f(x_k + \sigma_k(d)\Delta_k d)^T d + \rho_k(\Delta_k) \text{ for all } d \in \mathcal{G}_k.$$

Dividing through by Δ_k and subtracting $\nabla f(x_k)^T d$ from both sides leads to

$$-\nabla f(x_k)^T d \leq (\nabla f(x_k + \sigma_k(d)\Delta_k d) - \nabla f(x_k))^T d + \frac{\rho_k(\Delta_k)}{\Delta_k} \text{ for all } d \in \mathcal{G}_k.$$

Since $0 < \sigma_k(d) < 1$ and $\| d \| = 1$ for all $d \in \mathcal{G}_k$, from the assumption that ∇f is Lipschitz continuous with constant M on Ω we obtain

$$-\nabla f(x_k)^T d \leq M\Delta_k + \frac{\rho_k(\Delta_k)}{\Delta_k} \text{ for all } d \in \mathcal{G}_k. \tag{4.11}$$

Since (4.11) holds for all $d \in \mathcal{G}_k$, (4.9) tells us that for some $\hat{d} \in \mathcal{G}_k$,

$$\kappa_{\min} \| [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \| \leq M\Delta_k + \frac{\rho_k(\Delta_k)}{\Delta_k}.$$

Using (4.4) finishes the proof:

$$\| [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \| \leq \frac{1}{\kappa_{\min}} (M + \alpha_k) \Delta_k. \tag{4.12}$$

□

From (4.12) it is easy to see that the choice of α_k in (4.5) yields a bound on the *relative* size of the projection of $-\nabla f(x_k)$ onto $T(x_k, \varepsilon_k)$:

$$\| [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \| \leq \frac{1}{\kappa_{\min}} \left(M + \alpha \max\left\{|f_{\text{typ}}|, |f(x_k)|\right\}\right) \Delta_k.$$

The next result strengthens [13, Theorem 3.4] while replacing the function ρ from [13, 15, 16] by the functions ρ_k.

THEOREM 4.3. *Consider the iterates produced by Algorithm 4.1. Then either* $\lim_{k\to\infty} \Delta_k = 0$ *or* $\lim_{k\to\infty} f(x_k) = -\infty$.

Proof. Suppose $\lim_{k\to\infty} \Delta_k \neq 0$. Observe that there must be at least one successful iteration, since otherwise every iteration would be unsuccessful and the updating rule (4.7) would ensure that $\lim_{k\to\infty} \Delta_k = 0$. Let $\bar{k}$ be the first successful iteration.

Let $\Delta^\star = \limsup_{k\to\infty} \Delta_k$. Since $\lim_{k\to\infty} \Delta_k \neq 0$ we know that $\Delta^\star > 0$. Recalling that Step 0 in Algorithm 4.1 requires $1 \leq \phi_{\max} < \infty$, define

$$\begin{aligned} \mathcal{Q} &= \left\{ \, k \mid k \geq \bar{k}, \ \Delta_k \geq \Delta^\star/2 \, \right\} \\ \mathcal{Q}' &= \left\{ \, k \mid k \geq \bar{k}, \ \Delta_k \geq \Delta^\star/(2\phi_{\max}) \, \right\}. \end{aligned}$$

We claim that there are infinitely many successful iterations in $\mathcal{Q}'$.

To prove the claim, note that $\mathcal{Q}$ is infinite. If there are infinitely many successful iterations in $\mathcal{Q}$ then there is nothing to prove since $\mathcal{Q} \subseteq \mathcal{Q}'$. On the other hand, suppose there are infinitely many unsuccessful iterations in $\mathcal{Q}$. Let k be any such iteration, and let m be such that iteration $k-m-1$ was the last successful iteration preceding k (there must be such an iteration since $k > \bar{k}$). Since iteration $k-m-1$ was successful and iterations $k-m$ to k were unsuccessful, the update rules say that

$$\begin{aligned} \Delta_k &= \theta_{k-1}\theta_{k-2}\cdots\theta_{k-m}\phi_{k-m-1}\Delta_{k-m-1} \\ \Delta_{k-m-1} &= (\theta_{k-1}\theta_{k-2}\cdots\theta_{k-m}\phi_{k-m-1})^{-1}\Delta_k. \end{aligned}$$

From (4.6) and (4.7), along with the fact that $k \in \mathcal{Q}$, we obtain

$$\Delta_{\max} \geq \Delta_{k-m-1} \geq \frac{1}{(\theta_{\max})^m}\frac{\Delta^\star}{2\phi_{\max}} \geq \frac{\Delta^\star}{2\phi_{\max}}.$$

Since $k-m-1 \geq \bar{k}$, we conclude that $k-m-1 \in \mathcal{Q}'$. Moreover,

$$(\theta_{\max})^m \geq \frac{\Delta^\star}{2\phi_{\max}\Delta_{\max}}.$$

Using the fact $\log\theta_{\max} < 0$ because $\theta_{\max} < 1$ we obtain

$$m \leq \bar{m} = \left\lceil \frac{\log\frac{\Delta^\star}{2\phi_{\max}\Delta_{\max}}}{\log\theta_{\max}} \right\rceil.$$

Note that $\bar{m}$ does not depend on k. Thus, for each of the infinitely many unsuccessful iterations $k \in \mathcal{Q}$ there is a successful iteration in $\mathcal{Q}'$ that precedes k by no more than $\bar{m}+1$ iterations, so there must be infinitely many successful iterations in $\mathcal{Q}'$.

Next, let $\rho^\star = \alpha \left| f_{\mathrm{typ}} \right| (\Delta^\star/(2\phi_{\max}))^2$. Then $\rho^\star > 0$. If $k \in \mathcal{Q}'$ is a successful iteration, it follows from the step acceptance rule (4.3), the definitions (4.4) and (4.5), and the definition of $\mathcal{Q}'$ that

$$f(x_{k+1}) - f(x_k) < -\rho_k(\Delta_k) < -\rho^\star < 0.$$

Meanwhile, for all other iterations (successful iterations not in $\mathcal{Q}'$ and unsuccessful iterations) we have $f(x_{k+1}) \leq f(x_k)$. Since there are infinitely many successful iterations in $\mathcal{Q}'$, it follows that $\lim_{k\to\infty} f(x_k) = -\infty$, and the theorem is proved. ∎

Theorems 4.2 and 4.3 then yield the following.

THEOREM 4.4. *Let ∇f be Lipschitz continuous with constant M on Ω and let $\{x_k\}$ be the sequence of iterates produced by Algorithm 4.1 with ε_k satisfying $\varepsilon_k = \Delta_k$. If $\{x_k\}$ is bounded, or if f is bounded below on Ω, then*

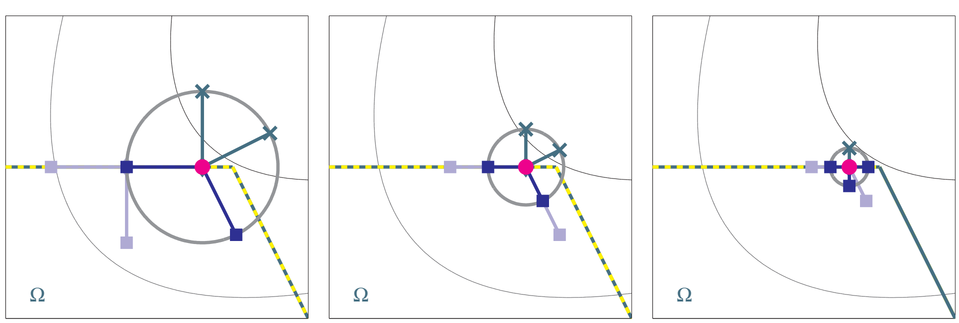
$$\lim_{k \in \mathcal{U}_T \to \infty} \left\| \, [-\nabla f(x_k)]_{T(x_k, \varepsilon_k)} \, \right\| = 0.$$

5. Active set identification properties of GSS. It is straightforward to specify Algorithm 4.1 so the conclusions of Theorems 4.2–4.4 apply. For instance, the choice $\phi_{\max} = 1$ ensures that $\{\Delta_k\}$ is monotonically nonincreasing. Then, the choice $\varepsilon_{\max} \geq \Delta_0$ ensures that $\varepsilon_k = \Delta_k$ for all k.

In this case, any convergent subsequence $\{x_k\}_{\mathcal{K}}$, $\mathcal{K} \subseteq \mathcal{U}_T$, of tangentially unsuccessful iterations produced by Algorithm 4.1 satisfies Assumption 3.1, and the identification results of Section 3 hold. Thus, for problem (1.1) GSS has active set identification properties like those of gradient projection—even though GSS does *not* explicitly rely on derivatives of the objective. Theorem 3.4 allows us to make inferences about the constraints active at $x_\star$ from the working sets $I(x_k, \varepsilon_k)$ at tangentially unsuccessful iterates x_k. The stronger results of Theorem 3.7 hold for the auxiliary sequence $\{\hat{x}_k\}_{\mathcal{K}}$, defined in (3.4): under mild assumptions, in a finite number of iterations $\hat{x}_k$ will identify the face containing $x_\star$.

We next turn to strategies for using the results of Section 3 to accelerate GSS algorithms. We first illustrate what can slow progress towards a solution if the search does not make better use of information about the working set $I(x_k, \varepsilon_k)$.

Figures 4.3–4.4 show GSS making slow but steady progress toward the KKT point. Figure 5.1 magnifies the next three iterations. Notice the pattern that emerges: if the working set contains both the constraints that are active at the solution, then no progress is made since there is no search direction that yields feasible improvement. But as soon as the working set drops back down to a single constraint, a feasible step to the East is possible—which is exactly what is needed to move the current iterate toward the solution at the vertex. This behavior is especially contrary, since the search fails to make progress at those steps where it actually has information that suggests where the solution lies.



(a) $k = 6$. New working set (w.r.t. $k = 5$). Change set of search directions. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$.

(b) $k = 7$. Same working set. Keep set of search directions. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$.

(c) $k = 8$. New working set (w.r.t. $k = 7$). Change set of search directions. Move East; $k \in \mathcal{S}$.

FIG. 5.1. *A continuation (at a finer scale) of the example given in Figures 4.3–4.4.*

If k is sufficiently large, then at a tangentially unsuccessful step $-\nabla f(x_k)$ lies primarily in the direction of $N(x_k, \varepsilon_k)$ in the sense that $\| [-\nabla f(x_k)]_{T(x_k,\varepsilon_k)} \| \ll \| [-\nabla f(x_k)]_{N(x_k,\varepsilon_k)} \|$. This suggests we include the generators of $N(x_k, \varepsilon_k)$ in the set of additional search directions $\mathcal{H}_k$ [16, Section 6], as illustrated in Figures 4.3–5.1. However, in this example there are no feasible steps along the generators of $N(x_k, \varepsilon_k)$ once $k > 1$. In general, steps along individual generators of $N(x_k, \varepsilon_k)$ capture only one constraint in $\mathcal{A}(x_\star)$ at a time, as seen in Figure 4.3(b). Active set strategies that change one constraint in the working set at a time are generally not as efficient as those that allow multiple constraints to change [11].

Our resolution is to use $\hat{x}_k$ from (3.4). We now outline some active set strategies for GSS that make use of the sequence $\{\hat{x}_k\}$ and its properties described in Theorem 3.7, as well as other information we can infer from the working set $I(x_k, \varepsilon_k)$.

5.1. Jumping onto the face identified by the working set. If at iteration k we find that (4.3) is not satisfied by any $d \in \mathcal{G}_k$, then the step is tangentially unsuccessful. We then could try the point $\hat{x}_k$ defined by (3.4). If this trial point exists and satisfies (4.3), we can accept it. If it is not acceptable, we could still track $\mathcal{A}(\hat{x}_k)$ as an estimate of $\mathcal{A}(x_\star)$. The rationale for the latter is that we may actually have $\mathcal{A}(\hat{x}_k) = \mathcal{A}(x_\star)$ but might not have taken the right step into this face.

A more aggressive strategy, which we have found to be more effective in our tests, is to try the step $\hat{x}_k$ for all iterations, tangentially unsuccessful or not. If this trial point exists and satisfies the acceptance criterion (4.3), we immediately accept it. This strategy is consistent with our stationarity results since it is equivalent to including the vector for this step (scaled by $1/\Delta_k$) in $\mathcal{H}_k$. In this aggressive strategy we venture that constraints that appear in the working set are active at the solution. This is only guaranteed to be true asymptotically, and then only for the subsequence $k \in \mathcal{U}_T$. However, in our tests we found that GSS had surprisingly few iterations that were tangentially unsuccessful (i.e., $k \in \mathcal{U}_T$) while also being successful (i.e., $k \in \mathcal{S}$). In general, $S \cap \mathcal{U}_T = \emptyset$. Moreover, to ascertain that an iteration is tangentially unsuccessful we must verify that (4.3) is not satisfied for any $d \in \mathcal{G}_k$, which requires $|\mathcal{G}_k|$ evaluations of the objective. This can be expensive when evaluation of the objective is costly.

On the other hand, determining whether the projection onto a face is a successful step requires only one evaluation of the objective. If the step is successful, the work for the iteration is done; otherwise, we revert to the usual strategy after this single speculative objective evaluation. We accept the potential cost of a rejected step since Theorem 3.7 assures us that asymptotically it will work consistently in our favor.

This aggressive strategy might not succeed in the initial iterations if we start far from a solution. For instance, consider a problem with only lower and upper bounds on the variables. Suppose $x_\star$ is the vertex where all the upper bounds are active and x_0 is a feasible point near the vertex where all the lower bounds are active. Choose Δ_0 so that all of the lower bounds, but none of the upper bounds, are contained in the working set $I(x_0, \varepsilon_0)$. Then at $k = 0$, the projection will produce the vertex defined by the lower bounds and most likely the objective function at that point will not satisfy (4.3). However, once the upper bounds begin to enter the working set, Theorem 3.7 tells us this strategy is sound.

5.2. Restricting the search to the face identified by the working set. Theorem 3.7 tells us that a step to $\hat{x}_k$ is a step into a face of Ω that contains $x_\star$ once k is sufficiently large. Once in such a face we have the option of giving priority to searching inside that face. To do so, we (re)order the directions in $\mathcal{G}_k$ so that the trial steps inside the face are evaluated first. If one of these trial steps satisfies (4.3), then we accept it and declare the iteration successful without requiring any further evaluations. However, if none of the steps restricted to the face satisfy (4.3), then we continue the search using the directions remaining in $\mathcal{G}_k$ and/or $\mathcal{H}_k$, which may possibly move the search off of the current face.

Figure 5.2 illustrates this strategy. Rather than automatically attempting to evaluate f at all four of the trial points defined by Δ_k and the set of directions $\mathcal{D}_k$, we start by evaluating the objective at trial steps in the face identified by the single constraint in the working set. If one of these two trial points is successful, we do not consider the trial steps defined by the two directions normal to this constraint. The reasoning here is that, in general, the performance of direct search methods suffers as the number of optimization variables increases. If we can, with reasonable confidence, restrict the search to a face, then the reduced dimension of the problem should make the search more efficient.

If the working set is a reasonable approximation of the active set, then searching first in the face for improvement makes sense. Furthermore, as long as the iterations remain successful, we do not risk premature convergence so long as once we cannot find a successful step in the face we return to a search in the full space.

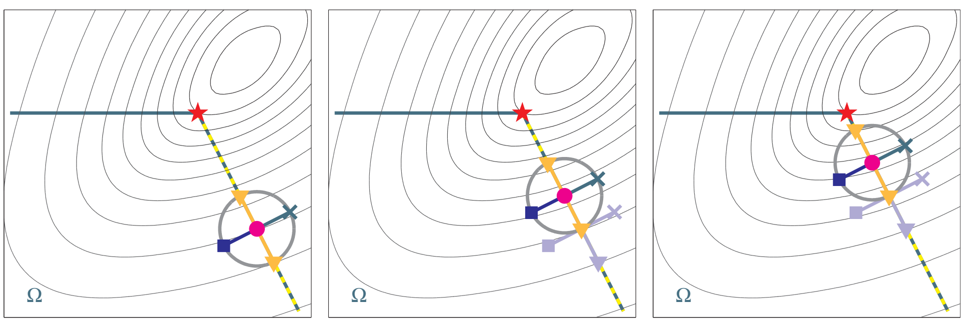
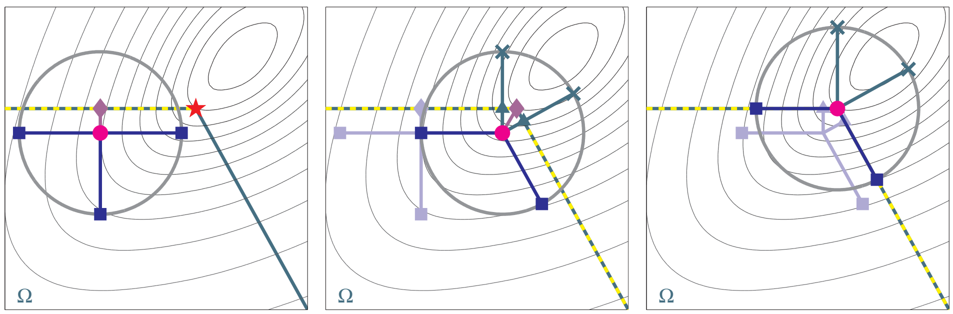


FIG. 5.2. *Giving priority to searching along directions inside a face believed to contain $x_\star$ with ▼ used to denote the steps within the face defined by the constraint in the working set. In this case the third generator of $T(x_k, \varepsilon_k)$ is not needed for the search to progress.*

5.3. Identifying vertex solutions. Another benefit of computing $\hat{x}_k$ is that Theorem 3.7 says that if $\{x_k\}$ is converging to a KKT point $x_\star$ which is a vertex, then $\{x_k\}$ converges to $x_\star$ in a finite number of iterations. If we find a successful step to a vertex and this is followed by an unbroken sequence of unsuccessful iterations, then this suggests that we have arrived at a KKT point.

This gives us another mechanism for terminating the search. Various analytical results, of which Theorem 4.2 is an example, bound appropriately chosen measures of stationarity in terms of the step-length control parameter Δ_k [6, 18, 15]. This makes the magnitude of Δ_k a practical measure of stationarity; hence our use of the test $\theta_k \Delta_k < \Delta_{\text{tol}}$, for some $\Delta_{\text{tol}} > 0$, for termination in Step 4 in Algorithm 4.1. However, we only decrease Δ at the end of an unsuccessful iteration, and to determine that an iteration is unsuccessful requires verifying that (4.3) is not satisfied for any $d \in \mathcal{G}_k$. Experience has shown that GSS methods make good progress toward the solution, but they can be slow and costly in verifying that a solution has been found—precisely because they lack explicit derivative information. In the case of vertex solutions the strategy outlined gives a new stopping rule that might require far fewer objective evaluations, as illustrated by the example in Figure 5.3.

5.4. Illustration of the active set strategies. Figures 5.3–5.4 illustrate what happens if the strategies in Sections 5.1–5.3 are incorporated into Algorithm 4.1 and applied to the same problem used in Figures 4.3–4.4. In Section 6 these same strategies are successfully applied to far more difficult problems from the CUTEr test suite.

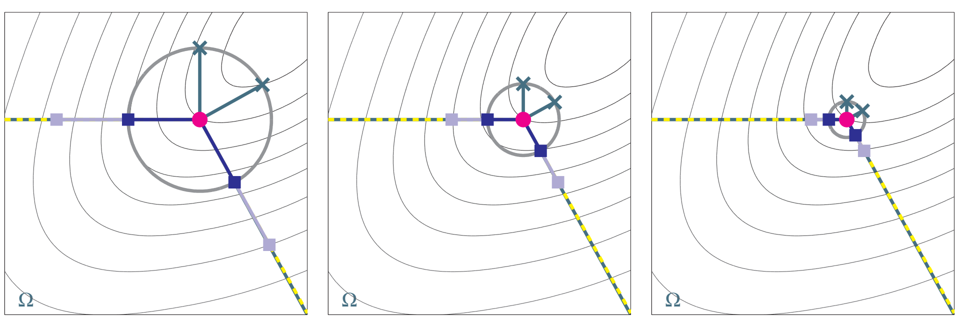


(a) $k = 0$. New working set. Try but reject the speculative step $\hat{x}_k$. Move East; $k \in \mathcal{S}$.

(b) $k = 1$. New working set. Try and accept the speculative step $\hat{x}_k$; $k \in \mathcal{S}$ and $k \in \mathcal{U}_T$.

(c) $k = 2$. Same working set. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$.

FIG. 5.3. *The effect of employing the strategies in Sections 5.1–5.3 on the example used in Figures 4.3–4.4. The speculative step $\hat{x}_k$ defined in (3.4) is denoted by* ♦.



(a) $k = 3$. Same working set. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$.

(b) $k = 4$. Same working set. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$.

(c) $k = 5$. Same working set. No decrease; halve Δ_k; $k \in \mathcal{U}$ and $k \in \mathcal{U}_T$. Possibly stop.

FIG. 5.4. *A continuation (at a finer scale) of the example started in Figure 5.3. For any $k > 1$, Theorem 3.7 supports the conclusion for this example that $x_k = \hat{x}_k = x_\star$ since x_k is at a vertex defined by the constraints in $I(x_k, \varepsilon_k)$, $I(x_k, \varepsilon_k)$ remains the same, and $k \in \mathcal{U}_T$.*

6. Numerical illustration. We demonstrate the effect of the active set strategies outlined in Section 5 on eight problems drawn from the CUTEr test suite [12]. Basic characteristics of these eight problems are given in Table 6.1. In seven of the eight problems the search encountered (at least once) working sets for which $N(x_k, \varepsilon_k)$ has a degenerate vertex at the origin; to handle this situation we employed the strategy outlined in [16, Section 5.4.2], which makes use of state-of-the-art algorithms from computational geometry to deal with the degeneracy in a computationally effective way. We included the problem BLEACHNG, even though it only has bounds on the variables, because it is representative of the type of simulation-based problems to which direct search methods are often applied [13, Section 1.2.1]. For BLEACHNG, evaluation of the objective involves the numerical integration of a system of differential equations. Second derivatives of the objective are not available, computing the objective is relatively expensive, and the values obtained are only moderately accurate.

problem	n	# of linear			# of bounds		CUTEr
		network eq.	eq.	ineq.	lower	upper	classification
AVION2	49	0	15	0	49	49	`OLR2-RN-49-15`
BLEACHNG	17	0	0	0	9	8	`SBR1-RN-17-0`
DALLASM	196	151	0	0	196	196	`ONR2-MN-196-151`
DALLASS	46	31	0	0	46	46	`ONR2-MN-46-31`
HIMMELBI	100	0	0	12	100	100	`OLR2-MN-100-12`
LOADBAL	31	11	0	20	31	11	`OLR2-MN-31-31`
SPANHYD	97	33	16	0	97	97	`ONR2-RN-97-33`
WATER	31	10	0	0	31	31	`ONR2-MN-31-10`

TABLE 6.1. *Summary of problem characteristics*

6.1. Starting values and tolerances for the results reported here. For all eight test problems, we started with the value of x_0 specified by CUTEr, though we often had to project into the feasible region, as discussed in [16, Section 8.1]. The objective values at the feasible points used to initiate the search are given in Table 6.2. The optimal feasible value of the objective function given in Table 6.2 was either extracted from the CUTEr file, when that information is included, or was defined as the best objective value found using SNOPT [10], when that information is not included in the CUTEr file. An exception was made for BLEACHNG since the low solution reported in the CUTEr file is almost twice the value of the best solutions found by both our method and SNOPT; the value reported by SNOPT is treated here as optimal. It is worth noting that on BLEACHNG, SNOPT terminates due to numerical difficulties, indicating that the current point cannot be improved upon; we suspect that at some point the derivatives are not sufficiently accurate due to the ODESSA ODE solver used to evaluate the objective.

problem	$f(x_0)$, $x_0 \in \Omega$	$f(x_*)$, $x_* \in \Omega$	max evals.
AVION2	`9.46803e+07`	`9.46801e+07`	1500
BLEACHNG	`1.59238e+04`	`9.17676e+03`	540
DALLASM	`6.19530e+07`	`-4.81981e+04`	5910
DALLASS	`1.24987e+07`	`-3.2393 e+04`	1410
HIMMELBI	`-7.84832e+02`	`-1.73557e+03`	3030
LOADBAL	`1.54669e+00`	`4.52851e-01`	960
SPANHYD	`2.26888e+10`	`2.39738e+02`	2940
WATER	`1.71709e+04`	`1.05494e+04`	960

TABLE 6.2. *Starting feasible values, best known optimal feasible values, and limits on the number of function evaluations we allowed*

When both lower and upper bounds are given for all the variables in the problem, we scale the variables to lie between -1 and 1 as described in [16, Section 8.4]. An advantage of scaling is that it makes it easier to define the reasonable default value $\Delta_0 = 2$ since 2 is the longest possible feasible step along a unit coordinate direction. Thus we used $\Delta_0 = 2$ as our default, even when we had no prior knowledge of reasonable ranges for the variables and so did not scale the problem.

The algorithm terminated either when the value of Δ_k fell below $\Delta_{\text{tol}} = \texttt{1.0e-05}$ or when the number of evaluations of the objective reached that which would be needed, at a minimum, for 30 iterations of a higher-order nonlinear programming method (such as SNOPT) using forward-difference estimates of the gradient; these limits are given in Table 6.2. The specific value chosen for Δ_{tol} is an acknowledgment of the fact that there are no more than seven significant decimal digits in the specification of the problems drawn from the CUTEr test suite (in fact, the data for several of the problems are specified to only two or three decimal digits). We used a factor of $\frac{1}{2}$ to reduce Δ_k after an unsuccessful iteration; we left Δ_k unchanged after a successful iteration, meaning that $\theta_k = \frac{1}{2}$ and $\phi_k = 1$ for all k. Thus the sequence $\{\Delta_k\}$ was monotonically nonincreasing. This, together with the choices of Δ_0 and Δ_{tol}, meant that Δ_k could be reduced at most eighteen times. The second stopping criterion imposes a fairly stringent limit on the number of evaluations of the objective. Our implementation employs a caching scheme, described in [16, Section 8.5], to avoid reevaluating the objective at points for which the objective has already been computed. Each run starts with the cache empty and only increments the number of evaluations of the objective *once* for evaluating the objective at any given point; there is no increment if in subsequent iterations the value of the objective at that point is found in the cache.

We set $\varepsilon_{\max} = 2^5\Delta_0$ so that it played no role in the results reported here. We set $\alpha = \texttt{1.0e-04}$ and $f_{\text{typ}} = 1$.

When the problem looked unconstrained locally (i.e., the working set was empty), coordinate search was used. Otherwise, the core set of search directions $\mathcal{G}_k$ consisted of generators for the ε-tangent cone $T(x_k, \varepsilon_k)$. A description of the construction of the set $\mathcal{G}_k$ in the presence of linear constraints can be found in [16, Section 5]. The set $\mathcal{H}_k$ consisted of generators for the ε-normal cone $N(x_k, \varepsilon_k)$. The generators for both $T(x_k, \varepsilon_k)$ and $N(x_k, \varepsilon_k)$ were normalized. In addition, whenever $I(x_k, \varepsilon_k)$ was not empty, the set $\mathcal{H}_k$ contained the vector from x_k to the $\hat{x}_k$ defined by (3.4).

The tests were run on an Apple MacBook with a 2 GHz Intel Core 2 Duo processor and 1 GB memory running Mac OS X, Version 10.4.10 and using Matlab 7.4.0 R2007a. With the exception of BLEACHNG, all the runs reported here took no more than one minute to execute, even with graphical and printed output to the screen during execution. The elapsed times for BLEACHNG were around three minutes since invoking ODESSA appreciably increases the cost of an objective evaluation.

6.2. The results. We include a summary and illustrations of the iteration history for each problem. Tables 6.3–6.6 give a summary of the solutions obtained. The active set strategies were designed with two goals: to obtain better solutions by obtaining as large as possible a fraction of optimal improvement, and to identify potential solutions more quickly by driving Δ_k below Δ_{tol} more quickly. The value `f_sol` was the best objective value found by our method given Δ_{tol} and the limit on the number of objective evaluations allowed. The value `f0` was the value at the initial feasible point at which the search commenced. The value `f_opt` was the optimal solution, as defined in Section 6.1. Fraction of optimal improvement was defined as

$$\frac{|\texttt{f0} - \texttt{f\_sol}|}{|\texttt{f0} - \texttt{f\_opt}|}.$$

In Table 6.3 it is clear that the active set strategies, when employed, obtained fractions of optimal improvement that were at least comparable—and generally better—than those obtained without employing the active set strategies.

problem	active set?	objective value at termination	fraction of opt. improvement
AVION2	no	9.46801e+07	0.99983
AVION2	yes	9.46801e+07	0.99999
BLEACHNG	no	9.17676e+03	1.00000
BLEACHNG	yes	9.17676e+03	1.00000
DALLASM	no	-4.81525e+04	0.99999
DALLASM	yes	-4.81311e+04	0.99999
DALLASS	no	-3.22905e+04	0.99999
DALLASS	yes	-3.22406e+04	0.99999
HIMMELBI	no	-1.72387e+03	0.98770
HIMMELBI	yes	-1.73488e+03	0.99928
LOADBAL	no	4.52897e-01	0.99996
LOADBAL	yes	4.52851e-01	1.00000
SPANHYD	no	2.39738e+02	1.00000
SPANHYD	yes	2.39739e+02	1.00000
WATER	no	1.05992e+04	0.99247
WATER	yes	1.05494e+04	1.00000

TABLE 6.3. *Summary of solutions obtained*

The results in Table 6.4 show that the active set strategies can help identify solutions more quickly by driving Δ_k below Δ_{tol} more quickly. In four of the instances, the run using the active set strategies terminated with $\Delta_k < \Delta_{\mathrm{tol}}$, which occurred for only two of the runs that did not employ the active set strategies. Furthermore, even when the runs using the active set strategies terminated because the limit on the number of objective evaluations was reached, the value of Δ_k obtained using the active set strategies was at least as low, if not lower, than the value obtained when the active set strategies were not employed.

problem	active set?	reason terminated	Δ_k at termination	reductions of Δ_k	# of obj. evals.	cache hits
AVION2	no	max evals.	1.220703e-04	14	1500	882
AVION2	yes	$\Delta_k < \Delta_{\mathrm{tol}}$	7.629395e-06	18	852	560
BLEACHNG	no	$\Delta_k < \Delta_{\mathrm{tol}}$	7.629395e-06	18	368	195
BLEACHNG	yes	$\Delta_k < \Delta_{\mathrm{tol}}$	7.629395e-06	18	329	115
DALLASM	no	max evals.	7.812500e-03	8	5910	931
DALLASM	yes	max evals.	7.812500e-03	8	5910	701
DALLASS	no	max evals.	1.562500e-02	7	1410	273
DALLASS	yes	max evals.	7.812500e-03	8	1410	275
HIMMELBI	no	max evals.	7.812500e-03	8	3030	1288
HIMMELBI	yes	max evals.	7.812500e-03	8	3030	1187
LOADBAL	no	max evals.	3.125000e-02	6	960	466
LOADBAL	yes	max evals.	3.051758e-05	16	960	357
SPANHYD	no	$\Delta_k < \Delta_{\mathrm{tol}}$	7.629395e-06	18	1110	2023
SPANHYD	yes	$\Delta_k < \Delta_{\mathrm{tol}}$	7.629395e-06	18	1121	1519
WATER	no	max evals.	7.812500e-03	8	960	899
WATER	yes	$\Delta_k < \Delta_{\mathrm{tol}}$	7.629395e-06	18	868	585

TABLE 6.4. *Summary of state at termination*

Figures 6.2–6.9 give the iteration histories of the test problems. For each problem there is a pair of graphs, the legends for which are given in Figure 6.1. Each pair of graphs shows the progress made by the algorithm, with and without the active set strategies employed, along with the value of Δ_k relative to the number of objective evaluations. In every case, at least one of the two active set steps, either a step to the face identified by the working set of constraints or a step within the face identified by the working set, was accepted during the course of the search.

Value of objective/Value of Δ	
—	$f(x_\star)$
◆	$f(x_k)$, $k \in \mathcal{S}$; accepted step $\hat{x}_k$ to face identified by $I(x_k, \varepsilon_k)$
▼	$f(x_k)$, $k \in \mathcal{S}$; accepted step within face identified by $I(x_k, 0)$
○	$f(x_k)$, $k \in \mathcal{S}$, accepted regular search step
×	$f(x_k)$, $k \in \mathcal{U}$
△	magnitude of Δ_k

FIG. 6.1. *Legend for Figures 6.2–6.9.*

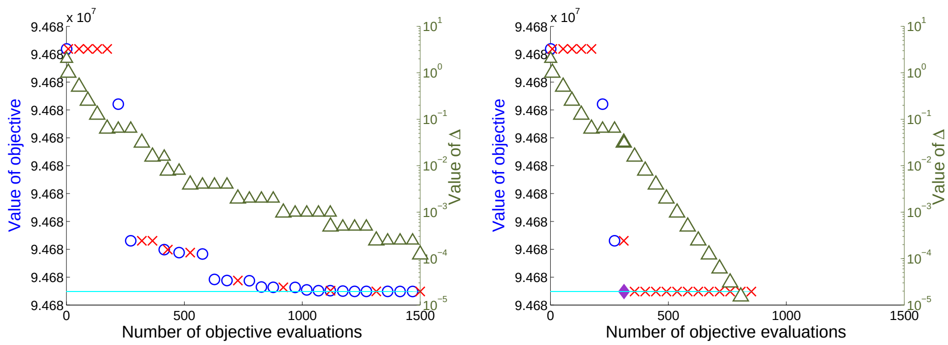


FIG. 6.2. *Progress on* AVION2. *On the left, without the active set strategies enabled. On the right, with the active set strategies enabled.*

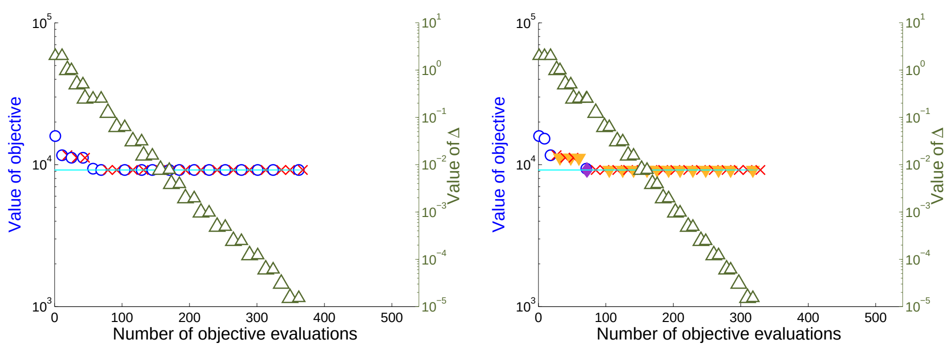


FIG. 6.3. *Progress on* BLEACHNG. *On the left, without the active set strategies enabled. On the right, with the active set strategies enabled.*

In addition to the results illustrated in Figures 6.2–6.9, Figures 6.10–6.13 show how our results compare with those found using SNOPT. The runs from SNOPT were made using the finite-difference option so that the count of objective evaluations can be compared. In all cases our algorithm exhibits the good global behavior one would expect from a gradient-related method and is competitive with SNOPT in this regard.

7. Conclusions. We have presented some general results on active set identification and shown that as a consequence the generating set search class of direct search algorithms possess active set identification properties that are as strong as those of

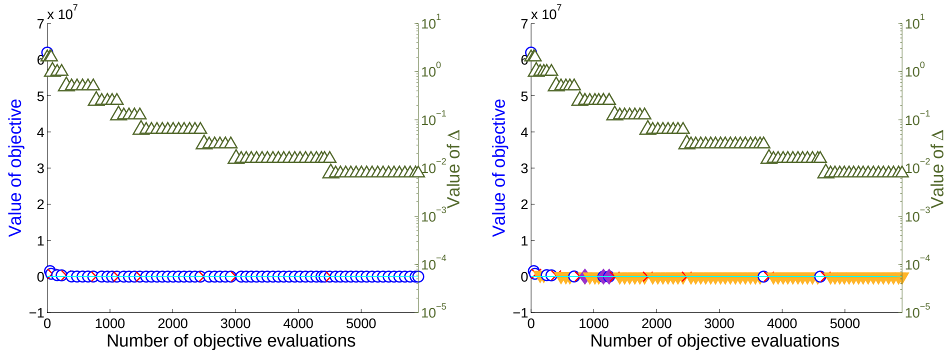


FIG. 6.4. *Progress on* DALLASM. *On the left, without the active set strategies enabled. On the right, with the active set strategies enabled.*

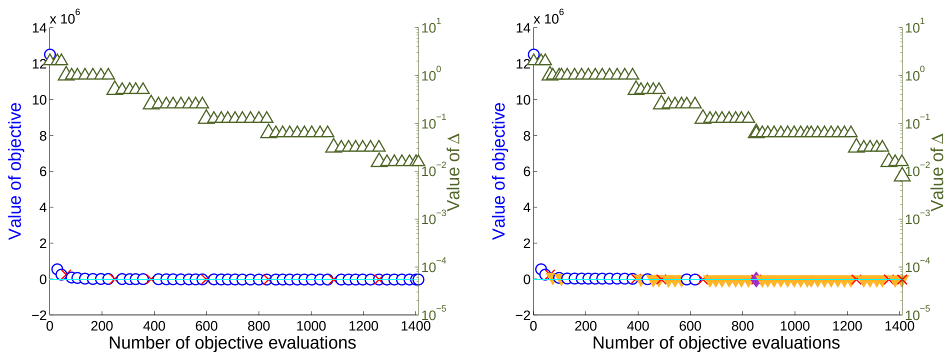


FIG. 6.5. *Progress on* DALLASS. *On the left, without the active set strategies enabled. On the right, with the active set strategies enabled.*

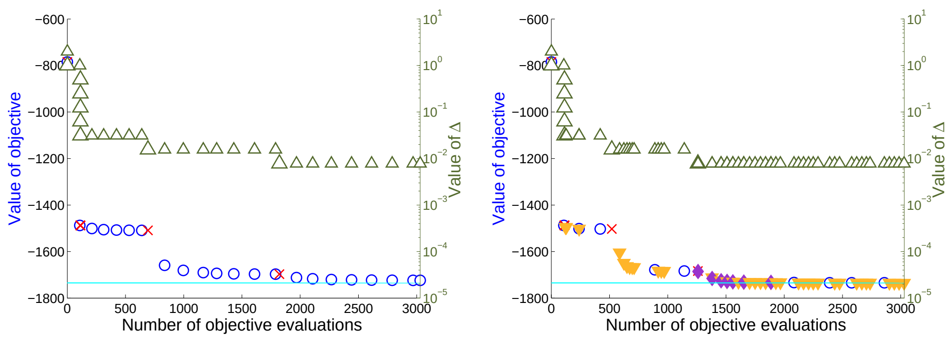


FIG. 6.6. *Progress on* HIMMELBI. *On the left, without the active set strategies enabled. On the right, with the active set strategies enabled.*

the gradient projection method. This is the case even though generating set search does not have explicit recourse to derivatives.

Our general analytical results show that the working sets of constraints at the standard sequence of iterates produced by generating set search will, under relatively mild assumptions, identify those constraints active at a KKT point for which there

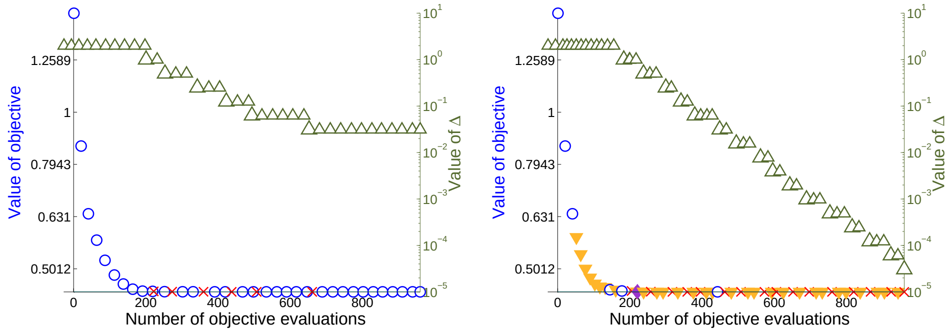


FIG. 6.7. *Progress on* LOADBAL. *On the left, without the active set strategies enabled. On the right, with the active set strategies enabled.*

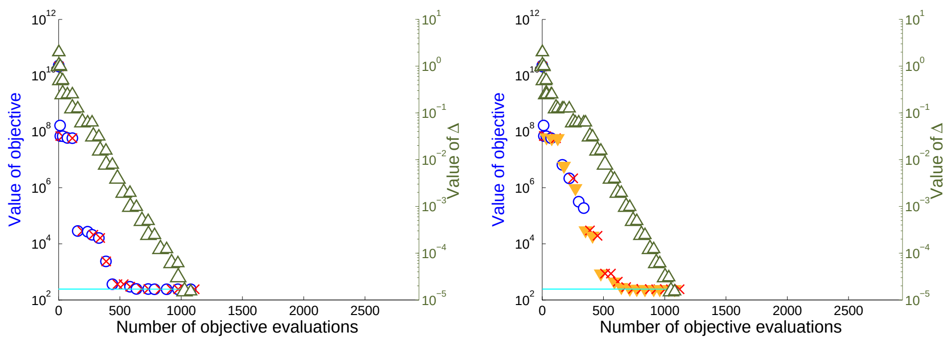


FIG. 6.8. *Progress on* SPANHYD. *On the left, without the active set strategies enabled. On the right, with the active set strategies enabled.*

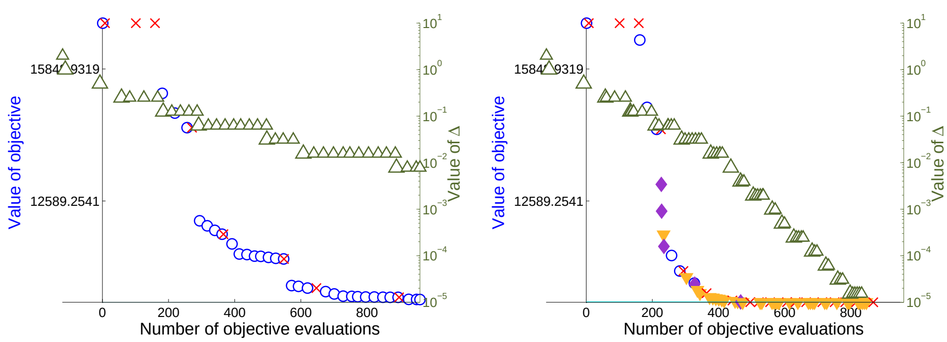


FIG. 6.9. *Progress on* WATER. *On the left, without the active set strategies enabled. On the right, with the active set strategies enabled.*

exist nonzero Lagrange multipliers. The general results also show how a straightforward modification of generating set search will enjoy active set identification properties comparable to the gradient projection algorithm.

We have used these ideas on active set identification to modify our implementation of GSS for solving (1.1) [16] and thus accelerate the progress of the search. This claim is substantiated with results obtained by applying the new variant of GSS for linearly constrained problems to a subset of difficult problems from the CUTEr test suite.

problem	active set?	maximum violation at solution lower bounds	upper bounds
AVION2	no	0.00000e+00	0.00000e+00
AVION2	yes	1.77636e-15	0.00000e+00
BLEACHNG	no	0.00000e+00	0.00000e+00
BLEACHNG	yes	0.00000e+00	0.00000e+00
DALLASM	no	3.55271e-14	0.00000e+00
DALLASM	yes	1.77636e-14	0.00000e+00
DALLASS	no	0.00000e+00	0.00000e+00
DALLASS	yes	0.00000e+00	0.00000e+00
HIMMELBI	no	3.41061e-13	0.00000e+00
HIMMELBI	yes	2.67164e-12	0.00000e+00
LOADBAL	no	3.27056e-15	0.00000e+00
LOADBAL	yes	1.67351e-14	0.00000e+00
SPANHYD	no	1.42109e-14	0.00000e+00
SPANHYD	yes	0.00000e+00	0.00000e+00
WATER	no	9.09495e-13	0.00000e+00
WATER	yes	5.68434e-13	0.00000e+00

TABLE 6.5. *Summary of feasibility of simple bounds at termination.*

problem	active set?	maximum violation at solution linear lower bounds	linear upper bounds
AVION2	no	3.63798e-12	4.54747e-13
AVION2	yes	2.54659e-11	3.92220e-12
DALLASM	no	4.17028e-13	3.62377e-13
DALLASM	yes	2.23821e-13	1.77636e-13
DALLASS	no	2.90323e-13	2.93099e-13
DALLASS	yes	1.56319e-13	9.76996e-14
HIMMELBI	no	0.00000e+00	2.67164e-12
HIMMELBI	yes	1.70530e-13	5.57066e-12
LOADBAL	no	1.77636e-14	1.13687e-13
LOADBAL	yes	2.84217e-14	8.52651e-14
SPANHYD	no	2.16716e-13	7.74492e-13
SPANHYD	yes	2.52243e-13	5.45342e-13
WATER	no	2.27374e-13	2.27374e-13
WATER	yes	2.16005e-12	1.08002e-12

TABLE 6.6. *Summary of feasibility of linear constraints at termination.*

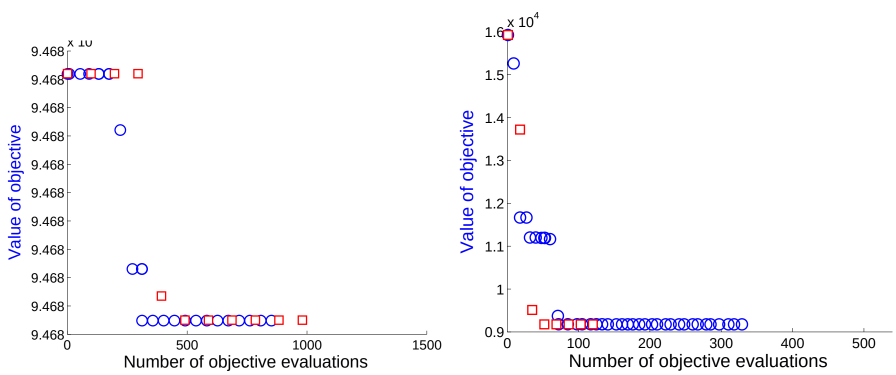


FIG. 6.10. *Progress on* AVION2*, on the left, and* BLEACHNG*, on the right, with the active set strategies enabled (○) vs. progress for* SNOPT *with finite-differencing (□).*

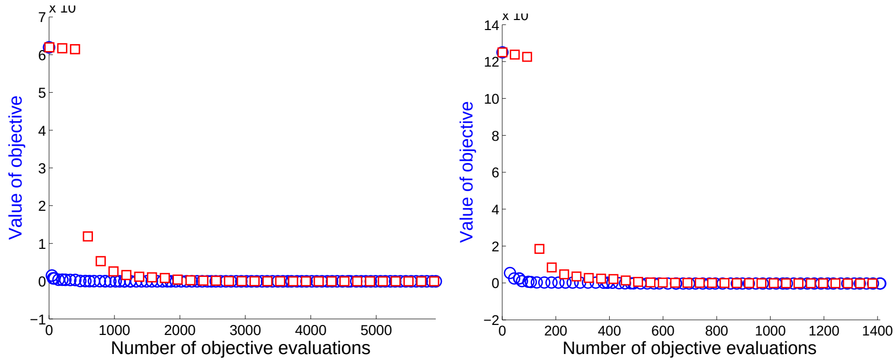


FIG. 6.11. *Progress on* DALLASM,*on the left, and* DALLASS, *on the right, with the active set strategies enabled* (○) *vs. progress for* SNOPT *with finite-differencing* (□).

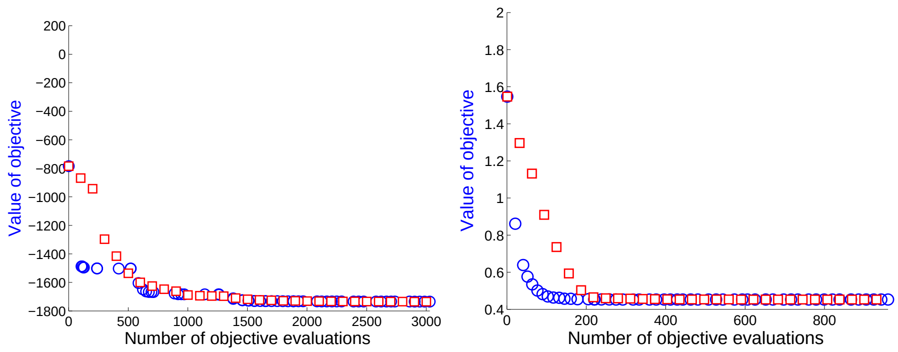


FIG. 6.12. *Progress on* HIMMELBI,*on the left, and* LOADBAL, *on the right, with the active set strategies enabled* (○) *vs. progress for* SNOPT *with finite-differencing* (□).

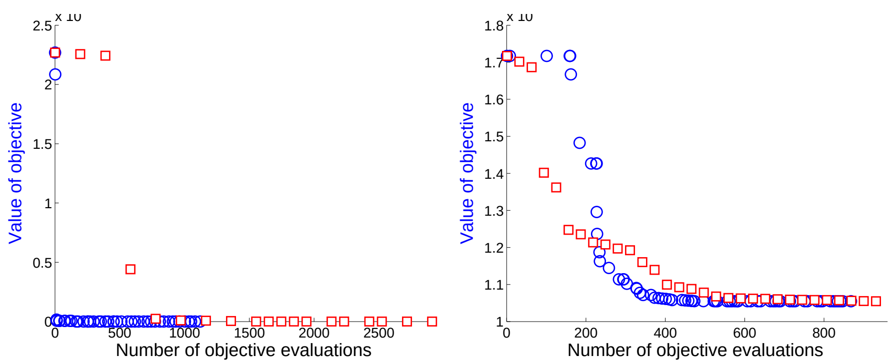


FIG. 6.13. *Progress on* SPANHYD,*on the left, and* WATER, *on the right, with the active set strategies enabled* (○) *vs. progress for* SNOPT *with finite-differencing* (□).

We plan to extend this approach to handle problems with nonlinear constraints for which derivatives can be obtained. Under appropriate constraint qualifications, similar active set results should hold for GSS methods should there be sufficiently accurate gradients to allow the linearization of nonlinear constraints, as in the approaches considered in [19]. In addition, the active set identification results may also be useful in conjunction with the treatment of inequality constraints via nonnegative slacks, as in the augmented Lagrangian approaches described in [18, 14].

Acknowledgments. The authors are deeply grateful to Philip Gill for making SNOPT available for use in the numerical tests reported in this paper. We also sincerely thank the anonymous referee and Jim Burke (the second referee, who revealed his identity), for their careful reading of the paper and their numerous corrections, suggestions, and questions, all of which greatly improved the paper. Finally, we thank Margaret Wright, Associate Editor, for her gracious help throughout the review process.

REFERENCES

[1] C. G. Broyden, *A class of methods for solving nonlinear simultaneous equations*, Mathematics of Computation, 19 (1965), pp. 577–593.

[2] J. Burke, *On the identification of active constraints II: The nonconvex case*, SIAM J. Numer. Anal., 27 (1990), pp. 1081–1102.

[3] J. V. Burke and J. J. Moré, *On the identification of active constraints*, SIAM J. Numer. Anal., 25 (1988), pp. 1197–1211.

[4] ———, *Exposing constraints*, SIAM J. Optim., 4 (1994), pp. 573–595.

[5] P. H. Calamai and J. J. Moré, *Projected gradient methods for linearly constrained problems*, Mathematical Programming, 39 (1987), pp. 93–116.

[6] E. D. Dolan, R. M. Lewis, and V. J. Torczon, *On the local convergence properties of pattern search*, SIAM J. Optim., 14 (2003), pp. 567–583.

[7] F. Facchinei, A. Fischer, and C. Kanzow, *On the accurate identification of active constraints*, SIAM J. Optim., 9 (1998), pp. 14–32.

[8] K. Fukuda, *cddlib*, 2005. http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html.

[9] K. Fukuda and A. Prodon, *Double description method revisited*, in Combinatorics and Computer Science, M. Deza, R. Euler, and I. Manoussakis, eds., vol. 1120 of Lecture Notes in Computer Science, Springer-Verlag, 1997, pp. 91–111.

[10] P. E. Gill, W. Murray, and M. A. Saunders, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM J. Optim., 12 (2002), pp. 979–1006. See also http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm.

[11] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, London, 1981.

[12] N. I. M. Gould, D. Orban, and P. L. Toint, *CUTEr (and SifDec), a constrained and unconstrained testing environment, revisited*, ACM Trans. Math. Software, 29 (2003), pp. 373–394. See also http://cuter.rl.ac.uk/cuter-www/.

[13] T. G. Kolda, R. M. Lewis, and V. Torczon, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Review, 45 (2003), pp. 385–482.

[14] ———, *A generating set direct search augmented Lagrangian algorithm for optimization with a combination of general and linear constraints*, Tech. Rep. 2006–5315, Sandia National Laboratories, Albuquerque, NM 87185 and Livermore, CA 94550, August 2006.

[15] ———, *Stationarity results for generating set search for linearly constrained optimization*, SIAM J. Optim., 17 (2006), pp. 943–968.

[16] R. M. Lewis, A. Shepherd, and V. Torczon, *Implementing generating set search methods for linearly constrained minimization*, SIAM J. Sci. Comput., 29 (2007), pp. 2507–2530.

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

[18] ———, *A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds*, SIAM J. Optim., 12 (2002), pp. 1075–1089.

[19] S. Lucidi, M. Sciandrone, and P. Tseng, *Objective-derivative-free methods for constrained optimization*, Mathematical Programming, 92 (2002), pp. 37–59.

[20] J. H. May, *Linearly Constrained Nonlinear Programming: A Solution Method That Does Not Require Analytic Derivatives*, PhD thesis, Yale University, New Haven, Connecticut, December 1974.

[21] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *Testing unconstrained optimization software*, ACM Trans. Math. Software, 7 (1981), pp. 17–41.

[22] C. Oberlin and S. J. Wright, *Active set identification in nonlinear programming*, SIAM J. Optim., 17 (2006), pp. 577–605.