Bulletin of the Iranian Mathematical Society Vol. 37 *No. 1* (*2011*), *pp 171-198.*

AN EFFECTIVE OPTIMIZATION ALGORITHM FOR LOCALLY NONCONVEX LIPSCHITZ FUNCTIONS BASED ON MOLLIFIER SUBGRADIENTS

N. MAHDAVI-AMIRI* AND R. YOUSEFPOUR

Communicated by Francis Clarke

Abstract. We present an effective algorithm for minimization of locally nonconvex Lipschitz functions based on mollifier functions approximating the Clarke generalized gradient. To this aim, first we approximate the Clarke generalized gradient by mollifier subgradients. To construct this approximation, we use a set of averaged functions gradients. Then, we show that the convex hull of this set serves as a good approximation for the Clarke generalized gradient. Using this approximation of the Clarke generalized gradient, we establish an algorithm for minimization of locally Lipschitz functions. Based on mollifier subgradient approximation, we propose a dynamic algorithm for finding a direction satisfying the Armijo condition without needing many subgradient evaluations. We prove that the search direction procedure terminates after finitely many iterations and show how to reduce the objective function value in the obtained search direction. We also prove that the first order optimality conditions are satisfied for any accumulation point of the sequence constructed by the algorithm. Finally, we implement our algorithm with MATLAB codes and approximate averaged functions gradients by the Monte-Carlo method. The numerical results show that our algorithm is effectively more efficient and also more robust than the GS algorithm, currently perceived to be a competitive algorithm for minimization of nonconvex Lipschitz functions.

MSC(2010): Primary: 90C26; Secondary: 47N10, 49J52.

Keywords: Locally nonconvex Lipschitz function, mollifier subgradient, Monte-Carlo method, Clarke generalized gradient, dynamic algorithm.

Received: 8 October 2009, Accepted: 23 December 2009.

* Corresponding author

© 2011 Iranian Mathematical Society.

Archive of SID

www.SID.ir

1. Introduction

Some nonsmooth optimization algorithms find the search direction using the Clarke generalized gradient. Most of these algorithms can be efficient only for certain types of functions; see [2, 9, 12, 13] for examples. Some modifications of these algorithms have been applied to nonconvex and general types of locally Lipschitz functions [12, 16, 18], but the search directions obtained are not usually sharp enough. The method for finding a search direction is the main difference among these algorithms. We can classify these algorithms in two main classes. The first class uses a single element of the Clarke generalized gradient or a modification of it as the search direction [18]. The algorithms in this class are simple, but their search directions are not very sharp and at times are not even a descent direction. Thus, these algorithms converge very slowly. The second class chooses a minimal element of an approximation of the Clarke generalized gradient as the search direction [2, 9, 12, 3, 16]. The approximation is the convex hull of some subdifferentials in a neighborhood of the current estimate of the solution. This search direction is often sharper than the search direction in the first class, but computing the direction is very time consuming. Hence, the algorithms in the second class can not usually be applied to high dimensional problems effectively.

Nonsmooth minimization algorithms using the Clarke generalized gradient for finding a search direction has a common drawback of needing an explicit formula for derivative [5] or at least one element of the Clarke generalized gradient [9, 12, 13], so that, with the absence of this requirement, they can not be applied to general types of functions. Thus, an efficient nonsmooth algorithm should have two important properties:

- An efficient method for approximating the Clarke generalized gradient of an arbitrary function with an acceptable error.
- An algorithm for finding a sharp enough descent direction without the need to compute too many subdifferentials.

A simple and practical method for approximating the Clarke generalized gradient makes use of the convex hull of some elements of the Clarke generalized gradient or of their approximations. To improve upon this type of the Clarke generalized gradient approximation, the elements of the Clarke generalized gradient should be selected such that their convex hull covers as much space of the Clarke generalized gradient as possible. Here, to achieve this improvement, we use the mollifier subgradients.

Archive of SID

www.SID.ir

Mollifier subderivative is a set of the upper limits of averaged functions gradient [8]. By convolution of the function f with a smooth class of mollifier functions, $\{\phi_\nu\}$, $\nu \in (0,1)$, a new class of smooth functions, f_ν, named averaged functions, is created [8, 19]. If f is locally Lipschitz, then the mollifier subderivative is a subset of the Clarke generalized gradient and its convex hull equals the Clarke generalized gradient. Usually, the convex hull of some elements of mollifier subderivative covers more space of the Clarke generalized gradient (set) than the convex hull of same number of the Clarke generalized gradient elements selected randomly. To approximate the Clarke generalized gradient, we use the mollifier subgradients. For this, we construct a set of averaged functions gradient and then show that for the locally Lipschitz case, the convex hull of this approximation is an approximation of the Clarke generalized gradient. In numerical experiments, we use y as a perturbation of the current estimate x and approximate $\nabla f_\nu(y)$ by the Monte-Carlo method, in place of $\nabla f_\nu(x)$ for small enough $\nu \in (0,1)$. By selecting an appropriate size of the Monte-Carlo method and a close perturbation of x and a small enough ν, we can find an approximation with an acceptable error.

For an efficient computation of the search direction, instead of approximating the complete Clarke generalized gradient, we give a dynamic algorithm for computing the search direction. In this algorithm, at each iteration we approximate one element of the mollifier subgradient and improve upon the approximation. Then, based on this approximation of mollifier subgradient (the minimum norm of the approximation), a search direction is constructed. If this search direction reduces the objective function sufficiently, then the descent direction is decided by the algorithm. Otherwise, in this direction, we perturb x and compute another approximation of mollifier subgradient element. We show that for locally Lipschitz functions this algorithm finds a descent direction after finitely many iterations. The numerical experiments show that we compute sharp descent directions while not needing to compute too many subgradients. We find a sharp descent direction by setting a sharp condition for the objective function reduction. Finally, using this descent direction, a minimization algorithm for locally Lipschitz functions is constructed.

We apply our algorithm to the test functions in [5], and compare our results with the ones obtained by the GS algorithm in [5]. The GS algorithm uses sampling gradient method [3] for approximating the

Archive of SID

www.SID.ir

Clarke generalized gradient. If n is the size of the problem, then in the GS algorithm, first the the Clarke generalized gradient is approximated by $2n$ sampling gradients and then the search direction is determined. Some test functions are locally nonconvex Lipschitz and others are not. The results show that our algorithm finds a minimizer in significantly smaller number of iterations and smaller averaged number of computed subdifferentials than the ones required by the GS algorithm.

Therefore, our proposed algorithm has two main merits as compared to other nonsmooth optimization algorithms, since

- we approximate the averaged functions gradient by the Monte-Carlo method, showing that the convex hull of the approximation is a good approximation of the Clarke generalized gradient, and
- the dynamic algorithm finds sharp enough descent directions without needing too many subdifferential evaluations.

In Section 2, we give a short review of mollifier function and its application to nonsmooth analysis. In Section 3, we use the mollifier subgradient to approximate the Clarke generalized gradient. First, we approximate the mollifier subgradient by Monte-Carlo method and then show that a collection of these approximations would be a good approximation for the Clarke generalized gradient. In Section 4, we use the approximation to establish an algorithm for finding descent directions. We prove that this algorithm finds the decent direction after finitely many iterations satisfying the Armijo condition. Using this decent direction, we develop a minimization algorithm and prove that any accumulation point of the sequence constructed by the algorithm satisfies the first order optimality conditions. Section 5 presents the numerical results. Conclusions are given in Section 6.

1.1. **Notations.** Let $\mathbb{R}$ be the real number line, N be the set of positive integer numbers, and $S_r(x) = \{y \in \mathbb{R}^n : \|x - y\|_2 \leq r\}$. We make use of co for convex hull and $\overline{co}$ for closure of convex hull throughout the paper. We denote the Hausdorff distance between the subsets $A, B \subseteq \mathbb{R}^n$ by:

$$d_H(A, B) = \max \left\{ \sup_{u \in A} \inf_{v \in B} \|u - v\|_2, \sup_{v \in B} \inf_{u \in A} \|u - v\|_2 \right\}.$$

Archive of SID

www.SID.ir

2. Smooth Approximation

A smooth approximation of a nonsmooth function f is constructed by convolution of the function with a smooth mollifier function. This new function is named as averaged functions. Here, we give a short review of mollifier functions and their applications to approximation of nonsmooth functions. We first define the mollifier function and then use it to define the mollifier subgradient and show its relation with the Clarke generalized gradient [8].

2.1. **Mollifier function.** We first define the averaged functions.

Definition 2.1. [8] *Given a locally integrable function $f : \mathbb{R}^n \to \mathbb{R}$ and a family of bounded mollifier functions $\{\psi_\nu : \mathbb{R}^n \to \mathbb{R}_+,\ \nu \in \mathbb{R}_+\}$ that satisfy*

$$\int_{\mathbb{R}^n} \psi_\nu(z)dz = 1,\ \mathsf{supp}\psi_\nu := \{z \in \mathbb{R}^n :\ \psi_\nu(z) > 0\} \subseteq \rho_\nu S_1(0)$$
$$\text{with } \rho_\nu \downarrow 0 \text{ as } \nu \downarrow 0,$$

the associated family $\{f_\nu,\ \nu \in \mathbb{R}\}$ of averaged functions is given by

$$f_\nu(x) := \int_{\mathbb{R}^n} f(x-z)\psi_\nu(z)dz = \int_{\mathbb{R}^n} f(z)\psi_\nu(x-z)dz.$$

A popular approach for constructing a family of mollifier functions makes use of the density functions. Let ψ be a density function such that $\mathsf{supp}\psi$ is bounded and $\alpha_\nu \downarrow 0$ as $\nu \downarrow 0$. Then, with this density function and the sequence, the family of mollifier functions is constructed by the following formula [8]:

$$\psi_\nu(z) = \frac{\psi(z/\alpha_\nu)}{(\alpha_\nu)^n}.$$

In fact, we can express the averaged functions f_ν by the following convolution formula:

$$f_\nu = f * \psi_\nu.$$

The following theorem establishes the conditions, under which the averaged functions converge to f.

Theorem 2.2. [8] *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuous and $\{f_\nu, \nu \in R_+\}$ be an associated family of averaged functions. Then, the averaged functions f_ν converge continuously to f, that is, $f_\nu(x_\nu) \to f(x)$, for all $x_\nu \to x$. In*

Archive of SID

www.SID.ir

fact, the averaged functions f_ν converge uniformly to f on every bounded subset of $\mathbb{R}^n$.

Here, we need the uniformly convergent property that the averaged functions converge to a continuous function. Weaker convergence conditions of averaged functions were established in [8].

In our numerical experiments, we construct averaged functions by a density function, but these averaged functions are very sensitive with respect to the density function. The numerical results show that the Steklov (averaged) functions are more stable than other well known ones based on density functions. The Steklov function is defined next [10, 11, 14].

Definition 2.3. *Given a locally integrable function $f : \mathbb{R}^n \to \mathbb{R}$, the Steklov (averaged) functions are defined as:*

$$f_\alpha(x) = \int_{\mathbb{R}^n} f(x-z)\psi_\alpha(z)dz,\ \alpha > 0,$$

where,

$$\psi_\alpha(z) = \begin{cases} \frac{1}{\alpha^n} & \text{if } \max\limits_{i=1,\dots,m} |z_i| \le \frac{\alpha}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Equivalently,

$$f_\alpha(x) = \frac{1}{\alpha^n} \int_{x_1-\frac{\alpha}{2}}^{x_1+\frac{\alpha}{2}} dy_1 \dots \int_{x_n-\frac{\alpha}{2}}^{x_n+\frac{\alpha}{2}} dy_n f(y).$$

As stated before, for approximating the Clarke generalized gradient, we need to compute the gradient of averaged functions. Using the next theorem, we can compute the gradient of averaged functions.

Theorem 2.4. [17, 19] *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally integrable. If the mollifier ψ_θ is smooth (of class C^1), then the gradient of the associated averaged functions f_θ is:*

$$\nabla f_\theta(x) = \int_{\mathbb{R}^n} f(y)\nabla\psi_\theta(x-y)dy.$$

The following theorem gives a special formula for the gradients of the Steklov functions.

Archive of SID

www.SID.ir

Theorem 2.5. [10] *Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuous. Then, the Steklov function f_α is continuously differentiable, and its gradient is given by*

$$\begin{aligned}
&\nabla f_\alpha(x) \\
&= \sum_{i=1}^{n} e_i \frac{1}{\alpha^{n-1}} \\
&\qquad \int_{x_1-\alpha/2}^{x_1+\alpha/2} dy_1 \ldots \int_{x_{i-1}-\alpha/2}^{x_{i-1}+\alpha/2} dy_{i-1} \int_{x_{i+1}-\alpha/2}^{x_{i+1}+\alpha/2} dy_{i+1} \ldots \int_{x_n-\alpha/2}^{x_n+\alpha/2} dy_n \\
&= \frac{1}{\alpha} \Big[f(y_1, \ldots, y_{i-1}, x_i + \frac{1}{2}\alpha, y_{i+1}, \ldots, y_n) - \\
&\qquad\qquad f(y_1, \ldots, y_{i-1}, x_i - \frac{1}{2}\alpha, y_{i+1}, \ldots, y_n) \Big],
\end{aligned}$$

where, e_i is the ith unit coordinate vector. This gradient can also be expressed as:

(2.1)

$$\nabla f_\alpha(x) = \sum_{i=1}^{n} e_i \int_{-\frac{1}{2}}^{\frac{1}{2}} d\xi_1 \ldots \int_{-\frac{1}{2}}^{\frac{1}{2}} d\xi_{i-1} \int_{-\frac{1}{2}}^{\frac{1}{2}} d\xi_{i+1} \ldots \int_{-\frac{1}{2}}^{\frac{1}{2}} d\xi_n \lambda_\alpha^i(x, \xi),$$

where,

(2.2)

$$\begin{aligned}
&\lambda_\alpha^i(x, \xi) = \\
&\tfrac{1}{\alpha} \big[f(x_1 + \alpha\xi_1, \ldots, x_{i-1} + \alpha\xi_{i-1}, x_i + \tfrac{1}{2}\alpha, x_{i+1} + \alpha\xi_{i+1}, \ldots, x_n + \alpha\xi_n) \\
&- f(x_1 + \alpha\xi_1, \ldots, x_{i-1} + \alpha\xi_{i-1}, x_i - \tfrac{1}{2}\alpha, x_{i+1} + \alpha\xi_{i+1}, \ldots, x_n + \alpha\xi_n) \big].
\end{aligned}$$

2.2. **Mollifier subgradient.** Here, we give the the definition of mollifier subgradient and its relationship to the Clarke generalized gradient [8].

Definition 2.6. [8] *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally integrable and let*

$$\{f_\nu := f_{\theta_\nu}, \nu \in N\}$$

be a sequence of averaged functions obtained from f by convolution with the sequence of mollifiers $\{\psi_\nu := \psi_{\theta_\nu} : \mathbb{R}^n \to \mathbb{R}_+, \nu \in N\}$, where, $\theta_\nu \downarrow 0$ as $\nu \to \infty$. Assume that the mollifiers are such that the averaged functions f_ν are smooth (of class C^1), as would be the case if the mollifiers ψ_ν are smooth. The subgradient set of Ψ-mollifier of f at x is:

$$\partial_\psi f(x) := \limsup_{\nu \to \infty} \{\nabla f_\nu(x_\nu) | x_\nu \to x\},$$

Archive of SID

www.SID.ir

that is, the cluster points of all possible sequences $\{\nabla f^{\nu}(x^{\nu})\}$ *such that* $x^{\nu} \to x$. *The full* Ψ*-subgradient set is:*

$$\partial_{\Psi} f(x) := \bigcup_{\psi} \partial_{\psi} f(x),$$

where, ψ *ranges over all possible sequences of mollifiers that generate smooth averaged functions.*

The following theorem establishes the relationship between mollifier subgradient and the Clarke generalized gradient.

Theorem 2.7. [8] *If* $f : \mathbb{R}^n \to \mathbb{R}$ *is lower semicontinuous and locally integrable, then*

$$co\ \partial_{\psi} f(x) \subseteq \partial_{\Psi} f(x) \subseteq \partial f(x),$$

where, $\partial f(x)$ *is the Clarke generalized gradient. If, in addition,* f *is locally Lipschitz, then*

$$co\ \partial_{\psi} f(x) = \partial_{\Psi} f(x) = \partial f(x).$$

Some nonsmooth optimization algorithms were constructed based on a mollifier subgradient approximation [8, 10, 11]. Our approach here, to be explained next, is to consider a convex hull of a number of mollifer subgradient approximations.

3. Approximation of the Clarke Generalized Gradient by Mollifier Subgradients

Here, we present a new approximation of the Clarke generalized gradient based on mollifier subgradient. We prove this approximation to be both outer and inner approximation for the Clarke generalized gradient. Finally, we show this approximation to be upper semicontinuous.

Suppose the averaged functions, f_{ν}, to be the convolution of f with a smooth family of mollifier functions. In numerical experiments, we use the Steklov functions. Now, we define the following set,

$$W(x, \nu_0, \lambda_0) = \{\nabla f_{\nu}(x + \lambda g)\ :\ \nu \in (0, \nu_0], \lambda \in (0, \lambda_0], g \in S_1(0)\}.$$

We show that, for some $\nu, \lambda \in (0, 1)$, $co\ W(x, \nu, \lambda)$ is a good approximation for the Clarke generalized gradient. Now, consider the following set,

$$W(x) = \bigcap_{\substack{\nu \geq 0 \\ \lambda \geq 0}} \overline{co}\, W(x, \nu, \lambda).$$

Archive of SID

www.SID.ir

Theorem 3.1. *If $f : \mathbb{R}^n \to \mathbb{R}$ is a locally Lipschitz function around x, then,*

$$\partial f(x) = W(x).$$

Proof. This is easily proved using Theorem 2.7. □

By the following propositions, we show that, $\overline{co}\, W(x, \nu, \lambda)$ can be a good approximation for the Clarke generalized gradient.

Proposition 3.2. *(Inner Approximation) Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function around x. Then, for each $\varepsilon > 0$, there exist $\lambda_0 \in (0, 1)$ and $\nu_0 \in (0, 1)$ such that*

$$\overline{co}\, W(x, \nu, \lambda) \subseteq \partial f(x) + \varepsilon S_1(0),$$

for all $\lambda \in (0, \lambda_0]$ and $\nu \in (0, \nu_0]$.

Proof. Suppose that the proposition is not true. Then there exists $\varepsilon > 0$ such that for all $n \in N$, there exist $\lambda_n, \nu_n \leq \frac{1}{n}$ and $g_n \in S_1(0)$, such that, we have,

$$\nabla f_{\nu_n}(x + \lambda_n g_n) \notin \partial f(x) + \varepsilon S_1(0). \tag{3.1}$$

On the other hand, $\{\nabla f_{\nu_n}(x + \lambda_n g_n)\}$ is a bounded sequence, and thus it has a convergent subsequence, say $\left\{\nabla f_{\nu_{n_k}}(x + \lambda_{n_k} g_{n_k})\right\}_{k=1}^{\infty}$. Let $\lim_{n_k \to \infty} \nabla f_{n_k}(x + \lambda_k g_k) = v$. By Theorem 2.7, we have $v \in \partial f(x)$, therefore there exists $K > 0$ such that for all $k \geq K$, $\nabla f_{\nu_{n_k}}(x + \lambda_{n_k} g_{n_k}) \in \partial f(x) + \varepsilon S_1(0)$, and this contradicts (3.1). □

Proposition 3.3. *(Outer Approximation) Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function around x. Then, for each $\varepsilon > 0$, there exist $\lambda_0 \in (0, 1)$ and $\nu_0 \in (0, 1)$ such that*

$$\partial f(x) \subseteq \overline{co}\, W(x, \nu, \lambda) + \varepsilon S_1(0),$$

for all $\lambda \in (0, \lambda_0]$ and $\nu \in (0, \nu_0)$.

Proof. Since $\partial f(x) \subseteq \overline{co}\, W(x, \nu, \lambda)$, then the proof is trivial. □

Archive of SID

www.SID.ir

Now, for each $\varepsilon > 0$, by Proposition 3.2, there exist $\lambda_\varepsilon \in (0,1)$ and $\nu_\varepsilon \in (0,1)$ such that

$$\overline{co}\, W(x,\nu,\lambda) \subseteq \partial f(x) + \varepsilon S_1(0), \tag{3.2}$$

for all $\lambda \in (0,\lambda_\varepsilon]$ and $\nu \in (0,\nu_\varepsilon]$. Define $C(x,\varepsilon) = \overline{co}\, W(x,\nu_\varepsilon,\lambda_\varepsilon)$, for $\varepsilon > 0$ and $C(x,0) = W(x)$. We show that $C(y,\varepsilon)$ is upper semicontinuous at $(x,0)$.

Proposition 3.4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz. Then, $C(y,\varepsilon)$ is upper semicontinuous at $(x,0)$.*

Proof. Let a sequence $\{(x_k,\varepsilon_k)\}_{k=1}^{\infty}$ be such that $(x_k,\varepsilon_k) \to (x,0)$ as $k \to \infty$. We show that if $v_k \in C(x_k,\varepsilon_k)$, for all k, and $v_k \to v$, then $v \in C(x,0)$. By equation (3.2), we have $v_k \in \partial f(x_k) + \varepsilon_k S_1(0)$. Thus, there exist $\zeta_k \in \partial f(x_k)$ and $g_k \in S_1(0)$ such that $v_k = \zeta_k + \varepsilon_k g_k$. So, we have

$$\lim_{k\to\infty} v_k = \lim_{k\to\infty} \zeta_k + \varepsilon_k g_k = \lim_{k\to\infty} \zeta_k = v.$$

On the other hand, since $\partial f(.)$ is upper semicontinuous, then we have $v = \lim_{k\to\infty} \zeta_k \in \partial f(x)$. Therefore, $v \in \partial f(x) = C(x,0)$ and this completes the proof. □

Corollary 3.5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz. Then, for each $\varepsilon > 0$, there exist $\delta > 0$, $\lambda_\varepsilon \in (0,1)$ and $\nu_\varepsilon \in (0,1)$ such that*

$$\overline{co}\, W(y,\nu,\lambda) \subseteq \partial f(x) + \varepsilon S_1(0),$$

for all $\lambda \in (0,\lambda_\varepsilon]$, $\nu \in (0,\nu_\varepsilon]$ and $y \in S_\delta(x)$.

Proof. The proof easily follows from upper semicontinuously of $C(y,\varepsilon)$ at $(x,0)$. □

4. **Minimization Algorithm**

Here, we develop an algorithm for solving

$$\min_{x\in\mathbb{R}^n} f(x), \tag{4.1}$$

where, f is a locally Lipschitz function. We first show how to compute a descent direction using $\overline{co}\, W(x,\nu,\lambda)$, as an approximation of $\partial f(x)$. In

Archive of SID

www.SID.ir

fact, instead of $\min_{v\in\partial f(x)} \|v\|_2$, we propose solving the following problem,

$$\begin{array}{ll} \min & \|v\|_2 \\ s.t. & \\ & v \in \overline{co}\, W(x,\nu,\lambda). \end{array} \tag{4.2}$$

But, solving (4.2) being impractical, we approximate $W(x,\nu,\lambda)$ by a finite number of its elements, and thus instead of (4.2), we solve the following problem,

$$\begin{array}{ll} \min & \|v\|_2 \\ s.t. & \\ & v \in W_k, \end{array} \tag{4.3}$$

where, W_k is the convex hull of k given points in $W(x,\nu,\lambda)$. Problem (4.3) is a quadratic optimization one, and there exist several efficient methods for solving it [7, 15, 21].

4.1. **Computing a descent direction.** We first show that the solution of (4.2) is a descent direction for f at x, for some $\nu \in (0,1)$ and $\lambda \in (0,1)$.

Proposition 4.1. *Let f be a locally Lipschitz function, $0 \notin \partial f(x)$ and $c \in (0,1)$. Then, there exist $\nu_0 \in (0,1)$ and $\lambda_0 \in (0,1)$ such that*

$$f(x+\lambda g) - f(x) \leq -c\lambda\|w\|_2,$$

for all $\nu \in (0,\nu_0]$ and $\lambda \in (0,\lambda_0]$, where,

$$w = \mathit{argmin}\left\{\|v\|_2 : v \in \overline{co}\, W(x,\nu,\lambda)\right\},$$

and $g = -\frac{w}{\|w\|_2}$.

Proof. By Proposition 3.2, there exist $\nu_0 \in (0,1)$ and $\lambda_0 \in (0,1)$ such that

$$\frac{\|w_0\|_2}{2} \leq \|w\|_2, \tag{4.4}$$

where, $w = \mathit{argmin}\{\|v\|_2 : v \in \overline{co}\, W(x,\nu,\lambda)\}$, for all $\nu \in (0,\nu_0]$, $\lambda \in (0,\lambda_0]$ and $w_0 = \mathit{argmin}\{\|v\|_2 : v \in \partial f(x)\}$. Select arbitrary $\lambda \in (0,\lambda_0]$ and $\nu \in (0,\nu_0]$. It now suffices to show that

$$f(x+\lambda g) - f(x) \leq -c\lambda\|w\|_2,$$

Archive of SID

www.SID.ir

where, $w = argmin\{\|v\|_2 : v \in \overline{co}\, W(x,\nu,\lambda)\}$ and $g = -\frac{w}{\|w\|_2}$. Set $\varepsilon = \frac{\lambda(1-c)}{2}\|w_0\|_2$. By Theorem 2.2, there exists $\nu_1 \in (0,\nu]$ such that

$$f(z) - f(y) \leq f_{\nu_1}(z) - f_{\nu_1}(y) + 2\varepsilon, \tag{4.5}$$

for all $y, z \in S_1(x)$. By the Mean Value Theorem and equation (4.5), we have

$$f(x+\lambda g) - f(x) \leq \lambda \nabla f_{\nu_1}(x + t\lambda g)^T g + 2\varepsilon,$$

for some $t \in (0,1)$. Since $t \in (0,1)$, then $\nabla f_{\nu_1}(x+t\lambda g) \in W(x,\nu,\lambda)$. On the other hand, $w = argmin\{\|v\|_2 : v \in \overline{co}\, W(x,\nu,\lambda)\}$ and $g = -\frac{w}{\|w\|_2}$. Therefore, by equation (4.4), we have

$$\begin{aligned} f(x+\lambda g) - f(x) &\leq -\lambda\|w\|_2 + 2\varepsilon \\ &\leq -c\lambda\|w\|_2. \end{aligned} \tag{4.6}$$

□

Next, we present a descent direction algorithm.

Algorithm 1: Descent Direction Algorithm (DDA).

Step 0: (Initialization)
Let $\gamma_\nu \in (0,1)$, $\nu_0 \in (0,1)$, $\nu_{\min} \in (0,1)$ $\gamma_\lambda \in (0,1)$, $\lambda_0 \in (0,1)$, $\eta_0 \in (0,1)$, $\alpha \in (0,1)$, $c \in (0,.2)$, $\delta > 0$, $g_0 \in S_1(0)$, $W_0 = \emptyset$, and $l = 0$.

Step 1: (Approximate the Clarke generalized gradient by mollifier subgradient)
Let $v_{l+1} = \nabla f_{\nu_l}(x + \lambda_l g_l)$ and $W_{l+1} = W_l \cup \{v_{l+1}\}$.

Step 2: (Compute a descent direction)
Solve the following minimization problem and let w_{l+1} be its solution:

$$\begin{aligned} &\min \ \|v\|_2 \\ &s.t. \\ &\quad v \in co\ W_{l+1}. \end{aligned}$$

If $\|w_{l+1}\|_2 \leq \delta$ **then stop else** let $g_{l+1} = -\frac{w_{l+1}}{\|w_{l+1}\|_2}$.

Step 3: (Stopping conditions)
If

$$f(x + \eta_l g_{l+1}) - f(x) \leq -c\eta_l \|w_{l+1}\|_2 \tag{4.7}$$

Archive of SID

www.SID.ir

then Stop.

Step 4: (Update parameters)
If $\nu_l < \nu_{\min}$ **then** let $\nu_{l+1} = \nu_0$, $\lambda_{l+1} = \lambda_0$, $\eta_{l+1} = \eta_l \times \alpha$, $l = l+1$ **else** let $\nu_{l+1} = \nu_l \times \gamma_\nu$, $\lambda_{l+1} = \lambda_l \times \gamma_\lambda$, $\eta_{l+1} = \eta_l$, $l = l+1$. **Go to Step 1**.

In DDA, at each iteration, an approximation of mollifier subgradient is improved upon by adding a new element of mollifier subgradient, and hence the approximation of the Clarke generalized gradient is improved. We show that DDA is well defined by proving that the algorithm terminates after finitely many iterations. The DDA's parameters are fixed for each test function at the start, and in each iteration of the minimization algorithm these parameters are updated.

Proposition 4.2. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function and $x \in \mathbb{R}^n$ be such that $0 \notin \partial f(x)$. Then, there exist $\lambda_0 \in (0,1)$ and $\nu_0 \in (0,1)$ such that DDA, using x, λ_0, ν_0 and other parameters arbitrary, terminates after finitely many iterations.*

Proof. Since $f_\nu(.)$ is continuously differentiable, then there exists $\bar{t} > 0$ such that

(4.8)
$$\|\nabla f_{\nu_l}(x) - \nabla f_{\nu_l}(x+tg)\|_2 \leq \frac{\|w_0\|_2}{8}, \ \forall t \in (0,\bar{t}),\ l = 0,1,2,\ldots, \forall g \in S_1(0),$$

where, $\|w_0\|_2 = \min_{v \in \partial f(x)} \|v\|_2$ and ν_l is defined as in DDA. Now, let $K \in N$ be such that $\eta_K \leq \bar{t}$. We show that if $\|w_{l+1}\|_2 > \delta$, for $l = 1,\ldots,K$, then the condition (4.7) will be satisfied for $l = K$. By Proposition 3.2, there exist $\nu_1 \in (0,1)$ and $\lambda_1 \in (0,1)$ such that

$$\frac{\|w_0\|_2}{2} \leq \|w\|_2, \tag{4.9}$$

where, $w = \mathit{argmin}\{\|v\|_2 : v \in co\ W(x,\nu,\lambda)\}$, for all $\nu \in (0,\nu_1]$, and $\lambda \in (0,\lambda_1]$. Let $\varepsilon = c\eta_K \frac{\|w_0\|_2}{4}$ and $\lambda_0 = \min\{\bar{t}, \lambda_1\}$. By Theorem 2.2, there exists $\nu_2 \in (0,1)$ such that

$$|f_\nu(y) - f(y)| \leq \varepsilon, \tag{4.10}$$

for all $\nu \in (0,\nu_2]$ and $y \in S_1(x)$. Let $\nu_0 = \min\{\nu_1, \nu_2\}$. Since

$$\|w_{l+1}\|_2 = \min\{\|v\|_2 :\ v \in conW_{l+1}\},$$

Archive of SID

www.SID.ir

then we have

$$v_1^T g_{l+1} \leq -\|w_{l+1}\|_2 \leq -\frac{\|w_0\|_2}{2}, \tag{4.11}$$

for $l = 0, 1, \ldots,$ where, $v_1 = \nabla f_{\nu_0}(x + \lambda_0 g_0)$. By the Mean Value Theorem, we have

$$f_{\nu_0}(x + \eta_K g_{K+1}) - f_{\nu_0}(x) = \eta_K \nabla f_{\nu_0}(x + \beta\eta_K g_{K+1})^T g_{K+1}, \tag{4.12}$$

for some $\beta \in (0, 1)$. By (4.8), (4.11), (4.12) and since $c \in (0, 0.2)$, we have

$$\begin{aligned} f_{\nu_0}(x + \eta_K g_{K+1}) - f_{\nu_0}(x) &\leq \eta_K \nabla f_{\nu_0}(x + \lambda_0 g_0)^T g_{K+1} + \eta_K \frac{\|w_0\|_2}{4} \\ &\leq \eta_K v_1^T g_{K+1} + \eta_K (1 - \frac{3}{2}c) \frac{\|w_0\|_2}{2} \\ &\leq \frac{3}{2} c \eta_K v_1^T g_{K+1}, \end{aligned} \tag{4.13}$$

for all $l = 0, \ldots,$ and $t \in (0, \bar{t}]$. Now, by (4.10), (4.11) and (4.13), we have

$$\begin{aligned} f(x + \eta_K g_{K+1}) - f(x) &\leq f_{\nu_0}(x + \eta_K g_{K+1}) - f_{\nu_0}(x) + 2\varepsilon \\ &\leq \frac{3}{2} c \eta_K v_1^T g_{K+1}^T + 2\varepsilon \\ &\leq -\frac{3}{2} c \eta_K \|w_{K+1}\|_2 + c\eta_K \frac{\|w_0\|_2}{2} \\ &\leq -c\eta_K \|w_{K+1}\|_2, \end{aligned}$$

and this completes the proof. □

Therefore, DDA terminates after finitely many iterations and at termination, one of the condition (4.7) or $\|w_{l+1}\|_2 \leq \delta$ is satisfied.

Corollary 4.3. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz on a compact set $X \subset \mathbb{R}^n$. Then, there exist $\lambda_0 \in (0, 1)$, $\nu_0 \in (0, 1)$ and $K \in N$ such that, for each $x \in X$, $\eta_0 \in (0, 1)$ and $\alpha \in (0, 1)$, DDA terminates after at most K iterations, by using λ_0, ν_0 and other parameters arbitrary.*

Proof. This is easily established using the proof of Proposition 4.2. □

Now, we show that if the stopping conditions are not satisfied in iteration l, then the new mollifier subgradient improves the approximation

Archive of SID

www.SID.ir

of $\partial f(x)$; i.e., we have $\nabla f(x+\lambda_{l+1}g_{l+1}) \notin W_{l+1}$. To achieve this aim, we prove the following proposition.

Proposition 4.4. *Let $f:\mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function and $x \in \mathbb{R}^n$ be such that $0 \notin \partial f(x)$. Then, there exist $\eta_0 \in (0,1)$, $\lambda_0 \in (0,1)$ and $\nu_0 \in (0,1)$ such that if DDA uses x, λ_0, η_0, ν_0 and other parameters arbitrary and the stopping conditions in DDA are not satisfied at iteration l, then $\nabla f(x+\lambda_{l+1}g_{l+1}) \notin W_{l+1}$.*

Proof. Consider all parameters as defined in Proposition 4.2 and let $\eta_0 \le \bar{t}$. If at iteration $l < K$, the condition (4.7) is not satisfied, then

$$-c\eta_l\|w_{l+1}\|_2 < f(x+\eta_l g_{l+1}) - f(x),$$

and by (4.10) and the Mean-Value Theorem, we have

$$\begin{aligned} -c\eta_l\|w_{l+1}\|_2 &< f_{\nu_{l+1}}(x+\eta_l g_{l+1}) - f_{\nu_{l+1}}(x) + 2\varepsilon \\ &\le \eta_l \nabla f_{\nu_{l+1}}(x+\theta\eta_l g_{l+1})^T g_{l+1} + 2\varepsilon, \end{aligned}$$

for some $\theta \in (0,1)$. Since $\theta\eta_l, \lambda_{l+1} \le \bar{t}$, then by (4.8), we have

$$\begin{aligned} -c\eta_l\|w_{l+1}\|_2 &< \eta_l \nabla f_{\nu_{l+1}}(x+\lambda_{l+1}g_{l+1})^T g_{l+1} + 2\varepsilon + \eta_l \frac{\|w_{l+1}\|_2}{4} \\ &\le \eta_l v_{l+1} g_{l+1} + c\eta_K \frac{\|w_{l+1}\|_2}{2} + \eta_l \frac{\|w_{l+1}\|_2}{4} \\ &\le \eta_l v_{l+1}^T g_{l+1} + \eta_l c' \|w_{l+1}\|_2, \end{aligned}$$

where, $c' = \frac{c}{2} + \frac{1}{4}$. Thus, we have

$$v_{l+1}^T w_{l+1} < (c+c')\|w_{l+1}\|_2^2 < \|w_{l+1}\|_2^2. \tag{4.14}$$

On the other hand, since $w_{l+1} = \mathit{argmin}\{v : v \in co\ W_{l+1}\}$, then

$$v^T w_{l+1} \ge \|w_{l+1}\|_2^2,\ \forall v \in co\ W_{k+1}.$$

Therefore, by (4.14), we have $v_{l+1} \notin W_{l+1}$ and this completes the proof. □

In our numerical testing, we approximate $\nabla f_\nu(.)$ by Monte-Carlo method and describe the details in Section 5.

Archive of SID

www.SID.ir

TABLE 1. The parameters of MA

k	Current iteration
λ_0^k	Perturbation parameter
s	Current inner iteration
γ_λ	Reduction parameter for perturbation parameter
x_k^s	Current point in inner iteration s
W_k^s	Current approximation for the Clarke generalized gradient
γ_k	Initial step size
w_s^k	The minimum point of W_k^s
α	Reduction parameter for initial step size
g_k^s	Current search direction in inner iteration s
c	the Armijo parameter
ν_1^k	Lower bound for averaged functions index
ν_0^k	Initial averaged functions index
γ_ν	Reduction parameter for averaged functions indices
δ_k	Update criteria
θ_λ	Reduction parameter for updating perturbation parameter

4.2. **Minimization algorithm and its convergence.** Now, we use the descent direction found in DDA to reduce f. The parameters of the minimization algorithm (MA) are as defined in Table 1. Algorithm 2 below stops when a stopping condition is satisfied; we explain this condition in Section 5, where we discuss our numerical experiments.

Algorithm 2: Minimization Algorithm (MA).

Step 0: (Initialization)
Let $x_1 \in \mathbb{R}^n$ and set $k = 1$.

Step 1: (Set new parameters)
Set $s = 1$ and $x_k^s = x_k$.

Step 2: (Descent direction)
Apply DDA with point x_k^s, parameters $\delta = \delta_k$, $(\nu_0, \nu_{\min}) = (\nu_0^k, \nu_{\min}^k)$, $\lambda_0 = \lambda_0^k$ and $\eta_0 = \gamma_k$. Let n_k^s be the number of iterations needed for termination of DDA, and let $\|w_k^s\|_2 = \min\left\{\|w\|_2 : w \in conW_{n_k^s+1}^s\right\}$. **If** $\|w_k^s\|_2 = 0$ **then Stop else** let $g_s^k = -\frac{w_k^s}{\|w_k^s\|_2}$ be the descent direction.

Step 3: (Line search)
If the stopping condition (4.7) is satisfied **then** compute the line

Archive of SID

www.SID.ir

search parameter, that is, the solution of the following problem,

$$\sigma = argmin\left\{f(x_k^s + \sigma g_k^s) : \sigma > 0\right\},$$

construct the next iteration $x_k^{s+1} = x_k^s + \sigma g_k^s$, set $s = s + 1$, and **go to Step 2**
else (Update parameters)
{we must have $\|w_k^s\|_2 \leq \delta_k$}
set $\lambda_0^{k+1} = \lambda_0^k \times \theta_\lambda$, $(\nu_0^{k+1}, \nu_{\min}^{k+1}) = (\nu_0^k, \nu_{\min}^k) \times \theta_\nu$, $\delta_{k+1} = \delta_k \times \theta_\delta$, $\gamma_{k+1} = \gamma_k \times \theta_\gamma$, $x_{k+1} = x_k^s$, $k = k + 1$ and **go to Step 1**.

Step 4: (Stopping condition)
If $\frac{\lambda_0^k}{\lambda_{\min}^k} < 10^{-5}$ **then stop else go to Step 1**.

In MA, we have two loops: The first one is the main loop, controlled by k, that counts the number of occurrences of parameter updating during the running of the algorithm and the second one is the inner loop, controlled by s, that counts the number of DDA calls in each step of the main loop. The following theorem shows when the stopping criterion (Step 4) of MA is disregarded, then every accumulation point of the sequence $\{x_k\}$, generated by MA, belongs to the set

$$X = \left\{x \in \mathbb{R}^n : 0 \in \partial f(x)\right\}.$$

Theorem 4.5. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a locally Lipschitz function and for the starting point $x_1 \in \mathbb{R}^n$, the level set $M = \{x : f(x) \leq f(x_1)\}$ be bounded. Suppose that MA does not terminate after a finite number of iterations and the stopping criterion (Step 4) of MA is disregarded. Then, every accumulation point of the sequence $\{x_k\}$, generated by MA, belongs to the set*

$$X = \left\{x \in \mathbb{R}^n : 0 \in \partial f(x)\right\}.$$

Proof. Since M is bounded and f is locally Lipschitz, then we have

$$f^* = \inf\left\{f(x) : x \in \mathbb{R}^n\right\} > -\infty.$$

First, we show that after finitely many iterations, the condition $\|w_k^s\|_2 \leq \delta_k$ will be satisfied. At iteration k, DDA starts with $\eta_0 = \gamma_k$, and by Corollary 4.3, there exists $J_k \in N$ such that DDA terminates after at most J_k iterations and we set $\eta_k^{J_k} = \gamma_k \times \alpha^{J_k}$ and $\eta_k^{n_k^s} = \gamma_k \times \alpha^{n_k^s}$, where n_k^s is the number of iterations needed for the termination of DDA with

Archive of SID

www.SID.ir

x_k^s. It is obvious that $\gamma_k^{J_k} \leq \gamma_k^{n_k^s}$, for all s. Suppose that the condition (4.7) is satisfied. Then, we have

$$f(x_k^{s+1}) - f(x_k^s) \leq f(x_k^s + \gamma_k^{n_k^s} g_k^s) - f(x_k^s) \leq -c\gamma_k^{n_k^s} \|w_k^s\|_2 < 0. \tag{4.15}$$

Therefore, $f(x_k^{s+1}) \leq f(x_k^s)$, for $s = 1, 2, \ldots$. Thus, there exists s_k such that

$$f(x_k^s) - f(x_k^{s+1}) \leq c\gamma_k^{J_k}\delta_k, \tag{4.16}$$

for all $s \geq s_k$, since if this equation is not satisfied, then we have $\lim_{s\to\infty} f(x_k^s) = -\infty$, and this contradicts $f(x_k^s) \geq f^* > -\infty$. Thus, by (4.15) and (4.16), we have

$$\|w_k^s\|_2 \leq \delta_k,$$

for all $s \geq s_k$. Hence, after finitely many iterations, there exists s_k such that $x_{k+1} = x_k^{s_k}$ and

$$\min\left\{\|v\|_2 : v \in conW_{n_k^{s_k}+1}^{s_k}\right\} \leq \delta_k. \tag{4.17}$$

Since $\{x_k\} \subseteq M$ and M is bounded, then $\{x_k\}$ has an accumulation point, namely x^*, and there is a subsequence $\{x_{k_i}\}$ such that $x_{k_i} \to x^*$, as $k_i \to \infty$. For each $\varepsilon > 0$, by Corollary 3.5, there exists $K > 0$ such that

$$\overline{con}W(x_{k_i}, \nu_0^{k_i}, \lambda_0^{k_i}) \subseteq \partial f(x^*) + S_\varepsilon(0), \tag{4.18}$$

for all $k_i \geq K$. On the other hand, we have,

$$conW_{n_{k_i}^{s_{k_i}}+1}^{s_{k_i}} \subseteq \overline{co}\, W(x_{k_i}, \nu_0^{k_i}, \lambda_0^{k_i}). \tag{4.19}$$

Now, by (4.17), (4.18) and (4.19), we have

$$\|w^*\|_2 = \min\{\|v\|_2 : \ v \in \partial f(x^*)\} \leq \delta_{k_i} + \varepsilon.$$

Since $\delta_{k_i} \to 0$, as $k_i \to \infty$, and ε is an arbitrary number, then we have.

$$\|w^*\|_2 = 0.$$

□

Therefore, $0 \in \partial f(x^*)$ and this completes the proof.

Archive of SID

www.SID.ir

5. **Numerical Experiments**

We apply our algorithm to test functions in [5] and compare the results with the ones obtained by the GS algorithm in [5]. We first describe how the parameters are selected.

(a) Averaged functions indices. Since $\nabla f_\nu(y)$ converges to one element of the Clarke generalized gradient at x, when $\nu \to 0$ and $y \to x$, then if y is close enough to x, for small enough values of ν, $\nabla f_\nu(y)$ can be considered as an approximation of the Clarke generalized gradient at x. To find a good approximation of the Clarke generalized gradient, an appropriate value for ν is very important. In our numerical experiments, we set $\nu_0 = 10^{-5}$, as an initial index value. With larger values of ν_0, for some easy problems, MA converged to the optimal value fast. For general problems, however, the algorithm approached approximate optimal values very fast, but without achieving high accuracies. On the other hand, ν_0 should not be chosen to be very small, since in that case, when the algorithm gets close to the optimal point, the norm of approximated mollifier subgradient reduces to zero fast, and therefore the algorithm fails to find an accurate enough optimal solution. Thus, we used a lower bound for the value of ν_1. Of course, the number of mollifier subgradients computed in each interaction of DDA and the sharpness of the descent direction depend on the value of the lower bound, ν_1. If we select ν_1 to be small, then the number of computed mollifier subgradients increases, while if we select ν_1 close to ν_0, then we lose the sharpness of the descent direction. We set $\nu_1 = 10^{-10}$. We also fix ν_0 and ν_1 in all iterations and do not update them, that is, we set $\theta_\nu = 1$. DDA starts with an initial value, ν_0, and this value is reduced in each iteration until reaching the lower bound, ν_1. In this situation, we reset λ_0 and ν_0 to their initial values and reduce the search direction step, λ_s.

(b) Perturbation parameters. As indicated before, we do not calculate $\nabla f_\nu(x)$ as an approximation of the subdifferential at x, but rather compute it at a perturbed x in some direction in the unit sphere. For the perturbation, we used the parameter λ_0 as the step size in a unit direction, g_0. We started with an initial value λ_0, and reduced this value in each iteration in DDA. The value $\nabla f_\nu(x+\lambda g)$ is also very sensitive with respect to λ_0. A large value of λ_0 slows down the DDA in finding the descent direction and the number of computed subdifferentials increases very fast. On the other hand, if λ_0 is very small, then the convergence

Archive of SID

www.SID.ir

to the optimal value slows down significantly. For low dimensional problems, we set $\lambda_0 = 0.1$, and set $\lambda_0 = 1$ for high dimensional problems. We update λ_0, by a reduction parameter γ_λ, reducing λ_0 in each iteration of DDA until ν_0 reaches its lower bound. Then, λ_0 is reset to its initial value. We set $\gamma_\lambda = 0.8$. When the minimum norm of mollifier subgradient approximation is less than the update criteria, using the reduction parameters, we reduce λ_0 in Step 3 of MA by the reduction factor θ_λ. Numerical experiments showed a good value for the reduction factor to be 0.1 and for achieving an accurate solution when $\lambda_0 = 10^{-5}$, we set $\theta_\lambda = 0.8$. We explain our rationale, when we discuss the *update criteria* in *(f)* below.

(c) Monte-Carlo dimension. The numerical experiments showed that the algorithm was not very sensitive to the size of m, and thus we selected $m = 1$. For the approximation of averaged functions gradient by Monte-Carlo method, we generated n random vectors, with $n-1$ components each, independent and uniformly distributed on $[-\frac{1}{2}, \frac{1}{2}]$, and then approximated each component of averaged functions gradient using (2.2).

(d) Reduction factor for averaged functions indices. To reduce the averaged functions indices in DDA, we used the parameter γ_ν. In numerical testing, we set $\gamma_\nu = 0.01$. This parameter shows how much the mollifier subgradient should be computed before resetting the parameters.

(e) Initial step size. λ_0^s is the initial step size for the descent direction. Regarding the proof of Proposition 4.2, since $\lambda_0^s \leq \lambda_0$, then we set $\lambda_0^s = \lambda_0$ and when ν_0 reaches its lower bound, we reduce its initial value by the reduction parameter α. We set $\alpha = 0.8$ in our numerical experiments.

(f) Update criteria. When the norm of the minimal point of the computed Clarke generalized gradient approximation set is small, the algorithm may not reduce the function sufficiently. In this case, considering the definition of $\lambda_\nu(x + \lambda_k g_k)$, we understand that λ_k is not small enough, and thus we must update this parameter. Hence, we apply DDA at the current point using the new initial value of λ_0 and with $\lambda_0^s = \lambda_0$, as described in *(b)* above. We set δ as an update criterion. This parameter affects the presumed amount of acceptable reduction of the descent direction. We set $\delta = 10^{-4}$ in all iterations in MA, that is, we set $\theta_\delta = 1$.

Archive of SID

www.SID.ir

(g) Line search method. We set the Armijo parameter as $c = 0.2$, and use the simple line search strategy,

$$\sigma = \inf \{ f(x_k^s + \sigma g_k^s) : \sigma > 0 \}.$$

We start with $\sigma = 1$ and backtrack with reduction factor $\gamma = 0.5$.

(h) Stopping criteria. We used the size of λ_0 as the stopping criterion. We do not use the norm of mollifier subgradient, because its size depends on the values of ν_0 and λ_0, and these values vary for different functions, and so it can not serve as a good stopping criterion. The algorithm terminates when $\frac{\lambda_0^k}{\lambda_0^1} < 10^{-5}$, where λ_0^1 is the initial value of λ_0 at Step 1 and λ_0^k is the initial value of λ_0 at iteration k.

Our algorithm having a stochastic behavior, we ran the algorithm for each test function 10 times until we found the optimal solution obtained by the GS in [5] or the stopping condition was satisfied. In all testings, we achieved the optimal value even before satisfying the stopping condition. Next, we applied our algorithm to solve various instances of several test functions. We observed that the algorithm was able to solve some large instances of the test functions, that could not be solved by the GS algorithm in a reasonable amount of time. We computed the average number of required iterations and the average number of subgradients evaluations over the 10 runs for each test function, denoting them by *Ave.Iter* and *Ave.Sub*, respectively. In order to have an indication of the variations in the number of required iterations and computed subgradients in any run of the algorithm for a test function, we computed the standard deviation of the number of required iterations and computed subgradients for each test function, denoting them by *Dev.Iter* and *Dev.Sub*. We also showed the best run of our algorithm as *B.Iter* and its computed subgradient as *B.Sub* to be compared with the best reported results of the GS algorithm in [5]. For the test functions that the GS algorithm could not solve, we computed the average and standard deviation of minimal values, showing these parameters by *Ave.Opt.Val* and *Dev.Opt.Val*. In our results, *GS.Iter* is the required number of iterations and *GS.Sub* is the number of computed gradient samplings needed in the GS algorithm, as reported in [5] (the *GS* algorithm needs $2n$ gradient samplings in each iteration, where n is the size of the problem).

The algorithm was implemented using MATLAB R2007b. The MATLAB codes are available in

http://mehr.sharif.edu/∼yosefpoor/archive/smalg.zip.

Archive of SID

www.SID.ir

TABLE 2. Results for exponential Chebyshev approximation, starting with $x = 0$.

n	Ave.Iter	B.Iter	Dev.Iter	Ave.Sub	B.Sub	Dev.Sub	GS.Iter	GS.Sub
2	18.4	14	2.67	50.7	36	7.83	42	168
4	41.5	33	3.47	177.9	154	13.16	63	504
6	87.4	75	12.02	527.1	450	99.37	166	1992
8	134.9	94	28.59	868.4	604	203.88	282	4512

5.1. **Chebyshev approximation by exponential sums.** Consider the following function,

$$f(x) = \sup_{s\in[l,u]} |h(s,x)|,$$

where, $[l,u] = [1,10]$ and $h(s,x) = \frac{1}{s} - \sum_{j=1}^{n/2} x_{2j-1}\exp(-x_{2j}s)$. We divided $[l,u] = [1,10]$ into 2000 grid points and approximated $\sup_{s\in[l,u]} |h(s,x)|$ using these 2000 grid points, and set $\lambda_0 = 0.1$. More details about this function are found in [5]. Table 2 shows the results obtained. Like the GS, our algorithm can not find an accurate solution for this problem, when $n > 8$.

5.2. **Minimization of eigenvalue products.** The aim is to minimize the product of the largest k eigenvalues of a Hadamard (componentwise) matrix product $A\circ X$, where A is a fixed positive semidefinite symmetric matrix and X is a variable symmetric matrix constrained to have ones on its diagonal and be positive semidefinite. We solved the following minimization problem,

$$\min f(x) = \prod_{j=1}^{k} \lambda_j(A\circ X) - \rho\min(0,\lambda_N(X)),$$

where λ_j stands for the jth largest eigenvalue and the $N\times N$ symmetric matrix X has ones on its diagonal and $n = N(N-1)/2$ variables from the vector x in its off-diagonal positions. We set $\rho = 100$. The function f is differentiable at a vector x corresponding to a matrix X if X is positive definite and $\lambda_k(A\circ X) > \lambda_{k+1}(A\circ X)$. The matrices A are the leading $N\times N$ submatrices of a specific 63×63 covariance data matrix arising from an environmental application [1]. More details about this problem are given in [5]. Our results are shown in Table 3.

Archive of SID

www.SID.ir

TABLE 3. Results for minimizing eigenvalue product, using random starting points.

n	N	Ave.Iter	B.Iter	Dev.Iter	Ave.Sub	B.Iter	Dev.Sub	GS.Iter	GS.Sub
1	2	7.2	6	1.75	54	53	1.83	10	20
6	4	66.7	56	8.26	365.1	311	26.10	68	816
15	6	60.1	56	3.07	347.1	341	31.11	150	4500
28	8	80.9	62	52.36	761.3	578	698.34	600	33600
45	10	260	221	20.74	2802.1	2421	287.20	227	20430
66	12	150.2	140	13.4	1840.8	1603	280.10	432	57024
91	14	419.4	377	30.43	9965.1	8142	1315.64	309	56238
120	16	355.2	289	39.63	9896.6	7147	1814.07	595	142800

TABLE 4. Results for minimization of eigenvalue product, using random starting points for $N \geq 20$.

n	N	Ave.Iter	Dev.Iter	Ave.Sub	Dev.Sub	Ave.Opt.Val	Dev.Opt.Val
190	20	235.3	28.72	4532.8	660.72	1.25470e-03	7.00200e-06
435	30	359	35.30	9331.2	984.30	2.27124e-04	4.06656e-06
780	40	19.6	0.70	110	5.14	8.81554e-05	9.23976e-06
1225	50	28	1.49	206	16.77	9.70365e-06	8.44910e-07
1770	60	35.7	0.95	323.6	21.46	6.38074e-07	6.26068e-08

We set $\lambda_0 = 1$ and solved this problem for higher dimensions, $N \geq 20$, where the GS algorithm failed to do so. The results are shown in Table 4.

5.3. **Spectral and pseudospectral minimization.** Consider the pseudospectral abscissa of a matrix, $\alpha_\delta(X)$, defined, for any given $\delta \geq 0$, as the maximum of the real parts of the δ-pseudospectrum of X, that is, the set of all z in the complex plane such that z is an eigenvalue of some complex matrix within a distance δ of X [20]. An algorithm for computing α_δ is given in [4]. We got its MATLAB code from http://www.cs.nyu.edu/faculty/overton/software/index.html.

We considered minimizing the following function,

$$f(x) = \alpha_\delta(X(x)), \tag{5.1}$$

where,

$$X(x) = \begin{bmatrix} -x_1 & 1 & 0 & . & . & 0 \\ x_1 & 0 & 1 & 0 & . & 0 \\ x_2 & 0 & . & . & . & . \\ . & . & . & . & . & 0 \\ . & . & . & . & . & 1 \\ x_n & 0 & . & . & . & 0 \end{bmatrix}, \tag{5.2}$$

Archive of SID

www.SID.ir

TABLE 5. Results for minimizing pseudospectral abscissa $\alpha_\delta(X(x))$ for $n = 4$ $(N = 5)$, starting with $x = 0$ (except for pure spectral abscissa case $\delta = 0$, started randomly).

δ	Ave.Iter	B.Iter	Dev.Iter	Ave.Sub	B.Iter	Dev.Sub	GS.Iter	GS.Sub
1	25.1	21	3.51	74.1	60	14.19	81	648
1.0e-001	36.8	24	13.27	114.4	79	34.70	105	840
1.0e-002	55.9	38	25.76	190.1	126	78.76	112	896
1.0e-003	102.2	80	15.65	455.8	370	64.75	163	1304
1.0e-004	202.4	175	23.85	745.2	643	84.99	236	1888
1.0e-005	223.1	184	29.26	865.3	721	100.70	322	2576
1.0e-006	200.4	143	51.04	788.9	552	225.35	403	3224
0	186.9	172	15.42	660.1	613	52.31	157	1256

TABLE 6. Results for minimizing pseudospectral abscissa $\alpha_\delta(X(x))$ for $n = 9$ $(N = 10)$, starting with $x = 0$ (except for pure spectral abscissa case $\delta = 0$, started randomly).

δ	Ave.Iter	Dev.Iter	Ave.Sub	Dev.Sub	Ave.Opt.Val	Dev.Opt.Val
1	5.80	1.75	20.00	4.29	1.97538e+00	1.05935e-08
1.00e-01	63.10	5.04	420.60	31.04	9.53867e-01	2.86390e-04
1.00e-02	237.75	88.25	1981.50	506.60	6.42977e-01	5.88233e-04
1.00e-03	515.50	225.64	2974.30	1147.52	1.64820e-01	1.50363e-03
1.00e-04	406.20	263.59	2411.20	1421.77	1.31202e-01	1.00716e-03
1.00e-05	990.90	413.89	5736.60	2314.74	1.20137e-01	6.75566e-04
1.00e-06	319.00	96.24	2142.80	577.63	1.17700e-01	1.36771e-03
0	283.30	35.25	1826.70	221.46	1.16136e-01	9.41200e-04

and n is the number of parameters, which is one less than the order of the matrix, say N. For $\delta = 0$, the global minimizer of f is zero and f is not Lipschitz at this point [5]. More details and the application of pseudospectral functions can be seen in [5]. Similar to [5], we applied our algorithm to f for $N = 5$ and set $\lambda_0 = 0.1$. The results are summarized in Table 5. We set $\lambda_0 = 1$ and solved the problem for $N = 10$, similar to the eigenvalue products minimization problem. We note that the GS algorithm was not able to solve this problem for this dimension. Our results are presented in Table 6.

5.4. **Maximization of distance to instability.** The distance to instability for a given matrix X is the least value δ such that some complex matrix Y within distance δ of x is not stable, and we denote it by $d_{inst}(X)$ [5]. We got the MATLAB code for computing $d_{inst}(X)$ from

http://www.cs.nyu.edu/faculty/overton/software/index.html.

We minimized $-d_{inst}\,(X(x) - sI)$ over x, given the parameter $s > 0$ and $X(x)$, as defined in (5.2). We used 0 as a starting point. We set $\lambda_0 = 0.1$ for $n = 4$ $(N = 5)$. The results are summarized in Table 7.

Archive of SID

www.SID.ir

TABLE 7. Results for maximizing distance to instability by minimizing $-f(x) = -d_{inst}(X(x)-sI)$ for $n = 4$ ($N = 5$), starting with $x = 0$.

s	Ave.Iter	B.Iter	Dev.Iter	Ave.Sub	B.Sub	Dev.Sub	GS.Iter	GS.Sub
1	37.9	30	10.69	110.1	89	64.08	55	440
3.16228e-01	37.6	32	10.50	156.6	99	43.85	71	568
1.00000e-01	87.5	60	64.48	351.5	235	97.90	110	880
3.16228e-02	125.4	69	35.18	539.4	301	161.71	141	1128

TABLE 8. Results for maximizing distance to instability by minimizing $-f(x) = -d_{inst}(X(x)-sI)$ for $n = 9$ ($N = 10$), starting with $x = 0$.

s	Ave.Iter	Dev.Iter	Ave.Sub	Dev.Sub	Ave.Opt.Val	Dev.Opt.Val
1	37.90	3.96	165.90	12.67	-2.54119e-01	2.16531e-04
3.16228e-01	390.20	44.29	1872.60	199.18	-6.37732e-03	9.84366e-05
1.00000e-01	1000.00	0.00	4800.00	256.82	-1.89869e-7	5.13051e-8
3.16228e-02	1500.00	0.00	5797.20	206.85	-3.28901e-11	1.41604e-11

The result on this problem is only available for $n = 4$ for the GS algorithm in [5]. We tried $n = 9$ with $\lambda_0 = 1$. The results are presented in Table 8. Since satisfying the stopping condition was very time consuming in the cases $s = 1.0e-1$ and $s = 3.16228e-2$, we limited the number of iterations, for the case $s = 1.0e-1$ to 1000 and scaled this problem by the factor 10^5, and for $s = 3.16228e-2$, we limited the algorithm to 1500 iterations and scaled the function by the factor 10^{10}. This function is very sensitive, and when the size of λ_0 is not small enough, the Monte-Carlo approximation of mollifier gradient turns to equal zero. So, to prevent this situation, in DDA, we multiplied λ_0 by the reduction factor θ_λ until the Monte-Carlo approximation would become nonzero, and while updating, if $\lambda_0 < \nu_0$, then we set $\nu_0 = \lambda_0$.

The numerical results show that our algorithm has two advantages, as compared to the GS algorithm. Firstly, in every case in our algorithm, the average number of iterations for finding the minimum point of the test function is less than the number of iterations needed by the GS algorithm (not to mention our better results). Also, the average number of computed subdifferentials in our algorithm for each function is much less than the number of computed gradient samplings in the GS algorithm. Having these properties, we applied our algorithm to problems of higher dimensions, for which the GS algorithm failed to produce solutions in a reasonable amount of time. In the numerical experiments, the computed standard deviation for the number of iterations and the

Archive of SID

www.SID.ir

number of computed subgradients are not large, and thus for each test function, the computing time is not expected to be large either. At the same time, the deviation in the optimal values is not large either, and hence we expect the algorithm to be reliable in finding a good estimate of the optimal point when applied to arbitrary functions.

Note. Here, we fixed the parameters of our algorithm to be the same in all of our testing. Of course, if one could adjust parameters according to the specifics of a problem, even better results can be attained.

6. Conclusions

We presented an algorithm for optimization of nonconvex Lipschitz functions. We first approximated the Clarke generalized gradient by mollifier subgradients, and then used the approximation to construct a dynamic algorithm for finding a descent direction. The minimization algorithm was constructed based on this approximation. We proved the convergence of our algorithm only requiring the function to be locally Lipschitz. To show the effectiveness of the algorithm, we implemented our algorithm and compared our results with the ones obtained by the GS algorithm as reported in [5]. The numerical results showed our algorithm to be more robust and significantly more efficient than the GS algorithm in the required number of iterations and the number of computed subgradients. In fact, a descent direction is computed dynamically, without needing too many iterations. This feature makes the algorithm practical for solving high dimensional problems.

Acknowledgments

We are grateful to Michael Overton for his explanation of spectral and pseudospectral minimization test functions. The authors also thank the Research Council of Sharif University of Technology for supporting this work.

Archive of SID

References

[1] K. Anstreicher and J. Lee, A masked spectral bound for maximum-entropy sampling, in MODA 7–Advances in Model-Oriented Design and Analysis, A. di Bucchianico, H. Lamuter, and H. P. Wynn, eds., Springer-Verlag, Berlin, (2004), pp. 1-12.

www.SID.ir

[2] A. M. Bagirov, Continuous subdifferential approximations and their applications, *J. Math. Sci. (N. Y.)* **115** (2003) 2567–2609.
[3] J. V. Burke, A. S. Lewis and M. L. Overton, Approximating subdifferentials by random sampling of gradients, *Math. Oper. Res.* **27** (2002) 567-584.
[4] J. V. Burke, A. S. Lewis and M. L. Overton, Robust stability and a criss-cross algorithm for pseudospectra, *IMA J. Numer. Anal.* **23** (2003) 359-375.
[5] J. V. Burke, A. S. Lewis and M. L. Overton, A robust gradient sampling algorithm for nonsmooth, nonconvex optimization, *SIAM J. Optim.* **15** (2005) 751-779.
[6] F. H. Clarke, *Optimization and Nonsmooth Analysis*, Classics in Applied Mathematics, Vol. 5, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990.
[7] V. A. Daugavet, Modification of the Wolfe method, *Zh. Vychisl. Mat. i Mat. Fiz.* **21** (1981) 504-508.
[8] Y. M. Ermoliev, V. I. Norkin and R. J.-B. Wets, The minimization of semicontinuous functions: mollifier subgradients, *SIAM J. Control Optim.* **33** (1995) 149-167.
[9] J. B. Hiriart-Urruty and C. Lemaréchal, *Convex Analysis and Minimization Algorithms II. Advanced Theory and Bundle Methods*, Springer-Verlag, Berlin, 1993.
[10] A. M. Gupal, A method for the minimization of almost differentiable functions,Translated from Kibernetika (Kiev) **1** (1977) 114-116, Cybernetics and System Analysis **13** (1977) 115-117.
[11] A. M. Gupal and V. I. Norkin, An algorithm for the minimization of discontnuous functions, Translated from Kibernetika (Kiev) **2** (1977) 73-75, Cybernetics and System Analysis **13** (1977) 220-223.
[12] K. C. Kiwiel, *Methods of Descent for Nondifferentiable Optimization*, Lecture Notes in Mathematics, 1133, Springer-Verlag, Berlin, 1985.
[13] M. M. Mäkelä and P. Neittaanmäki, *Nonsmooth Optimization*, World Scientific Publishing Co., Inc., River Edge, NJ, 1992.
[14] D. Q. Mayne and E. Polak, Nondifferential optimization via adaptive smoothing, *J. Optim. Theory Appl.* **43** (1984) 601-613.
[15] B. F. Mitchell, V. F. Demyanov and V. N. Malozemov, Finding the point of a polyhedron closest to the origin, *SIAM J. Control* **12** (1974) 19-26.
[16] H. Schramm and J. Zowe, A version of the bundle idea for minimizing a nonsmooth function: Conceptual idea, convergence analysis, numerical results, *SIAM J. Optim.* **2** (1992) 121-152.
[17] L. Schwartz, *Théorie des Distributions*, Hermann, Paris, 1966.
[18] N. Z. Shor, *Minimization Methods for Non-differentiable Functions*, Springer-Verlag, Berlin, New York, 1985.
[19] S. L. Sobolev, *Some Applications of Functional Analysis in Mathematical Physics*, Third edition, Nuaka, Moscow, 1988.
[20] L. N. Trefethen, Pseudospectra of linear operators, *SIAM Rev.* **39** (1997) 383-406.
[21] P. Wolfe, Finding the nearest point in a polytope, *Math. Programming* **11** (1976) 128-149.

Archive of SID

www.SID.ir

Nezam Mahdavi-Amiri
Faculty of Mathematical Sciences, Sharif University of Technology, Tehran, Iran
Email: nezamm@sharif.edu

Rohollah Yousefpour
Faculty of Mathematics, Mazandaran University, Babolsar, Iran
Email: r.yosefpor@gmail.com

Archive of SID

www.SID.ir