A Limited-Memory Quasi-Newton Algorithm for Bound-Constrained Nonsmooth Optimization

Nitish Shirish Keskar* Andreas Wächter†
Department of Industrial Engineering and Management Sciences,
Northwestern University, Evanston, Illinois, USA – 60208

December 21, 2016

Abstract

We consider the problem of minimizing a continuous function that may be nonsmooth and nonconvex, subject to bound constraints. We propose an algorithm that uses the L-BFGS quasi-Newton approximation of the problem's curvature together with a variant of the weak Wolfe line search. The key ingredient of the method is an active-set selection strategy that defines the subspace in which search directions are computed. To overcome the inherent shortsightedness of the gradient for a nonsmooth function, we propose two strategies. The first relies on an approximation of the ϵ-minimum norm subgradient, and the second uses an iterative corrective loop that augments the active set based on the resulting search directions. We describe a Python implementation of the proposed algorithm and present numerical results on a set of standard test problems to illustrate the efficacy of our approach.

Keywords: nonsmooth optimization; bound constraints; quasi-Newton; L-BFGS; active-set method; active-set correction

1 Introduction

We propose an algorithm for solving bound-constrained optimization problems of the form

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \; & f(x) \\ \text{s.t. } & l \le x \le u, \end{aligned} \tag{1}$$

where the objective function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuous but may not be differentiable everywhere. No assumptions are placed on the convexity of f. The lower bounds $l \in (\mathbb{R} \cup \{-\infty\})^n$ and upper bounds $u \in (\mathbb{R} \cup \{\infty\})^n$ can take values of $-\infty$ or ∞ whenever the variables are unbounded in those coordinates. We assume that the problem is feasible; i.e., $l \le u$.

Many algorithms have been proposed for solving (1) when x is unconstrained. Some of these methods include gradient-sampling methods [4, 11, 25], bundle methods [16, 17, 30], quasi-Newton methods [12, 20, 26, 27, 33], and hybrid methods [12]. Gradient-sampling methods randomly sample gradients in the vicinity of the iterate to calculate an estimate of the minimum-norm subgradient. In conjunction with an Armijo-like line search, global convergence can be proved using these minimum-norm subgradients as search directions. Bundle methods aggregate subgradients from previous iterates and iteratively solve piecewise-quadratic approximations of the objective function to generate steps. Recently, Lewis and Overton [26] observed that the unadulterated BFGS method works very well when applied to unconstrained nonsmooth optimization problems so long as the weak Wolfe line search is performed. Skajaa [33] reported similar results for L-BFGS [28]. For problems ranging from $n = 100$ to $n = 10000$, it was found that L-BFGS was not only more efficient in solving test problems, but it was also more reliable compared to other methods. However,

*Email: keskar.nitish@u.northwestern.edu
†Email: andreas.waechter@northwestern.edu

theoretical convergence guarantees (or the lack thereof) of (L-)BFGS for nonsmooth problems remain to be shown. Recent efforts (e.g., [12]) focused on the design of a hybrid strategy that retains the efficacy of standard L-BFGS but ensures convergence through a gradient-sampling approach. Other approaches for solving (1) include subgradient methods, quasi-secant methods, and discrete gradient methods. We refer the reader to [2, 8, 21, 30, 33] for a detailed summary of these methods and their numerical performance.

In certain applications, it is necessary to optimize a nonsmooth objective function subject to bound constraints. These include applications in many fields including statistics, optimal control, and as subproblems for certain robust optimization problems [2]. Some of the algorithms described above can be extended to solve problems with bound constraints. For instance, the LMBM-B [22, 23] method extends the limited-memory bundle method to (1). Gradient-sampling methods have also been extended to the case of constrained optimization [10]. A natural question is whether the surprising and remarkable success of the unadulterated (L-)BFGS method in the unconstrained case can be extended to problems with bound constraints.

The L-BFGS-B method is a variant of L-BFGS for minimizing a smooth objective function over box constraints. At an iterate x^k, the method first determines an active set by computing a Cauchy point $\tilde{x}^k$ as the first local minimizer $\alpha > 0$ of f along the gradient-projection path $\alpha \mapsto P(x^k - \alpha \nabla f(x^k))$. Here, $P(v)$ is the orthogonal projection of a vector $v \in \mathbb{R}^n$ onto the feasible hypercube $[l, u]$. The bound constraints that are tight at the Cauchy point $\tilde{x}^k$ then define an active set, $\mathcal{A}^k = \{i : \tilde{x}_i^k \in \{l_i, u_i\}\}$, and a subspace step $\tilde{p}^k$ is computed as the solution of the problem

$$\begin{aligned} \min_{p \in \mathbb{R}^n} \; & f(x^k) + \nabla f(x^k)^T p + \frac{1}{2} p^T B^k p \\ \text{s.t. } & p_i = 0 \text{ for all } i \in \mathcal{A}^k. \end{aligned} \tag{2}$$

The objective in this subproblem is a quadratic model of the original objective function. Its Hessian matrix B^k is defined by the L-BFGS update and therefore is positive definite. Subproblem (2) is solved efficiently using the compact-form representation of L-BFGS [7]. Finally, the overall step $p^k = (\tilde{x}^k + \tilde{p}^k) - x^k$ is computed and a projected line search is performed along the path $\alpha \mapsto P(x^k + \alpha p^k)$ to find a step size satisfying the strong Wolfe conditions.

Henao et al. [18] recently proposed L-BFGS-B-NS as a variant of L-BFGS-B for solving (1) with a nonsmooth objective function. The only difference to the original method is that the strong Wolfe line search is replaced with the weak Wolfe line search. This is the same modification that was suggested by Lewis and Overton [26] in the unconstrained case.

In this paper, we propose a different adaptation of the L-BFGS method. Our method first determines an active set based on the bound constraints that are tight at the current iterate, without referring to a Cauchy point. After computing the search direction from (2), a new iterate is determined using a variant of the weak Wolfe line search.

The key ingredients in our method are active-set selection strategies that take into account the nonsmoothness of the function. First, we propose the use of an approximation of the minimum-norm ϵ-subgradient instead of the gradient to determine which bound constraints are binding at the current iterate. Second, we explore an iterative corrective mechanism that augments the active set until the final search direction points inside the feasible region.

Throughout the paper, we assume that the function is differentiable at each iterate and trial point, and that its gradient can be computed. This is in line with the work by Lewis and Overton [26] who make the same assumption for their numerical experiments. Burke et al. [4] describe a mechanism that perturbs a trial point in case f is not differentiable at that point. In the box-constrained case, the boundary of the feasible region might align with a manifold of nondifferentiability. Then, the projections carried out during the line search might generate trial points that lie in this manifold. To circumvent this problem, we assume that there is an extension of the function beyond the feasible region that is differentiable at almost all boundary points. For example, consider the feasible set $[0, 1] \subset \mathbb{R}$ with objective function $f(x) = |x|$ which is not differentiable at the boundary point $x = 0$. When we replace the objective with $\tilde{f}(x) = x$, the objective values are identical within the feasible region, resulting in the same optimal solution, but the function is now differentiable at $x = 0$.

The paper is organized as follows. In the subsection to follow, we introduce some notation that is used throughout the paper. In Section 2, we describe our algorithm including the active-set prediction and

correction strategies, as well as the proposed weak Wolfe line search. Finally, in Section 3, we present details of our Python implementation and detailed numerical results examining the efficacy of our approach.

1.1 Notation

We use superscripts to denote the iteration index and subscripts to denote a specific element of a vector. For instance, x_j^k refers to the j^{th} element of the k^{th} iterate. We abbreviate $[\nabla f(x)]_i$ by $\nabla_i f(x)$. We define the instantaneous projection of a direction p at an iterate x as

$$[T(x,p)]_i = \begin{cases} p_i & \text{if } x_i \in (l_i, u_i) \\ \max(p_i, 0) & \text{if } x_i = l_i \\ \min(p_i, 0) & \text{if } x_i = u_i. \end{cases} \tag{3}$$

This operator zeroes out those components of p for which x is at its bounds with p pointing in the direction of infeasibility. We use the notation $B_\epsilon(x)$ to denote the closed ball of radius ϵ centered at x. Further, we denote the cardinality of a set A by $|A|$. Finally, given a vector v and a set of indices $\mathcal{A}$, $v_\mathcal{A}$ refers to the subvector corresponding to the indices in $\mathcal{A}$. Similarly, given a matrix M, then $M_{\mathcal{A},\mathcal{B}}$ denotes the submatrix with row indices given in $\mathcal{A}$ and column indices given in $\mathcal{B}$. Finally, we let $\mathcal{N} = \{1, \ldots, n\}$ be the set of all variable indices.

2 Proposed Algorithm

2.1 Active-Set Framework

The proposed algorithm is an active-set method which, at each iteration, determines an estimate $\mathcal{A}^k$ of the optimal active set $\mathcal{A}^* := \{i \in \mathcal{N} : x_i^* = l_i \text{ or } x_i^* = u_i\}$ of a local solution x^* of (1). We say that the bound constraints in $\mathcal{A}^*$ are tight at x^*. The L-BFGS-B algorithm chooses as active set $\mathcal{A}^k$ the bounds at which the Cauchy point is tight. In contrast, our method chooses from bounds for which the current iterate x^k itself is tight, without referring to a Cauchy point.

For a smooth objective function, we might consider the set of the tight constraints that are binding; i.e.,

$$\mathcal{B}^k(g^k) = \{i \in \mathcal{N} : x_i^k = l_i \text{ and } g_i^k \geq 0\} \cup \{i \in \mathcal{N} : x_i^k = u_i \text{ and } g_i^k \leq 0\} \tag{4}$$

with $g^k = \nabla f(x^k)$. These are the coordinates for which the gradient predicts no decrease in the objective if the corresponding components of the iterate are moved inside the feasible region. With this, $\nabla_i f(x^k) = 0$ for all $i \in \mathcal{N} \setminus \mathcal{B}(\nabla f(x^k))$ if and only if x^k satisfies the first-order optimality conditions for problem (1) at x^k; i.e.,

$$\begin{aligned} &\nabla_i f(x^k) = 0 \text{ for all } i \text{ with } l_i < x_i^k < u_i \\ &\nabla_i f(x^k) \geq 0 \text{ for all } i \text{ with } x_i^k = l_i \\ &\nabla_i f(x^k) \leq 0 \text{ for all } i \text{ with } x_i^k = u_i. \end{aligned} \tag{5}$$

Consequently, the subspace step p^k obtained from solving (2) with $\mathcal{A}^k = \mathcal{B}(\nabla f(x^k))$ is zero if and only if x^k is a first-order optimal point. In addition, it can be shown that p^k is a descent direction for the projected line search; i.e., the function $\alpha \mapsto f(P(x^k + \alpha p^k))$ is decreasing for $\alpha > 0$ sufficiently small. (This is a consequence of [3, Proposition 1].)

Consider a simple algorithm that computes search directions from (2) with $\mathcal{A}^k = \mathcal{B}(\nabla f(x^k))$ and performs a projected line search to determine the new iterate $x^{k+1} = P(x^k + \alpha^k p^k)$ with some step size $\alpha^k > 0$. Suppose that f is differentiable and that the iterates converge to a non-degenerate first-order optimal x^*; i.e., x^* satisfies (5) and $\nabla_i f(x^*) \neq 0$ for all $i \in \mathcal{A}^*$. Further assume that at some iterate x^k sufficiently close to x^*, the bounds that are tight at x^k are identical to the optimal active set; i.e., $\{i \in \mathcal{N} : x_i^k = l_i \text{ or } x_i^k = u_i\} = \mathcal{A}^*$. It is then not difficult to show that $\mathcal{A}^k = \mathcal{B}(\nabla f(x^*))$ for all large k. In other words, the optimal active set is identified in a finite number of iterations. This observation motivates the choice $\mathcal{A}^k = \mathcal{B}(\nabla f(x^k))$.

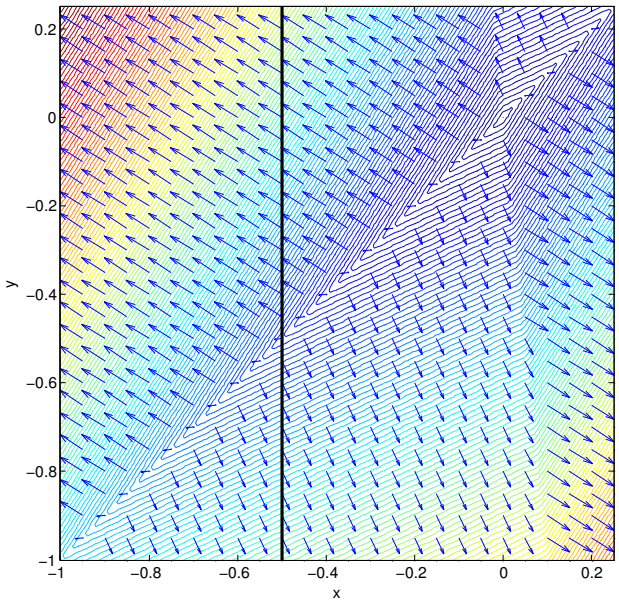

Figure 1: The contour lines of the objective function in (6). The arrows indicate the gradients of the function.

The conclusion in the previous paragraph was drawn under the strong assumption that an iterate is encountered at which all constraints in $\mathcal{A}^*$ are tight. To the best of our knowledge, no convergence proof has been established for the simple algorithm when this assumption is lifted, even when f is differentiable. Nevertheless, despite the lack of theoretical convergence guarantees, our proposed active-set selection strategy is based on (4) since it seems to perform well in practice in our setting. Recall that global convergence has not been proved for the unadulterated L-BFGS algorithm with a nonsmooth objective function even in the unconstrained case.

In the context of nonsmooth optimization, the gradient of the objective function can be very myopic in regions close to a manifold on which the function is nondifferentiable, and a gradient-based active-set identification can be quite misleading. We illustrate this with a simple example, depicted in Figure 1.

Consider the problem

$$\begin{aligned} &\min_{x\in\mathbb{R}^2} \ |x_1 - x_2| + \frac{1}{2}(x_1 + 0.1x_2)^2 \\ &\text{s.t. } x_1 \leq -0.5 \end{aligned} \tag{6}$$

with optimal solution $x^* = (-0.5, -0.5)^T$. Suppose we have an iterate $x^k = (-0.5, a)^T$ for a number $a \in (-5, -0.5)$. The function is differentiable at this point with gradient $\nabla f(x^k) = (1, -1)^T + (-0.5 + 0.1a)(1, 0.1)^T = (0.5 + 0.1a, -1.05 + 0.01a)^T$. Given that $a > -5$, we have $\nabla_1 f(x^k) > 0$. Therefore, the choice $\mathcal{A}_k = \mathcal{B}(\nabla f(x^k))$ predicts x_1 to be free, no matter how close x^k is to x^*. This determination is incorrect since x_1 is indeed at its bound at the solution. Notice that the failure of identification is caused by the inherent shortsightedness of the gradient and not due to some kind of degeneracy.

We point out that also the active-set identification of the L-BFGS-B method does not recognize that x_1 is tight at the solution. The L-BFGS-B method, as described previously, locates the first minimizer of the gradient-projection path, $\alpha \mapsto P(x^k - \alpha\nabla f(x^k))$, for the active-set identification. For problem (6), the ray $\{x^k - \alpha\nabla f(x^k) : \alpha > 0\}$, for an iterate $x^k = (-0.5, a)^T$ with $a \in (-5, -0.5)$, never intersects a bound. Thus, also in this case, x_1 is not recognized as active.

In order to compensate for the shortsightedness of the gradient, we propose two strategies: (i) an active-set prediction that considers an approximation $\tilde{g}^k$ of minimum-norm subgradients of nearby non-differentiable points to determine the binding constraints $\mathcal{B}(\tilde{g}^k)$ (Section 2.2); and (ii) a correction mechanism that augments the active set if the search direction, computed with L-BFGS approximation of the nonsmooth objective, indicates that a variable should be active (Section 2.4).

2.2 Active-Set Prediction Using a Subgradient Approximation

Continuing with problem (6), let us consider the scenario in which we use the ϵ-minimum-norm subgradient (ϵ-MNSG) based on the Clarke ϵ-subdifferential [4] for the active-set prediction. For a fixed $\epsilon > 0$, the ϵ-MNSG is defined as

$$\hat{g}_\epsilon(x) = \underset{y \in \text{cl conv } \nabla f(B_\epsilon(x))}{\arg\min} \quad \frac{1}{2}\|y\|_2^2. \tag{7}$$

Here, $\nabla f(B_\epsilon(x)) = \{\nabla f(y) : y \in B_\epsilon(x)\}$, and the term "cl conv" indicates the closure of the convex hull of a set. When ϵ is sufficiently small and a is close to -0.5, the ϵ-MNSG at $x^k = (-0.5, a)^T$ is approximately $\hat{g}_\epsilon(x^k) \approx (-0.3025, -0.3025)^T$. The active set $\mathcal{A}^k = \mathcal{B}(\hat{g}_\epsilon(x^k))$ correctly identifies that x_1 is tight at the solution. Indeed, the ϵ-MNSG attempts to be less myopic than the gradient and forms the basis for gradient-sampling methods [4]. This motivates us to base the active-set identification on the ϵ-MNSG instead of the gradient.

Computing the true ϵ-MNSG is usually not feasible, due to the complex nature of $\nabla f(B_\epsilon(x^k))$. Instead, gradient sampling methods work with an approximation $\tilde{g}^k$ that is based on the gradients at points from a finite random subsample $G^k = \{x^{k,1}, \ldots, x^{k,l^k}\}$ of the ball $B_\epsilon(x^k)$. More specifically, $\tilde{g}^k = \sum_{i=1}^{l^k} \nabla f(x^{k,i})\lambda_i^\star$ where $\lambda^\star$ is the solution of the convex quadratic problem

$$\begin{aligned} \min_{\lambda \in \mathbb{R}^{l^k}} \ & \frac{1}{2}\left\| \sum_{i=1}^{l^k} \nabla f(x^{k,i})\lambda_i \right\|_2^2 \\ \text{s.t.} \ & \sum_{i=1}^{l^k} \lambda_i = 1 \\ & 0 \le \lambda_i \le 1 \quad \text{for all } i = 1, \ldots, l^k. \end{aligned} \tag{8}$$

A good approximation of the ϵ-MNSG typically requires a large number l^k of gradient evaluations. For the purpose of determining the active set, however, an inexact estimate might suffice, since its main purpose is to capture roughly the geometry of the nonsmooth function. It is not used for the step computation itself. To avoid additional gradient evaluations, we simply choose $G^k = \{x^k, \ldots, x^{\max\{0,k-M\}}\}$ to contain the most recent M iterates. This strategy is motivated by two observations: (i) as we will describe in Section 2.5, the line search encourages the iterates to cross over manifolds of nondifferentiability and thus, the gradients for G^k represent different "pieces" of the nonsmooth function; and (ii) near the solution, where active-set prediction strategies are arguably more important, the steps taken by the algorithm are small and the points in G^k are then from a small neighborhood around the current iterate.

Motivated by these observations, our first active-set selection strategy chooses

$$\mathcal{A}^k = \mathcal{B}^k(\tilde{g}^k) \cup \mathcal{B}^k(\nabla f(x^k)). \tag{9}$$

Note that we include the bound constraints identified by the gradient $\nabla f(x^k)$ as well. We observed in our experiments that this led to better performance. We speculate that, in regions not close to a manifold on which the function is nonsmooth, the active set identified by the gradient is often reliable and the subgradient approximation might cause spurious identification, when the points in G^k are not close to each other.

2.3 Computation of the Search Direction

Our second active-set strategy loops over candidate choices for the active set that are evaluated based on the search directions they generate. We describe the step computation first.

The search directions are based on the BFGS method [31]. This method constructs and updates a convex second-order model of the objective function requiring only the first-order derivatives. Given an estimate, B^l, of the curvature of the objective function, the BFGS method revises the estimate using a rank-2 update as

$$B^{l+1} = B^l + \frac{y^l (y^l)^T}{(y^l)^T s^l} - \frac{B^l s^l (s^l)^T B^l}{(s^l)^T B^l s^l}, \tag{10}$$

where

$$s^l = x^{l+1} - x^l \tag{11a}$$

$$y^l = \nabla f(x^{l+1}) - \nabla f(x^l). \tag{11b}$$

One usually requires that the curvature condition

$$(s^l)^T y^l > 0 \tag{12}$$

holds, since then B^l remains positive definite if the initial matrix B^0 is positive definite. For smooth convex objective functions, the BFGS method possesses strong theoretical properties including global convergence and superlinear local convergence. Even though only limited theoretical convergence guarantees have been established for nonconvex objectives, many have noted good performance on a variety of problems.

The original BFGS method requires the storage and manipulation of an $n \times n$ matrix. For large-scale problems, this is unwieldy. The limited-memory BFGS (L-BFGS) method [28] attempts to alleviate this handicap by storing only the past m curvature pairs (s^l, y^l). The matrix B^l itself is never explicitly constructed. The value for m, also called as the L-BFGS memory, is often in the range of 5–20. This reduces the storage from $\mathcal{O}(n^2)$ to $\mathcal{O}(mn)$. The complexity of the search direction computation in the L-BFGS method is also reduced from $\mathcal{O}(n^2)$ to $\mathcal{O}(mn)$.

Given an active set $\mathcal{A}^k$ and an L-BFGS approximation B^k to the curvature of the problem, we generate the search direction p^k as the solution to subproblem (2). Here, all components of p^k belonging to $\mathcal{A}^k$ are set to zero, and the remaining entries are obtained from a linear system involving a symmetric submatrix of B^k. Making use of the L-BFGS compact representation matrices [6, 7], the step can be compute efficiently, using $2m^2t + 6mt + 4t + \mathcal{O}(m^3)$ operations where $t = |\mathcal{A}^k|$.

To specify the L-BFGS approximation in a given iteration k, it is necessary to provide an initial matrix $B^{k,0}$, from which $B^k = B^{k,m}$ is generated by repeatedly applying the update formula (10). The matrix $B^{k,0}$ is an estimate of the curvature of f. This choice, especially when the L-BFGS memory m is low, has direct consequences on the quality of the search direction. For smooth optimization, $\theta = \frac{(s^k)^T y^k}{(y^k)^T y^k}$ is often recommended and is found to work well in a variety of applications. Intuitively, the ratio is justified since it is a scalar approximation to $\nabla^2 f(x^k)$ [31]. However, for nonsmooth optimization this choice seems to lead to inferior performance. Instead, Curtis and Que [12] proposed $\theta = \max(1.0, \min(\|\nabla f(x^k)\|_\infty, 10^8))$. We use this choice in our implementation as well.

2.4 An Active-Set Correction Mechanism

In Section 2.2, we described an active-set identification mechanism that is based on an approximation of a subgradient. This strategy attempts to guess directly which bounds are tight at the optimal solution.

Next we describe another approach using an iterative correction procedure that judges the quality of a candidate active set and adjusts it if necessary. The quality of the active set is adjudged through the search direction generated using it. The goal is to obtain a direction that is feasible in the sense that a sufficiently small step into this direction does not leave the feasible region. If, for a given candidate active set, there is a variable that is tight at the current iterate and the candidate search direction points outside the feasible region, then this variable is added to the active set and the procedure is repeated. Similar mechanisms have been used previously, for example, for solving convex quadratic programs [9, 19] and ℓ_1-regularized convex optimization problems [5, 24].

For ease of notation, we drop the iteration index k for the remainder of this section. Given an iterate x, we let $g = \nabla f(x)$, and we define the set of interior variables, $\mathcal{F} = \{i : l_i < x_i < u_i\}$ and the set of variables with tight bound constraints, $\bar{\mathcal{F}} = \mathcal{N} \setminus \mathcal{F}$. Algorithm 1 formally states the proposed active-set correction procedure. When there are no tight bounds, the active set must be empty, and step 4 immediately returns the unrestricted search direction in the full space. Otherwise, the t-loop computes a potential search direction in step 7 and tests if there are any components, collected in the set $\mathcal{C}^t$, that would take a variable instantaneously outside its bound constraints. Such components are added to the active set, until a feasible direction is found. Clearly, the loop terminates in finite time, since $\mathcal{A}^t$ grows by at least one element per iteration.

Algorithm 1 ActiveSetCorrection

Inputs: Current iterate x; initial active set $\mathcal{A}^{\text{init}} \subseteq \bar{\mathcal{F}}$.
Output: Final active set $\mathcal{A}$ with corresponding search direction p.

1: Initialize $t \leftarrow 0$ and set $\mathcal{A}^0 = \mathcal{A}^{\text{init}}$.
2: **if** $\bar{\mathcal{F}} = \emptyset$ **then** ▷ Nothing to do if there are no tight bounds
3: Compute p as solution of (2) with $\mathcal{A}^k = \emptyset$.
4: **return** $\mathcal{A} = \emptyset$ and p.
5: **end if**
6: **for** $t = 0, 1, 2, \ldots$ **do**
7: Compute p^t as solution of (2) with $\mathcal{A}^k = \mathcal{A}^t$. ▷ Potential search direction
8: Set $\mathcal{C}^t = \{i \in \bar{\mathcal{F}} \setminus \mathcal{A}^t : T(x, p^t)_i \neq p_i^t\}$. ▷ Variables to be added
9: **if** $\mathcal{C}^t = \emptyset$ **then**
10: **return** $\mathcal{A} = \mathcal{A}^t$ and $p = p^t$. ▷ No more corrections necessary
11: **end if**
12: Set $\mathcal{A}^{t+1} = \mathcal{A}^t \cup \mathcal{C}^t$.
13: **end for**

The approach described in Section 2.2 uses a subgradient approximation to overcome the shortsightedness of the gradient when predicting the optimal active set. In contrast, the corrective procedure in Algorithm 1 exploits the fact that the L-BFGS approximation of the problem curvature contains information about the structure of the nonsmoothness. It has been observed that the (L-)BFGS approximation of a nonsmooth objective function is able to approximate the U- and V-spaces of the objective function [26, 33]. Roughly speaking, the U-space of f at a point x is the subspace tangent to the manifold of points at which f is not differentiable. The V-space is the orthogonal complement of the U-space. In [26], Lewis and Overton hypothesized that the V-space of a nonsmooth function can be numerically approximated within the unadulterated BFGS method through the eigenvectors of B^k corresponding to eigenvalues that converge to infinity.

In our example problem (6), the V-space at any point x with $x_1 = x_2$, including the optimal solution, is spanned by $(1, -1)^T$. When we apply the proposed method from random starting points, the iterates converge to the solution $(-0.5, -0.5)^T$. After some iterations, the BFGS matrix is approximately

$$B^k \approx y^k \cdot \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{13}$$

with some sequence y^k converging to infinity. Note that the eigenvectors of B^k suggest precise recovery of the U- and V-spaces of the objective function. In particular, the eigenvector $(1, -1)^T$ with respect to the asymptotically infinite eigenvalue indeed spans the V-space of f at the optimal solution. Even though the matrix on the right-hand side of (13) is singular, B^k itself is always nonsingular. Numerically we observe that the inverse matrix $H^k = (B^k)^{-1}$ is approximately

$$H^k \approx \tilde{y}^k \cdot \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \tag{14}$$

with some sequence $\tilde{y}^k$ converging to zero, and now $(1, -1)^T$ is an eigenvector with respect to the eigenvalue approaching zero.

Dropping the iteration index k, consider again an iterate of the form $x = (-0.5, a)^T$ with $a \in (-5, -0.5)$ and gradient $\nabla f(x) = (0.5 + 0.1a, -1.05 + 0.01a)^T$. In Section 2.1 we observed that the naïve choice $\mathcal{A} = \mathcal{B}(\nabla f(x)) = \emptyset$ fails to recognize that x_1 is active at the solution. If we choose $\mathcal{A}^{\text{init}} = \mathcal{B}(\nabla f(x))$ in Algorithm 1, we have $\mathcal{A}^0 = \emptyset$ in the first iteration. With the approximation (14), the search direction in step 7 becomes $p^0 \approx \tilde{y} \cdot (0.55 - 0.11a, 0.55 - 0.11a)^T$. Clearly, from the current iterate with $x_1 = -0.5$, this direction points out of the feasible region because $p_1^0 > 1$. Therefore, $T(x, p^0)_1 \neq p_1^0$ and $\mathcal{C}^0 = \{1\}$. In the next iteration of the correction loop, $\mathcal{A}^1 = \{1\}$ is accepted as the final active set, and correctly predicts that x_1 is active at the solution.

The following lemma shows that the correction mechanism cannot lead to a spurious termination of the overall algorithm. A zero step can be generated only when the current iterate is already a stationary point of the objective function (assuming that $\mathcal{A}^0$ is initialized as $\mathcal{B}(\nabla f(x^k))$).

Lemma 2.1. *Suppose $\mathcal{A}^0 = \mathcal{B}(\nabla f(x^k))$ and the corrective loop terminates with $p = 0$. Then, the current iterate x^k satisfies the first order optimality conditions* (5) *for problem* (1).

Proof of Lemma 2.1. For ease of exposition, we assume that $l = 0$ and $u = \infty$ in problem (1). Let $\hat{t}$ be the iteration in which the method terminates with $p = p^{\hat{t}} = 0$ and let $g = \nabla f(x)$.

Given the active set $\mathcal{A}^{\hat{t}} \subseteq \bar{\mathcal{F}}$ and defining $\bar{\mathcal{A}}^{\hat{t}} := \bar{\mathcal{F}} \setminus \mathcal{A}^{\hat{t}}$, the solution p to (2) (with $\mathcal{A}^k = \mathcal{A}^{\hat{t}}$) is obtained by solving the linear system

$$\begin{bmatrix} B_{\mathcal{F}\mathcal{F}} & B_{\mathcal{F}\bar{\mathcal{A}}^{\hat{t}}} \\ B_{\bar{\mathcal{A}}^{\hat{t}}\mathcal{F}} & B_{\bar{\mathcal{A}}^{\hat{t}}\bar{\mathcal{A}}^{\hat{t}}} \end{bmatrix} \begin{pmatrix} p_{\mathcal{F}} \\ p_{\bar{\mathcal{A}}^{\hat{t}}} \end{pmatrix} = - \begin{pmatrix} g_{\mathcal{F}} \\ g_{\bar{\mathcal{A}}^{\hat{t}}} \end{pmatrix} \tag{15}$$

and setting

$$p_{\mathcal{A}^{\hat{t}}} = 0. \tag{16}$$

Because the L-BFGS approximation B is positive definite, (15) together with $p^{\hat{t}} = 0$ implies

$$g_{\mathcal{F}} = 0 \text{ and } g_{\bar{\mathcal{A}}^{\hat{t}}} = 0. \tag{17}$$

For the purpose of deriving a contradiction suppose that $\hat{t} > 0$. Then $\mathcal{A}^{\hat{t}} = \mathcal{A}^{\hat{t}-1} \cup \mathcal{C}^{\hat{t}-1}$, and therefore $\bar{\mathcal{A}}^{\hat{t}-1} = \bar{\mathcal{A}}^{\hat{t}} \cup \mathcal{C}^{\hat{t}-1}$. By definition, $p_i^{\hat{t}-1} < 0$ for every $i \in \mathcal{C}^{\hat{t}-1}$. Since $\mathcal{C}^{\hat{t}-1} \neq \emptyset$, we have $p^{\hat{t}-1} \neq 0$. Consider the linear system from which $p^{\hat{t}-1}$ is computed:

$$\underbrace{\begin{bmatrix} B_{\mathcal{F}\mathcal{F}} & B_{\mathcal{F}\bar{\mathcal{A}}^{\hat{t}}} & B_{\mathcal{F}\mathcal{C}^{\hat{t}-1}} \\ B_{\bar{\mathcal{A}}^{\hat{t}}\mathcal{F}} & B_{\bar{\mathcal{A}}^{\hat{t}}\bar{\mathcal{A}}^{\hat{t}}} & B_{\bar{\mathcal{A}}^{\hat{t}}\mathcal{C}^{\hat{t}-1}} \\ B_{\mathcal{C}^{\hat{t}-1}\mathcal{F}} & B_{\mathcal{C}^{\hat{t}-1}\bar{\mathcal{A}}^{\hat{t}}} & B_{\mathcal{C}^{\hat{t}-1}\mathcal{C}^{\hat{t}-1}} \end{bmatrix}}_{=:\hat{B}^{\hat{t}-1}} \underbrace{\begin{pmatrix} p_{\mathcal{F}}^{\hat{t}-1} \\ p_{\bar{\mathcal{A}}^{\hat{t}}}^{\hat{t}-1} \\ p_{\mathcal{C}^{\hat{t}-1}}^{\hat{t}-1} \end{pmatrix}}_{=:\hat{p}^{\hat{t}-1}} = - \underbrace{\begin{pmatrix} g_{\mathcal{F}} \\ g_{\bar{\mathcal{A}}^{\hat{t}}} \\ g_{\mathcal{C}^{\hat{t}-1}} \end{pmatrix}}_{=:\hat{g}^{\hat{t}-1}}.$$

With (17) we obtain

$$(g_{\mathcal{C}^{\hat{t}-1}})^T p_{\mathcal{C}^{\hat{t}-1}}^{\hat{t}-1} = (g^{\hat{t}-1})^T \hat{p}^{\hat{t}-1} = -(\hat{p}^{\hat{t}-1})^T B^{\hat{t}-1} \hat{p}^{\hat{t}-1} < 0 \tag{18}$$

because B is positive definite and $p^{\hat{t}-1} \neq 0$.

On the other hand, $\mathcal{C}^{\hat{t}-1} \subseteq \bar{\mathcal{F}} \setminus \mathcal{A}^{\hat{t}-1} \subseteq \bar{\mathcal{F}} \setminus \mathcal{A}^{\hat{t}-2} \subseteq \ldots \subseteq \bar{\mathcal{F}} \setminus \mathcal{A}^0 = \bar{\mathcal{F}} \setminus \mathcal{B}(g)$. From (4) we then have $g_i < 0$ for all $i \in \mathcal{C}^{\hat{t}-1}$. Also, from the definition of $\mathcal{C}^{\hat{t}-1}$, it is $p_i^{\hat{t}-1} < 0$ for all $i \in \mathcal{C}^{\hat{t}-1}$. Therefore,

$$(g_{\mathcal{C}^{\hat{t}-1}})^T p_{\mathcal{C}^{\hat{t}-1}}^{\hat{t}-1} = \sum_{i \in \mathcal{C}^{\hat{t}-1}} g_i p_i^{\hat{t}-1} > 0,$$

in contradiction to (18).

It follows that $\hat{t}$ must be zero, and (17) yields that $g_i = 0$ for any $i \notin \mathcal{A}^0 = \mathcal{B}(\nabla f(x^k))$. Consequently, (5) holds. □

2.5 Line search

Once a search direction p^k at an iterate x^k has been calculated, the algorithm determines a step size $\alpha^k > 0$ to generate the next iterate, $x^{k+1} = P(x^k + \alpha^k p^k)$. The projection ensures that the new iterate is feasible.

For the unconstrained minimization of a nonsmooth function, Lewis and Overton [26] use the weak Wolfe conditions (19) and (20) to determine whether a trial point $x^{\text{trial}} = x^k + \alpha p^k$ is acceptable as a new iterate. Given fixed values for $c_1, c_2 \in (0, 1)$ with $c_1 < c_2$, the first condition,

$$f(x^{\text{trial}}) \leq f(x^k) + \alpha c_1 \nabla f(x^k)^T p^k, \tag{19}$$

ensures that the objective function decreases by at least a fraction of what is predicted by a linear approximation that is based on the gradient. Because the objective is nonsmooth, the linear model might be a good approximation only for very small step sizes α. Nevertheless, since f is assumed to be differentiable at x^k, condition (19) can be satisfied as long as α is sufficiently small.

The second condition,

$$\nabla f(x^{\text{trial}})^T p^k \geq c_2 \nabla f(x^k)^T p^k, \tag{20}$$

imposes that the slope of the function $\phi(\alpha) = f(x^k + \alpha p^k)$ is less steep at α than at 0, indicating that sufficient progress towards a local minimizer of $\phi(\alpha)$ is made. With a nonsmooth objective, a local minimizer of $\phi(\alpha)$ might be at a point where f (and hence ϕ) is nondifferentiable. If the current iterate is close to such a point, requiring (20) leads to a step size α that is beyond the point of nondifferentiability. This observation is crucial and provides the main intuition why the BFGS algorithm works well for nonsmooth problems. Because the next iterate lies on another "smooth piece" of the nonsmooth function, the new gradient is quite different from the current gradient, even when the next iterate is very nearby. Recalling the definition (11) of s^k and y^k in the BFGS update, we see that then the gradient difference y^k is much larger in size than the step s^k. Consequently, high curvature in the direction s^k is incorporated into the BFGS update B^{k+1}, approximating the infinite curvature at the point of nondifferentiability.

Lewis and Overton [26] prove that, in the absence of bounds, a step size α satisfying both (19) and (20) always exists, and they provide a bracketing procedure to find it. We point out that (20) also guarantees that the (s^k, y^k) pair for the BFGS update satisfies $(s^k)^T y^k > 0$ so that the update is well-defined.

On the other hand, for the minimization of a *smooth* objective function subject to bound constraints, Ferry [14] proposed a generalized Wolfe line search which replaces the search direction by $\bar{p}^k = T(x^k, p^k)$. Recalling the definition of T in (3), we see that $\bar{p}^k$ is the modified search direction that zeros out all components that would result in an immediate violation of a constraint. With this, the Wolfe conditions suggested by Ferry [14] are

$$f(x^{\text{trial}}) \leq f(x^k) + \alpha c_1 \nabla f(x^k)^T \bar{p}^k \tag{21}$$

$$\nabla f(x^{\text{trial}})^T T(x^{\text{trial}}, p^k) \geq c_2 \nabla f(x^k)^T \bar{p}^k. \tag{22}$$

We adopt these conditions in our context of minimizing a nonsmooth objective. Algorithm 2 describes the corresponding bracketing mechanism. Note that, in the absence of bounds, this procedure is identical to the line search algorithm proposed by Lewis and Overton [26]. When f is smooth, Ferry [14] showed that there always exists a step size α that satisfies both (21) and (22). For a nonsmooth objective, such a step size may not exist. In our method, when a suitable step size cannot be found in step 16, finite termination is ensured by the termination test in step 24.

The bracketing mechanism generates a sequence of values for U and L in a way that shrinks the length of the interval $[L, U]$ to zero (see steps 20 and 22 together with steps 11 and 14). Because it is not clear whether a step size satisfying both (21) and (22) can be found, the algorithm will attempt only a moderate number of trial step sizes, until the relative interval length is on the order of ϵ_{rel}. Whenever a step size is encountered that satisfies the sufficient decrease condition (21), step 14 sets L to this value (unless the search is terminated in step 16). Therefore, when step 26 returns a nonzero step size, it is guaranteed that the next iterate will have a smaller objective value, and so the overall optimization algorithm cannot cycle. On the other hand, if L is zero in step 28, the trial step size $U = \alpha$ has become smaller than ϵ_{abs}. In that case, we declare a line search error, which is likely caused by numerical issues in the search direction computation or round-off in the function evaluation. Finally, it may happen that the computed search direction is such that $P(x + \alpha p) = x$ for any $\alpha > 0$. Then there is no point in conducting a line search and we terminate the optimization with an error message in step 6. This may occur due to numerical problems during the step computation.

2.6 Main Algorithm

The overall optimization algorithm for solving (1) is given in Algorithm 3. Step 6 is purposely left vague, because we will explore different alternatives for the active-set selection. The experiments in the following section consider the following options:

Variant 1 Choose the active set based on the gradient at the current iterate, $\mathcal{A}^k = \mathcal{B}(\nabla f(x^k))$.

Variant 2 Choose the active set based on the subgradient approximation using (9).

Variant 3 Compute the active set from the correction procedure Algorithm 1 with initial guess $\mathcal{A}^{\text{init}} = \mathcal{B}(\nabla f(x^k))$.

Algorithm 2 ModifiedWolfe

Inputs: Current iterate x and a search direction p.
Output: Step size α to generate next iterate.
Parameters: $c_1, c_2 \in (0,1)$ with $c_1 < c_2$; $\epsilon_{\text{abs}}, \epsilon_{\text{rel}} > 0$.

1: Set $L = 0$
2: Set $U = \max_i\{\gamma_i\}$, where $\gamma_i \stackrel{\text{def}}{=} \begin{cases} \frac{u_i - x_i}{p_i} & p_i > 0 \text{ and } x_i \neq u_i; \\ \frac{x_i - l_i}{p_i} & p_i < 0 \text{ and } x_i \neq l_i; \\ \infty & \text{otherwise.} \end{cases}$
3: Set $\alpha = \min(1, U)$.
4: Compute $\bar{p} = T(x, p)$.
5: **if** $\bar{p} = 0$ **then**
6: **terminate** with "No search direction". ▷ Error due to bad search direction
7: **end if**
8: **for** $t = 0, 1, 2, \ldots$ **do**
9: Set $x^{\text{trial}} = P(x + \alpha\bar{p})$.
10: **if** (21) does not hold **then** ▷ Sufficient decrease condition does not hold
11: Set $U = \alpha$.
12: **else**
13: **if** (22) does not hold **then** ▷ Curvature condition does not hold
14: Set $L = \alpha$.
15: **else**
16: **return** α. ▷ Return step satisfying weak Wolfe conditions
17: **end if**
18: **end if**
19: **if** $U < \max_i\{\gamma_i\}$ **then** ▷ Update step-size
20: Set $\alpha = \frac{U+L}{2}$.
21: **else**
22: Set $\alpha = \min(2L, U)$.
23: **end if**
24: **if** $U - L < \epsilon_{\text{abs}} + \epsilon_{\text{rel}} L$ **then**
25: **if** $L > 0$ **then**
26: **return** L. ▷ Return step satisfying sufficient decrease condition
27: **else**
28: **terminate** with "Line Search Error". ▷ Error, no suitable step size found
29: **end if**
30: **end if**
31: **end for**

Variant 4 Compute the active set from the correction procedure Algorithm 1 with initial guess $\mathcal{A}^{\text{init}}$ based on the subgradient approximation using (9).

For the last two variants, the search direction is already computed as byproduct of the active set selection and step 7 does not require any actual work.

There is no guarantee that the pair (s^k, y^k) pair defined in (11) satisfies the curvature condition (12), even when the weak Wolfe conditions (21) and (22) are satisfied, since the actual step s^k might be different from $\alpha^k\bar{p}^k$, due to the projection in step 10. This is in contrast to the unconstrained case, where (22) implies that (11) holds. To handle this situation, the update is skipped in step 12 whenever $(s^k)^T(y^k) \leq \epsilon_{\text{skip}}\|s^k\|\|y^k\|$.

As mentioned in Section 1, we do not consider any theoretical convergence properties of this method, including the possibilities of stalling at non-stationary points and of spurious termination of the line search. We point out that convergence guarantees remain an open question even when no constraints are present.

Algorithm 3 Nonsmooth Quasi-Newton (NQN)

Inputs: Initial point $x^0 \in [l, u]$.
Parameters: Size of L-BFGS memory m; update tolerance ϵ_{skip}.

Initialize storage $\mathcal{S}$ of L-BFGS curvature pairs to be empty.
for $k = 0, 1, 2, \ldots$ **do**
  if $T(x^k, -\nabla f(x^k) = 0$ **then**
    return x^k ▷ Finite termination at stationary point
  end if
  Choose active set $\mathcal{A}^k$. ▷ Details specified elsewhere
  Compute search direction p^k.
  Compute $\alpha^k = \texttt{ModifiedWolfe}(x^k, p^k)$. ▷ Perform line search using Algorithm 2
  Set $\bar{p}^k = T(x^k, p^k)$.
  Set $x^{k+1} = P(x^k + \alpha^k \bar{p}^k)$.
  Compute curvature pair (s^k, y^k) from (11).
  if $(s^k)^T y^k > \epsilon_{\text{skip}} \|s^k\| \|y^k\|$ **then** ▷ Discard pair if curvature condition not satisfied
    Store (s^k, y^k) in $\mathcal{S}$. ▷ Update L-BFGS memory
    If $|\mathcal{S}| > m$ then discard oldest curvature pair.
  end if
end for

Parameter	Value	Description
m	20	L-BFGS memory
M	20	Maximum size of sample set G^k
(c_1, c_2)	$(10^{-8}, 0.9)$	Parameters on Wolfe conditions
ϵ_{abs}	10^{-16}	Absolute bracketing tolerance
ϵ_{rel}	10^{-6}	Relative bracketing tolerance
ϵ_{skip}	10^{-8}	Tolerance for skipping L-BFGS update

Table 1: Parameter values used for numerical experiments.

3 Numerical Experiments

3.1 Implementation and Problem Set

We implemented Algorithm 3 in Python. We will refer to it as NQN. The code for our algorithm can be found in our GitHub repository: `https://github.com/keskarnitish/NQN`. The values for the various parameters used are summarized in Table 1. In order to solve the quadratic program (8) for the subgradient approximation, we used the CVXOPT package [1]. We rely on the NumPy package [35] for linear algebra operations, and Theano [34] is used to compute derivatives of the objective functions using algorithmic differentiation.

We compare NQN with other codes for solving (1), namely (i) L-BFGS-B [6]; (ii) L-BFGS-B-NS [18]; and (iii) LMBM-B [23]. While L-BFGS-B is not designed to solve nonsmooth problems, we nonetheless include it owing to its documented success at solving smooth bound-constrained problems. We include L-BFGS-B-NS and LMBM-B since they are specifically designed to solve (1) and have shown competitive performance on variety of tasks. The former is identical to the L-BFGS-B algorithm except that it uses the weak Wolfe line search as opposed to the strong Wolfe line search. LMBM-B [23] combines the limited-memory bundle method (LMBM) [16, 17] with a Cauchy-point-based active-set selection strategy similar to the one in L-BFGS-B. The LMBM-B method generates steps using a subgradient bundle in conjunction with an L-BFGS/SR-1 updating scheme to gain curvature information.

To make sure each solver obtains the same function and derivative information, we implemented Python wrappers around the Fortran codes written by the respective authors. The original Fortran codes can be found at `users.iems.northwestern.edu/~nocedal/lbfgsb.html`, `github.com/wilmerhenao/L-BFGS-B-NS` and `napsu.karmitsa.fi/lmbm/` for L-BFGS-B, L-BFGS-B-NS and LMBM-B respectively. We exclude other

Problem Number	Test Problem	Reference
1	`Active_Faces`	[16]
2	`Chained_CB3_1`	[16]
3	`Chained_CB3_2`	[16]
4	`Chained_Crescent_1`	[16]
5	`Chained_Crescent_2`	[16]
6	`Chained_LQ`	[16]
7	`Chained_Mifflin_2`	[16]
8	`Convex_Nonsmooth`	[33]
9	`L1`	[33]
10	`L1HILB`	[16]
11	`L2`	[26]
12	`MAXHILB`	[16]
13	`MAXQ`	[16]
14	`Modified_Rosenbrock_1`	[18]
15	`Modified_Rosenbrock_2`	[18]
16	`Myopic_Coupled`	(24)
17	`Myopic_Decoupled`	(23)
18	`Nesterov_1`	[15]
19	`Nesterov_2`	[15]
20	`Nesterov_3`	[32]
21	`Nonsmooth_Brown`	[16]
22	`TEST29_2`	[29]
23	`TEST29_6`	[29]
24	`TEST29_22`	[29]
25	`TEST29_24`	[29]

Table 2: Test problems used in numerical experiments.

methods, including gradient-sampling methods, since we found their performance to be inferior to the methods listed above.

To explore the effect of the different active-set identification mechanisms, we propose two generalizations of (6) as test problems for nonsmooth optimization. The two problems are defined for even values of n; we call them `Myopic_Decoupled` and `Myopic_Coupled`. They are given as

$$\min_{x\in\mathbb{R}^n} \sum_{i\in\{1,3,\cdots,n-1\}} |x_i - x_{i+1}| + (x_i + 0.1x_{i+1})^2, \tag{23}$$

and

$$\min_{x\in\mathbb{R}^n} \sum_{i=1}^{n-1} |x_i - x_{i+1}| + (x_i + 0.1x_{i+1})^2, \tag{24}$$

respectively; the bound constraints for these problems are discussed below. The attributes `Decoupled` and `Coupled` refer to whether or not the problem is separable.

In addition, we use several test problems from the literature that are listed in Table 2 along with their references[1]. Since these test problems are for unconstrained optimization, we follow an approach similar to

[1]The objective function for problem 20, suggested by Michael Overton [32], is $\max\left\{|x_1|, \max_{i\in\{2,3,\cdots,n\}} |x_{i-1} - x_i|\right\}$.

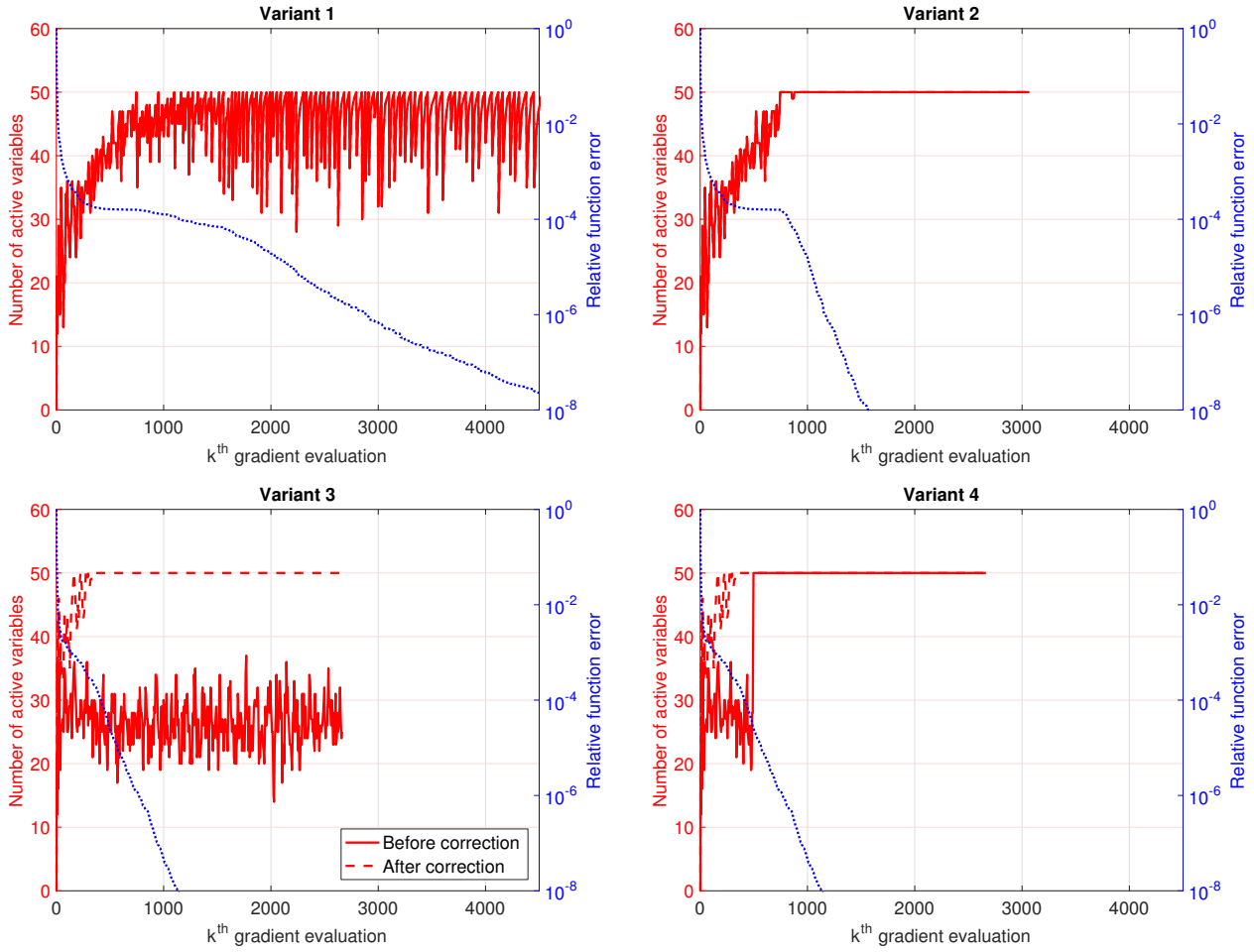

Figure 2: A comparison of the four variants of the algorithm on the `Myopic_Decoupled` problem.

[22] in that we add the following bounds to all problems:

$$l_i = \begin{cases} [x^\star_{\text{uncon}}]_i - 5.5 & \text{if} \quad \text{mod } (i,2) = 0 \\ -100 & \text{if} \quad \text{mod } (i,2) = 1 \end{cases}$$

$$u_i = \begin{cases} [x^\star_{\text{uncon}}]_i - 0.5 & \text{if} \quad \text{mod } (i,2) = 0 \\ 100 & \text{if} \quad \text{mod } (i,2) = 1 \end{cases}$$

where $x^\star_{\text{uncon}}$ is the unconstrained global minimizer which is known for all problems in closed form. By construction, for all problems, the unconstrained minimizer lies outside the bounds. For each of the 25 problems, ten starting points were generated randomly via a uniform distribution $U(-2, 2)$ centered at the midpoint of the bounds, giving rise to a total of 250 instances. Each code was run until the number of gradient evaluations exceeded $100n$ or an error occurred.

3.2 Effect of Active-Set Prediction and Correction Mechanism

We begin by investigating the efficacy of the active-set prediction, the correction mechanism, and their interplay, using the four variants given in Section 2.6.

Let us first consider the `Myopic_Decoupled` and `Myopic_Coupled` problems with $n = 100$ for one particular starting point. These problems are designed specifically to highlight the failure of gradient-based active-set prediction strategies. Figures 2 and 3 detail the behavior of the algorithm over the course of

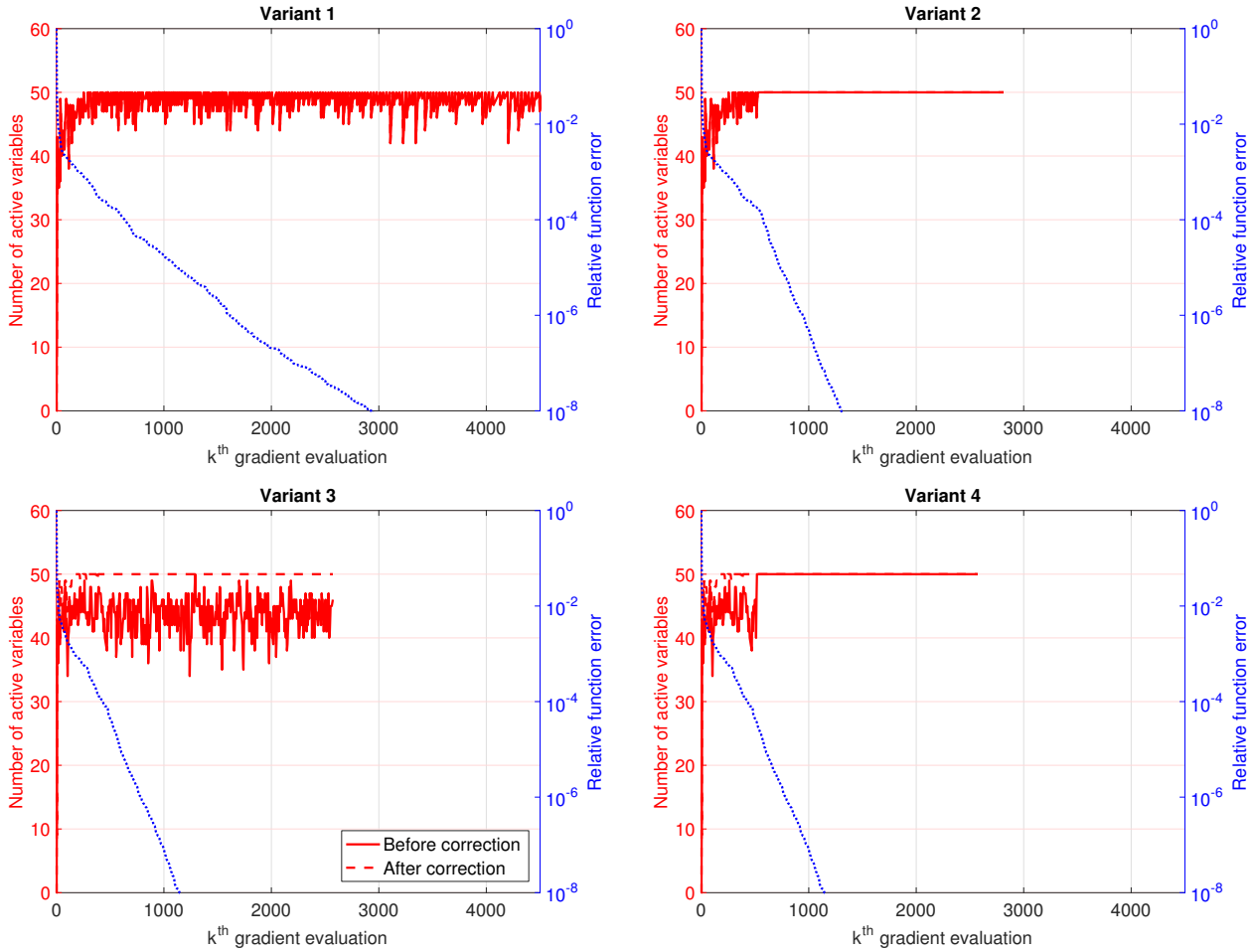

Figure 3: A comparison of the four variants of the algorithm on the `Myopic_Coupled` problem.

the optimization. In each plot, the dotted blue line depicts the progress in objective function, measured as $(f(x^k) - f(x^\star))/(f(x^0) - f(x^\star))$, where $x^\star$ is the optimal solution. The solid red line gives the size of the (initial) active set $\mathcal{A}^k$. For the variants that employ the active set correction strategy, the dashed red line gives the size of the active set at the end of Algorithm 1.

As one might expect, Variant 1, which uses the myopic gradient and no correction strategy, fails to identify the optimal active set (which contains 50 variables) and its guess $\mathcal{A}^k$ keeps fluctuating. The reduction in the objective function is significantly slower compared to the other variants. When the subgradient approximation is used in Variant 2, the objective function decreases faster, and after a certain number of iterations, the active set settles to the optimal active set. When the correction strategy is used in Variants 3 and 4, the optimal active set is identified more quickly, and the reduction in the objective is even faster. In these experiments, there is very little difference in performance between the two initializations of $\mathcal{A}^{\text{init}}$ in Algorithm 1. We observe similar behavior for larger dimensions of this problem. Further, rapid fluctuations in the active set are also seen in the other algorithms (L-BFGS-B, L-BFGS-B-NS and LMBM-B) which also employ gradient-based identification strategies.

Next we assess the relative performance of the different variants for the entire set of 250 instances with $n = 100$. Figure 4 presents Dolan-Moré performance profiles [13] with respect to the number of gradient evaluations. These profiles rely on a condition to determine when a run is deemed converged. For this purpose, given a tolerance $\epsilon > 0$, we use the test

$$\frac{f(x^k) - f^\star}{f(x^0) - f^\star} < \epsilon \tag{25}$$

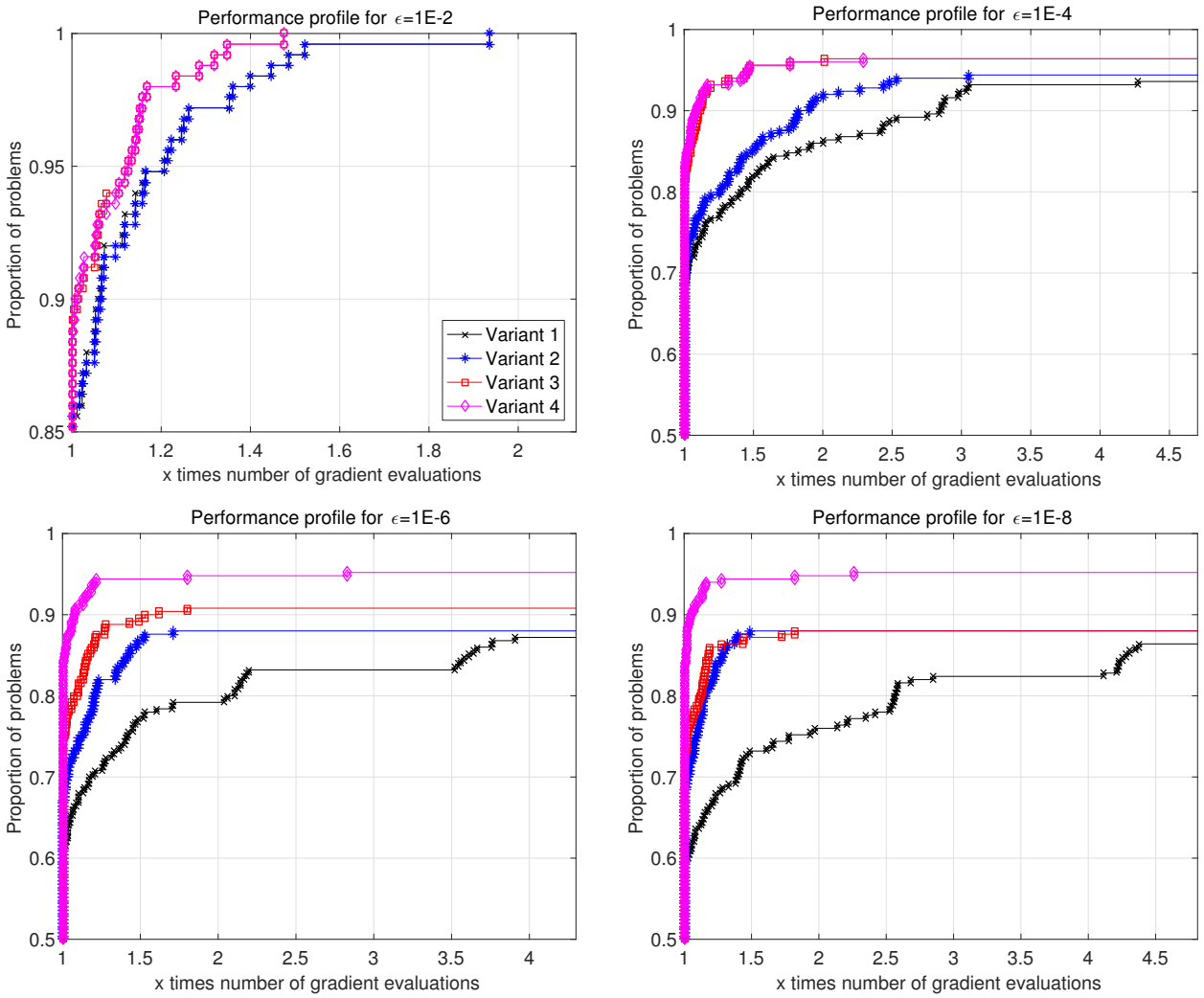

Figure 4: Dolan-Moré performance profiles comparing the four variants of the algorithm on 250 test problems for $\epsilon = 10^{-2}, 10^{-4}, 10^{-6}$ and $\epsilon = 10^{-8}$.

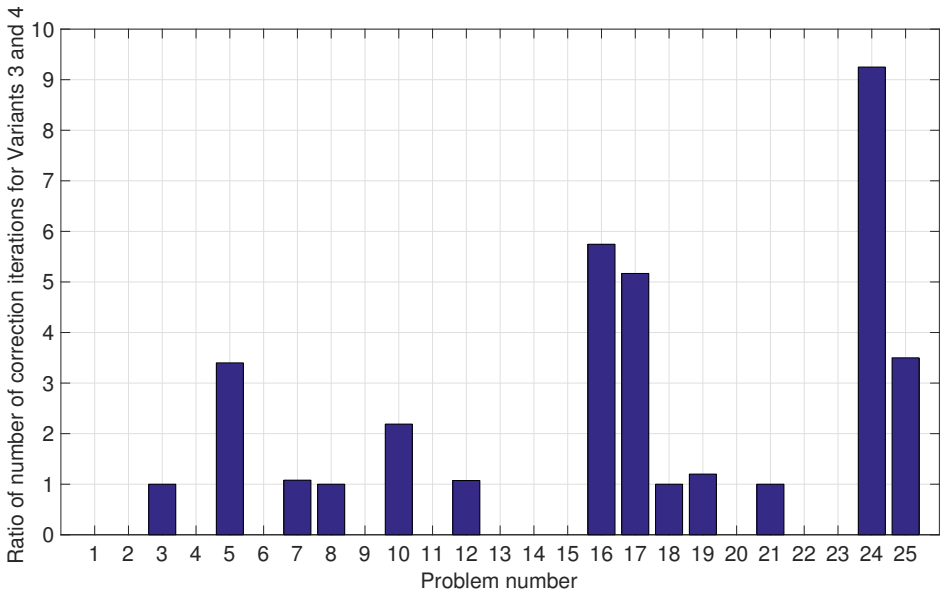

Figure 5: Average ratio of number of corrections for Variants 3 and 4 for all problems for $\epsilon = 10^{-4}$ and $n = 100$. A ratio of 0 indicates that both methods did not need any corrections.

where $f^\star$ is the best value found by any of the methods for the same instance. We present plots for four values of ϵ viz., $10^{-2}, 10^{-4}$, 10^{-6} and 10^{-8}.

This experiment reveals results similar to those found in Figures 2 and 3. Variant 1 shows the worst behavior, and the use of the subgradient approximation in Variant 2 improves the convergence rate. Using the corrective strategy gives the best performance, with an advantage for Variant 4 when a very tight tolerance is used.

Variants 3 and 4 incur different computational costs per iteration in Algorithm 3. In addition to the cost of the corrective loop, Variant 4 relies on the solution of the quadratic program (9) for the computation of the subgradient approximation. Figures 2 and 3 suggest that Variant 4 might require fewer iterations in the corrective loop in Algorithm 1 than Variant 3, since its initial active set $\mathcal{A}^{\text{init}}$ is a better guess of the final active set returned by the correction procedure. In Figure 5, we present the average ratio of the number of correction iterations for Variants 3 over 4. Indeed, Variant 3 needs up to 10 times as many correction iterations as Variant 4 to achieve similar performance.

Nevertheless, since the solution of the quadratic program (9) comes at a significant computational cost, we used Variant 3 for the remaining experiments. Also, Variant 3 is consistent with Lemma 2.1, so that we would encounter a zero step from the correction loop only when the current iterate is already stationary.

3.3 Comparison with Other Methods

We now compare NQN with L-BFGS-B, L-BFGS-B-NS, and LMBM-B on the 250 test instances. Figures 6, 7, and 8 correspond to three sets of experiments, with $n = 100$, $n = 1000$, and $n = 10000$, respectively. We present performance profiles for two values of ϵ, viz. 10^{-2} and 10^{-4}. We do not report experiments for $\epsilon = 10^{-6}$ and $\epsilon = 10^{-8}$ since the relative error is based on the best function value obtained by any of the methods. Tighter tolerances for ϵ would magnify insignificant differences between methods when neither are very close to an optimal solution. It can be seen that the proposed algorithm performs better than the other methods across different tolerances ϵ and problem sizes n. The figures show that NQN is able to find a lower objective on more problems and requires fewer gradient evaluations. This difference is particularly pronounced for tight tolerances and large problem sizes.

In Table 3, we summarize the occurrence for failures of the various methods for tolerances of $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$. The flag `OK` indicates that the termination criterion was satisfied at some iteration, `MAX` corresponds to reaching maximum number of gradient evaluations, and `OTHER` implies other failures which

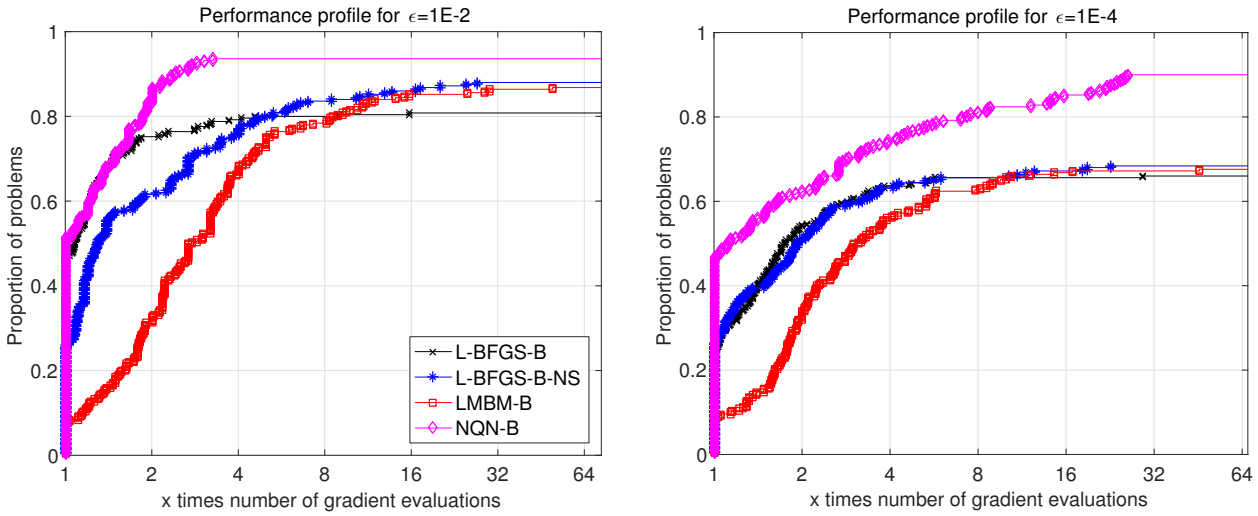

Figure 6: Dolan-Moré performance profiles of gradient evaluations for 250 test problems for $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$ with $n = 100$.

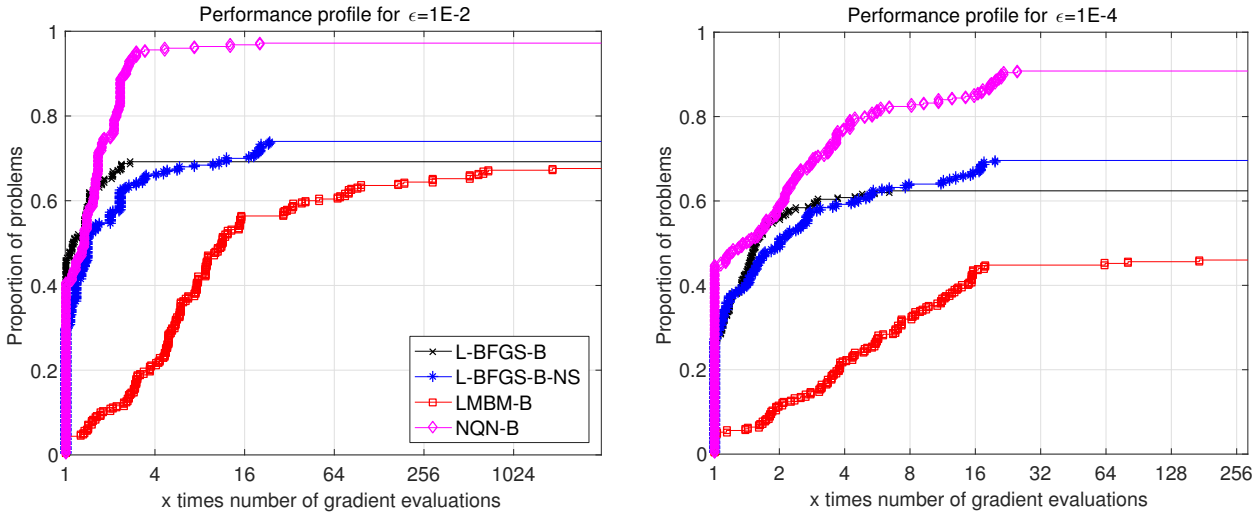

Figure 7: Dolan-Moré performance profiles of gradient evaluations for 250 test problems for $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$ with $n = 1000$.

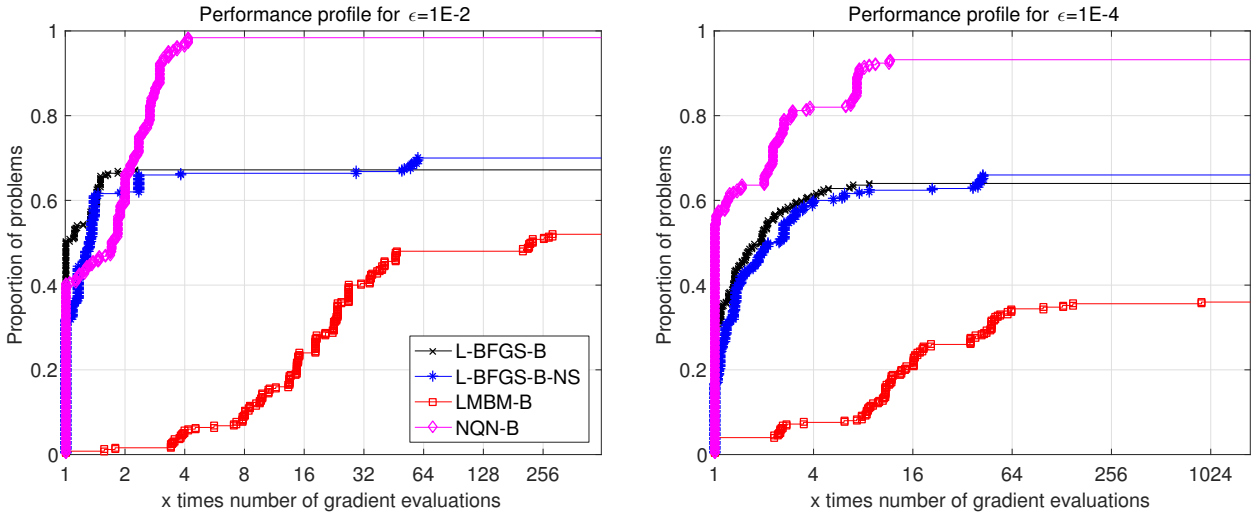

Figure 8: Dolan-Moré performance profiles of gradient evaluations for 250 test problems for $\epsilon = 10^{-2}$ and $\epsilon = 10^{-4}$ with $n = 10000$.

include solver-specific causes such as spurious termination of the line search, numerical issues, or convergence to a non-stationary point. For NQN we break down the number of `OTHER` failures into convergence to a point with no feasible direction in step 6 of Algorithm 2 (first number) and line search failure in step 28 of Algorithm 2 (second number).

As can be seen from Table 3, failures for NQN are more often due to budget exhaustion rather than another type of failure. In total, there were 10 instances in which numerical issues led to a bad search direction. A line search error was observed only once. The cause for budget exhaustion in NQN is, in part, due to the tight tolerance of ϵ_{abs}; close to a solution, the bracketing procedure takes many iterations in order to find points providing sufficient function decrease. Most of the large number of failures for L-BFGS-B occur due to a breakdown in the line search. This is to be expected since L-BFGS-B employs a strong Wolfe line search which is difficult to be satisfied with a nonsmooth objective. When the weak Wolfe line search is used in L-BFGS-B-NS instead, the number of line search failures is reduced significantly. Nevertheless, the overall number of successfully solved problems increases only marginally. LMBM-B is the least robust method, with a noticeable increase in the failure rate as the problem size grows.

Acknowledgements

The first author was partially supported by Office of Naval Research grant N00014-14-1-0313. The second author was partially supported by National Science Foundation grant DMS-1522747. The authors are grateful to Jorge Nocedal and Michael Overton for their insightful comments.

References

[1] M. S. Andersen, J. Dahl, and L. Vandenberghe. CVXOPT: A Python package for convex optimization, version 1.1.8. *Available at cvxopt.org*, 2013.

[2] A. Bagirov, N. Karmitsa, and M. M. Mäkelä. *Introduction to nonsmooth optimization: Theory, practice and software*. Springer, 2014.

[3] D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on control and Optimization*, 20(2):221–246, 1982.

[4] 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.

Flag	OK	MAX	OTHER	OK	MAX	OTHER
	$\epsilon = 10^{-2}$			$\epsilon = 10^{-4}$		
$n = 100$						
L-BFGS-B	202	0	48	165	0	85
L-BFGS-B-NS	220	25	5	171	58	21
LMBM-B	217	16	17	169	64	17
NQN	234	12	4 + 0	225	17	7 + 1
$n = 1000$						
L-BFGS-B	173	4	73	156	8	86
L-BFGS-B-NS	185	61	4	174	68	8
LMBM-B	169	11	70	115	54	81
NQN	243	6	1 + 0	227	22	1 + 0
$n = 10000$						
L-BFGS-B	168	17	65	160	21	69
L-BFGS-B-NS	175	73	2	165	82	3
LMBM-B	130	30	90	90	70	90
NQN	246	2	2 + 0	233	15	2 + 0

Table 3: Number of outcomes with different termination messages.

[5] R. H. Byrd, G. M. Chin, J. Nocedal, and F. Oztoprak. A family of second-order methods for convex ℓ_1-regularized optimization. *Mathematical Programming*, 159(1):435–467, 2016.

[6] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208, 1995.

[7] R. H. Byrd, J. Nocedal, and R. B. Schnabel. Representations of quasi-Newton matrices and their use in limited memory methods. *Mathematical Programming*, 63(1):129–156, 1994.

[8] F. H. Clarke. *Optimization and nonsmooth analysis*, volume 5. SIAM, 1990.

[9] F. E. Curtis, Z. Han, and D. P. Robinson. A globally convergent primal-dual active-set framework for large-scale convex quadratic optimization. *Computational Optimization and Applications*, 60(2):311–341, 2015.

[10] F. E. Curtis and M. L. Overton. A sequential quadratic programming algorithm for nonconvex, nonsmooth constrained optimization. *SIAM Journal on Optimization*, 22(2):474–500, 2012.

[11] F. E. Curtis and X. Que. An adaptive gradient sampling algorithm for non-smooth optimization. *Optimization Methods and Software*, 28(6):1302–1324, 2013.

[12] F. E. Curtis and X. Que. A quasi-Newton algorithm for nonconvex, nonsmooth optimization with global convergence guarantees. *Mathematical Programming Computation*, 7(4):399–428, 2015.

[13] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[14] M. W. Ferry. *Projected-search methods for box-constrained optimization*. PhD thesis, Department of Mathematics, University of California at San Diego, 2011.

[15] M. Gürbüzbalaban and M. L. Overton. On Nesterovs nonsmooth Chebyshev–Rosenbrock functions. *Nonlinear Analysis: Theory, Methods and Applications*, 75(3):1282–1289, 2012.

[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] N. Haarala, K. Miettinen, and M. M. Mäkelä. Globally convergent limited memory bundle method for large-scale nonsmooth optimization. *Mathematical Programming*, 109(1):181–205, 2007.

[18] W. Henao. An L-BFGS-B-NS optimizer for non-smooth functions. Master's thesis, Courant Institute of Mathematical Science, New York University, 2014.

[19] P. Hungerländer and F. Rendl. A feasible active set method for strictly convex quadratic problems with simple bounds. *SIAM Journal on Optimization*, 25(3):1633–1659, 2015.

[20] A. Kaku. Implementation of high precision arithmetic in the BFGS method for nonsmooth optimization. Master's thesis, Courant Institute of Mathematical Science, New York University, 2011.

[21] N. Karmitsa, A. Bagirov, and M. M. Mäkelä. Comparing different nonsmooth minimization methods and software. *Optimization Methods and Software*, 27(1):131–153, 2012.

[22] N. Karmitsa and M. M. Mäkelä. Adaptive limited memory bundle method for bound constrained large-scale nonsmooth optimization. *Optimization*, 59(6):945–962, 2010.

[23] N. Karmitsa and M. M. Mäkelä. Limited memory bundle method for large bound constrained nonsmooth optimization: Convergence analysis. *Optimization Methods and Software*, 25(6):895–916, 2010.

[24] N. Keskar, J. Nocedal, F. Öztoprak, and A. Wächter. A second-order method for convex ℓ_1-regularized optimization with active-set prediction. *Optimization Methods and Software*, 31(3):605–621, 2016.

[25] K. C. Kiwiel. Convergence of the gradient sampling algorithm for nonsmooth nonconvex optimization. *SIAM Journal on Optimization*, 18(2):379–388, 2007.

[26] A. S. Lewis and M. L. Overton. Nonsmooth optimization via quasi-Newton methods. *Mathematical Programming*, 141(1):135–163, 2013.

[27] A. S. Lewis and S. Zhang. Nonsmoothness and a variable metric method. *Journal of Optimization Theory and Applications*, 165(1):151–171, 2015.

[28] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[29] L. Lukšan, M. Tuma, J. Vlcek, N. Ramešová, M. Šiška, J. Hartman, and C. Matonoha. UFO 2004 - interactive system for universal functional optimization. Technical Report 1218, Institute of Computer Science, Academy of Science of the Czech Republic, 2014.

[30] M. M. Mäkelä. Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software*, 17:1, 2002.

[31] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, New York, 2nd edition, 2006.

[32] M. Overton. private communication, 02/23/2016.

[33] A. Skajaa. Limited memory BFGS for nonsmooth optimization. Master's thesis, Courant Institute of Mathematical Science, New York University, 2010.

[34] The Theano Development Team, R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, et al. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*, 2016.

[35] S. Walt, S. C. Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13(2):22–30, 2011.