ISE Industrial and Systems Engineering

An Adaptive Gradient Sampling Algorithm for Nonsmooth Optimization

Frank E. Curtis
Lehigh University

Xiaocun Quez
Lehigh University

Report: 11T-008

Optimization Methods and Software
Vol. , No. , Month , 1–23

RESEARCH ARTICLE

An Adaptive Gradient Sampling Algorithm for Nonsmooth Optimization

Frank E. Curtis[†] and Xiaocun Que[‡]

(*Received 00 Month 200x; in final form 00 Month 200x*)

We present an algorithm for the minimization of $f : \mathbb{R}^n \to \mathbb{R}$, assumed to be locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D}$ of $\mathbb{R}^n$. The objective f may be nonsmooth and/or nonconvex. The method is based on the gradient sampling algorithm (GS) of Burke, Lewis, and Overton [*SIAM J. Optim.*, 15 (2005), pp. 751-779]. It differs, however, from previously proposed versions of GS in that it is variable-metric and only $O(1)$ gradient evaluations are required per iteration. Numerical experiments illustrate that the algorithm is much more efficient than GS in that it consistently requires significantly fewer gradient evaluations. In addition, the adaptive sampling procedure allows for warm-starting of the quadratic subproblem solver so that the number of subproblem iterations per nonlinear iteration is also reduced. Global convergence of the algorithm is proved assuming that the Hessian approximations are positive definite and bounded, an assumption shown to be true for the proposed Hessian approximation updating strategies.

Keywords: unconstrained optimization, nonsmooth optimization, nonconvex optimization, gradient sampling, line search methods, quadratic optimization, warm-starting

AMS Subject Classification: 49M05, 65K05, 65K10, 90C26, 90C30

1. Introduction

The gradient sampling algorithm (GS), introduced and analyzed by Burke, Lewis, and Overton [4, 5], is a method for minimizing an objective function $f : \mathbb{R}^n \to \mathbb{R}$ that is locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D}$ of $\mathbb{R}^n$. The approach is widely applicable and robust [3, 6, 24], and it is intuitively appealing in that theoretical convergence guarantees hold with probability one without requiring algorithmic modifications to handle nonconvexity.

The theoretical foundations for GS, as well as various extensions, are developing rapidly. Stronger theoretical results than in [5] for both the original algorithm and for various extensions were provided in [22], an extension of the ideas for solving constrained problems was presented in [8], and a variant using only gradient estimates derived via function evaluations appeared in [23]. Continued developments along these lines suggest that GS techniques have the potential to rival bundle methods (BM) [17, 18, 20] in terms of theoretical might and practical performance.

The main goal of this paper is to address three practical limitations of GS as it is presented in [5, 22]. Consider the following remarks.

(1) GS produces approximate ϵ-steepest descent directions by evaluating the gradient of f at $n + 1$ (or more) randomly generated points during each

[†]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA. This author was supported by National Science Foundation grant DMS-1016291. E-mail: frank.e.curtis@gmail.com
[‡]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA, USA. This author was supported by National Science Foundation grant DMS-1016291. E-mail: xiq209@lehigh.edu

ISSN: print/ISSN online
© Taylor & Francis
DOI:
http://www.informaworld.com

iteration. This results in a high computational cost that is especially detrimental when search directions turn out to be unproductive.

(2) Each ϵ-steepest descent direction produced by GS is obtained by the solution of a quadratic optimization subproblem (QO). As the subproblem data is computed afresh for every iteration, the computational effort required to solve each of these subproblems can be significant for large-scale problems.

(3) GS may behave, at best, as a steepest descent method. The use of second order information of the problem functions may be useful, but it is not clear how to incorporate this information effectively in nonsmooth regions.

We address both remarks 1 and 2 by the *adaptive* sampling of gradients over the course of the optimization process. That is, rather than evaluate gradients at a completely new set of points during every iteration k, we maintain a history and reuse any recently stored gradients that were obtained in an ϵ-neighborhood of x_k. This reduces the computational effort of gradient evaluations per iteration, and also provides a clear strategy for warm-starting the QO solver. That is, any gradients corresponding to *active* subproblem constraints during iteration $k-1$ that remain in the set of sample gradients are included in the initial active set when solving the QO during iteration k. We show in our numerical experiments in §5 that adaptive sampling significantly reduces the total number of gradient evaluations required.

We address remark 3 by proposing two novel strategies for updating approximations of second order terms. The first strategy is similar to a limited-memory Broyden-Fletcher-Goldfarb-Shanno (LBFGS) update typical in smooth optimization [27]. Our method is unique, however, in that we incorporate gradient information from sample points instead of that solely at algorithm iterates. We also control the updates so that bounds on the Hessian approximations required for our convergence analysis are obtained. The second strategy we propose — intended solely for nonconvex problems — is entirely novel as far as we are aware. It also involves the incorporation of function information at sample points, but is based on the desire to produce model functions that overestimate the true objective f. Bounds required for our convergence analysis are also proved for this latter strategy. Our numerical experiments in §5 illustrate that reductions in nonlinear iterations, overall QO solver iterations, and function/gradient evaluations are obtained by employing either of our updating strategies for approximating second order information.

The paper is organized as follows. A description of our Adaptive Gradient Sampling algorithm (AGS) is presented in §2. Our updating strategies for approximating second order information are presented and analyzed in §3. Global convergence of a generic AGS algorithm is analyzed in §4. Numerical experiments comparing implementations of GS and variants of AGS on a large test set are presented in §5. This implementation involves a specialized QO solver that has been implemented by enhancing the method proposed in [21]; the details of this solver are described in the Appendix. Finally, concluding remarks are provided in §6.

The analysis in this paper builds on that of Kiwiel in [22]. It should also be noted that ideas of “incremental sampling” and “bundling past information” were briefly mentioned by Kiwiel in [23]. However, our methods are unique from those appearing in these papers as adaptive sampling was not considered in [22], exact gradient information was not used in [23], and our algorithm involves Hessian approximations that were not considered in either article. Still, in addition to the original work by Burke, Lewis, and Overton [5], it is clear that the works of Kiwiel have been inspirational for the work in this paper, not to mention the QO algorithm from [21] that has found a new area of applicability in the context of AGS.

2. Algorithm Description

Consider the unconstrained problem

$$\min_{x} \; f(x) \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz and continuously differentiable in an open dense subset $\mathcal{D}$ of $\mathbb{R}^n$. Letting $\operatorname{cl}\operatorname{conv} S$ denote the closure of the convex hull of a set $S \subseteq \mathbb{R}^n$ and defining the multifunction $\mathbb{G}_\epsilon(x) := \operatorname{cl}\operatorname{conv} \nabla f(\mathbb{B}_\epsilon(x) \cap \mathcal{D})$ where $\mathbb{B}_\epsilon(x) := \{\overline{x} : \|\overline{x} - x\| \leq \epsilon\}$ is the Euclidean ϵ-ball about x, we have the following representation of the Clarke subdifferential [7] of f at x:

$$\overline{\partial} f(x) = \bigcap_{\epsilon > 0} \mathbb{G}_\epsilon(x).$$

Similarly, the Clarke ϵ-subdifferential [12] is given by

$$\overline{\partial}_\epsilon f(x) := \operatorname{cl}\operatorname{conv} \overline{\partial} f(\mathbb{B}_\epsilon(x)).$$

A point x is stationary for f if $0 \in \overline{\partial} f(x)$ and ϵ-stationary if $0 \in \overline{\partial}_\epsilon f(x)$.

At a given iterate x_k and for a given sampling radius $\epsilon_k > 0$, the central idea behind gradient sampling techniques is to approximate $\mathbb{G}_{\epsilon_k}(x_k)$ through the random sampling of gradients in $\mathbb{B}_{\epsilon_k}(x_k) \cap \mathcal{D}$. This set, in turn, approximates the Clarke ϵ_k-subdifferential since, at any x, $\mathbb{G}_\epsilon(x) \subset \overline{\partial}_\epsilon f(x)$ for any $\epsilon \geq 0$ and $\overline{\partial}_{\epsilon'} f(x) \subset \mathbb{G}_{\epsilon''}(x)$ for any $\epsilon'' > \epsilon' \geq 0$. Thus, by locating x_k at which the minimum-norm element of $\mathbb{G}_{\epsilon_k}(x_k)$ is small, reducing the sampling radius, and then repeating the process for $\epsilon_k \to 0$, gradient sampling techniques locate stationary points of f by repeatedly locating (approximate) ϵ_k-stationary points for decreasing values of ϵ_k.

We now present a generic AGS algorithm of which GS is a special case. During iteration k, let $X_k := \{x_{k,1}, \ldots, x_{k,p_k}\}$ (with $x_{k,i} = x_k$ for some i) denote a set of points that have been generated in $B_k := \mathbb{B}_{\epsilon_k}(x_k) \cap \mathcal{D}$, let

$$G_k := \begin{bmatrix} g_{k,1} \cdots g_{k,p_k} \end{bmatrix} \tag{2.2}$$

denote the matrix whose columns are the gradients of f at the points in X_k, and let $H_k \succ 0$. The main computational component of the generic algorithm is the solution of the following QO subproblem:

$$\begin{aligned} \min_{z,d} \;& z + \tfrac{1}{2} d^T H_k d \\ \text{s.t. } & f(x_k)e + G_k^T d \leq ze. \end{aligned} \tag{2.3}$$

(Here, and throughout the paper, e denotes a vector of ones whose length is determined by the context.) Alternatively, one may solve the dual of (2.3), namely

$$\begin{aligned} \max_{\pi} \;& -\tfrac{1}{2} \pi^T G_k^T W_k G_k \pi \\ \text{s.t. } & e^T \pi = 1, \; \pi \geq 0, \end{aligned} \tag{2.4}$$

where $W_k := H_k^{-1} \succ 0$. The solution (z_k, d_k, π_k) of (2.3)–(2.4) has $d_k = -W_k G_k \pi_k$.

The only other major computational component of the algorithm is a backtracking line search, performed after the computation of the search direction d_k. For this purpose, we define the sufficient decrease condition

$$f(x_k + \alpha_k d_k) \leq f(x_k) - \eta \alpha_k d_k^T H_k d_k. \tag{2.5}$$

We set $x_{k+1} \leftarrow x_k + \alpha_k d_k$ for α_k chosen to satisfy (2.5), but in order to ensure that all iterates remain within the set $\mathcal{D}$, it may be necessary to perturb such an x_{k+1}; in such cases, we make use of the perturbed line search conditions

$$\begin{aligned} f(x_{k+1}) \leq{}& f(x_k) - \eta \alpha_k d_k^T H_k d_k & \text{(2.6a)} \\ \text{and } \|x_k + \alpha_k d_k - x_{k+1}\| \leq{}& \min\{\alpha_k, \epsilon_k\} \|d_k\|. & \text{(2.6b)} \end{aligned}$$

See [22] for motivation of these line search conditions and a description of how, given α_k and d_k satisfying (2.5), an x_{k+1} satisfying (2.6) can be found in a finite number of operations.

Our algorithmic framework, AGS, is presented as Algorithm 2.1 below. As it is the approach employed in our numerical experiments, we assume that search directions are obtained by solving the dual problem (2.4) and that both H_k and W_k are computed for all k. It should be clear, however, that in general it is not necessary to compute and store both H_k and W_k.

Algorithm 2.1 Adaptive Gradient Sampling (AGS) Algorithm

```
1: (Initialization): Choose a number of sample points to compute each iteration $\overline{p}$, number of sample points required for a complete line search $p \geq n+1$, sampling radius reduction factor $\psi \in (0,1)$, number of backtracks for an incomplete line search $q \geq 0$, sufficient decrease constant $\eta \in (0,1)$, line search backtracking constant $\kappa \in (0,1)$, and tolerance parameter $\nu > 0$. Choose an initial iterate $x_0 \in \mathcal{D}$, set $X_{-1} \leftarrow \emptyset$, choose an initial sampling radius $\epsilon_0 > 0$, and set $k \leftarrow 0$.
2: (Sample set update): Set $X_k \leftarrow (X_{k-1} \cap B_k) \cup x_k \cup \overline{X}_k$, where the sample set $\overline{X}_k := \{\overline{x}_{k,1}, \ldots, \overline{x}_{k,\overline{p}}\}$ is composed of $\overline{p}$ points generated uniformly in $B_k$. Set $p_k \leftarrow |X_k|$. If $p_k > p$, then remove the $p_k - p$ eldest members of $X_k$ and set $p_k \leftarrow p$. Compute any unknown columns of the matrix $G_k$ defined in (2.2).
3: (Hessian update): Set $H_k \succ 0$ and $W_k = H_k^{-1} \succ 0$, respectively, as approximations of the Hessian and inverse Hessian of $f$ at $x_k$.
4: (Search direction computation): Set $d_k \leftarrow -W_k G_k \pi_k$, where $\pi_k$ solves (2.4).
5: (Stationarity test): If $d_k^T H_k d_k \leq \epsilon_k \leq \nu$, then stop. Otherwise, if $d_k^T H_k d_k \leq \epsilon_k$, then set $x_{k+1} \leftarrow x_k$, $\alpha_k \leftarrow 1$, and $\epsilon_{k+1} \leftarrow \psi \epsilon_k$ and go to step 8.
6: (Backtracking line search): If $|X_k| < p$, then set $\alpha_k$ as the largest value in $\{\kappa^0, \kappa^1, \ldots, \kappa^q\}$ such that (2.5) is satisfied, or set $\alpha_k \leftarrow 0$ if (2.5) is not satisfied for any of these values of $\alpha_k$. If $|X_k| = p$, then set $\alpha_k$ as the largest value in $\{\kappa^0, \kappa^1, \kappa^2, \ldots\}$ such that (2.5) is satisfied.
7: (Iterate update): Set $\epsilon_{k+1} \leftarrow \epsilon_k$. If $x_k + \alpha_k d_k \in \mathcal{D}$, then set $x_{k+1} \leftarrow x_k + \alpha_k d_k$. Otherwise, set $x_{k+1}$ as any point in $\mathcal{D}$ satisfying (2.6).
8: (Iteration increment): Set $k \leftarrow k + 1$ and go to step 2.
```

If $\overline{p} = p \geq n+1$ and $H_k = W_k = I$ for all k, then AGS reduces to GS as proposed in [22]; specifically, it reduces to the variant involving nonnormalized search directions in §4.1 of that paper. We use AGS, therefore, to refer to instantiations of AGS where $\overline{p} < p$ with (potentially) variable H_k. Our numerical experiments in §5 illustrate a variety of practical advantages of AGS over GS, while the analysis in §4 shows that nothing is lost in terms of convergence guarantees when $\overline{p} < p$.

3. Hessian Approximation Strategies

In this section, we present novel techniques for choosing H_k and W_k in the context of AGS. We refer to H_k and W_k, respectively, as approximations of the Hessian and inverse Hessian of f at x_k. These are accurate descriptions for our first strategy as we employ gradient information at sample points to approximate the Hessian and inverse Hessian of f at x_k. However, the descriptions are not entirely accurate for our second strategy as in that case our intention is to form models that overestimate f, and not necessarily to have $H_k \approx \nabla^2 f(x_k)$. Still, for ease of exposition, it will be convenient to refer to H_k and W_k as Hessian and inverse Hessian approximations, respectively, in that context as well.

A critical motivating factor in the design of our Hessian updating strategies is the following assumption needed for our global convergence guarantees in §4.

Assumption 3.1 *There exist $\overline{\xi} \geq \underline{\xi} > 0$ such that, for all k and $d \in \mathbb{R}^n$, we have*

$$\underline{\xi}\|d\|^2 \leq d^T H_k d \leq \overline{\xi}\|d\|^2.$$

For each of our updating strategies, we show that Assumption 3.1 is satisfied. We remark, however, that numerical experience has shown that for nonsmooth problems it is often beneficial to allow Hessian approximations to approach singularity [25]. Thus, our numerical experiments include forms of our updates that ensure Assumption 3.1 is satisfied as well as forms that do not. Either of these forms can be obtained through choices of the user-defined constants defined for each update. Note also that the bounds we provide are worst case bounds that typically would not be tight in practice.

Both of the following strategies employ gradient information — and, in the latter case, function value information — evaluated at points in the sample set X_k. The matrices H_k and W_k are constructed by applying a series of updates to the initial approximations $H_k \leftarrow \mu_k I$ and $W_k \leftarrow \mu_k^{-1} I$ based on this information. As is often the case for Hessian updating strategies, we have found in our numerical experiments that the value μ_k (chosen in our case during each iteration k) is critical for the performance of the algorithm. See §5 for our approach for setting μ_k; for now, all that is required in this section is that $\mu_k > 0$ for all k.

3.1. *LBFGS Updates on Sample Directions*

We consider an updating strategy based on the well-known BFGS formula [2, 10, 11, 30]. During iteration k, the main idea of our update is to use gradient information at the points in X_k to construct H_k and W_k. We begin by initializing $H_k \leftarrow \mu_k I$ and $W_k \leftarrow \mu_k^{-1} I$ and then perform a series of (at most) $p_k := |X_k|$ updates based on $d_{k,i} := x_{k,i} - x_k$ and $y_{k,i} := \nabla f(x_{k,i}) - \nabla f(x_k)$ for $i = 1, \ldots, p_k$. As at most p_k updates are performed, this strategy is most accurately described as a LBFGS approach for setting H_k and W_k [27]. In the end, after all p_k updates are performed, we obtain bounds of the type required in Assumption 3.1 where the constants $\overline{\xi} \geq \underline{\xi} > 0$ depend only on p and user-defined constants $\gamma > 0$ and $\sigma > 0$.

Suppose that updates have been performed for sample points 1 through $i - 1$ and consider the update for sample point i. We know from step 2 of AGS that

$$\|d_{k,i}\|^2 \leq \epsilon_k^2. \tag{3.1}$$

Moreover, we will require that

$$d_{k,i}^T y_{k,i} \geq \gamma \epsilon_k^2 \tag{3.2a}$$

$$\text{and } \|y_{k,i}\|^2 \leq \sigma \epsilon_k^2 \tag{3.2b}$$

for the constants $\gamma > 0$ and $\sigma > 0$ provided by the user. We skip the update for sample point i if (3.2) fails to hold. (For instance, for some i we have $x_{k,i} = x_k$, meaning that $d_{k,i} = y_{k,i} = 0$ and (3.2a) is not satisfied.) For ease of exposition, however, we suppose throughout the remainder of this subsection that no updates are skipped, this assumption not invalidating our main result, Theorem 3.3.

The update formulas for H_k and W_k for sample point i are the following:

$$H_k \leftarrow H_k - \frac{H_k d_{k,i} d_{k,i}^T H_k}{d_{k,i}^T H_k d_{k,i}} + \frac{y_{k,i} y_{k,i}^T}{y_{k,i}^T d_{k,i}} \tag{3.3a}$$

$$W_k \leftarrow \left(I - \frac{y_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}}\right)^T W_k \left(I - \frac{y_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}}\right) + \frac{d_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}}. \tag{3.3b}$$

The following lemma reveals bounds on inner products with H_k and W_k after the updates for sample point i has been performed.

LEMMA 3.2 *Suppose that after updates have been performed for sample points* 1 *through* $i-1$, *we have* $H_k \succ 0$, $W_k \succ 0$, *and for any* $d \in \mathbb{R}^n$ *we have* $d^T H_k d \leq \theta \|d\|^2$ *and* $d^T W_k d \leq \beta \|d\|^2$ *for some* $\theta > 0$ *and* $\beta > 0$. *Then, after applying* (3.3), *we maintain* $H_k \succ 0$ *and* $W_k \succ 0$ *and have*

$$d^T H_k d \leq \left(\theta + \frac{\sigma}{\gamma}\right) \|d\|^2 \tag{3.4a}$$

$$\textit{and } \; d^T W_k d \leq \left(2\beta\left(1 + \frac{\sigma}{\gamma^2}\right) + \frac{1}{\gamma}\right) \|d\|^2. \tag{3.4b}$$

Proof That BFGS updates of the form (3.3) maintain positive-definiteness of H_k and W_k if $d_{k,i}^T y_{k,i} > 0$ is well known; e.g., see [28, Chapter 6]. Thus, as (3.2a) ensures $d_{k,i}^T y_{k,i} > 0$ and AGS has $\epsilon_k > 0$ for all k, it remains only to prove (3.4).

Since prior to the update we have $H_k \succ 0$ and (3.2a) with $\epsilon_k > 0$ ensures that $d_{k,i} \neq 0$, we have from (3.2) and (3.3a) that

$$\begin{aligned} d^T H_k d \leftarrow \; & d^T H_k d - \frac{(d_{k,i}^T H_k d)^2}{d_{k,i}^T H_k d_{k,i}} + \frac{(y_{k,i}^T d)^2}{d_{k,i}^T y_{k,i}} \\ \leq \; & d^T H_k d + \frac{(y_{k,i}^T d)^2}{d_{k,i}^T y_{k,i}} \\ \leq \; & \left(\theta + \frac{\sigma}{\gamma}\right) \|d\|^2. \end{aligned}$$

This is precisely (3.4a). As for (3.4b), first note that from (3.1) and (3.2a), we have

$$d^T \left(\frac{d_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}}\right) d = \frac{(d_{k,i}^T d)^2}{d_{k,i}^T y_{k,i}} \leq \frac{\epsilon_k^2 \|d\|^2}{\gamma \epsilon_k^2} = \frac{1}{\gamma} \|d\|^2. \tag{3.5}$$

Since prior to the update we have $W_k \succ 0$, we may write $W_k = N_k^T N_k$ for some nonsingular $N_k \in \mathbb{R}^{n\times n}$. From the statement of the lemma, we have that for any d

$$d^T W_k d = \|N_k d\|^2 \le \beta \|d\|^2,$$

which, along with (3.1) and (3.2), yields

$$\begin{aligned} \left\| \left(\frac{d_{k,i}^T d}{d_{k,i}^T y_{k,i}} \right) N_k y_{k,i} \right\|^2 &= \left| \frac{d_{k,i}^T d}{d_{k,i}^T y_{k,i}} \right|^2 \|N_k y_{k,i}\|^2 \\ &\le \left(\frac{\epsilon_k^2 \|d\|^2}{\gamma^2 \epsilon_k^4} \right) \beta \sigma \epsilon_k^2 \qquad (3.6) \\ &= \beta \frac{\sigma}{\gamma^2} \|d\|^2. \end{aligned}$$

Together, (3.5), (3.6), and the fact that for any vectors a and b of equal length we have $\|a-b\|^2 \le 2(\|a\|^2 + \|b\|^2)$ show that

$$\begin{aligned} \left\| N_k \left(I - \frac{y_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}} \right) d \right\|^2 &\le 2 \left(\|N_k d\|^2 + \left\| \left(\frac{d_{k,i}^T d}{d_{k,i}^T y_{k,i}} \right) N_k y_{k,i} \right\|^2 \right) \\ &\le 2\beta \left(1 + \frac{\sigma}{\gamma^2} \right) \|d\|^2. \end{aligned}$$

Overall, the above and (3.3b) yield

$$\begin{aligned} d^T W_k d \leftarrow & \left\| N_k \left(I - \frac{y_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}} \right) d \right\|^2 + d^T \left(\frac{d_{k,i} d_{k,i}^T}{d_{k,i}^T y_{k,i}} \right) d \\ \le & \left(2\beta \left(1 + \frac{\sigma}{\gamma^2} \right) + \frac{1}{\gamma} \right) \|d\|^2, \end{aligned}$$

which is precisely (3.4b). ■

We now have the following theorem revealing bounds for products with H_k.

THEOREM 3.3 *For any k, after all updates have been performed via* (3.3a) *for sample points* 1 *through* p_k*, the following holds for any $d \in \mathbb{R}^n$:*

$$\left(2^p \left(1 + \frac{\sigma}{\gamma^2} \right)^p \mu_k + \frac{1}{\gamma} \left(\frac{2^p \left(1 + \frac{\sigma}{\gamma^2} \right)^p - 1}{2 \left(1 + \frac{\sigma}{\gamma^2} \right) - 1} \right) \right)^{-1} \|d\|^2 \le d^T H_k d \le \left(\mu_k + \frac{p\sigma}{\gamma} \right) \|d\|^2. \qquad (3.7)$$

Proof Since H_k and W_k are initialized to the positive definite matrices $\mu_k I$ and $\mu_k^{-1} I$, respectively, we have that prior to the first update the bounds in Lemma 3.2 hold with $\theta = \mu_k$ and $\beta = \mu_k^{-1}$. Thus, the result of Lemma 3.2 can be applied repeatedly to produce upper bounds for $d^T H_k d$ and $d^T W_k d$ that hold for any $d \ne 0$ after all p_k updates have been performed. (Note that the bounds are trivially satisfied for $d = 0$.) Specifically, the upper bound for $d^T H_k d / \|d\|^2$ increases by the constant factor σ/γ for every update, implying that after all p_k updates have been performed, we have

$$d^T H_k d \leq \left(\mu_k + \frac{p_k \sigma}{\gamma}\right) \|d\|^2$$

Then, since $p \geq p_k$, we have the upper bound in (3.7).

As for the lower bound in (3.7), we begin by letting $\delta := \left(1 + \frac{\sigma}{\gamma^2}\right)$ and $\omega := \frac{1}{\gamma}$. Then, by applying the result in Lemma 3.2 repeatedly for $i = 1, \ldots, p_k$, we find that after all updates have been performed we have the following upper bound for inner products with W_k:

$$\begin{aligned} d^T W_k d \leq\ & (2^{p_k}\delta^{p_k}\mu_k + 2^{p_k-1}\delta^{p_k-1}\omega + \cdots + 2\delta\omega + \omega)\|d\|^2 \\ =\ & \left(2^{p_k}\delta^{p_k}\mu_k + \omega\left(\frac{2^{p_k}\delta^{p_k} - 1}{2\delta - 1}\right)\right)\|d\|^2. \end{aligned}$$

This implies via the Rayleigh-Ritz Theorem [19] that after all updates have been performed we have

$$d^T H_k d \geq \left(2^{p_k}\delta^{p_k}\mu_k + \omega\left(\frac{2^{p_k}\delta^{p_k} - 1}{2\delta - 1}\right)\right)^{-1}\|d\|^2.$$

As before, since $p \geq p_k$, this implies the lower bound in (3.7). ■

We note that the following corollary follows by applying the Rayleigh-Ritz Theorem to the result of Theorem 3.3.

Corollary 3.4 *For any k, after all updates have been performed via* (3.3b) *for sample points* 1 *through p_k, the following holds for any $d \in \mathbb{R}^n$:*

$$\left(\mu_k + \frac{p\sigma}{\gamma}\right)^{-1}\|d\|^2 \leq d^T W_k d \leq \left(2^p\left(1 + \frac{\sigma}{\gamma^2}\right)^p \mu_k + \frac{1}{\gamma}\left(\frac{2^p\left(1 + \frac{\sigma}{\gamma^2}\right)^p - 1}{2\left(1 + \frac{\sigma}{\gamma^2}\right) - 1}\right)\right)\|d\|^2. \tag{3.8}$$

3.2. *Updates to Promote Model Overestimation*

During iteration k, the primal subproblem (2.3) is equivalent to the following:

$$\min_d \ m_k(d) := f(x_k) + \max_{x \in X_k}\{\nabla f(x)^T d\} + \tfrac{1}{2} d^T H_k d.$$

If $m_k(d) \geq f(x_k + d)$ for all $d \in \mathbb{R}^n$, then a reduction in f is obtained after a step along $d_k \neq 0$ computed from (2.3)–(2.4). Thus, it is desirable to choose H_k so that m_k overestimates f to guarantee that such reductions occur in AGS.

It is not economical to ensure through the choice of H_k that m_k overestimates f for any given $d \in \mathbb{R}^n$. However, we can promote overestimation by evaluating $f(x_{k,i})$ at each sample point $x_{k,i} = x_k + d_{k,i}$ and performing a series of updates of H_k to increase, when appropriate, the value of $m_k(d_{k,i})$. Specifically, we set

$$H_k \leftarrow M_{k,p_k}^T \cdots M_{k,1}^T(\mu_k I)M_{k,1} \cdots M_{k,p_k} \tag{3.9}$$

where $M_{k,i}$ is chosen based on information obtained along $d_{k,i}$. (Note that such an H_k can be obtained by initializing $H_k \leftarrow \mu_k I$ and updating $H_k \leftarrow M_{k,i}^T H_k M_{k,i}$ for

$i = 1, \ldots, p_k$.) We choose $M_{k,i}$ in such a way that H_k remains well-conditioned and obtain bounds of the type required in Assumption 3.1 where $\overline{\xi} \geq \underline{\xi} > 0$ depend only on p and a user-defined constant $\rho \geq \frac{1}{2}$.

Suppose that updates have been performed for sample points 1 through $i-1$ and consider the update for sample point i. We consider $M_{k,i}$ of the form

$$M_{k,i} \leftarrow I + \frac{\rho_{k,i}}{d_{k,i}^T d_{k,i}} d_{k,i} d_{k,i}^T \succ 0 \tag{3.10}$$

where $d_{k,i} = x_{k,i} - x_k$ is the ith sample direction and the value for $\rho_{k,i}$ depends on the relationship between $f(x_{k,i})$ and the model value

$$m_k(d_{k,i}) = f(x_k) + \max_{x \in X_k} \{\nabla f(x)^T d_{k,i}\} + \tfrac{1}{2} d_{k,i}^T H_k d_{k,i}.$$

Specifically, if $m_k(d_{k,i}) \geq f(x_{k,i})$, then we choose $\rho_{k,i} \leftarrow 0$, which by (3.10) means that $M_{k,i} \leftarrow I$. Otherwise, we set

$$\rho_{k,i} \leftarrow -1 + \sqrt{\frac{2\Delta_{k,i}}{d_{k,i}^T H_k d_{k,i}}} \tag{3.11}$$

where

$$\Delta_{k,i} \leftarrow \min\left\{ f(x_{k,i}) - f(x_k) - \max_{x \in X_k}\{\nabla f(x)^T d_{k,i}\}, \rho d_{k,i}^T H_k d_{k,i} \right\}. \tag{3.12}$$

In this latter case when $m_k(d_{k,i}) < f(x_{k,i})$, it is easily verified from the definition of $m_k(d_{k,i})$ and (3.12) that $\Delta_{k,i} \geq \frac{1}{2} d_{k,i}^T H_k d_{k,i}$, implying that $\rho_{k,i} \geq 0$. Moreover, as $\Delta_{k,i} \leq \rho d_{k,i}^T H_k d_{k,i}$, it follows from (3.11) that $\rho_{k,i} \leq \sqrt{2\rho} - 1$. Thus, $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$.

The following lemma reveals useful bounds for inner products with $M_{k,i}$.

LEMMA 3.5 *Let $M_{k,i}$ be defined by* (3.10). *Then, for any $d \in \mathbb{R}^n$, we have*

$$\|d\|^2 \leq d^T M_{k,i}^T M_{k,i} d \leq (1 + \rho_{k,i})^2 \|d\|^2. \tag{3.13}$$

Proof The result is trivial for $d = 0$. Moreover, for d with $\|d\| \neq 0$, we find

$$d^T M_{k,i}^T M_{k,i} d = d^T d + \frac{2\rho_{k,i} + \rho_{k,i}^2}{d_{k,i}^T d_{k,i}} (d_{k,i}^T d)^2.$$

Thus, (3.13) follows by the extreme cases $d_{k,i}^T d = 0$ and $d = (\|d\|/\|d_{k,i}\|) d_{k,i}$. ■

We then have the following theorem revealing bounds for products with H_k.

THEOREM 3.6 *For any k, with H_k defined by* (3.9), *$M_{k,i}$ defined by* (3.10), *and $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$ for $i = 1, \ldots, p_k$, the following holds for any $d \in \mathbb{R}^n$:*

$$\mu_k \|d\|^2 \leq d^T H_k d \leq \mu_k (2\rho)^p \|d\|^2. \tag{3.14}$$

Proof First, we repeatedly apply the result of Lemma 3.5 to obtain

$$\begin{aligned} d^T H_k d = &\ d^T M_{k,p_k}^T \cdots M_{k,1}^T (\mu_k I) M_{k,1} \cdots M_{k,p_k} d \\ = &\ \mu_k (M_{k,2} \cdots M_{k,p_k} d)^T M_{k,1}^T M_{k,1} (M_{k,2} \cdots M_{k,p_k} d) \\ \le &\ \mu_k (1+\rho_{k,1})^2 (M_{k,2} \cdots M_{k,p_k} d)^T (M_{k,2} \cdots M_{k,p_k} d) \\ \cdots \le &\ \mu_k \left(\prod_{i=1}^{p_k} (1+\rho_{k,i})^2 \right) \|d\|^2. \end{aligned}$$

Since $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$ for $i = 1, \ldots, p_k$ and since AGS maintains $p_k \ge p$ for all k, we then find

$$\mu_k \left(\prod_{i=1}^{p_k} (1+\rho_{k,i})^2 \right) \|d\|^2 \le \mu_k (2\rho)^{p_k} \|d\|^2 \le \mu_k (2\rho)^p \|d\|^2.$$

The lower bound on $d^T H_k d$ in (3.14) is found in a similar fashion. ■

The approximation $W_k = H_k^{-1}$ for the inverse Hessian corresponding to (3.9) is

$$W_k \leftarrow M_{k,p_k}^{-T} \cdots M_{k,1}^{-T} (\mu_k^{-1} I) M_{k,1}^{-1} \cdots M_{k,p_k}^{-1} \tag{3.15}$$

where the Sherman-Morrison-Woodbury formula [13] reveals that for each $i = 1, \ldots, p_k$ we have

$$M_{k,i}^{-1} = I - \frac{\rho_{k,i}}{(1+\rho_{k,i}) d_{k,i}^T d_{k,i}} d_{k,i} d_{k,i}^T. \tag{3.16}$$

The following corollary follows by applying the Rayleigh-Ritz Theorem to the result of Theorem 3.6.

Corollary 3.7 *For any k, with W_k defined by* (3.15)*, $M_{k,i}^{-1}$ defined by* (3.16)*, and $\rho_{k,i} \in [0, \sqrt{2\rho} - 1]$ for $i = 1, \ldots, p_k$, the following holds for any $d \in \mathbb{R}^n$:*

$$\mu_k^{-1} (2\rho)^{-p} \|d\|^2 \le d^T W_k d \le \mu_k^{-1} \|d\|^2. \tag{3.17}$$

We conclude this subsection by showing that the updating strategy described here is intended solely for nonconvex problems. That is, if f is convex, then the updates will maintain $H_k = \mu_k I$ and $W_k = \mu_k^{-1} I$.

Theorem 3.8 *Suppose f is convex. Then, for any k, the matrices H_k and W_k described in Theorems 3.6 and 3.7, respectively, satisfy $H_k = \mu_k I$ and $W_k = \mu_k^{-1} I$.*

Proof It suffices to show that for any k and i we have $m_k(d_{k,i}) \ge f(x_{k,i})$, meaning that the method will always choose $\rho_{k,i} \leftarrow 0$ and $M_{k,i} \leftarrow I$ and so H_k and W_k will never be altered from their initial values. Since f is convex, we have

$$\begin{aligned} m_k(d_{k,i}) = &\ f(x_k) + \max_{x \in X_k} \{\nabla f(x)^T d_{k,i}\} + \tfrac{1}{2} d_{k,i}^T H_k d_{k,i} \\ \ge &\ f(x_k) + \nabla f(x_{k,i})^T d_{k,i} + \tfrac{1}{2} d_{k,i}^T H_k d_{k,i} \\ \ge &\ f(x_{k,i}) + \tfrac{1}{2} d_{k,i}^T H_k d_{k,i}. \end{aligned}$$

Thus, since $H_k = \mu_k I \succ 0$, it follows that $m_k(d_{k,i}) \ge f(x_{k,i})$ for all k and i. ■

4. Global Convergence Analysis

We make the following assumption about the objective function f of (2.1) throughout our global convergence analysis.

Assumption 4.1 *The objective function $f : \mathbb{R}^n \to \mathbb{R}$ is locally Lipschitz and continuously differentiable on an open dense subset $\mathcal{D} \subset \mathbb{R}^n$.*

We also make Assumption 3.1 stated previously at the beginning of §3.

The result we aim to prove is the following.

Theorem 4.2 *Let the optimality tolerance parameter be $\nu = 0$. Then, AGS produces an infinite sequence of iterates $\{x_k\}$ and, with probability one, either $f(x_k) \to -\infty$ or $\{\epsilon_k\} \to 0$ and every cluster point of $\{x_k\}$ is stationary for f.*

Our analysis follows closely that of Kiwiel in [22]. However, there are subtle differences due to the adaptive sampling procedure and the variable-metric Hessian approximations. Thus, we analyze the global convergence behavior of AGS for the sake of completeness.

We begin our analysis for proving Theorem 4.2 by showing that AGS is well-posed in the sense that each iteration terminates finitely. It is clear that this will be true as long as the backtracking line search in step 6 terminates finitely.

Lemma 4.3 *If $|X_k| \geq p$ in step 6, then $\alpha_k > 0$ is computed satisfying* (2.5).

Proof If $|X_k| \geq p$, then since step 2 implies $\nabla f(x_k) \in G_k$, it follows from the constraints of subproblem (2.3) that

$$\nabla f(x_k)^T d_k \leq \max_{g \in G_k} g^T d_k \leq z_k - f(x_k).$$

Moreover, since $(z, d) = (f(x_k), 0)$ is a feasible solution for (2.3) yielding an objective value of $f(x_k)$, it follows that $z_k + \frac{1}{2} d_k^T H_k d_k \leq f(x_k)$, which, along with the above, means

$$\nabla f(x_k)^T d_k \leq -\tfrac{1}{2} d_k^T H_k d_k. \tag{4.1}$$

Since by step 5 we must have $d_k^T H_k d_k > 0$ in step 6, it follows that d_k is a direction of strict descent for f at x_k, so there exists $\alpha_k > 0$ such that (2.5) holds. ■

Our next result builds on the previous one and shows that there will be an infinite number of iterations during which the algorithm produces a nonzero steplength.

Lemma 4.4 *There exists an infinite subsequence of iterations in which $\alpha_k > 0$.*

Proof By step (5), if $d_k^T H_k d_k \leq \epsilon_k$ an infinite number of times, then the result follows as the algorithm sets $\alpha_k \leftarrow 1$ for such iterations. Otherwise, to derive a contradiction, suppose there exists $k' \geq 0$ such that for $k \geq k'$, step 6 is reached and sets $\alpha_k \leftarrow 0$. By Lemma 4.3, this means that for $k \geq k'$, we have $p_k < p$. However, by steps 7, 8, and then 2, it is clear that if $\alpha_k \leftarrow 0$, then $p_{k+1} = \min\{p, p_k + \overline{p}\}$, contradicting the conclusion that $\{p_k\}$ is bounded above by $p - 1$. ■

We now show a critical result about the sequence of decreases produced in f. A similar result was proved in [22].

Lemma 4.5 *The following inequality holds for all k:*

$$f(x_{k+1}) \leq f(x_k) - \tfrac{1}{4} \eta \underline{\xi} \|x_{k+1} - x_k\| \|d_k\|.$$

Proof By the triangle inequality, condition (2.6b) ensures that

$$\|x_{k+1} - x_k\| \leq \min\{\alpha_k, \epsilon_k\}\|d_k\| + \alpha_k\|d_k\| \leq 2\alpha_k\|d_k\|. \tag{4.2}$$

Indeed, this inequality holds trivially if the algorithm sets $x_{k+1} \leftarrow x_k$ in step 5 or sets $\alpha_k \leftarrow 0$ in step 6, and holds by the triangle inequality if step 6 yields $x_{k+1} \leftarrow x_k + \alpha_k d_k$. Thus, by (2.5), (2.6), and (4.2), we find that for all k,

$$\begin{aligned} f(x_{k+1}) - f(x_k) &\leq -\eta\alpha_k d_k^T H_k d_k \\ &\leq -\eta\alpha_k(\tfrac{1}{2}\underline{\xi}\|d_k\|^2) \\ &\leq -\tfrac{1}{4}\eta\underline{\xi}\|x_{k+1} - x_k\|\|d_k\|, \end{aligned}$$

as desired. ■

We now consider the ability of the algorithm to approximate the set $\mathbb{G}_{\epsilon_k}(x')$ when x_k is close to a given point x'. For this purpose, consider the following subproblem:

$$\inf_d \; q(d; x', \mathbb{G}_{\epsilon_k}(x'), H_k) := f(x') + \sup_{x \in \mathbb{G}_{\epsilon_k}(x')} \{\nabla f(x)^T d\} + \tfrac{1}{2} d_k^T H_k d_k. \tag{4.3}$$

Given a solution d' of (4.3), we have the following reduction in its objective:

$$\Delta q(d'; x', \mathbb{G}_{\epsilon_k}(x'), H_k) := q(0; x', \mathbb{G}_{\epsilon_k}(x'), H_k) - q(d'; x', \mathbb{G}_{\epsilon_k}(x'), H_k) \geq 0.$$

Similarly, writing (2.3) in the form

$$\min_d \; q(d; x_k, G_k, H_k),$$

we have the following reduction produced by the search direction d_k:

$$\Delta q(d_k; x_k, G_k, H_k) = q(0; x_k, G_k, H_k) - q(d_k; x_k, G_k, H_k) \geq 0.$$

The purpose of our next lemma is to show that for any desired level of accuracy (though not necessarily perfect accuracy), as long as x_k is sufficiently close to x', there exists a sample set such that the reduction $\Delta q(d_k; x_k, G_k, H_k)$ produced by the solution d_k of (2.3) will be sufficiently close to the reduction $\Delta q(d'; x', \mathbb{G}_{\epsilon_k}(x'), H_k)$ produced by the solution d' of (4.3). For a given x' and tolerance ω, we define

$$\mathcal{T}_k(x', \omega) := \left\{ X_k \in \prod_1^{p_k} B_k : \Delta q(d_k; x_k, G_k, H_k) \leq \Delta q(d'; x', \mathbb{G}_{\epsilon_k}(x'), H_k) + \omega \right\}.$$

This set plays a critical role in the following lemma. A similar result was proved in the context of constrained optimization in [8].

Lemma 4.6 *If $p_k \geq p$, then for any $\omega > 0$, there exists $\zeta > 0$ and a nonempty set $\mathcal{T}$ such that for all $x_k \in \mathbb{B}_\zeta(x')$ we have $\mathcal{T} \subset \mathcal{T}_k(x', \omega)$.*

Proof Under Assumption 4.1 and since $\omega > 0$, there exists a vector d satisfying

$$\Delta q(d; x', \mathbb{G}_{\epsilon_k}(x'), H_k) < \Delta q(d'; x', \mathbb{G}_{\epsilon_k}(x'), H_k) + \omega$$

such that for some $g \in \operatorname{conv} \nabla f(\mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D})$ we have

$$q(d; x', \mathbb{G}_{\epsilon_k}(x'), H_k) = f(x') + g^T d + \tfrac{1}{2} d^T H_k d.$$

Then, since $p_k \geq p \geq n+1$, Carathéodory's theorem [29] implies that there exists $y_1, \dots, y_{p_k} \subset \mathbb{B}_{\epsilon_k}(x') \cap \mathcal{D}$ and a nonnegative set of scalars $\lambda_1, \dots, \lambda_{p_k}$ such that

$$\sum_{i=1}^{p_k} \lambda_i = 1 \qquad \text{and} \qquad \sum_{i=1}^{p_k} \lambda_i \nabla f(y_i) = g'.$$

Since f is continuously differentiable in $\mathcal{D}$, there exists $\zeta \in (0, \epsilon_k)$ such that the set

$$\mathcal{T} := \prod_{i=1}^{p_k} \operatorname{int} \mathbb{B}_\zeta(y_i)$$

lies in $\mathbb{B}_{\epsilon_k - \zeta}(x')$ and the solution d_k to (2.3) with $G_k \in \mathcal{T}$ satisfies

$$\Delta q(d_k; x_k, G_k, H_k) \leq \Delta q(d'; x', \mathbb{G}_{\epsilon_k}(x'), H_k) + \omega.$$

Thus, for all $x_k \in \mathbb{B}_\zeta(x')$, $\mathbb{B}_{\epsilon_k - \zeta}(x') \subset \mathbb{B}_{\epsilon_k}(x_k)$ and hence $\mathcal{T} \subset \mathcal{T}_k(x', \omega)$. ■

We are now prepared to prove Theorem 4.2. Our proof follows closely that of [22, Theorem 3.3]. We provide a proof for the sake of completeness and since subtle changes to the proof are required due to our adaptive sampling strategy.

Proof If $f(x_k) \to -\infty$, there is nothing to prove, so suppose that

$$\inf_{k \to \infty} f(x_k) > -\infty.$$

Then, we have from (2.5), (2.6), and Lemma 4.5 that

$$\sum_{k=0}^{\infty} \alpha_k d_k^T H_k d_k < \infty, \quad \text{and} \tag{4.4a}$$

$$\sum_{k=0}^{\infty} \|x_{k+1} - x_k\| \|d_k\| < \infty. \tag{4.4b}$$

We continue by considering two cases, the first of which has two subcases.

Case 1: Suppose that there exists $k' \geq 0$ such that $\epsilon_k = \epsilon' > 0$ for all $k \geq k'$. According to step 5, this occurs only if

$$d_k^T H_k d_k > \epsilon' \text{ for all } k \geq k'. \tag{4.5}$$

In conjunction with (4.4), this implies $\alpha_k \to 0$ and $x_k \to x'$ for some x'. Note also that, applying a similar argument as in the proof of Lemma 4.4, it follows that in this case there exists an infinite subsequence of iterations $\mathcal{K}$ in which $p_k \geq p$.

Case 1a: If x' is ϵ'-stationary for f, then for all $k \geq k'$, the solution d' to (4.3) satisfies $\Delta q(d'; x', \mathbb{G}_{\epsilon'}(x'), H_k) = 0$. Thus, with $\omega = (\epsilon')^2/2$ and $(\zeta, \mathcal{T})$ chosen as in Lemma 4.6, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_\zeta(x')$ for all $k \geq k''$ and

$$\Delta q(d_k; x_k, G_k, H_k) \leq (\epsilon')^2/2 \tag{4.6}$$

whenever $k \geq k''$, $k \in \mathcal{K}$, and $X_k \in \mathcal{T}$. Together, (4.5) and (4.6) imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}$. However, this is a probability zero event since for all such k the set X_k continually collects points each generated uniformly from B_k, meaning that it will eventually include an element of the set $\mathcal{T}$ yielding (4.6).

Case 1b: If x' is not ϵ'-stationary, then for all $k \geq k'$, any α not satisfying the sufficient decrease condition (2.5) yields

$$f(x_k + \alpha d_k) - f(x_k) > -\eta \alpha d_k^T H_k d_k,$$

and along with (4.1) yields

$$f(x_k + \alpha d_k) - f(x_k) \leq -\eta \alpha d_k^T H_k d_k + \alpha^2 L_k \|d_k\|^2.$$

Here, L_k is a finite upper bound for $(f(x_k + \alpha d_k) - f(x_k))/\|d_k\|$ on the interval $[x_k, x_k + \alpha d_k]$ whose existence follows from Assumption 4.1. Combining the above inequalities yields a lower bound on any α not satisfying (2.5), which, since step 6 invokes the backtracking factor κ, yields the bound

$$\alpha_k > \kappa(1-\eta) d_k^T H_k d_k / (L_k \|d_k\|^2).$$

However, with $\omega = \Delta q(d'; x', \mathbb{G}_{\epsilon'}(x'), H_k)$ (which is strictly positive since x' is not ϵ'-stationary) and $(\zeta, \mathcal{T})$ again chosen as in Lemma 4.6, there exists $k'' \geq k'$ such that $x_k \in \mathbb{B}_\zeta(x')$ for all $k \geq k''$ and

$$\Delta q(d_k; x_k, G_k, H_k) \leq 2 \Delta q(d'; x', \mathbb{G}_{\epsilon'}(x'), H_k)$$

whenever $k \geq k''$, $k \in \mathcal{K}$, and $X_k \in \mathcal{T}$. Under Assumptions 4.1 and 3.1 and since $x_k \to x'$, we have that for all k sufficiently large, $L_k \|d_k\|^2 \leq L$ for some constant $L > 0$, implying that for all $k \geq k''$ with $k \in \mathcal{K}$ such that $X_k \in \mathcal{T}$, α_k is bounded away from zero. Together, this and the fact that $\alpha_k \to 0$ imply that $X_k \notin \mathcal{T}$ for all $k \geq k''$ with $k \in \mathcal{K}$. Again, this is a probability zero event.

Case 2: Suppose $\{\epsilon_k\} \to 0$ and $\{x_k\}$ has a cluster point x'. First, we show that

$$\liminf_{k \to \infty} \max\{\|x_k - x'\|, \|d_k\|\} = 0. \tag{4.7}$$

If $x_k \to x'$, then by construction in the algorithm, $\{\epsilon_k\} \to 0$ if and only if there exists an infinite subsequence $\mathcal{K}'$ of iterations where

$$\tfrac{1}{2}\underline{\xi}\|d_k\|^2 \leq \tfrac{1}{2} d_k^T H_k d_k \leq \epsilon_k^2.$$

Thus, since $\{\epsilon_k\} \to 0$, we have

$$\lim_{k \in \mathcal{K}'} \|d_k\| = 0,$$

yielding (4.7). On the other hand, if $x_k \nrightarrow x'$, then we proceed by contradiction and suppose that (4.7) does not hold. Since x' is a cluster point of $\{x_k\}$, there is an $\epsilon' > 0$ and an index $k' \geq 0$ such that the set $K' := \{k : k \geq k', \|x_k - x'\| \leq \epsilon', \|d_k\| > \epsilon'\}$ is infinite. By (4.4b), this means

$$\sum_{k \in K'} \|x_{k+1} - x_k\| < \infty. \tag{4.8}$$

Since $x_k \nrightarrow x'$, there exists an $\epsilon > 0$ such that for all $k_1 \in K'$ with $\|x_{k_1} - x'\| \leq \epsilon'/2$ there is $k_2 > k_1$ satisfying $\|x_{k_1} - x_{k_2}\| > \epsilon$ and $\|x_k - x'\| \leq \epsilon'$ for all $k_1 \leq k \leq k_2$. Thus, by the triangle inequality, we have $\epsilon < \|x_{k_1} - x_{k_2}\| \leq \sum_{k=k_1}^{k_2-1} \|x_{k+1} - x_k\|$. However, for $k_1 \in K'$ sufficiently large, (4.8) implies that the right-hand side of this inequality must be strictly less than ϵ, a contradiction.

Finally, since for all k the elements of X_k lie in B_k, equation (4.7) and $\{\epsilon_k\} \to 0$ imply that the cluster point x' is stationary for f. ■

5. An Implementation

We have implemented Algorithm 2.1 in Matlab along with the QO subproblem solver described in the Appendix. In this section, we describe the algorithm variations that we have tested, the test problems that we have solved, and the results of our numerical experiments. All tests were performed on a machine running Debian 2.6.32 with two 8-Core AMD Opteron 6128 2.0GHz processors and 32GB RAM.

5.1. *Algorithm Variations*

Given varying values for the input parameters, our implementation of Algorithm 2.1 yields the algorithm variations described below.

- `GS`. This is a basic gradient sampling algorithm with nonnormalized search directions [22, §4.1], obtained by choosing $\overline{p} = p \geq n+1$ with $H_k = W_k = I$ for all k. We consider this variant of GS for comparison purposes as it is the most similar with the AGS variations described below. The global convergence analysis in §4 applies for this algorithm as long as Assumption 4.1 holds.
- `AGS`. This algorithm samples gradients adaptively as we choose $\overline{p} < p$, but it does not use either Hessian updating strategy as we choose $H_k = W_k = I$ for all k. The global convergence analysis in §4 applies for this algorithm as long as Assumption 4.1 holds.
- `AGS-LBFGS`. This algorithm is an enhanced version of `AGS` where the Hessian updating strategy in §3.1 is used to set H_k and W_k for all k. We choose $\gamma = 0.1$ and $\sigma = 100$ in the Hessian updates, which by Theorem 3.3 means that the global convergence analysis in §4 applies as long as Assumption 4.1 holds.
- `AGS-LBFGS-ill`. This algorithm is similar to `AGS-LBFGS`, except that we choose $\gamma = 0$ and $\sigma = \infty$ so that H_k and W_k may become ill-conditioned. The global convergence analysis in §4 does *not* apply for this method.
- `AGS-over`. This algorithm is an enhanced version of `AGS` where the Hessian updating strategy in §3.2 is used to set H_k and W_k for all k. We choose $\rho = 100$ in the Hessian updates, which by Theorem 3.6 means that the global convergence analysis in §4 applies as long as Assumption 4.1 holds.
- `AGS-over-ill`. This algorithm is similar to `AGS-over`, except that we choose $\rho = \infty$ so that H_k and W_k may become ill-conditioned. The global convergence analysis in §4 does *not* apply for this method.

We summarize the differing inputs for these six algorithm variations in Table 5.1.

Specific values for the input parameters mentioned above, as well as for the remaining parameters that were set consistently for all algorithm variations, were chosen as those that yielded the best overall results in our experiments. As recommended in [5, 22], we choose $p = 2n$ as the number of sample points required for a complete line search. (Note that this is also the number of sample points and sample gradients computed per iteration for `GS`.) For `AGS` and the remaining vari-

Name	Samples per Iteration	Hessian updates	γ	σ	ρ
`GS`	$\overline{p} = p \geq n+1$	None	-	-	-
`AGS`	$\overline{p} < p \geq n+1$	None	-	-	-
`AGS-LBFGS`	$\overline{p} < p \geq n+1$	Strategy in §3.1	0.1	100	-
`AGS-LBFGS-ill`	$\overline{p} < p \geq n+1$	Strategy in §3.1	0	∞	-
`AGS-over`	$\overline{p} < p \geq n+1$	Strategy in §3.2	-	-	100
`AGS-over-ill`	$\overline{p} < p \geq n+1$	Strategy in §3.2	-	-	∞

Table 5.1. Summary of six algorithm variations used to test the adaptive sampling procedure in Algorithm 2.1 along with the Hessian updating strategies described in §3.1 and §3.2.

ants, we experimented with various values for $\overline{p}$, eventually finding that $\overline{p} = n/10$ yielded nice results. Our convergence analysis in §4 requires only $O(1)$ gradients per iteration, but we suggest that setting $\overline{p}$ as a fraction of n will generally yield a good balance between overall gradient evaluations and iterations; see §5.3 below. We set the line search backtracking constant to be $\kappa = 0.5$, sufficient decrease constant to be $\eta = 10^{-8}$, and number of backtracks in an incomplete line search to be $q = 7$. The initial sampling radius is chosen to be $\epsilon_0 = 0.1$ and $\psi = 0.5$ is set as the sampling radius reduction factor. We limit the number of iterations to $10,000$ and set the stationarity tolerance to 10^{-4}.

The Hessian and inverse Hessian approximations are initialized during iteration k as $H_k = \mu_k I$ and $W_k = \mu_k^{-1} I$, respectively. The scalar value μ_k itself is initialized at the start of a run of the algorithm as $\mu_0 = 1$ and is updated dynamically at the end of each iteration k based on the steplength α_k. Specifically, we set

$$\mu_{k+1} \leftarrow \begin{cases} \min\{2\mu_k, \overline{\mu}\} & \text{if } \alpha_k < 1 \\ \max\{\frac{1}{2}\mu_k, \underline{\mu}\} & \text{if } \alpha_k = 1 \end{cases}$$

where we choose $\underline{\mu} = 10^{-2}$ and $\overline{\mu} = 10^3$. This strategy increases the eigenvalues of the initialized Hessian if, during the current iteration, the line search had to backtrack from $\alpha_k = 1$, thus promoting a shorter search direction in the next iteration. Similarly, if the current iteration yielded $\alpha_k = 1$, then the eigenvalues of the initialized Hessian are decreased to promote longer search directions.

We implemented Algorithm 2.1 along with the QO subproblem solver described in the Appendix. We set the subproblem optimality tolerance to 10^{-8} and maximum number of iterations to $\min\{1000, 2^{\max\{n,p_k\}}\}$.

5.2. *Test Problems*

We tested the algorithm variations with 26 nonsmooth minimizaton problems, some convex and some nonconvex. The first 20 of these problems were considered in [15] and the last 6 were considered in [31]. All problems are scalable in the sense that they can be defined to have different numbers of variables n. The first 10 problems, introduced in [16], are all nonsmooth at their respective minimizers: `MAXQ`, `MXHILB`, `CHAINED_LQ`, `CHAINED_CB3_I`, `CHAINED_CB3_II`, `ACTIVE_FACES`, `BROWN_FUNCTION_2`, `CHAINED_MIFFLIN_2`, `CHAINED_CRESCENT_I`, and `CHAINED_CRESCENT_II`. The first 5 of these problems are convex and the second 5 are nonconvex. The second 10 problems in our set, some of which are nonconvex, were introduced in the test library TEST29 [26]: `TEST29_2`, `TEST29_5`, `TEST29_6`, `TEST29_11`, `TEST29_13`, `TEST29_17`, `TEST29_19`, `TEST29_20`, `TEST29_22`, and `TEST29_24`. Of the 6 remaining problems, the first four were introduced in [25], the fifth was introduced in [14], and the sixth is a problem to minimize the Schatten norm [31]: `TILTED_NORM_COND`, `CPSF`, `NCPSF`, `EIG_PROD`, `GREIF_FUN`, and `NUC_NORM`.

5.3. *Numerical Results*

We chose $n = 50$ for all problems. The only exception was `EIG_PROD`, for which we choose $n = 64$, as the variables for this problem need to compose a square matrix. We ran each problem 10 times, the first time using a fixed initial point x_0' and the remaining nine times using a starting point generated randomly from a ball about x_0' with radius $\|x_0'\|$. (We choose $x_0' \neq 0$ for all problems, so the initial points for each run were unique.) For the first 20 problems, we choose x_0' as the initial point defined in [15]. For the remaining 6 problems, we choose $x_0' = e$. The input parameters we use for `TILTED_NORM_COND`, `CPSF`, `NCPSF`, and `NUC_NORM` are those used in [31]. The only remaining problem inputs that require specification are the matrices involved in `EIG_PROD` and `GREIF_FUN`. For the former we used the leading 8×8 submatrix of A from [1] and for the latter we used a randomly generated 10×10 symmetric positive definite matrix A (with the $n = 50$ variables composing a 10×5 matrix X so that the product $X^T AX$ is well defined).

First, we compare the results obtained by applying the algorithms `GS` and `AGS` to the $26 \times 10 = 260$ test problems. Performance profiles [9] for nonlinear iterations, function evaluations, gradient evaluations, and overall QO iterations are given in Figure 5.1 comparing these two solvers. The profiles illustrate clearly the tradeoffs between the two algorithm variations. `GS`, by virtue of computing more local information of f during each iteration, consistently requires fewer nonlinear iterations to obtain the required accuracy. This comes at the cost, however, of significantly more function and gradient evaluations for the overall run of the algorithm. That is, despite lacking in a comparable amount of local information of f during each iteration, `AGS` benefits by cheaper iterations and still solves each problem quickly enough so that, overall, the number of function and gradient evaluations are fewer than those for `GS`. Finally, in terms of overall QO iterations, `GS` has an advantage, but that advantage is relatively small compared to the differences seen in the other performance measures.

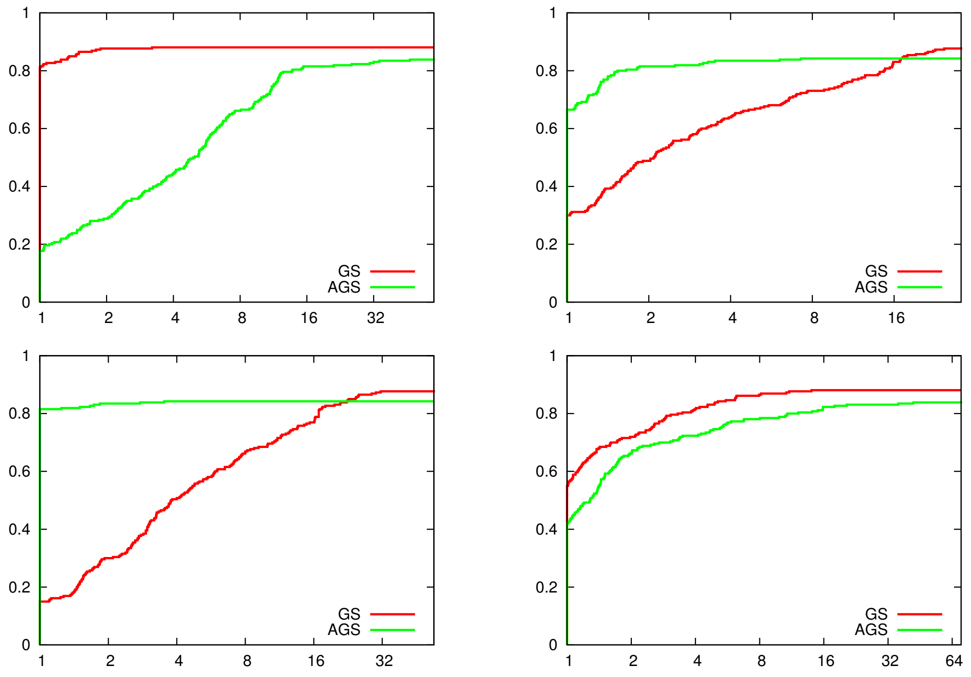

Figure 5.1. Performance profiles for nonlinear iterations (upper left), function evaluations (upper right), gradient evaluations (lower left), and overall QO iterations (lower right) comparing algorithms `GS` and `AGS`.

In summary, the choice between `GS` and `AGS` for a particular application depends on which is the source of the dominant computational costs: the calculations within the algorithm itself or that within the function/gradient evaluations. If function/gradient evaluations are expensive compared to the computational cost of the nonlinear and QO iterations (involving primarily matrix-vector and vector-vector operations with $O(n)$-dimensional quantities), then `AGS` appears to be the algorithm of choice. Otherwise, if function/gradient evaluations are cheap enough, then one should choose `GS` over `AGS` as the former clearly benefits by obtaining more local information of f during each iteration.

One additional remark to make about the performance profiles in Figure 5.1 are that the profiles for nonlinear and overall QO iterations illustrate the benefits obtained by warm-starting the QO solver in `AGS`. In particular, while `AGS` generally requires many more nonlinear iterations than `GS`, it does not require many more overall QO iterations. In order for this to occur, the number of QO iterations per nonlinear iteration must be significantly fewer for `AGS` as compared to `GS`. This can be attributed to the fact that the subproblems in `AGS` are often smaller than those in `GS`, and when they are the same size as in `GS`, warm-starting the solver reduces the number of QO iterations required.

Our second set of performance profiles illustrate the benefits of the Hessian updating strategies in §3 by comparing the results for `AGS-over` and `AGS-LBFGS` with those for `AGS`. This time, the profiles clearly indicate that the Hessian updates provide benefits in terms of all performance measures. These improvements can be attributed to both the updating strategy for μ_k and the two strategies for manipulating H_k and W_k. If anything, there appears to be a slight advantage in terms of overall QO iterations in using `AGS-over`, but this benefit is slight and may be dependent on the choice of test set and initial points.

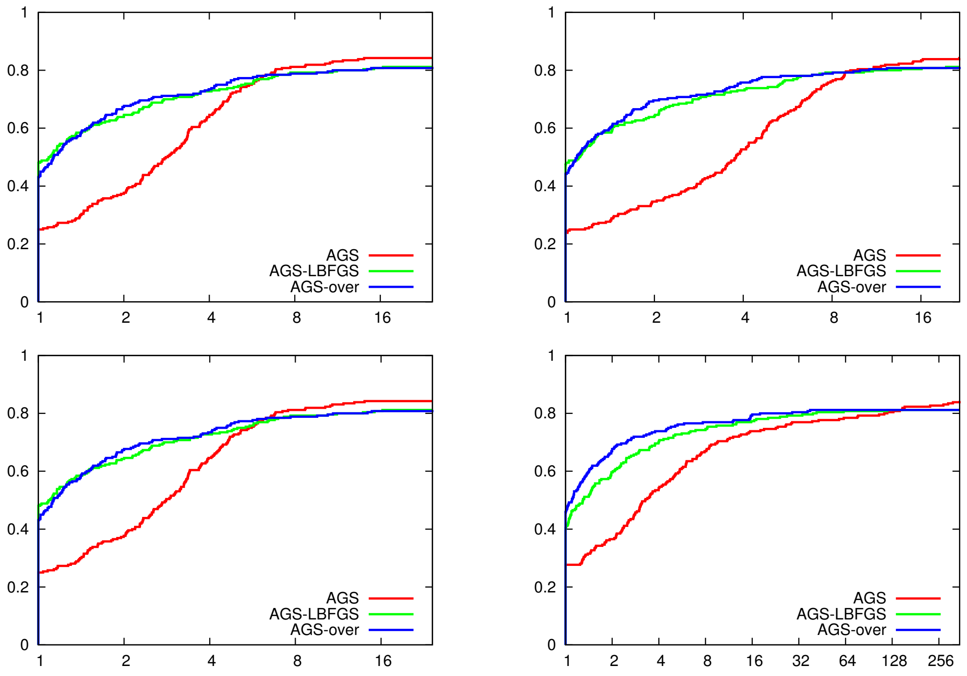

Figure 5.2. Performance profiles for nonlinear iterations (upper left), function evaluations (upper right), gradient evaluations (lower left), and overall QO iterations (lower right) comparing algorithms `AGS`, `AGS-LBFGS`, and `AGS-over`.

We close this section with performance profiles comparing `AGS-over` and `AGS-LBFGS` with `AGS-over-ill` and `AGS-LBFGS-ill`, the latter two being variants for which our global convergence analysis in §4 does not apply. In contrast to the commonly-held belief that algorithms for nonsmooth optimization benefit by having Hessian matrices that tend to singularity during the solution process, we do not see much of an impact in our numerical results. (In fact, there appears to be a disadvantage of allowing ill-conditioning, at least in terms of `AGS-LBFGS-ill`.) There are at least a couple possible explanations for this phenomenon. First, due to the fact that we reinitialize H_k and W_k during each iteration and perform only a finite number of updates based on sample point information, our Hessian approximations may naturally remain better conditioned than those obtained by standard quasi-Newton updating techniques that continually build these matrices based on gradient information obtained at all previous algorithm iterates. Second, our input parameter choices may be generous enough that, e.g., `AGS-LBFGS` and `AGS-LBFGS-ill` produce similar Hessian approximations during most iterations.

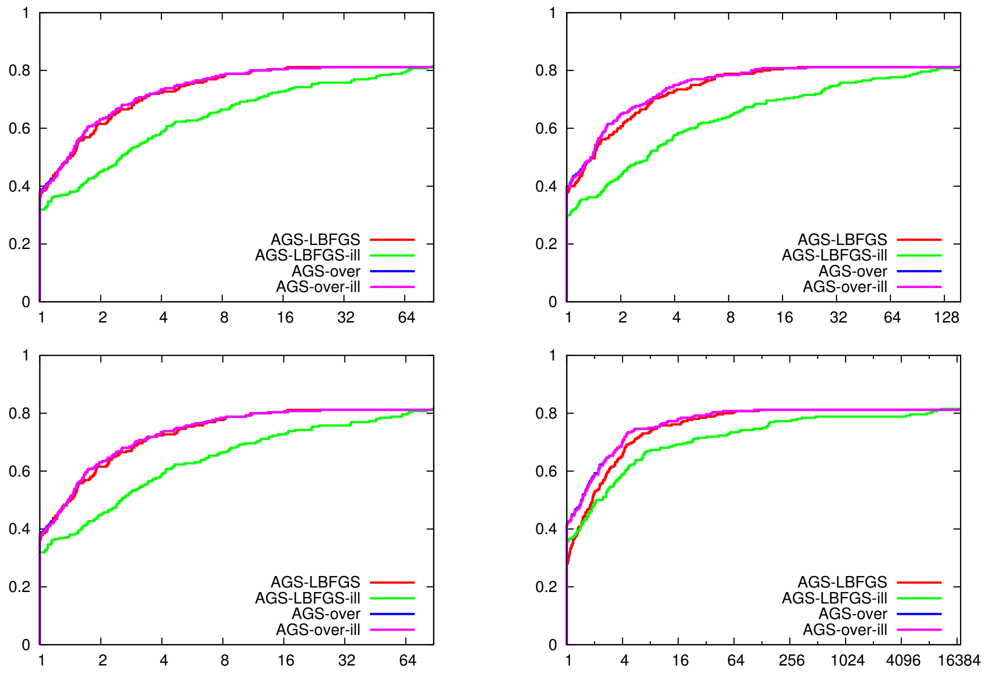

Figure 5.3. Performance profiles for nonlinear iterations (upper left), function evaluations (upper right), gradient evaluations (lower left), and overall QO iterations (lower right) comparing algorithms `AGS-LBFGS`, `AGS-LBFGS-ill`, `AGS-over` and `AGS-over-ill`.

6. Conclusion

In this paper, we have addressed major practical limitations of the rapidly-developing class of gradient sampling (GS) algorithms for nonsmooth optimization. Our proposed enhancements to GS that attempt to correct for these limitations take the form of an adaptive sampling procedure and variable-metric Hessian updating strategies. We have shown that our enhanced framework, AGS, maintains the global convergence guarantees of GS while providing many practical advantages, especially in terms of significant reductions in overall function/gradient evaluations. These advantages have been illustrated via numerical experiments on a diverse set of test problems without requiring tailored inputs for each test problem.

In addition to representing an enhanced version of GS, we believe that the development of AGS represents a step toward merging the algorithmic frameworks of gradient sampling and bundle methods. Indeed, by incorporating information obtained during previous iterations, the subproblems formed and solved in AGS closely resemble those typically found in bundle methods (after a "descent" or "serious" step has been made). We intend to investigate the marriage of gradient sampling and bundle method strategies in our future work.

Appendix: QO Subproblem Solver

In this appendix, we discuss a specialized technique for solving the primal-dual pair (2.3) and (2.4) during step 4 of AGS. Specifically, we present an approach for solving (2.4) that follows the technique described in [21] for solving a similar dual problem. The differences are that, in our subproblem, there is no linear term in the objective, and we allow for the use of general positive definite Hessian approximations. We drop iteration number subscripts in this section. Now, subscripts are used only to indicate column number of a matrix or element number in a vector.

The benefits of our QO solver are that it can produce more accurate solutions than, say, an interior-point method, and we can easily warm-start the approach to take advantage of the fact that the columns of G often do not change drastically between iterations of AGS. The algorithm also carefully handles ill-conditioning in G, which is extremely important in our context as the columns of G come from the calculation of gradients of f at points that may be very close to one another.

Necessary and sufficient conditions for both subproblems (2.3) and (2.4) are

$$g_j^T W G\pi - (\pi^T G^T W G\pi) \geq\ 0,\ j = 1, \ldots, q \tag{6.1a}$$

$$e^T\pi = 1,\ \pi \geq\ 0. \tag{6.1b}$$

A vector π satisfying these conditions is the unique optimal solution to (2.4), with which the unique optimal solution d to (2.3) can be computed as $d \leftarrow -WG\pi$.

It is clear from (6.1b) that any solution π to (6.1) must have at least one positive entry. Thus, the method commences with a nonempty estimate

$$\mathcal{A} \subseteq \{1, \ldots, q\}$$

of the optimal *active set*, i.e., an estimate of the indices corresponding to positive values of π in the solution to (6.1). Denoting $\hat{G}$ and $\hat{\pi}$, respectively, as the ordered submatrix of G and the ordered subvector of π corresponding to the indices in the active set estimate $\mathcal{A}$, we begin with $\hat{\pi}$ satisfying

$$e^T\hat{\pi} = 1,\ \hat{\pi} \geq 0.$$

Assuming always that the remaining elements in the corresponding π are set to zero, this solution is either optimal or, for some $j \notin \mathcal{A}$, we have

$$g_j^T W\hat{G}\hat{\pi} - (\hat{\pi}^T\hat{G}^T W\hat{G}\hat{\pi}) < 0. \tag{6.2}$$

An improvement in the objective of (2.4) can then be obtained by including j in $\mathcal{A}$. If the direct inclusion of j in $\mathcal{A}$ yields a new $\hat{G}$ such that $\begin{bmatrix} e\ \hat{G}^T \end{bmatrix}$ has full row rank, then $\mathcal{A}$ is simply augmented to include j. (Determining whether or not this matrix has full row rank can be done by solving the least-squares system

$$(\hat{G}^T W\hat{G} + ee^T)\widetilde{\pi} = e + \hat{G}^T W g_j \tag{6.3}$$

and then determining whether $e^T\widetilde{\pi} = 1$ and $\hat{G}\widetilde{\pi} = g_j$.) Otherwise, j is swapped with an appropriate element in $\mathcal{A}$ to avoid rank-deficiency. In either case, a new trial solution $\overline{\pi}$ is obtained by solving the linear system

$$\begin{bmatrix} \hat{G}^T W \hat{G} & e \\ e^T & 0 \end{bmatrix} \begin{bmatrix} \overline{\pi} \\ v \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \tag{6.4}$$

which yields the solution to the equality constrained subproblem

$$\begin{aligned} &\max_{\overline{\pi}} \tfrac{1}{2}\overline{\pi}^T \hat{G}^T W \hat{G} \overline{\pi} \\ &\text{s.t. } e^T \overline{\pi} = 1. \end{aligned}$$

If $\overline{\pi} > 0$, then $\hat{\pi} \leftarrow \overline{\pi}$ becomes the new solution estimate and the above procedures are repeated. Otherwise, a step from $\hat{\pi}$ in the direction of $\overline{\pi}$ is made until some element hits zero (say, corresponding to the $\overline{j}$th column of G), in which case $\overline{j}$ is removed from $\mathcal{A}$ and (6.4) is reformulated and resolved for the new active set.

A complete description of our subproblem solver, ASQO, is presented as Algorithm 6.1 below. The algorithm returns a vector $\widetilde{\pi}$ corresponding to $\mathcal{A}$, i.e., corresponding to the nonzero elements of the optimal solution to (2.4). This vector is to be permuted and augmented with zeros in the appropriate entries to construct the optimal π from which the optimal primal solution d is obtained.

Algorithm 6.1 Active-Set Quadratic Optimization Subproblem Solver (ASQO)

```
1: (Initialization) Given $\mathcal{A}$ as an estimate of the optimal active set, let $\hat{G}$ be the
   submatrix of $G$ corresponding to the indices in $\mathcal{A}$ and let $\hat{\pi}$ be an $|\mathcal{A}|$-vector
   satisfying $e^T\hat{\pi} = 1$ and $\hat{\pi} \geq 0$.
2: (Termination check) If (6.1a) holds, then terminate. Otherwise, set $j \in \{1, \ldots, q\}$
   as any index not in $\mathcal{A}$ such that (6.2) holds.
3: (Rank-deficiency check) Solve (6.3) for $\widetilde{\pi}$. If $\widetilde{\pi}^T[e \ \hat{G}^T] = [1 \ g_j^T]$, go to step 5.
4: (Column augmentation) Append 0 to $\hat{\pi}$, $j$ to $\mathcal{A}$, and $g_j$ to $\hat{G}$, go to step 6.
5: (Column exchange) Replace $\hat{\pi}$ by $\hat{\pi} - t\widetilde{\pi}$ where

       $t = \min_i \{\hat{\pi}_i / \widetilde{\pi}_i : \widetilde{\pi}_i > 0\}.$

   Find some $i$ such that $\hat{\pi}_i = 0$. Delete the $i$th index from $\mathcal{A}$, the $i$th component
   from $\hat{\pi}$, and the $i$th column from $\hat{G}$. Append $t$ to $\hat{\pi}$, $j$ to $\mathcal{A}$, and $g_j$ to $\hat{G}$.
6: (Subproblem solution) Solve (6.4) for $\overline{\pi}$. If $\overline{\pi} > 0$, set $\hat{\pi} = \overline{\pi}$ and go to step 2.
7: (Column deletion) Replace $\hat{\pi}$ by $t\overline{\pi} + (1-t)\hat{\pi}$ where

       $t = \min\{1, \min_i \{\hat{\pi}_i / (\hat{\pi}_i - \overline{\pi}_i) : \overline{\pi}_i < 0\}\}.$

   Find $i$ such that $\hat{\pi}_i = 0$. Delete the $i$th index from $\mathcal{A}$, the $i$th component from
   $\hat{\pi}$, and the $i$th column from $\hat{G}$, then go to step 6.
```

Our implementation of ASQO actually maintains a Cholesky factorization of $(\hat{G}^T W \hat{G} + ee^T)$ that is updated during each iteration. Specifically, we maintain an upper triangular singular matrix R satisfying

$$R^T R = \hat{G}^T W \hat{G} + ee^T$$

with which it can easily be verified that the solutions to (6.3) and (6.4) can be obtained, respectively, by solving the pairs of systems

$$\left\{\begin{array}{rl} R^T r_1 = & e + \hat{G}^T W g_j \\ R\widetilde{\pi} = & r_1 \end{array}\right\} \quad \text{and} \quad \left\{\begin{array}{rl} R^T r_2 = & e \\ R\overline{\pi} = & r_2/\|r_2\|^2. \end{array}\right\}.$$

The maintenance of R and calculation of the intermediate vectors above allows sophisticated extensions of the rank-deficiency check in step (3) of ASQO so that it is less susceptible to numerical errors. We have implemented these extensions in our code, but suppress the details for simplicity in our presentation here; see [21] for details.

References

[1] K. M. Anstreicher and J. Lee. A Masked Spectral Bound for Maximum-Entropy Sampling. In *MODA 7 - Advances in Model-Oriented Design and Analysis*, pages 1–10, Berlin, Germany, 2004.

[2] C. G. Broyden. The Convergence of a Class of Double-Rank Minimization Algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6(1):76–90, 1970.

[3] J. V. Burke, D. Henrion, A. S. Lewis, and M. L. Overton. HIFOO - A MATLAB Package for Fixed-order Controller Design and H-infinity Optimization. In *Proceedings of the IFAC Symposium on Robust Control Design*, Haifa, Israel, 2006.

[4] J. V. Burke, A. S. Lewis, and M. L. Overton. Approximating Subdifferentials by Random Sampling of Gradients. *Mathematics of Operations Research*, 27(3):567–584, 2002.

[5] J. V. Burke, A. S. Lewis, and M. L. Overton. A Robust Gradient Sampling Algorithm for Nonsmooth, Nonconvex Optimization. *SIAM Journal on Optimization*, 15(3):751–779, 2005.

[6] J. V. Burke, A. S. Lewis, and M. L. Overton. Pseudospectral Components and the Distance to Uncontrollability. *SIAM Journal on Matrix Analysis and Applications*, 26(2):350–361, 2005.

[7] F. H. Clarke. *Optimization and Nonsmooth Analysis*. Canadian Mathematical Society Series of Monographs and Advanced Texts. John Wiley & Sons, New York, NY, USA, 1983.

[8] F. E. Curtis and M. L. Overton. A Sequential Quadratic Programming Algorithm for Nonconvex, Nonsmooth Constrained Optimization. *SIAM Journal on Optimization*, in second round of review; original submission, 2009.

[9] E. Dolan and J. Moré. Benchmarking Optimization Software with Performance Profiles. *Mathematical Programming*, 91:201–213, 2002.

[10] R. Fletcher. A New Approach to Variable Metric Algorithms. *Computer Journal*, 13(3):317–322, 1970.

[11] D. Goldfarb. A Family of Variable Metric Updates Derived by Variational Means. *Mathematics of Computation*, 24(109):23–26, 1970.

[12] A. A. Goldstein. Optimization of Lipschitz Continuous Functions. *Mathematical Programming*, 13(1):14–22, 1977.

[13] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, third edition, 1996.

[14] C. Greif and J. Varah. Minimizing the Condition Number for Small Rank Modifications. *SIAM Journal on Matrix Analysis and Applications*, 29(1):82–97, 2006.

[15] M. Haarala. *Large-Scale Nonsmooth Optimization: Variable Metric Bundle Method with Limited Memory*. PhD thesis, University of Jyväskylä, 2004.

[16] M. Haarala, K. Miettinen, and M. M. Mäkelä. New Limited Memory Bundle Method for Large-Scale Nonsmooth Optimization. *Optimization Methods and Software*, 19(6):673–692, 2004.

[17] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I*. A Series of Comprehensive Studies in Mathematics. Springer-Verlag, New York, NY, USA, 1993.

[18] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms II*. A Series of Comprehensive Studies in Mathematics. Springer-Verlag, New York, NY, USA, 1993.

[19] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[20] K. C. Kiwiel. *Methods of Descent for Nondifferentiable Optimization*. Lecture Notes in Mathematics. Springer-Verlag, New York, NY, USA, 1985.

[21] K. C. Kiwiel. A Method for Solving Certain Quadratic Programming Problems Arising in Nonsmooth Optimization. *IMA Journal of Numerical Analysis*, 6(2):137–152, 1986.

[22] K. C. Kiwiel. Convergence of the Gradient Sampling Algorithm for Nonsmooth Nonconvex Optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

[23] K. C. Kiwiel. A Nonderivative Version of the Gradient Sampling Algorithm for Nonsmooth Nonconvex Optimization. *SIAM Journal on Optimization*, 20(4):1983–1994, 2010.

[24] A. S. Lewis. Local Structure and Algorithms in Nonsmooth Optimization. In F. Jarre, C. Lemaréchal, and J. Zowe, editors, *Optimization and Applications*, pages 104–106. Mathematisches Forschungsinstitut Oberwolfach, Oberwolfach, Germany, 2005.

[25] A. S. Lewis and M. L. Overton. Nonsmooth Optimization via BFGS. *SIAM Journal on Optimization*, in review; original submission, 2010.

[26] L. Lukšan, M. Tůma, M. Šiška, J. Vlček, and N. Ramešová. UFO 2002: Interactive System for Universal Functional Optimization. Technical Report 883, Institute of Computer Science, Academy of Sciences of the Czech Republic, 2002.

[27] J. Nocedal. Updating Quasi-Newton Matrices With Limited Storage. *Mathmatics of Computation*, 35(151):773–782, 1980.

[28] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, second edition, 2006.

[29] R. T. Rockafellar. *Convex Analysis*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, Princeton, NJ, USA, 1970.

[30] D. F. Shanno. Conditioning of Quasi-Newton Methods for Function Minimization. *Mathematics of Computation*, 24(111):647–656, 1970.

[31] A. Skajaa. Limited Memory BFGS for Nonsmooth Optimization. Master's thesis, New York University, 2010.