Draft

171240510 Yuheng Ma

DII, Nanjing University

Summer 2020

Abstract

The abstract goes here.

1 Introduction

Unconstraint composite optimization has been one of the most essential problems in non-smooth optimization. It is generally stated as follows:

$$\underset{x}{\text{minimize}}\, f(x) \triangleq \phi(x) + h(F(x)),\ x \in \mathbb{R}^n. \tag{1}$$

where $\phi : \mathbb{R}^n \to \mathbb{R}$ is smooth and sometimes 0 in convention. $F : \mathbb{R}^n \to \mathbb{R}^p$ is differentiable $h : \mathbb{R}^p \to \mathbb{R}$ is continuous but non-smooth and in most cases assumed to be nonconvex. Such problem often come with optimization of structured loss function, where $h \circ F$ performs as penalty in order to endow parameters desired structure. For instance, L^2 penalty, known as Tikhonov regularization, provides ridge regression and weight decay in neural network that decrease the scale of parameters. Lasso [25] providing sparsity comes from L^1 regularization and ElasticNet [26] comes from combination of two. There is also L^0 regularization which diminishes rank. When regularization is non-smooth, such as L^1 or L^∞ norm [10], the machine learning problems are exactly composite optimization.

Plenty of applications in control and signal processing involves composite optimization [5], such as distributed control system [2] and low complexity model [31]. Duchi and Feng [23] considered application in phase retrieval. For true signal $x^* \in \mathbb{R}^n$ and measurement vector $(a_1, \cdots, a_n)$, observation will be $b_i = \phi(< a_i, x^* >) + \varepsilon_i$, $i = 1, \cdots, n$ where ϕ is nonlinear. Ideally the retrieval of x^* is $min_x \frac{1}{n}\sum_{i=1}^n (\phi(\langle a_i, x\rangle) - b_i)^2$, which is unfortunately unpractical since observation is usually $b_i = \langle a_i, x^\star\rangle^2$. This leads to optimizing $\frac{1}{n}\sum_{i=1}^n \left|\langle a_i, x\rangle^2 - b_i\right|$, again a composite optimization.

Classical optimization also yield necessity of composite optimization. Consider a set of equations $F_i(x) = 0$, $i = 1, \cdots, n$. Solving this equation is equivalent to optimizing $||F(x)|| = ||(F_1, \cdots, F_n)||$, as $||\cdot||$ is any norm operator [14]. This is a typical composite optimization when the operator is non-smooth. Similarly, consider set of nonlinear inequality constraints $F_i(x) \le 0$, $i = 1, \cdots, n$. A feasible initial point can be obtained by solving $min\, \Sigma||F_i^+(x)||$, where $F_i^+(x) = max(F_i(x), 0)$.

Theory of composite optimization was first established by A. D. Ioffe [1], which has also given conditions for local minimum for sublinear h. J. V. Burke [27] relaxed restriction for h to convex. When F is Lipschitz and Gâteaux differentiable, Jeyakumar and X. Q. [30] provided first order Lagrange necessary condition and second order sufficient condition. For cases where ∂F is available, abundant analyses were provided [8, 9, 29].

In context of numerical algorithm, proximal gradient descent [4, 24] makes a great substitution after gradient descent fails, though it needs separability of non-smooth term with respect to each variable. By splitting component of smooth and non-smooth in f and impose equality constraint, problem is converted to constraint optimization and amounts of methods such as ADMM [21] become available. A consequent advantage is that distributed computation is now possible, as problem is decomposed into smooth subproblems or tractable non-smooth ones. However, there is no theoretical guarantees for ADMM when ϕ is non-convex. In opposite, method of multipliers [3] guarantees convergence in non-convex case as well as rigorous parameter tuning framework. But MM requires joint optimization of augmented Lagrange function with respect to all variables, which significantly limits computation speed. Dhingra and Khong [15] employ manifold obtained by minimizing variable with respect to non-smooth term, and project augmented Lagrange onto it. This gives continuously differentiable proximal augmented Lagrangian providing distributed computation, rigorous parameter tuning procedure and theoretical guarantees. Bundle methods [11,33,34] are a class of most commonly used algorithms in non-convex optimization, even when ∂F is unavailable (derivative free) [12,22].

However, in most cases, h is required to be some specific norm, such as L^2 [28] and L^∞ [19,29]. This naturally preclude non-convexity of h. Larson, Menickellly and Wild [18] developed method of **manifold sampling**, a trust-region framework estimating $f(x)$ utilizing estimation of $F(x)$ on nearby manifold and Clarke direction derivative of h. Though adopted $h = ||\cdot||_1$ as objective function, theoretical analysis and implementation required nothing about convexity. Meanwhile, the algorithm survive in derivative free case and surpass gradient sampling method [17,20] by nonstochasticness. Khan, Larson and Wild [16] extended this work by allowing h to be piecewise affine functions. Larson, Menickelly and Zhou [32] recently extend h to be any continuous function as long as it is Clarke sub-derivative is available, though require $\phi = 0$. To best of our knowledge, this is the only work for **any continuous** h and F with **unavailable Jacobian**.

For this graduate project, we expect to extend work of Larson, Menickelly and Zhou [32] to where ϕ is a smooth function. Theoretical results are expected to generalize properly, and a software package, ideally in Matlab/Python, is required. New simulations corresponding to the form should be done and, if possible, a real data application employ simply model (Lasso/ ElesticNet) will be done.

2 Preliminaries

2.1 Problem Setting

We tend to solve problem

$$\underset{x}{\text{minimize}}\, f(x) \triangleq \phi(x) + h(F(x)),\ x \in \mathbb{R}^n.$$

where $\phi : \mathbb{R}^n \to \mathbb{R}$ is smooth and second order Lipschitz continuous. $F : \mathbb{R}^n \to \mathbb{R}^p$ is differentiable, $h : \mathbb{R}^p \to \mathbb{R}$ is continuous but non-smooth and non-convex.

2.2 Notations

All norms in the project are assumed to be l^2 norm without specific notation. The closure of set S is denoted as $\mathbf{cl}(S)$. The interior of set S is denoted as $\mathbf{int}(S)$. The convex hull of set S is denoted as $\mathbf{co}(S)$. The image of set S through F is $\mathbf{Im}(S) =: \{F(x)|x \in S\}$. Balls $B(x,\delta)$ are assumed to be closed. The generalized Clarke subdifferential of a locally Lipschitz continuous function f at x is

$$\partial_C f(x) \triangleq \operatorname{co}\left(\left\{\lim_{y^j \to x} \nabla f\left(y^j\right) : y^j \in D\right\}\right)$$

where D is some neighborhood. We generally use k on footnote to represent variables in k^{th} iteration.

2.3 Definitions and Assumptions

Definition 2.1. For some constant δ_{max} and point x_0 in R^n, define

$$\mathscr{L}_{\max} \triangleq \bigcup_{x \in \mathscr{L}\left(x^0\right)} B\left(x; \Delta_{\max}\right)$$

where $\mathscr{L}\left(x^0\right)$ is some set we need. This definition will be used as domain of our consideration.

Assumption 2.2. We assume the following about f and F constraint on $\mathscr{L}_{max}$

- For any $x_0 \in R^n$, set $\mathscr{L}(x_0) =: \{x : f(x) \leq f(x_0)\}$ is bounded
- F_i is Lipschitz with constant L_{F_i} on $\mathscr{L}_{max}$
- ∇F_i is Lipschitz with constant $L_{\nabla F_i}$ on $\mathscr{L}_{max}$

Assumption 2.3. We assume the following about smooth function ϕ.

- ϕ and $\nabla\phi$ can be evaluated inexpensively
- For any $x_0 \in R^n$, set $\{x : \phi(x) \leq \phi(x_0)\}$ is bounded
- ϕ is Lipschitz with constant L_ϕ on $\mathscr{L}_{max}$
- $\nabla\phi$ is Lipschitz with constant $L_{\nabla\phi}$ on $\mathscr{L}_{max}$

Definition 2.4. A function h is a continuous selection on $U \subseteq R^p$ if it is continuous on U and

$$h(z) \in \left\{h_j(z) : h_j \in \mathfrak{H}\right\}, \quad \forall z \in U$$

where $\mathfrak{H}$ is a finite set of selection functions. For instance, $h(z) = ||z_i||_1$ is a continuous selection provided that it is continuous and $h(z) \in \{\sum \pm z_i\}$

Definition 2.5. A function h is a continuous selection. Then

$$S_j \triangleq \left\{z : h(z) = h_j(z)\right\}, \quad \tilde{S}_j \triangleq \mathrm{cl}\left(\mathrm{int}\left(S_i\right)\right), \quad \mathbb{A}(z) \triangleq \left\{j : z \in \tilde{S}_j\right\}$$

$\mathbb{A}(z)$ is active indices set at z. We refer any h_j for which $j \in \mathbb{A}(z)$ as active selection function for h at z. For any set Z, let $\mathbb{A}(\mathbb{Z}) = \bigcup_{z\in\mathbb{Z}} \mathbb{A}(z)$

Assumption 2.6. We assume the following about h.

- $\mathbb{A}(z)$ is computable for any z.
- h_i is Lipschitz with constant L_{h_i} on $\mathscr{L}_{max}$
- ∇h_i is Lipschitz with constant $h_{\nabla F_i}$ on $\mathscr{L}_{max}$

Definition 2.7. A function m^{F_i} is a fully linear model of F_i on $B(x,\Delta)$ if there exists constants $\kappa_{i,ef}$ and $\kappa_{i,eg}$ such that

$$\left|F_i(x+s) - m^{F_i}(x+s)\right| \leq \kappa_{i,\mathrm{ef}}\Delta^2, \;\; \forall s \in B(0;\Delta)$$
$$\left\|\nabla F_i(x+s) - \nabla m^{F_i}(x+s)\right\| \leq \kappa_{i,\mathrm{eg}}\Delta, \;\; \forall s \in B(0;\Delta)$$

Assumption 2.8. For $F = (F_1, \cdots, F_p)$ and its fully linear model $m^F = (m^{F_1}, \cdots, m^{F_p})$, m^F is twice continuously differentiable and Lipschitz continuous bounded by constant $\left\|\nabla^2 m^{F_i}(x)\right\| \leq \kappa_{i,\mathrm{mH}}$.

Definition 2.9. Here we define some constants to be used, mostly Lipschitz constants.

$$L_F \triangleq \sqrt{\textstyle\sum_{i=1}^p L_{F_i}^2} \quad L_{\nabla F} \triangleq \sqrt{\textstyle\sum_{i=1}^p L_{\nabla F_i}^2}, \quad L_h \triangleq \max_{j\in\{1,\ldots,|\mathfrak{H}|\}}\left\{L_{h_j}\right\},$$
$$\kappa_\mathrm{f} \triangleq \textstyle\sum_{i=1}^p \kappa_{i,\mathrm{ef}}, \quad \kappa_\mathrm{g} \triangleq \textstyle\sum_{i=1}^p \kappa_{i,\mathrm{eg}}, \quad \kappa_\mathrm{H} \triangleq \textstyle\sum_{i=1}^p \kappa_{i,\mathrm{mH}}$$
$$C \triangleq \left(2L_h\kappa_\mathrm{g} + 2L_{\nabla h}L_F^2 + L_hL_{\nabla F} + L_{\nabla\phi}\right)$$

Definition 2.10. Let $\mathbb{Z}^k$ be set of samples

$$\mathbb{Z}^k = \mathbf{co}\left(\left\{F(y) : y \in B(x^k, \Delta_k)\right\}\right)$$

That is, $\mathbb{Z}^k$ includes all points of the form $\alpha F\left(x^k\right) + (1-\alpha)F(y)$ for $\alpha \in [0,1]$. Thus, $\mathbb{Z}^k \subset \mathbf{co}\left(\mathrm{Im}\left(\mathscr{L}_{\mathrm{max}}\right)\right)$. Let $\mathbb{D}^k$ be the set of gradients of linearizations of selection functions.

$$\mathbb{D}^k = \left\{\nabla h_j(z) : z \in \mathbb{Z}^k, j \in \mathbb{A}(z)\right\}$$

Let $\mathbb{G}^k$ be the set of modeled gradients (or generator, as in [32]), of sample set $\mathbb{Z}^k$.

$$\mathbb{G}^k = \left\{\nabla M\left(x^k\right)\nabla h_j(z) + \nabla\phi : z \in \mathbb{Z}^k, j \in \mathbb{A}(z)\right\}$$

Let G^k and D^k are matrices with columns that are elements of corresponding set. Also, $G^k = \nabla M\left(x^k\right)D^k + (\nabla\phi, \cdots, \nabla\phi)$.

Definition 2.11. Define the approximated gradient g^k

$$g^k \triangleq \mathrm{proj}\left(0, \mathrm{co}\left(\mathbb{G}^k\right)\right) \in \mathrm{co}\left(\mathbb{G}^k\right) \tag{2}$$

which is the minimum-norm of element of $\mathbb{G}^k$

Assumption 2.12. For L_F in 2.9, we have

$$F\left(x^k\right) \subseteq \mathbb{Z}^k \subset \mathscr{B}\left(F\left(x^k\right); L_F\Delta_k\right)$$

3 Manifold Sampling

In this section, we describe manifold sampling algorithm we use. Intuitively, manifold sampling is a model-based derivative free optimization algorithm. Modeling of function F is done by m^F and active selection function h_j is utilized to model h. Thus,

$$\nabla f = \nabla \phi + \nabla h(F(x)) = \nabla \phi + \nabla F(x) \nabla h_j(F(x)) \approx \nabla \phi + \nabla m^F(x) \nabla h_j(F(x))$$

Thus by heuristics of gradient descent or trust region method, local minimum can be found. In 2.10, $\mathbb{Z}^k$ corresponds to derivative free model building part. $\mathbb{D}^k$ corresponds to possible ∇h. $\mathbb{G}^k$ corresponds to possible ∇f and g^k in 2.11 is final approximation of gradient. We now present some subproblem encountered in the algorithm.

3.1 Major Components

3.1.1 Derivative Free Model

To acquire model of F, we use model-based derivative free techniques. [6] utilize fully linearity and least square loss for model building. It worth noting that [6] requires only n+1 points to build model for n dimension input.

3.1.2 Master Model

To acquire projection $g^k \triangleq \operatorname{proj}\left(0, \operatorname{co}\left(\mathbb{G}^k\right)\right) \in \operatorname{co}\left(\mathbb{G}^k\right)$, we use convex quadratic optimization

$$\begin{array}{ll} \underset{\lambda}{\operatorname{minimize}} & \lambda^\top \left(G^k\right)^\top G^k \lambda \\ \text{subject to} & e^\top \lambda = 1, \lambda \geq 0 \end{array} \tag{3}$$

Solving 3 and we can set

$$g^k \triangleq G^k \lambda^* + \nabla \phi \quad \text{and} \quad d^k \triangleq D^k \lambda^* \tag{4}$$

Also, we can employ d^k as weights of smooth component m^{F_i}

$$m_k^f(x) \triangleq \sum_{i=1}^{p} \left[d^k\right]_i m^{F_i}(x) + \phi(x)$$

A simple observation will be

$$\nabla m_k^f\left(x^k\right) = \sum_{i=1}^{p} \left[d^k\right]_i \nabla m^{F_i}\left(x^k\right) + \nabla \phi(x^k) = \left(\nabla m^F\left(x^k\right) D^k + \nabla \phi\right) \lambda^* = G^k \lambda^* = g^k$$

3.1.3 Sufficient Decrease Condition

After model building, the master model can be locally optimized by trust-region methods.

$$\underset{s \in B(0;\Delta_k)}{\operatorname{minimize}} \quad m_k^f\left(x^k + s\right) \tag{5}$$

Exact solution is nor necessary. We provide an available sufficient decrease condition

$$\phi(x^k) - \phi(x^k + s^k) + \left\langle m^F\left(x^k\right) - m^F\left(x^k + s^k\right), d^k \right\rangle \geq \frac{\kappa_{\mathrm{d}}}{2} \left\|g^k\right\| \min\left\{\Delta_k, \frac{\|g^k\|}{L_h \kappa_{\mathrm{H}} + L_{\nabla\phi}}\right\} \tag{6}$$

Note that the sufficient decrease condition 6 differs from traditional conditions employed in classical trust-region methods. Instead of measuring the decrease in $m^f(x)$ and $m^f(x+s)$, the left-hand side measures the decrease in terms of the specific convex combination, defined by d^k, of the gradients of selection functions at points near $F(x^k)$

3.1.4 Manifold Sampling Loop

In 6, s^k suggested by local model based on $\mathbb{Z}^k$ may be defected if $\mathbb{Z}^k$ itself is poor, namely $\mathbb{A}(\mathbb{Z}^k)$ is much too incompatible to $\mathbb{A}(F(x))$. $\mathbb{Z}^k$ thus must be augmented.

We add new z to $\mathbb{Z}^k$ until the following is satisfied.

$$\exists z \in \mathbf{co}\left(F(x^k), F(x^k+s^k)\right) \textit{ and index } j \textit{ s.t.}$$

$$j \in \mathbb{A}(z) \textit{ and } \nabla h_j(z)^\top \left(F(x^k) - F(x^k+s^k)\right) - \nabla\phi(x^z)\cdot s^k \le h(F(x^k)) + \phi(x^k) - h(F(x^k+s^k)) - \phi(x^k+s^k) \tag{7}$$

Where x^z is some x that $||F(x)-z|| \le \frac{L_{\nabla F}}{4}||s^k||^2$. Existence of such x is guaranteed by Lemma 4.1.This condition typically states that there exists j such that h_j performs worse than d. We also require that there exists a secondary $z' \in \mathbb{Z}^k$ such that

$$j \in \mathbb{A}(z') \textit{ and } \left(s^k\right)^\top \nabla M\left(x^k\right)\left(\nabla h_j\left(z'\right) - d^k\right) \le 0 \tag{8}$$

Geometrically, this is saying that projection from s^k to generator produced by d^k is better than that produced by h_j.

3.1.5 Test

In common with classical trust-region methods, manifold sampling employs a ratio test as a merit criterion. The ρ_k used in the manifold sampling algorithm is the ratio of actual decrease in F to predicted decrease in m^F, as weighted by the convex combination of gradients d^k.

$$\rho_k \triangleq \frac{\left\langle F\left(x^k\right) - F\left(x^k+s^k\right), d^k\right\rangle + \phi(x^k) - \phi(x^k+s^k)}{\left\langle M\left(x^k\right) - M\left(x^k+s^k\right), d^k\right\rangle + \phi(x^k) - \phi(x^k+s^k)} \tag{9}$$

3.2 Algorithm

For detailed pseudo code, check A.

3.3 Some Statement

Here we comment on the algorithm. For upper bound of η_2, we assume

$$\eta_{\max} \triangleq \min\left\{\frac{1}{L_\phi + L_h\kappa_{\mathrm{H}}}, \frac{\eta_1\kappa_{\mathrm{d}}}{4C}\right\} \tag{10}$$

This might be violated in practice for efficiency. We also define acceptable iterations and successful iteration. If in iteration k, $\Delta_k < \eta_2||g^k||$, it is acceptable. If in iteration k, $\rho_k > \eta_1$, it is successful.

4 Theoretical Guarantees

In this section we present theoretical guarantees stating that under proper parameter settings and initializations, sequences $(x^k, f(x^k))$ converges to local optimum $(x^*, f(x^*))$. We now briefly introduce each lemma's role. Lemma 4.1 provide useful bound for sample set. Lemma 4.2 assures that the way we build master model has a "order-1 subgradient accuracy". Lemma 4.3 and 4.4 assures proper augmentation exists and can be found by algorithm provided. Lemma 4.5 assures that we can always exit loop with some choice of s. Lemma 4.6 evaluate decrease in each acceptable iteration. Lemma 15 provides sufficient condition for successful iteration. Lemma 4.8 4.9 4.10 together give a converging subsequence and Theorem 4.11 states that it converges to Clark sub-stationary point. Since proof is analogous to [32], we only present parts of proof that should be modified.

Lemma 4.1. For $z = \alpha F(x) + (1-\alpha)F(y)$, $\alpha \in [0,1]$, a path in $\mathbf{co}\left(\mathrm{Im}\left(\mathscr{L}_{\max}\right)\right)$, we can find a path $\sigma(\alpha)$, $\alpha \in [0,1]$, $\alpha(0) = x$, $\alpha(1) = y$ in $\mathscr{L}_{max}$ such that

$$||\alpha F(x) + (1-\alpha)F(y) - F(\sigma(\alpha))|| \le \frac{1}{4}L_{\nabla F}||x-y||^2$$

Proof. We take $\sigma(\alpha)=\alpha x+(1-\alpha)y$ as example.

$$\begin{aligned}
&||\alpha F(x)+(1-\alpha)F(y)-F(\alpha x+(1-\alpha)y)||\\
=&||F(x)+(1-\alpha)\nabla F(x)^T(y-x)+(1-\alpha)^2(y-x)^T\nabla^2F(x)(y-x)-(\alpha F(x)+(1-\alpha)F(y))+o(||y-x||^2)||\\
=&||(1-\alpha)(\nabla F(t(y-x))-\nabla F(x))^T(x-y)+(1-\alpha)^2(y-x)^T\nabla^2F(x)(y-x)+o(||y-x||^2)||\\
=&||((1-\alpha)t-(1-\alpha)^2)(y-x)^T\nabla^2F(x)(y-x)+o(||y-x||^2)||\\
\leq&\frac{1}{4}L_{\nabla F}||x-y||^2
\end{aligned} \tag{11}$$

□

Lemma 4.2. If all assumption hold, given $x,y\in\mathscr{L}_{max}$ s.t. $||x-y||\leq\Delta$, then for

$$\mathbb{G}\triangleq\{\nabla M(x)\nabla h_i(F(x))+\nabla\phi : i\in I\}$$

$$\mathbb{H}\triangleq\text{co}\left(\left\{\nabla F(y)\nabla h_j(F(y))+\nabla\phi : j\in I\right\}\right)$$

we have

$$\|g-v(g)\|\leq c_2\Delta$$

where

$$c_2\triangleq L_h\left(L_{\nabla F}+\kappa_{\mathrm{g}}\right)+L_F^2L_{\nabla h_i}$$

Lemma 4.3. All assumptions hold, we can find z and j satisfying 7.

Proof. We define

$$z(\alpha)\triangleq\alpha F(x)+(1-\alpha)F(x+s)$$

We show by contradiction. Suppose there exists no such α such that $z(\alpha)$ is qualified. Then

$$\begin{aligned}
\phi(x)+h(F(x))-\phi(x+s)-h(F(x+s))&\geq\int_0^1\inf_{j\in\mathbb{A}(z(\alpha))}\left\{\nabla h_j(z(\alpha))^\top(F(x)-F(x+s))\right\}-\nabla\phi(x+(1-\alpha)s)\cdot sd\alpha\\
&>\int_0^1(\phi(x)+h(F(x))-\phi(x+s)-h(F(x+s)))d\alpha\\
&=\phi(x)+h(F(x))-\phi(x+s)-h(F(x+s))
\end{aligned}$$

and thus a contradiction. □

Lemma 4.4. All assumptions hold, Algorithm 2 returns z and j in finite time.

Lemma 4.5. Let d^k and $g^k=\nabla m^Fd^k+\nabla\phi$ be as in 4. If all assumptions hold and 10 is satisfied, then

$$\hat{s}=-\Delta_k\frac{g^k}{||g^k||}$$

satisfies 6.

Proof. It is clear that

$$||d^k||\leq L_h \tag{12}$$

From assumption 2.8 and definition 2.9,

$$\begin{aligned}
&\left\|M\left(x^k\right)+\nabla M\left(x^k\right)^\top s-M\left(x^k+s\right)\right\|\\
\leq&\sum_{i=1}^p\left|m^{F_i}\left(x^k\right)+\nabla m^{F_i}\left(x^k\right)^\top s-m^{F_i}\left(x^k+s\right)\right|\\
\leq&\sum_{i=1}^p\frac{1}{2}\kappa_{i,\mathrm{mH}}||s||^2=\frac{1}{2}\kappa_{\mathrm{H}}||s||^2
\end{aligned} \tag{13}$$

Combining 12 and 13, we have

$$\begin{aligned}
&\left(d^k\right)^\top \left(M\left(x^k\right)+\nabla M\left(x^k\right)^\top \hat{s}-M\left(x^k+\hat{s}\right)\right)\\
\geq& -\left\|d^k\right\|\left\|M\left(x^k\right)+\nabla M\left(x^k\right)^\top \hat{s}-M\left(x^k+\hat{s}\right)\right\|\\
\geq& -\frac{1}{2}L_h\kappa_{\mathrm{H}}\|\hat{s}\|^2=-\frac{1}{2}L_h\kappa_{\mathrm{H}}\Delta_k^2
\end{aligned}$$

Using 10, we get

$$\begin{aligned}
&\phi(x^k)-\phi(x^k+s^k)+\left\langle M\left(x^k\right)-M\left(x^k+\hat{s}\right),d^k\right\rangle\\
\geq& -\nabla\phi(x^k)\cdot\hat{s}-\frac{1}{2}L_{\nabla\phi}\hat{s}^Ts-\left(d^k\right)^\top\nabla M\left(x^k\right)^\top\hat{s}-\frac{1}{2}L_h\kappa_{\mathrm{H}}\Delta_k^2\\
=& -\left(g^k\right)^\top\hat{s}-\frac{1}{2}L_h\kappa_{\mathrm{H}}\Delta_k^2=\left\|g^k\right\|\Delta_k-\frac{1}{2}L_h\kappa_{\mathrm{H}}\Delta_k^2-\frac{1}{2}L_{\nabla\phi}\Delta_k^2\\
\geq& \left\|g^k\right\|\Delta_k-\frac{1}{2}\left\|g^k\right\|\Delta_k=\frac{1}{2}\left\|g^k\right\|\Delta_k\\
\geq& \frac{\kappa_{\mathrm{d}}}{2}\left\|g^k\right\|\min\left\{\Delta_k,\frac{\|g^k\|}{L_{\nabla\phi}+L_h\kappa_{\mathrm{H}}}\right\}
\end{aligned}$$

■

Lemma 4.6. All assumption hold. If iteration k of Algorithm 1 is acceptable, then for C in 2.9

$$\phi(x^k)+h(F(x^k))-\phi(x^k+s^k)-h(F(x^k+s^k))\geq\left(d^k\right)^\top\left(F(x^k)-F(x^k+s^k)\right)-\nabla\phi(x^k)\cdot s-C\Delta_k^2 \tag{14}$$

Proof. We first show that 8 always holds. In 8, if z' is found then it is naturally true. Else, we have $s^k=-\Delta_k\frac{g^k}{\|g^k\|}$. From classical projection theory and 2

$$\begin{aligned}
&\left(\nabla h_j\left(z'\right)-d^k\right)^\top\nabla m^F(x^k)^\top s^k\\
=&-\left(\nabla h_j\left(z'\right)-d^k\right)^\top\nabla m^F(x^k)^\top\Delta_k\frac{g^k}{\|g^k\|}\\
=&positiveconst\cdot\left[\left(\nabla\phi(x^k)+d^k\nabla m^F(x^k)^\top\right)g^k-\left(\nabla\phi(x^k)+\nabla h_j\left(z'\right)\nabla m^F(x^k)^\top\right)g^k\right]\\
\leq&0
\end{aligned}$$

From assumption 2.2 and 2.9,

$$\left|F(x^k+s^k)-F(x^k)-\nabla F(x^k)^\top s^k\right|\leq\frac{L_{\nabla F}}{2}\left\|s^k\right\|^2$$

Thus

$$\begin{aligned}
&\left(\nabla h_j(z)-d^k\right)^\top\left(F(x^k+s^k)-F(x^k)-\nabla F(x^k)^\top s^k\right)\\
&\leq\left\|\nabla h_j(z)-d^k\right\|\left\|F(x^k+s^k)-F(x^k)-\nabla F(x^k)^\top s^k\right\|\\
&\leq(2L_h)\tfrac{L_{\nabla F}}{2}\left\|s^k\right\|^2
\end{aligned}$$

which leads to

$$
\begin{aligned}
&\phi(x^k+s^k)+h(F(x^k+s^k))-\phi(x^k)-h(F(x^k))-\nabla\phi(x^k)\cdot s-\left(d^k\right)^\top\left(F(x^k+s^k)-F(x^k)\right)\\
\leq&\left(\nabla h_j(z)-d^k\right)^\top\left(F(x^k+s^k)-F(x^k)\right)+\phi(x^k+s^k)-\phi(x^k)-\nabla\phi(x^z)\cdot s\\
=&\left(\nabla h_j(z)-d^k\right)^\top\left(F(x^k+s^k)-F(x^k)\right)+s^T\nabla^2\phi(s-x^z+x)\\
\leq&\left(\nabla h_j(z)-d^k\right)^\top\nabla F(x^k)^\top s^k+\left(L_hL_{\nabla F}+L_{\nabla\phi}\right)\left\|s^k\right\|^2\\
=&\left(\nabla h_j(z')-d^k\right)^\top\nabla F(x^k)^\top s^k+\left(\nabla h_j(z)-\nabla h_j(z')\right)^\top\nabla F(x^k)^\top s^k+\left(L_hL_{\nabla F}+L_{\nabla\phi}\right)\left\|s^k\right\|^2\\
\leq&\left(\nabla h_j(z')-d^k\right)^\top\left(\nabla m^F(x^k)^\top s^k+\left(\nabla F(x^k)-\nabla m^F(x^k)\right)^\top s^k\right)\\
&+2L_{\nabla h}L_F^2\Delta_k^2+L_{\nabla\phi}\Delta_k^2+L_hL_{\nabla F}\Delta_k^2\\
=&\left(\nabla h_j(z')-d^k\right)^\top\nabla m^F(x^k)^\top s^k+\left(\nabla h_j(z')-d^k\right)^\top\left(\nabla F(x^k)-\nabla m^F(x^k)\right)^\top s^k\\
&+2L_{\nabla h}L_F^2\Delta_k^2+L_{\nabla\phi}\Delta_k^2+L_hL_{\nabla F}\Delta_k^2\\
\leq&0+2L_h\kappa_g\Delta_k^2+2L_{\nabla h}L_F^2\Delta_k^2+L_{\nabla\phi}\Delta_k^2+L_hL_{\nabla F}\Delta_k^2\\
=&C\Delta_k^2
\end{aligned}
$$

■

Lemma 4.7. All assumptions hold. For acceptable iterations, if

$$\Delta_k<\frac{\kappa_{\mathrm{d}}(1-\eta_1)}{4\kappa_{\mathrm{f}}L_h}\left\|g^k\right\| \tag{15}$$

then $\rho_k>\eta_1$ in Algorithm 1, and the iteration is successful.

Lemma 4.8. All assumption hold. If $\{x^k,\Delta_k\}$ is generated by Algorithm 1, then the sequence $\{f(x^k)\}$ is nonincreasing and $\lim_k\Delta_k=0$.

Proof. We show here that f is non-increasing. Rest of the proof is analogous to [32]. If an iteration is unsuccessful, then $x^{k+1}=x^k$, therefore $f(x^{k+1})=f(x^k)$. If it is successful,

$$
\begin{aligned}
f(x^k)-f(x^{k+1})&\geq\left(d^k\right)^\top\left(F(x^k)-F(x^{k+1})\right)-\nabla\phi(x^z)\cdot s-C\Delta_k^2\\
&\geq\left(d^k\right)^\top\left(F(x^k)-F(x^{k+1})\right)+\phi(x^k)-\phi(x^k+s^k)-L_{\nabla\phi}\Delta_k^2-C\Delta_k^2\\
&=\rho_k\left[\left(d^k\right)^\top\left(m^F(x^k)-m^F(x^{k+1})\right)+\phi(x^k)-\phi(x^k+s^k)\right]-L_{\nabla\phi}\Delta_k^2-C\Delta_k^2\\
&\geq\rho_k\frac{\kappa_{\mathrm{d}}}{2}\left\|g^k\right\|\min\left\{\Delta_k,\frac{\|g^k\|}{L_h\kappa_{\mathrm{H}}+L_{\nabla\phi}}\right\}-L_{\nabla\phi}\Delta_k^2-C\Delta_k^2\\
&=\rho_k\frac{\kappa_{\mathrm{d}}}{2}\left\|g^k\right\|\Delta_k-C\Delta_k^2>\eta_1\frac{\kappa_{\mathrm{d}}}{2}\left\|g^k\right\|\Delta_k-L_{\nabla\phi}\Delta_k^2-C\Delta_k^2\\
&>\eta_1\frac{\kappa_{\mathrm{d}}}{2}\frac{4C}{\eta_1\kappa_{\mathrm{d}}}\Delta_k^2-L_{\nabla\phi}\Delta_k^2-C\Delta_k^2=C\Delta_k^2-L_{\nabla\phi}\Delta_k^2>0
\end{aligned}
$$

■

Lemma 4.9. For sequence in Lemma 4.8, we have

$$\liminf_{k\to\infty}||g^k||=0$$

Lemma 4.10. For sequence in Lemma 4.8, there is a infinite subsequence of acceptable iterations $\{k_j\}$ such that

$$\lim_{j\to\infty}\left\|g^{k_j}\right\|=0$$

and $\{x^{k_j}\}$ converge to some x^*. What is more, $0\in\partial_Cf(x^*)$.

Theorem 4.11. All assumptions hold. For a sequence generated by Algorithm 1, its cluster point is Clark substationary.

5 Simulations

We present numerical simulation results in the section. We first illustrate with classical examples that manifold sampling is computational efficient and considerably precise. We also adopt benchmark from [13] consisting of ten non-smooth optimization problems. We pay special attention to locally convergence and illustrate performance under far starting point. Then we test performance of MS under non-convex non-smooth situation. At last, we combine MS with neural network parameter optimization.

5.1 L^1 Norm

We first consider a simple situation

$$\phi(x) + h(F(x)) = ||x||_2 + ||((x_2 - x_1^2)^2 + (1-x_1)^2, x_3)||_1,\ x \in \mathbb{R}^3 \tag{16}$$

where h is L^1 norm and $\phi : \mathbb{R}^3 \to \mathbb{R}^2$ is Rosenbrock test function on its first element. The third element is additive separable and take 0 as optimum. Function value with respect to x_1 and x_2 is as Figure 1.

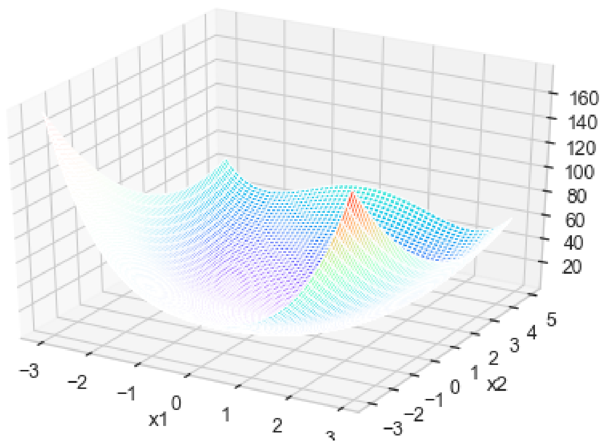

Figure 1: Function value w.r.t. x_1 and x_2.

Noting that this is generally convex, we don't post result about locally convergence here. For hyperparameters, we set $\eta_1 = 0.1$, $\eta_2 = 1$, $\kappa_d = 10^{-4}$, $\kappa_H = 1$, $\gamma_d = 0.5$, $\gamma_i = 1.5$, $\Delta_{max} = 10^4$, $\Delta_{min} = 10^{-13}$. Initial x_0 is chosen random from standard Gaussian. If algorithm performs at most 3000 iteration. From a perspective of practice, we also set termination condition that algorithm shuts if $\left|f(x^k) - f(x^{k+1})\right| < conv\_tol = 10^{-9}$ for several successive k (three for us). We especially care for number of points that is used to build local derivative free model since it greatly effect efficiency and precision. Figure 2 shows performance of model under different number setting. We set number of points to 10 in following experiments because it achieves both reasonable precision and is comparably fast.

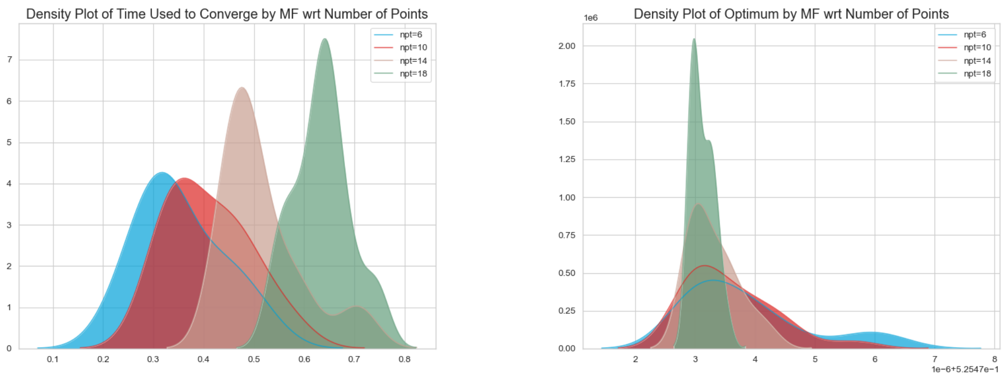

Figure 2: (Best viewed in color) Distribution of time and returned objective function value. Unsurprisingly, increasing number of points leads to more robust model with trade off in time.

We adopt two simple benchmarks. One is least square derivative free optimization tool box by [6], to compare manifold sampling and other derivative free methods. Another is simply gradient descent with implementation in PyTorch. Note that this is possible only if function h is piecewise smooth and that F(x) falling in non-smooth region has measure zero. Use of PyTorch is for further work of combining MF and neural network. Figure 3 presents their performance and shows superiority of MF.

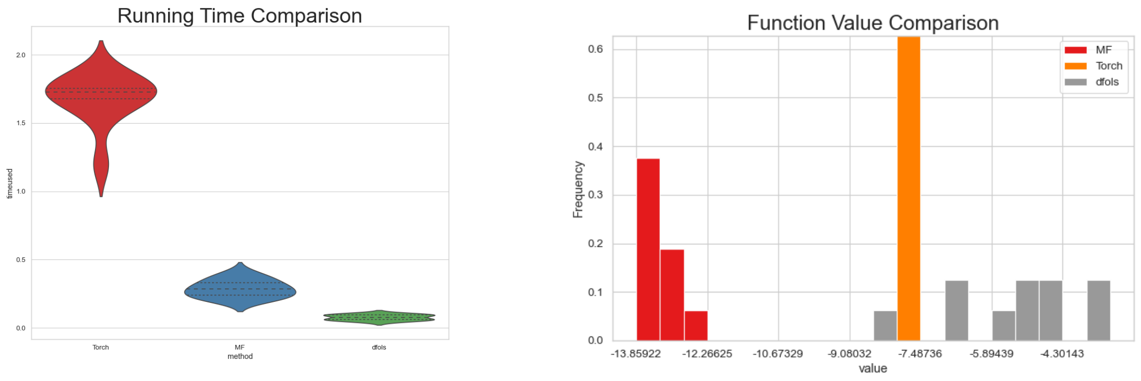

Figure 3: (Best viewed in color) Comparison result of three model. Function value is viewed in log scale of difference with oracle obtained by grid search. MF achieves best precision in relatively short time.

However, Rosenbrock function in 16 is non-negative and thus L^1 norm posed is meaningless. Thus we change the test problem to

$$\phi(x) + h(F(x)) = ||x||_2 + ||((x_2 - x_1^2)^2 - (1-x_1)^2, x_3)||_1,\ x \in \mathbb{R}^3 \tag{17}$$

This is a non-convex problem and its behavior around the origin is shown in 4.

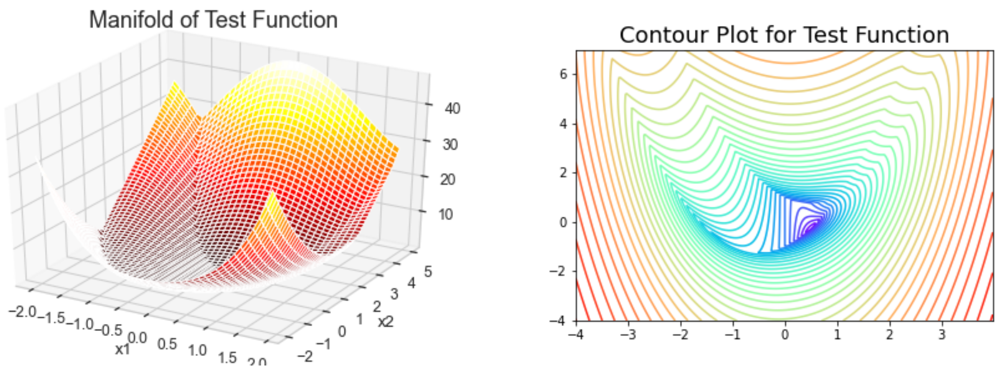

Figure 4: (Best viewed in color) Test function plotting. Left is plot of manifold around origin, right is contour plot of log value of test function.

For performance on this problem, check B.

5.2 Benchmark Problem Set

We adopt [13] as benchmark. The problem set is especially designed for non-smooth optimization and can provide problem of any dimension scale. For details, please check C. The result is displayed in Table 1. We emphasize that most, if not all, of the original problems was formed in low dimension with n=2, in which cases manifold sampling performed pretty well. Yet high dimension experiments were done for further exploration.

We note that in the experiment, some sequence non-convergent sequence will have a convergent subsequence to optimum. We believe the second reason, which is about F, accounts for most cases. For instance, in trials where error is large (or even not convergent) and liminf is small, ”Chained CB3 1” ”Number of Active Faces” and ”Chained Crescent 2”, the ”curvature” we presented in the table is respectively $12(n-1)$, ∞ and $2(n-1)$, which is large and makes the bounding effect of curvature useless. Thus it is reasonable for this to happen. For some other intuition, though the three problems all converges for n=2, ”Chained CB3 1” and ”Chained Crescent 2” has small curvature for small n, thus they still have small liminf for n=5, while ”Number of Active Faces” does not. We add here some discussion of performance sensitivity with respect to n. In test set, solution can still be acquired when n=100 for ”MAXQ”, ”MXHILB”, ”CB3 2” and ”Crescent 1”, all of which have curvature of constant scale.

5.3 Nonconvexity

We argue that, though non-convex, the test function does not have non-convex h, which is one reason that manifold sampling is powerful. Thus we check for general non-convex non smooth function.

$$\phi(x)+h(F(x)) = -exp(-||x||^2)+h((x_2-x_1^2)^2,(1-x_1)^2),\ x\in\mathbb{R}^2 \tag{18}$$

$$h(z)=\sum_{i=1}^{m}|1-z_i^2|$$

We present the problem as before in Figure 5 and solution in ??

Problem	Curvature	n	Convergence	Reach Optimum	Error	Liminf Error	Time(s)	Iteration
Generalization of MAXQ	2	2	Y	Y	-9	-9	1.71	150
		5	Y	Y	-9	-9	3.59	300
		10	Y	Y	-9	-9	6.57	500
		100	Y	Y	-5	-5	140	3000
Generalization of MXHILB	1	2	Y	Y	-6	-6	5.46	300
		5	Y	Y	-5	-5	51	3000
		10	Y	Y	-5	-5	107	3000
		100	Y	Y	-3	-5	5598	3000
Chained LQ	2(n-1)	2	Y	Y	-6	-6	2.74	100
		5	Y	Y	-3	-5	112	3000
		10	Y	N		-5	180	3000
		100	N	Y		-5		
Chained CB3 1	12(n-1)	2	Y	Y	-8	-8	21.4	1000
		5	Y	N	-1	-4	20.0	250
		10	N	N		-1		
		100	N	N		-1		
Chained CB3 2	12	2	Y	Y	-9	-9	27.3	1450
		5	Y	Y	-3	-4	61.6	3000
		10	Y	Y	-3	-3	80.4	3000
		100	Y	Y	-3	-3	590	3000
Number of Active Faces	∞	2	Y	Y	-9	-9	3.30	130
		5	N	N		-1		
		10	N	N		-1		
		100	N	N		-1		
Chained Milfflin	2(n-1)	2	Y	Y	-9	-9	31.3	1600
		5	Y	Y	-6	-6	71.7	3000
		10	N	N				
		100	N	N				
Chained Crescent 1	2	2	Y	Y	-9	-9	3.14	150
		5	Y	Y	-8	-8	22	700
		10	Y	Y	-7	-7	30	1200
		100	Y	Y		-3	90	3000
Chained Crescent 2	2(n-1)	2	Y	Y	-7	-7	2.11	150
		5	N	N		-4		
		10	N	N		-1		
		100	N	N		1		

Table 1: Experiment result on benchmark. We test MS on nine problem (there was ten problems in benchmark set but problem 7 is not compatible with our formulation by adopting inseparable non-smoothness and compositeness) with input dimension n=2, 5, 10 and 100 to see how MS perform on high dimension tasks. Midway dimension m is intrinsic as illustrate in C. We also list out upper bound of $h(L_{\nabla F_i}, \cdots)$ at optimum for further use and we call this ”curvature”. All results are acquired from single trial but every case is guaranteed not to be outlier by at least five additional trials. By convergence we mean whether x^k and $f(x^k)$ is convergent. By reaching optimum we mean that whether x^k and $f(x^k)$ is approaching the global optimum. For all problems, since function has unique global minimum, we would expect convergence implies reaching optimum. However, violation happened in “Chained CB3 1” and ”Chained CB3 2”. This is not due to occasional numerical error. We further check that curvature around optimum is relatively large, which we blame this situation on. By error we mean the log scale of difference between optimal value and convergent sequence. Note that for problem ”Chained Milfflin”, we do not have exact extreme value and thus we use the Cauchy limit of sequence if convergent and omit if not. We also add inferior limits here to illustrate potential convergence.Reason for jumping out of convergence lies in two possibilities. One is numerical explosion and the other is F itself. See below for further discussion. By time and iteration we mean period of time and iterations (less than 3000) spent by algorithm. Positions omitted is where filling this position is meaningless, e.g. time for a non-convergent trial.

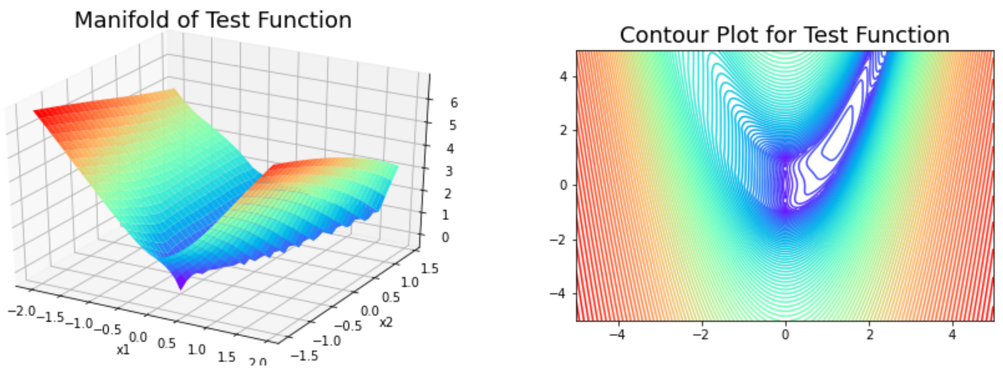

Figure 5: (Best viewed in color) Test function plotting. Left is plot of manifold around origin, right is contour plot of log value of test function. The function has five local minimum, (0,0)(0,1)(0,-1)(2,3)(2,5)

We generate 1000 initial points from uniform distribution $[-3,3] \times [-3,3]$ to test local convergence of MS on non-convex h. In most trials, sequence converges successfully and we present convergence result in Figure 6. Due to limitations in computation resources, we could only use large search step when samples are large. The converging shows territorial property that initial points will converge to closest stationary point.

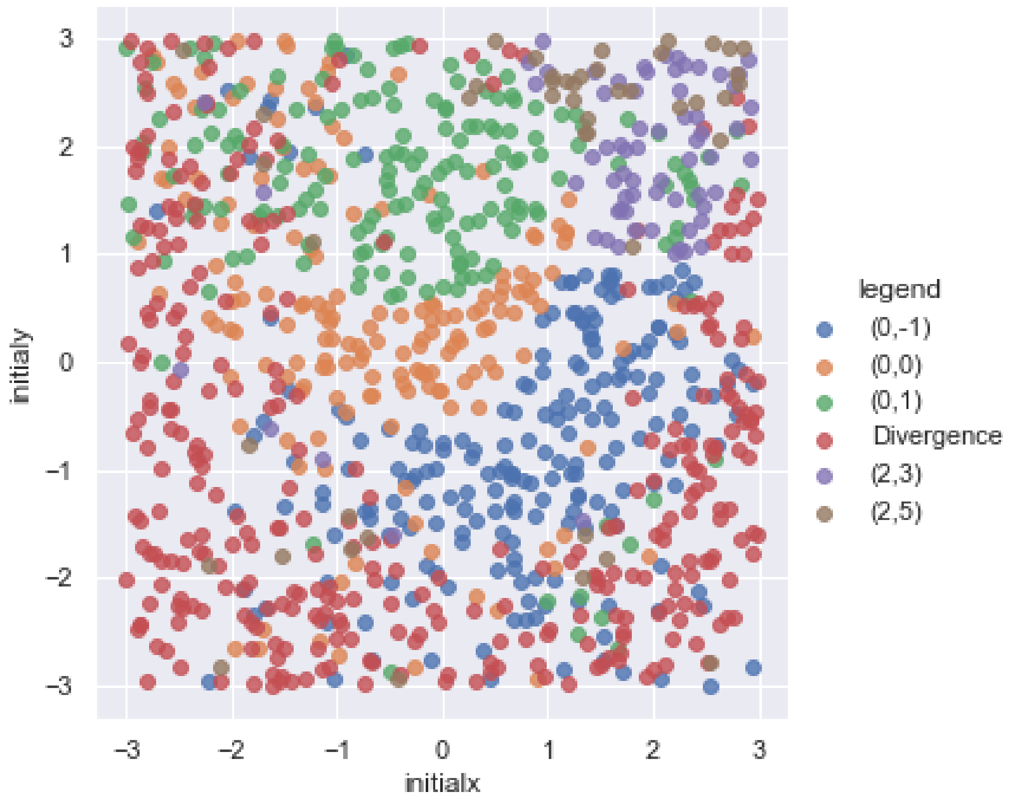

Figure 6: (Best viewed in color) We note that in left picture there are many divergent trials still. This is due to searching parameters in 5. Since 1000 experiments are expensive to run, we could only adopt large search steps, which leads to x^ks jumping between two "slope" of loss surface. Given enough time and small search step, then would surely converge.

6 Empirical Study

For real data application, we use MS to do regression task using feed forward neural network. The dataset we use is diabetes dataset from [7]. There are ten baseline variables—age, sex, body-mass index, average blood pressure, and six blood serum measurements serving as independent variables, with the response of interest being a quantitative measure of disease progression one year after baseline. 442 samples are included. We build feed forward neural network with one hidden layer including 10 hidden units using sigmoid function as activation. Since sample size is relatively small, we use gradient descent as a benchmark and examine change of loss through time as in Figure 7. In the figure, though unstable, MS is able to beat NN by a lower loss and faster speed.

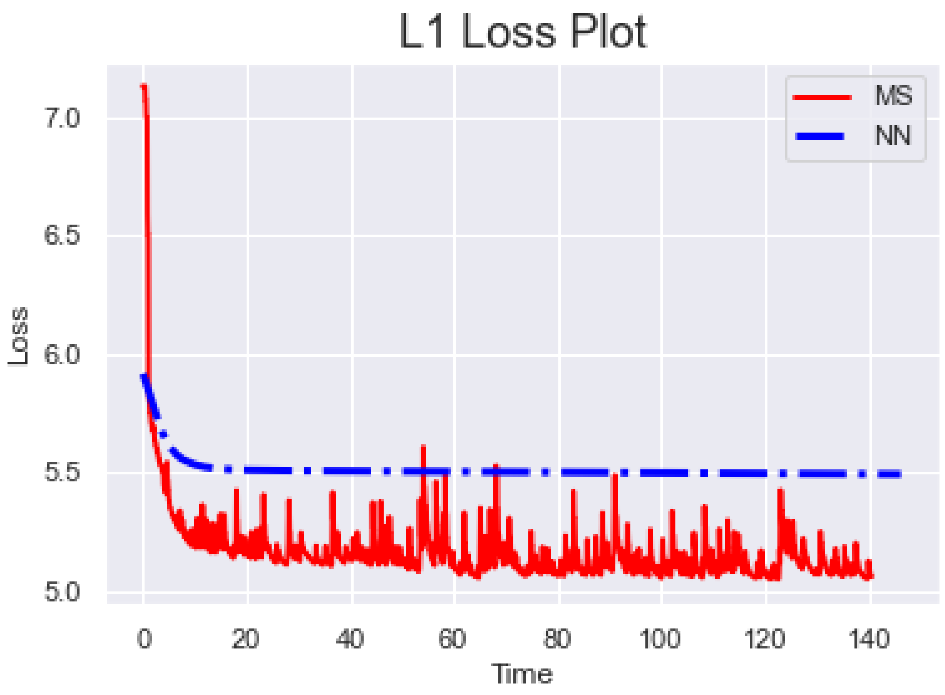

Figure 7: (Best viewed in color). Change of loss in log scale through time. MS rapidly finds minimum better than gradient descent. Note that as naive benchmark, we do not perform batch norm or learning rate tuning, which is unfair for MS since no hyper parameter tuning pipeline has been given. Oscillation of MS is due to that parameter set previously is no longer proper after it falls into region with higher ”curvature”.

References

[1] Ioffe A.D. Necessary and sufficient conditions for a local minimum, 1: A reduction theorem and first-order conditions. SIAM J Control Optim 17:245-250, 1979.

[2] R Arastoo. Optimal sparse output feedback controller design: A rank constrained optimization approach. https://arxiv.org/pdf/1412.8236.pdf, 2014.

[3] D. P. Bertsekas. Constrained optimization and lagrange multiplier methods. New York Academic Press, 1982.

[4] N. Parikh. S. Boyd. Proximal algorithms. Foundations and Trends in Optimization, 2013.

[5] Charalambous C. Acceleration of the least pth- algorithm for minimax optimization with engineering applications. Math Program, 1979.

[6] Fiala J. Marteau B. Cartis, C. and L. Roberts. Improving the flexibility and robustness of model-based derivative-free optimization solvers. ACM Transactions on Mathematical Software, 45:3:32:1–32:41, 2019.

[7] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. The Annals of Statistics, 32(2):407 – 499, 2004.

[8] R. Fletcher. A model algorithm for composite nondifferentiable optimization problems, in nondifferential and variational techniques in optimization. D. C. Sorensen and R. J.-B. Wets, eds., vol. 17 of Math- ematical Programming Studies, Springer, 1982.

[9] R. Fletcher. Second order corrections for non-differentiable optimization. Numerical Analysis, Springer, 1982.

[10] Wajeb Gharibi. A dual approach for solving nonlinear infinite-norm minimization problems with applications in separable cases. https://arxiv.org/pdf/1106.0851.pdf, 2011.

[11] W. L. Hare and C. Sagastiza´bal. A redistributed proximal bundle method for nonconvex optimization. SIAM Journal on Optimization, 2010.

[12] N. Karmitsa and A. M. Bagirov. Limited memory discrete gradient bundle method for nonsmooth derivative-free optimization. Optimization, 61, 2012.

[13] Napsu Karmitsa, K. Miettinen, and Marko Makela. New limited memory bundle method for large-scale nons- mooth optimization. Optimization Methods and Software - OPTIM METHOD SOFTW, 19:673–692, 12 2004.

[14] Andre A. Keller. lp-norm minimization method for solving nonlinear systems of equations. WSEAS TRANS- ACTIONS on MATHEMATICS, 2014.

[15] Neil K. Dhingra. Sei Zhen Khong and Mihailo R. Jovanovic. The proximal augmented lagrangian method for nonsmooth composite optimization. https://arxiv.org/pdf/1610.04514.pdf, 2016.

[16] K. A. Khan. J. Larson and S. M. Wild. Manifold sampling for optimization of nonconvex functions that are piecewise linear compositions of smooth components. SIAM Journal on Optimization 28, 2018.

[17] J. V. Burke. A. S. Lewis and M. L. Overton. Approximating subdifferentials by ran- dom sampling of gradients. Mathematics of Operations Research, 2002.

[18] J. Larson. M. Menickelly and S. M. Wild. Manifold sampling for l1 nonconvex optimization. SIAM Journal on Optimization, 26, 2016.

[19] M. Menickelly and S. M. Wild. Derivative-free robust optimization by outer approximations. Mathematical Programming, 179, 2020.

[20] J. V. Burke. F. E. Curtis. A. S. Lewis. M. L. Overton and L. E. A. Simo˜es. Gradient sampling methods for nonsmooth optimization. Numerical Nonsmooth Optimization, 2020.

[21] S. Boyd. N. Parikh. E. Chu. B. Peleato and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. Found. Trends Mach. Learning, 2011.

[22] G. Liuzzi. S. Lucidi. F. Rinaldi and L. N. Vicente. Trust-region methods for the derivative-free optimization of nonsmooth black-box functions. SIAM Journal on Optimization 29, 2019.

[23] John C. Duchi. Feng Ruan. Solving (most) of a set of quadratic equalities: Composite optimization for robust phase retrieval. https://arxiv.org/abs/1705.02356, 2017.

[24] A. Beck. M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci, 2009.

[25] Robert Tibshirani. Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (methodological) 58, 1996.

[26] Hui Zou. Hastie Trevor. Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society, Series B. 67, 2005.

[27] Berke J. V. Second-order necessary and sufficient con- ditions for convex composite ndo. Math Program 38:287-302, 1987.

[28] S. M. Wild. Solving derivative-free nonlinear least squares problems with pounders, in advances and trends in optimization with engineering applications. SIAM, 2017.

[29] R. Womersley and R. Fletcher. An algorithm for composite nonsmooth optimiza- tion problems. Journal of Optimization Theory and Applications, 1986.

[30] Jeyakumar V. Yang XQ. Convex composite multi- objective nonsmooth programming. Math Program 59, 1993.

[31] Xiaowei Zhou. Can Yang. Hongyu Zhao. Weichuan Yu. Low-rank modeling and its applications in image analysis. ACM Computing Surveys 47, 2014.

[32] J. Larson. M. Menickelly. Baoyu Zhou. Manifold sampling for optimizing nonsmooth nonconvex compositions. https://arxiv.org/abs/2011.01283, 2020.

[33] W. L. Hare. C. Sagastiza´bal and M. Solodov. A proximal bundle method for nonsmooth nonconvex functions with inexact information. Computational Optimization and Applications, 63, 2016.

[34] M. Ma¨kela¨. Survey of bundle methods for nonsmooth optimization. Optimization Methods and Software, 2002.

A Code

Pseudo code of main part of manifold sampling is in 1. 2 is grid search method necessary for Algorithm 1.

Algorithm 1 Manifold Sampling

```
1:  Set $\eta_1 \in (0,1), \kappa_{\mathrm{d}} \in (0,1), \kappa_{\mathrm{H}} \geq 0, \eta_2 \in (0, \eta_{\max}), 0 < \gamma_{\mathrm{d}} < 1 \leq \gamma_{\mathrm{i}},$ and $\Delta_{\max}, \Delta_{\min} > 0$
2:  Choose $\Delta_0$ in $(0, \Delta_{max}]$ and initialize $x^0$
3:  for $k = 0 \to K$ do
4:      Build master model $m^F$
5:      Initialize $\mathbb{Z}^k$ and Form $D^k$
6:      while true do
7:          Form $\nabla m^F$
8:          $G^k \leftarrow \nabla M\left(x^k\right) D^k + (\nabla\phi(x^k), \cdots, \nabla\phi(x^k))$
9:          Solve 3 for $\lambda^*$
10:         $d^k \leftarrow D^k \lambda^*$
11:         $g^k \leftarrow G^k \lambda^*$
12:         if $\Delta_k < \Delta_{min}$ then
13:             Break
14:         end if
15:         if $\Delta_k < \eta_2 ||g^k||$ then
16:             Solve 5 to get $s^k$
17:             Get z and j satisfying 7 using Algorithm 2
18:             if $j \in \mathbb{A}(\mathbb{Z}^k)$ and no z' satisfies 8 then
19:                 $s^k \leftarrow -\Delta_k \frac{g^k}{||g^k||}$
20:                 Get z and j satisfying 7 using Algorithm 2
21:             end if
22:             if $j \in \mathbb{A}(\mathbb{Z}^k)$ then
23:                 Calculate $\rho_k$ using 9
24:                 Break
25:             else
26:                 Update $m^F$ and $Z\mathbb{Z}^k$
27:                 Form $D^k$
28:             end if
29:         else
30:             $\rho_k \leftarrow 0$
31:             Break
32:         end if
33:     end while
34:     if $\rho_k > \eta_1 > 0$ then
35:         $x^{k+1} \leftarrow x^k + s^k$
36:         $\Delta_{k+1} \leftarrow min\{\gamma_i \Delta_k, \Delta_{max}\}$
37:     else
38:         $x^{k+1} \leftarrow x^k$
39:         $\Delta_{k+1} \leftarrow \gamma_d \Delta_k$
40:     end if
41: end for
```

B Test Performance

We adopt same procedure for performance test as in Figure 8 and 9.

Algorithm 2 Grid Search for z and j

```
1:  if $x^z = x$, F(x) and j satisfy 7 then
2:      return F(x) and j
3:  end if
4:  if $x^z = x + s$, F(x+s) and j satisfy 7 then
5:      return F(x+s) and j
6:  end if
7:  for $l = 0, \cdots$ do
8:      Generate $\{\frac{2k-1}{2^l} : k = 1, \dots, 2^{l-1}\}$
9:      for $k = 1, 2, \cdots 2^{l-1}$ do
10:         $\alpha \leftarrow \frac{2k-1}{2^l}$
11:         $z(\alpha) \leftarrow \alpha F(x) + (1-\alpha) F(x+s)$
12:         if $x^z = x + (1-\alpha)s$, $z(\alpha)$ and j satisfy 7 then
13:             return $z(\alpha)$and j
14:         end if
15:     end for
16: end for
```

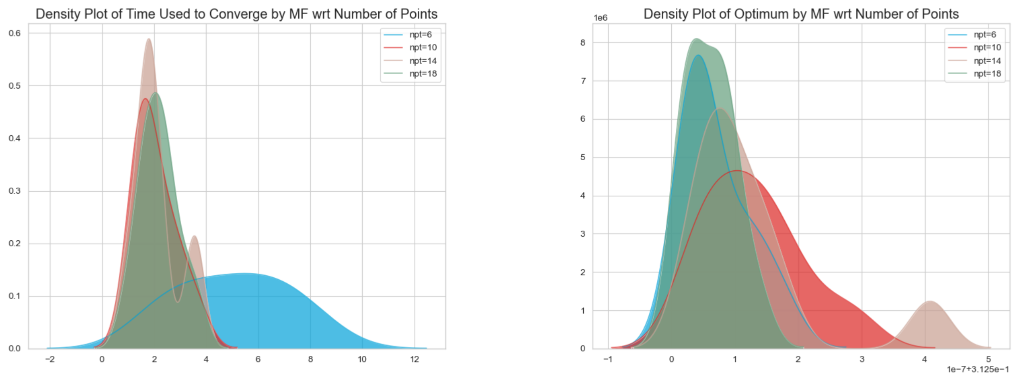

Figure 8: (Best viewed in color) Distribution of time and returned objective function value. There is one trial for npt=14 that broke down, which led to poor performance.

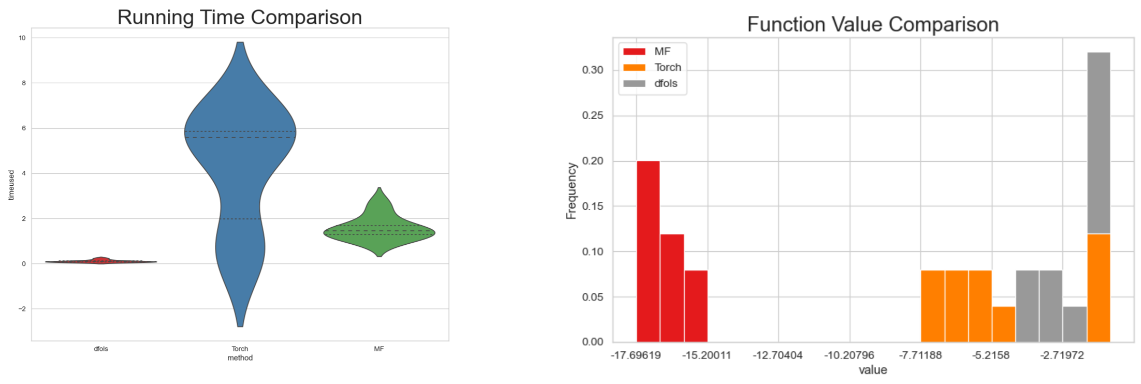

Figure 9: (Best viewed in color) Comparison result of three model. Function value is viewed in log scale of difference with oracle obtained by grid search. MF achieves best precision in relatively short time.

C Benchmark Problem Set

In this appendix, we provide details about benchmark from [13]. Rigorous formulation are as follows.

1. Generalization of MAXQ

$$f(\mathbf{x}) = \max_{1\le i\le n} x_i^2$$
$$x_i^{(1)} = i, \quad \text{for } i = 1,\ldots,n/2 \text{ and}$$
$$x_i^{(1)} = -i, \quad \text{for } i = n/2+1,\ldots,n.$$

2. Generalization of MXHILB

$$f(\mathbf{x}) = \max_{1\le i\le n} \left|\sum_{j=1}^{n} \frac{x_j}{i+j-1}\right|$$
$$x_i^{(1)} = 1, \quad \text{for all } i = 1,\ldots,n$$

3. Chained LQ

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \max\left\{-x_i - x_{i+1}, -x_i - x_{i+1} + \left(x_i^2 + x_{i+1}^2 - 1\right)\right\}$$
$$x_i^{(1)} = -0.5, \text{ for all } i = 1,\ldots,n$$

4. Chained CB3 1

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \max\left\{x_i^4 + x_{i+1}^2, (2-x_i)^2 + (2-x_{i+1})^2, 2e^{-x_i+x_{i+1}}\right\}$$
$$x_i^{(1)} = 2, \quad \text{for all } i = 1,\ldots,n$$

5. Chained CB3 2

$$f(\mathbf{x}) = \max\left\{\sum_{i=1}^{n-1}\left(x_i^4 + x_{i+1}^2\right), \sum_{i=1}^{n-1}\left((2-x_i)^2 + (2-x_{i+1})^2\right)\right.$$
$$\left.\sum_{i=1}^{n-1}\left(2e^{-x_i+x_{i+1}}\right)\right\}$$
$$x_i^{(1)} = 2, \quad \text{for all } i = 1,\ldots,n$$

6. Number of Active Faces

$$f(\mathbf{x}) = \max_{1\le i\le n}\left\{g\left(-\sum_{i=1}^{n} x_i\right), g\left(x_i\right)\right\}$$
where $g(y) = \ln(|y|+1)$
$$x_i^{(1)} = 1, \quad \text{for all } i = 1,\ldots,n$$

7. Chained Mifflin

$$f(\mathbf{x}) = \sum_{i=1}^{n-1}\left(-x_i + 2\left(x_i^2 + x_{i+1}^2 - 1\right) + 1.75\left|x_i^2 + x_{i+1}^2 - 1\right|\right)$$
$$x_i^{(1)} = -1, \quad \text{for all } i = 1,\ldots,n$$

8. Chained Crescent 1

$$f(\mathbf{x}) = \max\left\{\sum_{i=1}^{n-1}\left(x_i^2 + (x_{i+1}-1)^2 + x_{i+1} - 1\right)\right.$$
$$\left.\sum_{i=1}^{n-1}\left(-x_i^2 - (x_{i+1}-1)^2 + x_{i+1} + 1\right)\right\}$$

$$x_i^{(1)} = -1.5, \text{ when } \bmod(i,2) = 1, (i = 1,\ldots,n) \text{ and}$$
$$x_i^{(1)} = 2.0, \quad \text{when} \quad \bmod(i,2) = 0, (i = 1,\ldots,n).$$

9. Chained Crescent 2

$$f(\mathbf{x}) = \sum_{i=1}^{n-1} \max\left\{x_i^2 + (x_{i+1}-1)^2 + x_{i+1} - 1, -x_i^2 - (x_{i+1}-1)^2 + x_{i+1} + 1\right\}$$
$$x_i^{(1)} = -1.5, \text{ when } \bmod(i,2) = 1, (i = 1,\ldots,n) \text{ and}$$
$$x_i^{(1)} = 2.0, \quad \text{when} \quad \bmod(i,2) = 0, (i = 1,\ldots,n).$$

Note that problem 6 is in fact not proper here since it doesn't satisfy fully linearity 2.8, but we keep it for completeness of the test set. For properties of the test functions, we refer to Table 2.

Table 2: Details of test problem set.

Index	Optimal Value	Convexity	Non-smoothness(h)	L_h	m
1	0	Y	$max(\cdot)$	1	n
2	0	Y	$max(\|\cdot\|)$		n
3	$-(n-1)\sqrt{2}$	Y	$\sum max(\cdot)$		2(n-1)
4	2(n-1)	Y	$\sum max(\cdot)$		3(n-1)
5	2(n-1)	Y	$max(\cdot)$		3
6	0	N	$max(\cdot)$		n+1
7	Varies*	N	$\|\cdot\|$		n-1
8	0	N	$max(\cdot)$		2
9	0	N	$\sum max(\cdot)$		2(n-1)

* Optimal value is approximately -6.51 for n=10,
-70.15 for n=100 and -706.55 for n=1000