A SUBGRADIENT METHOD BASED ON GRADIENT SAMPLING FOR SOLVING CONVEX OPTIMIZATION PROBLEMS

Yaohua Hu[*], Chee-Khian Sim[†], Xiaoqi Yang[‡]

Abstract Based on the gradient sampling technique, we present a subgradient algorithm to solve the nondifferentiable convex optimization problem with an extended real-valued objective function. A feature of our algorithm is the approximation of subgradient at a point via random sampling of (relative) gradients at nearby points, and then taking convex combinations of these (relative) gradients. We prove that our algorithm converges to an optimal solution with probability 1. Numerical results demonstrate that our algorithm performs favorably compared with existing subgradient algorithms on applications considered.

Keywords Gradient sampling technique; Subgradient method; Projection; Convex optimization.

Mathematics Subject Classification 90C25; 65K05; 49M37.

1 Introduction

Subgradient methods are popular and practical techniques used to minimize a nondifferentiable convex function. Because of their simple formulations and low storage requirements, subgradient methods can potentially be applied to a wide variety of problems. Subgradient methods originated with the works of Shor [32], Polyak [27] and Ermoliev [9] in the 1960s. In the last 50 years, many properties of subgradient methods have been discovered, generalizations and extensions have been proposed, and many applications have been found (see [2, 3, 12, 14, 17, 18, 24, 25, 28, 32]). Nowadays, the subgradient method still remains an important tool for large-scale nonsmooth optimization and stochastic optimization problems, due to its simple formulation and low storage requirement.

Nedić and Ozdaglar [25] propose a dual subgradient method to solve the resource allocation problems in large-scale networks. Since these constrained primal problems have favorable dual structures, the dual subgradient method achieves a highly efficient performance. In the dual subgradient algorithm, the authors generate the subgradient information in the dual

[*]College of Mathematics and Statistics, Shenzhen University, Shenzhen 518060, P. R. China (mayhhu@szu.edu.cn).

[†]Department of Mathematics, University of Portsmouth, Lion Gate Building, Lion Terrace, Portsmouth PO1 3HF, United Kingdom (chee-khian.sim@port.ac.uk).

[‡]Department of Applied Mathematics, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong (mayangxq@polyu.edu.hk).

space. Using the subgradient information and an averaging scheme, they construct an approximate primal solution with an explicit error estimate. Along similar lines, Larsson et al. [22] use an averaging scheme to show that elements of the averaged subgradient sequence satisfy the optimality conditions in the limit, while the original generated sequence does not satisfy these optimality conditions.

Combining the primal and dual processes, Larsson et al. [23], Nedić and Ozdaglar [25] and Nesterov [26] develop primal-dual subgradient methods for nondifferentiable convex problems with several classical types of penalty function and stepsize. The primal-dual subgradient algorithm generates a sequence of primal and dual iterates for which some subsequence converges to a pair of primal-dual optimal solutions. An important improvement gained by these primal-dual subgradient algorithms is that they possess a natural stopping criterion, which is unavailable in purely primal or purely dual subgradient methods.

Nedić and Bertsekas [24] develop an incremental subgradient method to minimize a convex function that is a summation of a large number of component convex functions. This type of convex function appears in large-scale least squares problems such as in the training of neural networks. The main idea of incremental subgradient methods is to perform each iteration as a cycle of some subiterations. Using previous subiterates, each subiteration is a subgradient algorithm iteration on a component function. Numerical results in [24] indicate that the incremental subgradient method converges fast when iterates are far from the optimal solution.

Kim and Ahn [15] demonstrate the convergence of generalized subgradient method, which approximates the current subgradient by subgradients at previous iterations. Goffin and Kiwiel [11] and Kiwiel [20, 21] exhibit some useful properties of ballstep subgradient methods, which use projections onto successive approximations of level sets to evaluate the optimal value. Gasimov [10] and Burachik et al. [5] applied a modified subgradient algorithm to the dual problem defined by the sharp augmented Lagrangian, whose primal problem is a nonconvex minimization problem with equality constraints. The authors not only establish primal and dual convergence results, but also generate a strictly increasing sequence of dual values. This monotone property is impossible in other types of subgradient method. Beck and Teboulle [1] view the mirror descent algorithm as a nonlinear projected-subgradient type method with the usual Euclidean distance function replaced by a general distance-like function.

Recently, gradient sampling technique is used in designing algorithms for optimization problems. The gradient sampling (in short, GS) technique is first presented by Burke et al. [7] to solve typical matrix optimization problems. In another of their work [6], they demonstrate that the Clarke subdifferential at a point can be approximated by the convex hull of gradients sampled at random nearby points. Extending their previous works, in Burke et al. [8], they design a steepest descent GS algorithm to minimize a locally Lipschitz function.

In this paper, we consider a nondifferentiable convex optimization problem with an extended real-valued objective function, where the domain of the objective function may have an empty interior. To solve this problem, we incorporate the GS technique into the subgradient method, which is to approximate the subdifferential via random sampling of relative gradients at nearby points. If the domain has an empty interior, the random sampling process cannot be carried out on the whole space $\mathbb{R}^n$, but on the affine hull spanned by the domain of the objective function. Because a gradient cannot be defined in the domain whose interior is empty, we use the relative gradient (Definition 2.1) instead. Furthermore, as each iterate is not necessarily a relative interior point of the domain, we perform a perturbation step, which perturbs the projected vector to the relative interior of the domain, to ensure the sampling process can be carried out.

The motivation for introducing our algorithm is that in applications, choosing a suitable subgradient in the subdifferential at a point can be essential for good performance – "In the nondifferentiable case, we have some flexibility in the gradient selection and the choice of (sub)gradients may affect the quality of the bound." (pp. 14-15 of [4]) – but it may be difficult to find such a subgradient. Our algorithm circumvents this by considering gradients at nearby points instead. These gradients are unique and hence there is no issue of choosing a suitable subgradient among infinitely many subgradients in a subdifferential.

Our algorithm is an implementation of the approximate subgradient method as discussed in [19]. We introduce the GS procedure in our algorithm, which gets the approximate subgradient via convex combination of relative gradients sampled at random nearby points. Even though our algorithm can be seen as a special case of the approximate subgradient method, the numerical experiments show that our algorithm performs better than the classical subgradient method, as well as variants of the subgradient method. This is the advantage of our algorithm over the approximate subgradient method. Due to the convex structure, our algorithm, applying the GS technique, is easy to implement, which does not need to solve a subproblem to find the search direction and the stepsize, as in [8]. The easy implementation is the advantage of our algorithm over the steepest descent GS algorithm [8].

In Section 5, we illustrate our algorithm on three examples. Our numerical experiments show that the GS procedure does not take much time in the whole algorithm and our algorithm always requires fewer iterations, costs less time or achieves the better optimal values than existing subgradient algorithms. The numerical results show the promise of our method as compared to other types of subgradient method. Especially for solving the affine rank minimization problem using nuclear norm, our algorithm takes one third or half of the computation time that is required for the ordinary subgradient method.

This paper is organized as follows. In Section 2, we present notations used in the paper and also our subgradient algorithm based on the GS technique. In Section 3, we demonstrate the convergence of our algorithm. In Section 4, we compute the perturbation direction, which we introduce in this paper, for two common types of domain. This perturbation direction plays a key role in our algorithm. Finally we exhibit several numerical results in Section 5.

2 The subgradient method based on gradient sampling

In this paper, we consider the Euclidean space $\mathbb{R}^n$, view a vector as a column vector, and denote by $\langle x, y\rangle$ the inner product of two vectors $x, y \in \mathbb{R}^n$. We use $\|x\|$ to denote the standard Euclidean norm, $\|x\| = \sqrt{\langle x, x\rangle}$. For a set $Z \subseteq \mathbb{R}^n$, we denote the closure (resp. interior, convex hull, affine hull, relative interior, relative boundary) of Z by cl$Z$ (resp. int$Z$, conv$Z$, aff$Z$, ri$Z$, rbd$Z$). For $x \in \mathbb{R}^n$ and $\delta \in \mathbb{R}_+$, we use $B(x, \delta)$ to denote the closed Euclidean ball centered at x of radius δ. For a convex set Z, the Euclidean distance dist(x, Z) of x from Z, the projection $P_Z(x)$ of x onto Z and the normal cone to Z at x are respectively defined by

$$d_Z(x) := \inf_{z\in Z} \|x - z\|,$$

$$P_Z(x) := \{z \in Z : \|x - z\| = \operatorname{dist}(x, Z)\} = \arg\min_{z\in Z} \|x - z\|,$$

and

$$N_Z(x) := \{\nu \in \mathbb{R}^n : \langle \nu, z - x\rangle \leq 0, \forall z \in Z\}.$$

Given a nonsmooth convex function $f : \mathbb{R}^n \to \bar{\mathbb{R}}(:= \mathbb{R} \cup \{+\infty\})$, the (effective) domain and the subdifferential of f at $x \in \mathbb{R}^n$ are respectively defined by

$$\operatorname{dom} f := \{x \in \mathbb{R}^n : \ f(x) < +\infty\},$$

and

$$\partial f(x) := \{g : f(y) \geq f(x) + \langle g, y - x\rangle, \forall y \in \mathbb{R}^n\}.$$

In this paper, we consider the following convex optimization problem

$$\text{(P)} \qquad \min_{x\in\mathbb{R}^n} \quad f(x), \tag{2.1}$$

where $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ is a proper closed convex function which may be nonsmooth. Throughout this paper, we let $X := \operatorname{dom} f$, V be the subspace parallel to aff$X$, and X_* and f_* be the optimal solution set and optimal value of (P), respectively.

The main idea of the subgradient method is to generalize the gradient method by replacing the gradient with an arbitrary subgradient. The iteration formula is given by

$$x_{k+1} = P_X(x_k - v_k g_k), \tag{2.2}$$

where $v_k > 0$ is the stepsize and $g_k \in \partial f(x_k)$. When f is continues differentiable at x_k, the only choice for g_k is $\nabla f(x_k)$, and the subgradient method is reduced to the gradient method. In practice, the ϵ-subgradient is usually considered due to the application and computation errors

$$\partial_\epsilon f(x_k) := \{g : f(x) \geq f(x_k) + \langle g, x - x_k\rangle - \epsilon, \forall x \in \mathbb{R}^n\}.$$

When int(dom$f$) $= \varnothing$, the gradient of f cannot be defined. The relative gradient of f is considered instead in such case. The definition of relative gradient is given as follows.

Definition 2.1 ([13]). *Let $X = \mathrm{dom} f$, V be the subspace parallel to $\mathrm{aff} X$. A vector $g \in V$ is called the relative gradient of f at x with respect to X, denoted by $\nabla_X f(\bar{x})$, if*

$$f(x+h) = f(x) + \langle g, h \rangle + o(\|h\|), \quad \forall h \in V.$$

If such g exists, we say that f is relatively differentiable at x.

It is easy to see that the relative gradient, if exists, is unique. To understand Definition 2.1, for given $x_0 \in X$, we introduce a new convex function $f_0 := f(x_0 + \cdot)$. This transformation makes the domain of f_0 full-dimensional on the subspace V, and thus f_0 is differentiable almost everywhere on $\mathrm{int}(\mathrm{dom} f_0) \subset V$, that is, f is relatively differentiable almost everywhere on $\mathrm{ri} X$ (see [13, Page 17]). Actually, $\nabla_X f(x)$ is the gradient of f_0 at $x - x_0$.

Now we present a subgradient method based on gradient sampling technique (in short, sampling SGM) for solving (2.1). Since we calculate relative gradients at points in a certain neighborhood of the current iterate in the GS technique, we need all iterates to be relative interior points of the domain. Hence, if an iterate is not a relative interior point of X, we perform a perturbation step to guarantee the iterate be a relative interior point (Step 3 of algorithm) and to ensure the GS technique can succeed. Therefore, the sampling SGM consists of generating a sequence $\{x_k\}$, where x_{k+1} is obtained from x_k by first moving along a direction g_k, constructed via random sampling of relative gradients at nearby points of x_k, to a new point. x_{k+1} is then obtained from the new point by projection onto X and then taking a perturbation step.

In the following algorithm, $\mathcal{D}$ denotes the set of all points in X where f is relatively differentiable, and v_k (resp. δ_k, μ_{ki}, α_k) denotes the stepsize (resp. sampling radius, sampling directions, perturbation weight) at the k-th iteration.

Subgradient method based on the gradient sampling technique (sampling SGM).

Step 1. (Initialization)

Start from $k = 0$, select an initial point $x_0 \in \mathrm{ri} X$, a sample size s, stepsizes $\{v_k\}$ and perturbation weights $\{\alpha_k\}$ with $\alpha_k \in (0, 1)$.

Step 2. (Generate the approximate subdifferential by gradient sampling technique)

Let $\mu_{k1}, \cdots, \mu_{ks}$ be sampled independently and uniformly from $B(0,1) \cap V$. Choose the sampling radius to satisfy $0 < \delta_k \le d_{\mathrm{rbd} X}(x_k)$. Set

$$x_{ki} = x_k + \delta_k \mu_{ki}, \quad i = 1, \ldots, s. \tag{2.3}$$

If for some $i = 1, \ldots, s$, the point $x_{ki} \notin \mathcal{D}$, then STOP; otherwise, set

$$G_k = \mathrm{conv}\{\nabla_X f(x_{k1}), \ldots, \nabla_X f(x_{ks})\},$$

and choose an arbitrary element g_k in G_k as an approximate subgradient of f at x_k, i.e.,

$$g_k = \sum_{i=1}^{s} \lambda_{ki} \nabla_X f(x_{ki}), \text{ with } \sum_{i=1}^{s} \lambda_{ki} = 1 \text{ and } \lambda_{ki} \ge 0.$$

Go to Step 3.

Step 3. (Solution update and perturbation)

Compute $\bar{x}_{k+1} = P_X(x_k - v_k g_k)$ by solving the convex program

$$\begin{array}{ll} \min & \|x - (x_k - v_k g_k)\|^2 \\ \text{s.t.} & x \in X. \end{array}$$

Compute the perturbation vector y_k such that

$$y_k \in \{\bar{x}_{k+1} - N_X(\bar{x}_{k+1})\} \cap \mathrm{ri}X \cap B(\bar{x}_{k+1}, 1) \tag{2.4}$$

and set

$$x_{k+1} = (1 - \alpha_k)\bar{x}_{k+1} + \alpha_k y_k, \text{ with } 0 < \alpha_k < 1. \tag{2.5}$$

Set $k = k + 1$ and go back to Step 2.

The following remarks explain the choice of parameters and the design of this algorithm.

Remark 2.1. *In Step 2, we choose $\delta_k \le d_{\mathrm{rbd}X}(x_k)$ to keep all sampling points x_{ki} in X. Even though x_{k_i} is in X, x_{ki} may not be in $\mathcal{D}$. In this case, our algorithm will stop and it turns out to be failed. Fortunately, in the proof of Theorem 3.2, we show that our algorithm will not terminate in Step 2, that is, an infinite sequence $\{x_k\}$ will be generated by our algorithm, with probability 1.*

Remark 2.2. *In (2.4) of Step 3, y_k lies in $\mathrm{ri}X$ to ensure that $x_{k+1} \in \mathrm{ri}X$. Moreover, y_k belongs to $\{\bar{x}_{k+1} - N_X(\bar{x}_{k+1})\} \cap B(\bar{x}_{k+1}, 1)$ ensures convergence of our algorithm. We show the existence of such y_k in Lemma 3.3, and how it can be calculated for two common domains in Section 4.*

There are two main differences between the sampling SGM and the ordinary subgradient method (in short, ordinary SGM). The first difference is the random sampling and subgradient approximation processes. The ordinary SGM always directly calculates and utilizes the subgradient information, while our sampling SGM generates the subgradient by calculating the convex hull of relative gradients sampled at random nearby points. Secondly, it is the perturbation step. The ordinary SGM performs a projection operation after each solution updating step. It makes each iterate x_{k+1} a feasible point which might not be a relative interior point of X. On the other hand, the sampling SGM performs a perturbation (2.5) after each projection operation. It makes each iterate x_{k+1} a relative interior point of X. Note that if $\bar{x}_{k+1}$ is already a relative interior point of X, then we have $y_k = \bar{x}_{k+1}$. Therefore, if the optimal solution is a relative interior point of X, no perturbation is needed when close to the optimal solution.

3 Convergence analysis

In this section, we prove the convergence of the sampling SGM. We first state the following two lemmas, which show some basic properties of the approximate subgradient.

Lemma 3.1 ([16, Lemma 3.1 (i)]). *Let g_1 be a subgradient of f at $x_1 \in X$. Then, for any $x_2 \in X$, g_1 is a ϵ-subgradient of f at x_2 with $\epsilon = f(x_2) - f(x_1) - \langle g_1, x_2 - x_1\rangle$.*

Lemma 3.2 ([16, Lemma 3.1 (ii)]). *Let g_i be an ϵ_i-subgradient of f at $x \in X$ for $i = 1, \ldots, s$. Then, the convex combination $\sum_{i=1}^s \lambda_{ki} g_i$ is an ϵ-subgradient of f at x with $\epsilon = \sum_{i=1}^s \lambda_{ki}\epsilon_i$.*

From Lemmas 3.1-3.2, it follows that the direction $g_k \in G_k$, generated in Step 2 of the sampling SGM, is an approximate subgradient direction. Indeed, when $x_{ki} \in \mathcal{D}$, $\nabla_X f(x_{ki})$ is a relative gradient of f at x_{ki}. Thus, by using Lemma 3.1, we have that $\nabla_X f(x_{ki})$ is a ϵ_{ki}-subgradient of f at x_k with $\epsilon_{ki} = f(x_k) - f(x_{ki}) + \langle \nabla_X f(x_{ki}), \delta_k \mu_{ki}\rangle$. Furthermore, from Lemma 3.2, it follows that the convex combination $g_k = \sum_{i=1}^s \lambda_{ki} \nabla_X f(x_{ki})$ is also a ϵ_k-subgradient of f at x_k with $\epsilon_k = \sum_{i=1}^s \lambda_{ki}\epsilon_{ki}$, that is

$$\begin{aligned} f(x) &\geq f(x_k) + \langle g_k, x - x_k\rangle - \epsilon_k \\ &= \langle g_k, x - x_k\rangle + \textstyle\sum_{i=1}^s \lambda_{ki} f(x_{ki}) - \sum_{i=1}^s \lambda_{ki}\langle \nabla_X f(x_{ki}), \delta_k\mu_{ki}\rangle, \quad \forall x \in \mathbb{R}^n. \end{aligned} \tag{3.1}$$

The following lemma is very important for our algorithm. It demonstrates that Step 3 of the sampling SGM is well-defined in that it guarantees the existence of perturbation vector.

Lemma 3.3. *Let $C \subseteq \mathbb{R}^n$ be a nonempty closed convex set, then for each $x \in C$, the intersection $\{x - N_C(x)\} \cap \mathrm{ri}C \cap B(x, 1)$ is nonempty.*

Proof. This lemma follows if we show that

$$(-N_C(x)) \cap \big\{ -x + \mathrm{ri}(C \cap B(x,1))\big\} \neq \varnothing,$$

noting that $\mathrm{ri}C \cap B(x,1) = \mathrm{ri}(C \cap B(x,1))$ (see [13, Proposition 2.1.10]).
By contradiction, suppose that$(-N_C(x)) \cap \big(-x + \mathrm{ri}(C \cap B(x,1))\big) = \varnothing$. By the separation theorem for convex sets (see [13]), there exists a vector $s \neq 0$, such that

$$\begin{aligned} \langle s, -y\rangle \geq 0, &\qquad \forall y \in N_C(x), \\ \langle s, -x + z\rangle < 0, &\qquad \forall z \in \mathrm{ri}(C \cap B(x,1)) \end{aligned} \tag{3.2}$$

Taking z in the closure of $\mathrm{ri}(C \cap B(x,1))$, the last inequality holds as

$$\langle s, -x + z\rangle \leq 0, \quad \forall z \in C \cap B(x,1),$$

which is equivalent to $s \in N_C(x)$. From (3.2) it then follows that $\langle s, -s\rangle \geq 0$, which implies $s = 0$. This is a contradiction to the separation theorem for convex sets. ■

Throughout this paper, we have the following assumptions which are commonly used when we study convex programs.

(A1) The optimal solution set X_* is nonempty.

(A2) The relative gradients of f are bounded, i.e., there exists some scalar M such that $\|\nabla_X f(x)\| \le M$ for all $x \in \mathcal{D}$.

It is well-known that the stepsize rule is critical in subgradient methods. In this paper, we investigate convergence property of the sampling SGM using the following stepsize rules.

(a) *Constant stepsize rule.* The stepsize $v_k \equiv v > 0$.

(b) *Divergence stepsize rule.* The stepsize v_k satisfies

$$v_k > 0, \quad \sum_{k=0}^{+\infty} v_k = +\infty, \quad \sum_{k=0}^{+\infty} v_k^2 < +\infty. \tag{3.3}$$

Theorem 3.1. *Let Assumptions* (A1) *and* (A2) *hold. Suppose the sequence* $\{x_k\}$ *is generated by the sampling SGM with parameters* δ_k *and* α_k *satisfying* $\delta_k \le d_{\mathrm{rbd}X}(x_k)$, $\alpha_k \in (0,1)$, $\sum_{k=0}^{+\infty} \alpha_k^2 < +\infty$, *and the constant stepsize* v. *Then,* $\varliminf_{k\to\infty} f(x_k) \le f_* + vM^2/2$ *with probability 1.*

Proof. We begin the proof by making an observation concerning the stochastic structure of the sampling SGM. We first consider the case when the algorithm terminates finitely. Note that $x_k \in \mathrm{ri}X$, $\delta_k > 0$, and μ_{ki} is a realization of a random variable that is uniformly distributed on $B(0,1) \cap V$. Since f is relatively differentiable almost everywhere on $\mathrm{ri}X$ (also on X), by measure theory, the probability that $x_{ki} \notin \mathcal{D}$ is zero for each i and k. Therefore, with probability 1, the algorithm does not terminate in Step 2.
We now restrict our attention to the case when the algorithm does not terminate finitely. According to our sampling SGM, for all $x \in X$, we have

$$\begin{aligned} \|x_{k+1} - x\|^2 &= \|\bar{x}_{k+1} - x - \alpha_k(\bar{x}_{k+1} - y_k)\|^2 \\ &= \|\bar{x}_{k+1} - x\|^2 - 2\alpha_k\langle \bar{x}_{k+1} - x, \bar{x}_{k+1} - y_k\rangle + \alpha_k^2\|\bar{x}_{k+1} - y_k\|^2. \end{aligned} \tag{3.4}$$

Since $y_k \in \{\bar{x}_{k+1} - N_X(\bar{x}_{k+1})\}$, we have

$$\langle \bar{x}_{k+1} - x, \bar{x}_{k+1} - y_k\rangle \ge 0, \forall x \in X.$$

Furthermore, $y_k \in B(\bar{x}_{k+1}, 1)$ implies that $\|\bar{x}_{k+1} - y_k\| \le 1$. Therefore, (3.4) is reduced to

$$\begin{aligned} \|x_{k+1} - x\|^2 &\le \|\bar{x}_{k+1} - x\|^2 + \alpha_k^2 \\ &\le \|x_k - vg_k - x\|^2 + \alpha_k^2 \\ &= \|x_k - x\|^2 - 2v\langle g_k, x_k - x\rangle + v^2\|g_k\|^2 + \alpha_k^2, \forall x \in X. \end{aligned} \tag{3.5}$$

By (3.1) and (3.5), for all $x \in X$, we obtain

$$\begin{aligned} \|x_{k+1} - x\|^2 &\le \|x_k - x\|^2 - 2v(\textstyle\sum_{i=1}^s \lambda_{ki} f(x_{ki}) - f(x)) \\ &\quad + 2v(\textstyle\sum_{i=1}^s \lambda_{ki}\langle \nabla_X f(x_{ki}), \delta_k\mu_{ki}\rangle) + v^2\|g_k\|^2 + \alpha_k^2 \\ &\le \|x_k - x\|^2 - 2v(\textstyle\sum_{i=1}^s \lambda_{ki} f(x_{ki}) - f(x)) + 2v\delta_k M + v^2M^2 + \alpha_k^2, \end{aligned} \tag{3.6}$$

where the second inequality follows from (A2) and that sampling points are in the unit ball. We denote $x_k + \delta_k \mu_k = \sum_{i=1}^{s} \lambda_{ki} x_{ki}$, where $\mu_k := \sum_{i=1}^{s} \lambda_{ki} \mu_{ki}$. By the convexity of f and (3.6), for all $x \in X$, we obtain

$$\begin{aligned} 2v(f(x_k + \delta_k \mu_k) - f(x)) &\le 2v(\textstyle\sum_{i=1}^{s} \lambda_{ki} f(x_{ki}) - f(x)) \\ &\le \|x_k - x\|^2 - \|x_{k+1} - x\|^2 + 2v\delta_k M + v^2 M^2 + \alpha_k^2. \end{aligned} \tag{3.7}$$

Summing (3.7) over $k = 0, \ldots, n$, for all $x \in X$, we obtain

$$\frac{\sum_{k=0}^{n} f(x_k + \delta_k \mu_k)}{n} - f(x) \le \frac{\|x_0 - x\|^2}{2nv} + \frac{M \sum_{k=0}^{n} \delta_k}{n} + \frac{vM^2}{2} + \frac{\sum_{k=0}^{n} \alpha_k^2}{2nv}. \tag{3.8}$$

By the assumptions that $\sum_{k=0}^{\infty} \alpha_k^2 < +\infty$ and that $\delta_k \le d_{\mathrm{rbd}X}(x_k) \le \|x_k - \bar{x}_k\| \le \alpha_{k-1}$, we obtain $\lim \sum_{k=0}^{n} \delta_k / n = 0$ (cf. [19, Lemma 2.1]). Thus, by using [19, Lemma 2.1], the relation (3.8) implies

$$\varliminf_{k \to \infty} f(x_k) \le \varliminf_{n \to \infty} \frac{\sum_{k=0}^{n} f(x_k + \delta_k \mu_k)}{n} \le f(x) + \frac{vM^2}{2}, \quad \forall x \in X,$$

Therefore, we have proved that $\varliminf_{k \to \infty} f(x_k) \le f_* + vM^2/2$ with probability 1. ■

Theorem 3.2. *Let Assumptions* (A1) *and* (A2) *hold. Suppose the sequence* $\{x_k\}$ *is generated by the sampling SGM with parameters* δ_k *and* α_k *satisfying* $\delta_k \le d_{\mathrm{rbd}X}(x_k)$, $\alpha_k \in (0,1)$, $\sum_{k=0}^{+\infty} \alpha_k^2 < +\infty$, *and the divergence stepsize rule* (3.3). *Then,* x_k *converges to some* $x_* \in X_*$ *and* $\varliminf_{k \to \infty} f(x_k) = f_*$, *with probability 1.*

Proof. Similar to the proof of Theorem 3.1, by using (3.3), we have that the algorithm generates an infinite sequence and $\varliminf_{k \to \infty} f(x_k) = f_*$ with probability 1. Then we prove the convergence of sequence $\{x_k\}$ as follows.
From inequality (3.6), if we use any $\bar{x} \in X_*$ instead of x, we have

$$\begin{aligned} \|x_{k+1} - \bar{x}\|^2 &\le \|x_k - \bar{x}\|^2 - 2v_k(\textstyle\sum_{i=1}^{s} \lambda_{ki} f(x_{ki}) - f(\bar{x})) + 2v_k \delta_k M + v_k^2 M^2 + \alpha_k^2 \\ &\le \|x_k - \bar{x}\|^2 + 2v_k \delta_k M + v_k^2 M^2 + \alpha_k^2. \end{aligned} \tag{3.9}$$

The hypotheses, as well as (3.3), imply that

$$\sum_{k=0}^{\infty} \left(2v_k \delta_k M + v_k^2 M^2 + \alpha_k^2\right) < \infty. \tag{3.10}$$

This implies that the sequence $\{x_k\}$ is bounded. Furthermore, as we have proved that $\varliminf_{k \to \infty} f(x_k) = f_*$, one has that $\{x_k\}$ has a cluster point $x_* \in X_*$. Finally, $\{x_k\}$ converges to x_* from (3.9), using x_* in place of $\bar{x}$, noting that the tail sum $\sum_{i=k}^{\infty} \left(2v_i \delta_i M + v_i^2 M^2 + \alpha_i^2\right)$ vanishes as k tends to infinity. ■

4 Calculating the perturbation vector for two common domains

The perturbation vector, y_k, plays a key role in the sampling SGM to guarantee that we obtain relative gradients at nearby points of each iteration and to achieve the convergence properties. We have proved its existence in Lemma 3.3. In this section, we show how this vector can be calculated for two common cases of the domain X: X is a convex polyhedron or sublevel set of some convex quadratic functions. For simplicity, we denote $\bar{x} := \bar{x}_{k+1}$ in what follows. We need to find $\bar{s} \in \mathbb{R}^n$ to satisfy $\bar{x} - \bar{s} \in \mathrm{ri}X$, $\bar{s} \in N_X(\bar{x})$ and $\|\bar{s}\| \leq 1$. The perturbation vector y_k can then be found by setting $y_k := \bar{x} - \bar{s}$.

4.1 Convex Polyhedron

Let X be a convex polyhedron in $\mathbb{R}^n$, i.e.,

$$X := \{x : \langle a_i, x\rangle \leq b_i, \|a_i\| = 1, i = 1, 2, \ldots, m\}.$$

For each $\bar{x} \in X$, let the active index set be

$$J(\bar{x}) := \{i : \langle a_i, \bar{x}\rangle = b_i, i = 1, 2, \ldots, m\}.$$

It is well-known that the normal cone to X at $\bar{x}$ is given by

$$\begin{aligned} N_X(\bar{x}) &= \mathrm{cone}\{a_j : j \in J(\bar{x})\} \\ &= \{\textstyle\sum_{j\in J(\bar{x})} \beta_j a_j : \beta_j \geq 0\}. \end{aligned}$$

Since we need $\bar{s} \in N_X(\bar{x})$, $\bar{s} = \sum_{j\in J(\bar{x})} \beta_j a_j$ with $\beta_j \geq 0$ for $j \in J(\bar{x})$. We next deduce conditions on β_j such that $\bar{x} - \bar{s} \in \mathrm{ri}X$, that is, $\langle a_i, \bar{x} - \bar{s}\rangle = \langle a_i, \bar{x} - \sum_{j\in J(\bar{x})} \beta_j a_j\rangle < b_i$, for $i = 1, 2, \ldots, m$. The deduction is divided into two cases: (a) $\bar{x} \in \mathrm{ri}X$ and (b) $\bar{x} \in \mathrm{rbd}X$.

(a) $\bar{x} \in \mathrm{ri}X$.

In this case, $J(\bar{x}) = \varnothing$, $\bar{s} \in N_X(\bar{x}) = \{0\}$, and $\bar{x} - \bar{s} = \bar{x} \in \mathrm{ri}X$. Hence, $\beta_j = 0, j \in J(\bar{x})$.

(b) $\bar{x} \in \mathrm{rbd}X$.

Deduction of conditions on β_j is split into two subcases based on the index of a_i: (i) $i \notin J(\bar{x})$ and (ii) $i \in J(\bar{x})$, as follows.

(i) $i \notin J(\bar{x})$.

In this case, we have $\langle a_i, \bar{x}\rangle < b_i$. Choose $\beta_j, j \in J(\bar{x})$, to satisfy

$$0 < \beta_j < \frac{1}{|J(\bar{x})|} \min\{\min_{i\notin J(\bar{x})} \frac{b_i - \langle a_i, \bar{x}\rangle}{\max\{-\langle a_i, a_j\rangle, 0\}}, 1\}. \tag{4.1}$$

Then we obtain $\langle a_i, \bar{x} - \bar{s}\rangle = \langle a_i, \bar{x}\rangle - \langle a_i, \sum_{j\in J(\bar{x})} \beta_j a_j\rangle < b_i$.

(ii) $i \in J(\bar{x})$.

In this case, we have $\langle a_i, \bar{x}\rangle = b_i$. Hence, to guarantee $\langle a_i, \bar{x} - \bar{s}\rangle < b_i$, we need to choose $\beta_j \in J(\bar{x})$ such that

$$\sum_{j\in J(\bar{x})} \beta_j \langle a_i, a_j\rangle > 0. \tag{4.2}$$

Therefore, for (a), $\bar{s} = 0$ satisfies $\bar{x} - \bar{s} \in \mathrm{ri}X, \bar{s} \in N_X(\bar{x})$ and $\|\bar{s}\| \le 1$. Hence, y_k can be found. For (b), we can find $\beta_j, j \in J(\bar{x})$ that satisfy both (4.1) and (4.2). To find $\beta_j, j \in J(\bar{x})$ that satisfy (4.2) for all $i \in J(\bar{x})$, we solve the following system of linear inequalities

$$\sum_{j\in J(\bar{x})} \beta_j \langle a_i, a_j\rangle \ge 1, \forall i \in J(\bar{x}),$$

in practice, instead. By scaling these $\beta_j, j \in J(\bar{x})$, appropriately so that $\|\sum_{j\in J(\bar{x})} \beta_j a_j\| \le 1$, we can then find $\bar{s}$ and hence the perturbation vector y_k.

Remark 4.1. *$\bar{s}$ and hence the perturbation vector y_k in the box case is particularly easy to calculate. For example, if $X = \mathbb{R}^n_+$, we obtain*

$$\bar{s} = \frac{sign(\bar{x}) - e}{\sqrt{n}},$$

where $e = (1, 1, \dots, 1)^T$ and $sign(\cdot)$ denotes the sign function.

4.2 Sublevel set of some convex quadratic functions

Define

$$f_i(x) := \frac{1}{2} x^T Q_i x + c_i^T x + d_i, \quad i \in I = \{1, 2, \dots, m\},$$

where Q_i is symmetric, positive semi-definite matrix, $c_i \in \mathbb{R}^n$ and $d_i \in \mathbb{R}$. Let X be the sublevel set of these convex quadratic functions, i.e.,

$$X := \{x : f(x) \le 0\},$$

where

$$f(x) := \max_{i\in I} f_i(x).$$

For each $\bar{x} \in X$, let the active index set be

$$J(\bar{x}) := \{i : f_i(\bar{x}) = 0, i \in I\}.$$

Again, we want to find the perturbation vector $y_k := \bar{x} - \bar{s}$, where $\bar{s}$ satisfies $\bar{x} - \bar{s} \in \mathrm{ri}X, \bar{s} \in N_X(\bar{x})$ and $\|\bar{s}\| \le 1$. We do this by looking at the only two situations that $\bar{x}$ can be in.

(a) $\bar{x} \in \mathrm{ri}X$.

In this case, $J(\bar{x}) = \varnothing$, $\bar{s} \in N_X(\bar{x}) = \{0\}$, and $\bar{x} - \bar{s} = \bar{x} \in \mathrm{ri}X$.

(b) $\bar{x} \in \operatorname{rbd} X$.

From [30, Proposition 10.3], by the regularity of f, the normal cone to X at $\bar{x}$ is given by

$$\begin{aligned} N_X(\bar{x}) &= \operatorname{cone}\{\nabla f_i(\bar{x}) : i \in J(\bar{x})\} \\ &= \operatorname{cone}\{Q_i\bar{x} + c_i : i \in J(\bar{x})\} \\ &= \{\textstyle\sum_{j\in J(\bar{x})} \beta_j(Q_j\bar{x} + c_j) : \beta_j \geq 0\}. \end{aligned}$$

Since we need $\bar{s} \in N_X(\bar{x})$, it can be represented as $\bar{s} = \sum_{j\in J(\bar{x})} \beta_j(Q_j\bar{x} + c_j)$ with $\beta_j \geq 0$. Let $\beta := (\beta_1, \ldots, \beta_{|J(\bar{x})|})^T \in \mathbb{R}^{|J(\bar{x})|}$, $a_j := Q_j\bar{x} + c_j \in \mathbb{R}^n$ and $A := (a_1, \ldots, a_{|J(\bar{x})|}) \in \mathbb{R}^{n\times|J(\bar{x})|}$, then $\bar{s} = A\beta$. In the following, we deduce conditions on β such that $\bar{x} - \bar{s} \in \operatorname{ri} X$, that is, $f_i(\bar{x} - \bar{s}) < 0$, for all $i \in I$. Note that

$$\begin{aligned} f_i(\bar{x} - \bar{s}) &= \tfrac{1}{2}(\bar{x} - \bar{s})^T Q_i(\bar{x} - \bar{s}) + c_i^T(\bar{x} - \bar{s}) + d_i \\ &= \tfrac{1}{2}\bar{s}^T Q_i\bar{s} - \bar{x}^T Q_i\bar{s} - c_i^T\bar{s} + f_i(\bar{x}) \\ &= \tfrac{1}{2}\beta^T(A^T Q_i A)\beta - (\bar{x}^T Q_i + c_i^T)A\beta + f_i(\bar{x}). \end{aligned}$$

(i) $i \notin J(\bar{x})$.

In this case, we have $f_i(\bar{x}) < 0$. To ensure $f_i(\bar{x} - \bar{s}) < 0$, we shall choose $\beta \in \mathbb{R}^{|J(\bar{x})|}$ such that

$$\frac{1}{2}\beta^T(A^T Q_i A)\beta - (\bar{x}^T Q_i + c_i^T)A\beta + f_i(\bar{x}) < 0. \tag{4.3}$$

(ii) $i \in J(\bar{x})$.

In this case, we have $f_i(\bar{x}) = 0$. Then, as in (4.3), we need

$$\frac{1}{2}\beta^T(A^T Q_i A)\beta - (\bar{x}^T Q_i + c_i^T)A\beta < 0. \tag{4.4}$$

Thus, the vector $\bar{s} = \sum_{j\in J(\bar{x})} \beta_j(Q_j\bar{x} + c_j)$, satisfying (4.3) for all $i \notin J(\bar{x})$ and (4.4) for all $i \in J(\bar{x})$, has the properties that $\bar{s} \in N_X(\bar{x})$ and $\bar{x} - \bar{s} \in \operatorname{ri} X$. Let

$$F_i(\beta) := \frac{1}{2}\beta^T P_i\beta - h_i^T\beta + f_i(\bar{x}),$$

where $P_i := A^T Q_i A$, $h_i := (\bar{x}^T Q_i + c_i^T)A$. Then inequalities (4.3) for all $i \notin J(\bar{x})$ and (4.4) for all $i \in J(\bar{x})$ are equivalent to $F_i(\beta) < 0$ for all $i \in I$.

By choosing β small enough, terms involving β^2 can be ignored. Hence $F_i(\beta) < 0$ for all $i \in I$ is equivalent to

$$-h_i^T\beta + f_i(\bar{x}) < 0, \forall i \in I. \tag{4.5}$$

Therefore, by finding β small enough which satisfies (4.5), we can find β that satisfy (4.3) for all $i \notin J(\bar{x})$ and (4.4) for all $i \in J(\bar{x})$. Finding β that satisfies (4.5) can be achieved as in Case 1(b). By appropriately scaling β so that $\|A\beta\| \leq 1$, we can then find the required $\bar{s}$ and hence y_k.

5 Numerical Experiments

In this section, we show some numerical experiments to illustrate that our algorithm is comparable with existing subgradient algorithms. In the first experiment, cited in [31], we compare our algorithm with the algorithm proposed by Ruszczyński on a nonsmooth convex program. In the second experiment, cited in [24], we use our algorithm to solve the dual problem arising from the assignment problem and compare it with the incremental subgradient method. In the third experiment, we use our algorithm to deal with the affine rank minimization problem. Using our algorithm, we can recover the MIT logo and PolyU logo clearly.

Before we describe our numerical experiments in detail, we need to clarify some points in the numerical experiments.

Since the subgradient method is not a descent method, it is common to keep track of the best point found so far, i.e., the one with the least function value. Therefore, at each iteration, we keep track of the record value

$$f_{rec}^k := \min\{f(x_k), f_{rec}^{k-1}\}.$$

This technique makes the sequence $\{f_{rec}^k\}$ nonincreasing.

In the following three numerical examples, as domains of objective functions are all full dimensional, relative gradients, as introduced in Definition 2.1, reduce to gradients.

Similar to the strategy in [8], we do not attempt to check whether the iterates lie in the set $\mathcal{D}$, where f is differentiable, in Step 2 of the sampling SGM. This seems not to be an easy task for a complicated function. Hence, we skip the differentiability check and assume that we have the information whether the gradient of f exists or not at a given point.

Another issue is the stopping criterion. Besides the nondifferentiablity information, we do not set any stopping criterion in the sampling SGM. Lack of implementable stopping criterion is a major drawback of subgradient methods. This drawback comes from the nondescent property of subgradient direction. If we cannot obtain or estimate the optimal value, it is really hard to set an effective stopping criterion. One common trick is to check whether there is any improvement in the last 100 iterates. If f_{rec}^k does not decrease in the last 100 iterates, then we stop and obtain the best value found so far. Another idea is to use the primal-dual subgradient method (see [23, 25, 26]) whose natural stopping criterion is the gap between the primal function value and dual function value. In the following, we do not set any stopping criterion and just illustrate the performance of the sampling SGM and other algorithms in a specified number of iterations.

5.1 Nonsmooth convex optimization

Here we consider the nonsmooth convex optimization problem (see [31])

$$\min_{x\in\mathbb{R}_+^n} f(x), \tag{5.1}$$

where $f : \mathbb{R}^n_+ \to \mathbb{R}$ is defined by $f(x) = \max\{f_1(x), f_2(x)\}$, with

$$f_1(x) = \alpha - c^T x, \qquad f_2(x) = \frac{1}{2} x^T D x.$$

As in the numerical experiments in [31], we set $n = 100$ and

$$\alpha = n \cdot \mathrm{rand}(), \quad c = 2 \cdot \mathrm{rand}(n, 1) - e, \quad D = \mathrm{diag}(\mathrm{rand}(n, 1)).$$

Here rand() denotes a random value drawn from an uniform distribution on the unit interval, rand$(n, 1)$ denotes a column vector with n elements and all elements take random values on the unit interval, e denotes a vector in $\mathbb{R}^n$ with all elements 1, and D is a diagonal matrix with random diagonal entries.

Solved by CVX[1], the optimal value for an instance of the above problem is $f_* = 18.5166$.

To solve (5.1), the subgradient algorithm based on a merit function approach (in short, MFA-SGM) designed in [31] is presented as follows,

$$\begin{aligned} y_k &= P_X(x_k - z_k), \\ x_{k+1} &= x_k + \tau_k(y_k - x_k), \\ z_{k+1} &= z_k + a\tau_k(s_{k+1} - z_k), \end{aligned}$$

with $s_{k+1} \in \partial f(x_k)$, starting from $x_0 \in X, z_0 \in \partial f(x_0)$.

The MFA-SGM differs from our sampling SGM in two main ways. The first difference is the random sampling and subgradient approximation process. The MFA-SGM updates the subgradient by a convex combination of the current subgradient and successive direction, while the sampling SGM updates the subgradient by combining the relative gradients of the random sampling points. The second difference is the updating and projection steps. The MFA-SGM uses the stepsize τ_k in the updating step after the projection operation while the sampling SGM uses stepsize v_k in the updating step before the projection operation. This is the essential difference between the MFA-SGM and sampling SGM. Note that if the MFA-SGM starts from a relative interior point x_0 of X, then all iterates are relative interior points, which is similar to our sampling SGM.

In our numerical computation, we use the same parameter $a = 0.1$ and stepsize $\tau_k = \tau/(1+0.01k)$ in MFA-SGM as in [31]. For comparison, we choose several different divergent stepsizes $v_k = v/(1 + 0.01k)$ and parameters $s = 5$, $\alpha_k = \min\{v_k, 1\}$, $\delta_k = \min\{\alpha_{k-1}/2, d_{\mathrm{rbd}\mathbb{R}^n_+}(x_k)\}$, $\lambda_{ki} = 1/s$ in our sampling SGM algorithm. Figure 1 plots the difference $f^k_{rec} - f_*$ when $\tau = 0.5$ in MFA-SGM and $v = 0.3$, 1, and 1.5, respectively in our sampling SGM until 3000 iterates. It illustrates that our sampling SGM when $v = 1.5$ converges faster than MFA-SGM when $\tau = 0.5$, but slower than sampling SGM when $v = 1$.

[1]CVX, designed by Michael Grant and Stephen Boyd, is a Matlab-based modeling system for convex optimization. Detailed information is available at the website http://cvxr.com/cvx/.

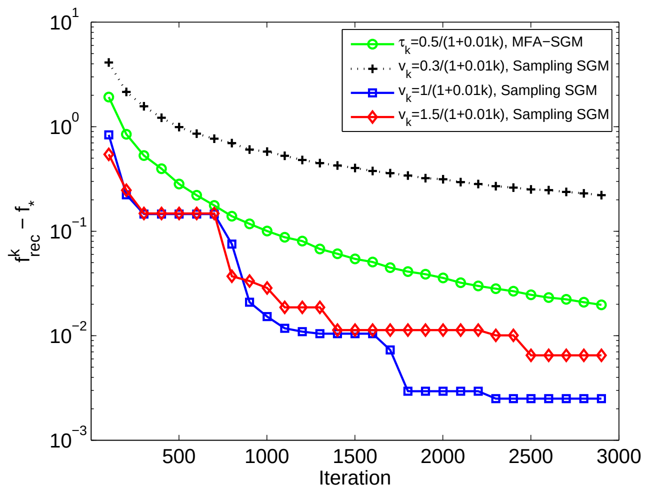


Figure 1: Comparison of MFA-SGM and sampling SGM for the nonsmooth convex optimization problem.

5.2 Assignment problem

The assignment problem is to assign m jobs to n machines such that the total cost is minimal (see [24]). Job i, performed at machine j, costs a_{ij} and requires p_{ij} time units.

Given the total available time t_j at each machine j, we want to find the minimum cost assignment of jobs to machines. Formally the problem can be written as

$$\begin{array}{rl} \min & \sum_{i=1}^m \sum_{j=1}^n a_{ij} y_{ij} \\ \text{s.t.} & \sum_{j=1}^n y_{ij} = 1, \quad i = 1, \dots, m, \\ & \sum_{i=1}^m p_{ij} y_{ij} \le t_j, \quad j = 1, \dots, n, \\ & y_{ij} = 0 \text{ or } 1, \quad \text{for all } i, j, \end{array}$$

where y_{ij} is the assignment variable, which is equal to 1 if the *i-th* job is assigned to the *j-th* machine and equal to 0 otherwise. In our numerical experiments we choose $n = 6$ and $m = 100$.

By relaxing the time constraints of machines, the following Lagrangian dual problem is obtained ([24])

$$\begin{array}{rl} \max & f(x) = \sum_{i=1}^m f_i(x) \\ \text{s.t.} & x \ge 0, \end{array} \tag{5.2}$$

where

$$f_i(x) = \min_{\sum_{j=1}^n y_{ij} = 1, y_{ij} \in \{0,1\}} \sum_{j=1}^n (a_{ij} + x_j p_{ij}) y_{ij} - \frac{1}{m} \sum_{j=1}^n t_j x_j. \tag{5.3}$$

A principal method for solving problem (5.2) is the subgradient method

$$x^{k+1} = P_{\mathbb{R}^m_+}[x^k + v_k \sum_{i=1}^m g_{i,k}],$$

where $g_{i,k}$ is a subgradient of f_i at x^k and v_k is the stepsize.

To solve problem (5.2), the main improvement of the incremental subgradient method (in short, incremental SGM) over subgradient method is that at each iteration, x is changed incrementally, through a sequence of m steps. Each step is a subgradient iteration for each single component function f_i. Thus, an iteration can be viewed as a cycle of m subiterations. Noting x^k as the vector obtained after k cycles, the vector x^{k+1} obtained after one more cycle is

$$x^{k+1} = \psi_{m,k},$$

where $\psi_{m,k}$ is obtained after m steps

$$\psi_{i,k} = P_{\mathbb{R}^m_+}[\psi_{i-1,k} + v_k g_{i,k}], \quad g_{i,k} \in \partial f_i(\psi_{i-1,k}), \quad i = 1, \ldots, m, \tag{5.4}$$

starting with

$$\psi_{0,k} = x^k.$$

Returning to (5.3), since $a_{ij} + x_j p_{ij} \geq 0$ for all i, j, we can easily evaluate $f_i(x)$ for each $x \geq 0$:

$$f_i(x) = a_{ij^*} + x_{j^*} p_{ij^*} - \frac{1}{m} \sum_{j=1}^{n} t_j x_{j^*},$$

where j^* is the index such that

$$a_{ij^*} + x_{j^*} p_{ij^*} = \min_{1 \leq j \leq n} \{a_{ij} + x_j p_{ij}\}.$$

Without additional cost, we obtain a subgradient g_i of f_i at x:

$$g_i = (g_{i1}, \ldots, g_{in})^T, \qquad g_{ij} = \begin{cases} -\frac{t_j}{m}, & \text{if } j \neq j^*, \\ p_{ij^*} - \frac{t_{j^*}}{m}, & \text{if } j = j^*. \end{cases}$$

The data for the problems (i.e., the matrices (a_{ij}), (p_{ij})) are randomly drawn from an uniform distribution on the unit interval.

$$A = (a_{ij}) = \text{rand}(m, n), \quad P = (p_{ij}) = \text{rand}(m, n).$$

The values t_j are calculated according to the formula

$$t_j = \frac{\bar{t}}{n} \sum_{i=1}^{m} p_{ij}, \quad j = 1, \ldots, n,$$

with $\bar{t}$ taking the value 0.4.

In our numerical computation, we choose the divergence stepsize $v_k = v/(1 + 0.01k)$ in both incremental SGM and sampling SGM and parameters $s = 5$, $\alpha_k = v_k$, $\delta_k = \min\{\alpha_{k-1}/2, d_{\text{rbd}\mathbb{R}^n_+}(x_k)\}$, $\lambda_{ki} = 1/s$ in our sampling SGM. Figure 2 shows the record value of f^k_{rec} when $v = 0.05$ in the incremental SGM and $v = 0.05,\ 0.1$ in our sampling SGM until 300 iterates. It illustrates that our sampling SGM results in a faster convergence of the dual objective values than the incremental SGM for the cases solved.

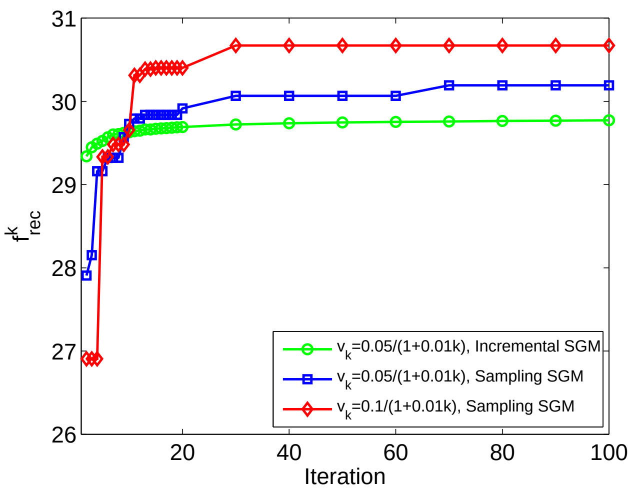


Figure 2: Comparison of incremental SGM and sampling SGM for the Lagrangian dual of assignment problem.

5.3 Affine rank minimization problem

Affine rank minimization problem has become an important problem in many applications in recent years. This problem can be stated as (see [29])

$$\begin{array}{ll} \min & \operatorname{rank} Z \\ \text{s.t.} & \mathcal{A}(Z) = b, \end{array} \tag{5.5}$$

where $Z \in \mathbb{R}^{m\times n}$ is the decision variable, the linear map $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ and vector $b \in \mathbb{R}^p$ are given. Let $K := mn$, the linear map $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ can always be written as its matrix representation,

$$\mathcal{A}(Z) = Avec(Z),$$

where $vec(Z) \in \mathbb{R}^K$ denotes the "vectorized" Z with its columns stacked in order on top of one another, and A is a $p \times K$ matrix.

An idea for affine rank minimization is to reformulate (5.5) as a nuclear norm minimization problem and solve it efficiently as a convex optimization problem. The corresponding nuclear norm minimization problem is

$$\begin{array}{ll} \min & \|Z\|_* \\ \text{s.t.} & \mathcal{A}(Z) = b. \end{array} \tag{5.6}$$

It is recalled that the nuclear norm of Z, $\|Z\|_*$, is defined as the sum of its singular values. Let $Z = U\Sigma V^T$ be an SVD where $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$, and Σ is an $r \times r$ diagonal matrix, with $\operatorname{rank} Z = r$. The subdifferential of the nuclear norm at Z is then given by (see [29])

$$\partial\|Z\|_* = \{UV^T + W : W \text{ and } Z \text{ have orthogonal row/column spaces and } \|W\| \le 1\},$$

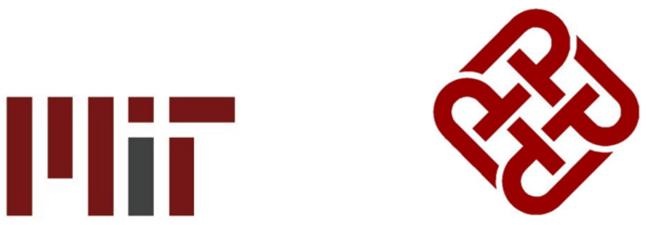

(a) MIT logo (b) PolyU logo

Figure 3: The original MIT and PolyU logos.

where $\|\cdot\|$ stands for the operator norm. When Z has no zero singular value (Z is full rank), the nuclear norm is differentiable and $\nabla\|Z\|_* = UV^T$.

We are interested in recovering the logos of Massachusetts Institute of Technology (MIT) and The Hong Kong Polytechnic University (PolyU), which are presented in Figure 3. Note that some numerical investigations are done in [29] for recovering the logo of MIT. We do some modification to the two logos. The modified MIT logo has three distinct colors white, gray, and black, with corresponding distinct nonzero numerical values, and rank equal to 5 ($r = 5$), while the modification of PolyU logo is a little more complex. Since the original PolyU logo is almost full rank, we rotate it by 45 degrees and then make it low rank. They are shown in Figure 4. The modified PolyU logo has two distinct colors white and black, with rank equal to 9 ($r = 9$).

(a) MIT logo (b) PolyU logo

Figure 4: The modified MIT and PolyU logos.

Consider the modified MIT and PolyU logos presented in Figure 4. The modified MIT logo has 46 rows, 81 columns and 3726 elements ($m = 46, n = 81, K = 3726$), with three distinct values corresponding to white, gray, and black, while the modified PolyU logo has 60 rows, 60 columns and 3600 elements ($m = 60, n = 60, K = 3600$), with two distinct values corresponding to white and black. For the linear map $\mathcal{A}$, we use the Gaussian ensemble and sample constraint matrices A with p ranging between 1000 and 2400 in our experiments.

Here we use the ordinary SGM and our sampling SGM to solve the nuclear norm minimization problem (5.6). Since the constraint set is an affine space, the perturbation step is skipped and thus we set $\alpha_k \equiv 0$. In the computation process, we choose divergence stepsize $v_k = 1/(1 + 0.1k)$, $\delta_k = v_k/2$, $\lambda_{ki} = 1/s$ and different sample size $s = 1,\ 5,\ 20,\ 50,\ 200$ in our sampling SGM. We display the numerical results for recovering the modified MIT logo when $p \geq 1300$ in Table 1 and the modified PolyU logo when $p \geq 2000$ in Table 2. In these tables, Δf denotes the required error on the objective value before termination, and NIT and time denote the corresponding number of iterations and total time taken to reach the specified precision of Δf respectively. It is illustrated that our sampling SGM arrives at the required level in fewer iterations and less time compared to ordinary SGM. When $s = 50$, it only takes one third or half of time that required for the ordinary SGM. Indeed, our sampling SGM provides a promising alternative subgradient method besides those suggested by Recht, et. al. in [29] that can be applied to the nuclear norm minimization problem.

Table 1: Computation results for recovering the modified MIT logo.

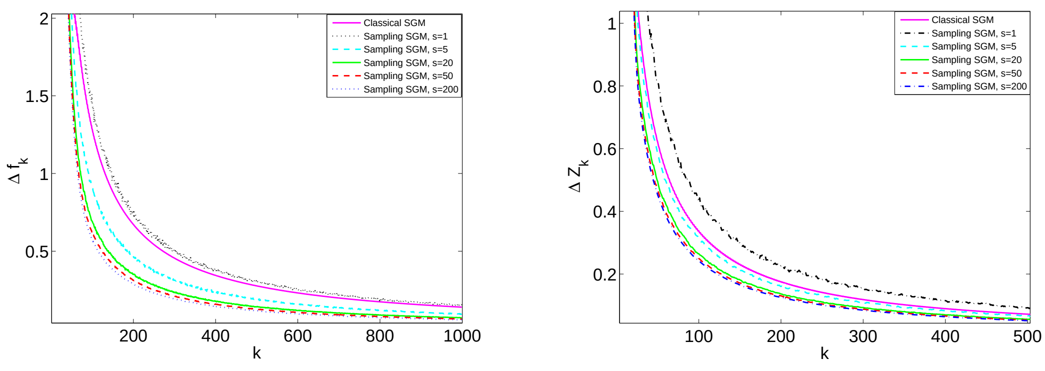
	$p = 1300$			$p = 1400$		
Algorithms/ sample size	Δf	NIT	time [min]	Δf	NIT	time [min]
Ordinary SGM	0.3	369	11	0.3	427	14
Sampling SGM/ $s = 1$	0.3	427	13	0.3	471	15
Sampling SGM/ $s = 5$	0.3	295	9	0.3	298	10
Sampling SGM/ $s = 20$	0.3	254	8	0.3	226	8
Sampling SGM/ $s = 50$	0.3	241	8	0.3	206	7
Sampling SGM/ $s = 200$	0.3	234	10	0.3	191	9
Sampling SGM/ $s = 500$	0.3	212	13	0.3	186	13
	$p = 1500$			$p = 1600$		
Ordinary SGM	0.3	460	16	0.3	486	18
Sampling SGM/ $s = 1$	0.3	449	16	0.3	510	20
Sampling SGM/ $s = 5$	0.3	313	11	0.3	320	12
Sampling SGM/ $s = 20$	0.3	223	8	0.3	236	9
Sampling SGM/ $s = 50$	0.3	207	8	0.3	211	8
Sampling SGM/ $s = 200$	0.3	191	10	0.3	186	10
Sampling SGM/ $s = 500$	0.3	187	13	0.3	185	14

Table 2: Computation results for recovering the modified PolyU logo.

	$p = 2000$			$p = 2100$		
Algorithms/ sample size	Δf	NIT	time [min]	Δf	NIT	time [min]
Ordinary SGM	0.3	472	18	0.3	495	19
Sampling SGM/ $s = 1$	0.3	424	16	0.3	421	17
Sampling SGM/ $s = 5$	0.3	252	10	0.3	225	9
Sampling SGM/ $s = 20$	0.3	193	8	0.3	188	8
Sampling SGM/ $s = 50$	0.3	168	7	0.3	163	7
Sampling SGM/ $s = 200$	0.3	153	9	0.3	149	8
Sampling SGM/ $s = 500$	0.3	150	12	0.3	143	11
	$p = 2200$			$p = 2300$		
Ordinary SGM	0.3	520	22	0.3	509	23
Sampling SGM/ $s = 1$	0.3	461	20	0.3	397	17
Sampling SGM/ $s = 5$	0.3	252	11	0.3	244	11
Sampling SGM/ $s = 20$	0.3	183	8	0.3	173	8
Sampling SGM/ $s = 50$	0.3	158	7	0.3	148	7
Sampling SGM/ $s = 200$	0.3	141	8	0.3	135	8
Sampling SGM/ $s = 500$	0.3	138	11	0.3	131	11

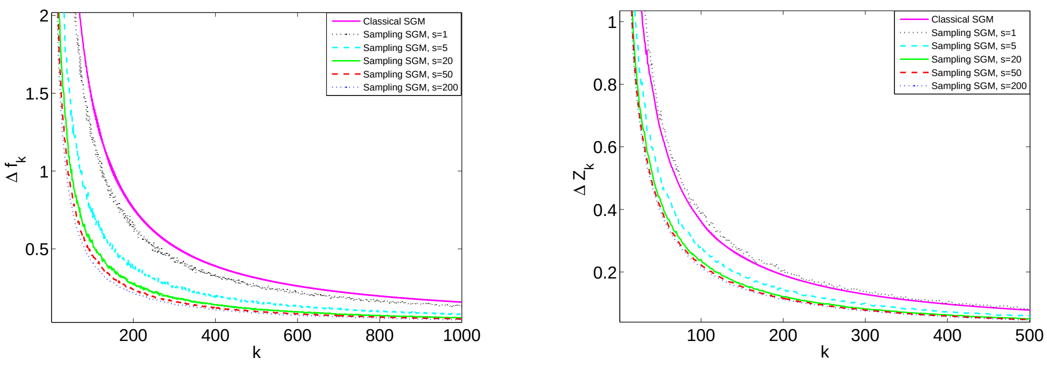
So, does bigger sample size lead to better result? The answer is negative. From Tables 1 and 2, we observe that the number of iterations decreases as the sample size increases. However, it takes much more time to compute gradients when $s = 200$ and 500. As such the total computation time becomes large again when s is over 200.

Figures 5 and 6 show the convergence behavior for recovering the modified MIT logo when $p = 1500$ and modified PolyU logo when $p = 2200$ using the ordinary SGM and our sampling SGM. In these figures, $\Delta f_k = f(x_k) - f_*$ denotes the error between the objective value and the optimal value, and $\Delta Z_k = \|Z_k - Z^*\|_F$ denotes the error between the k^{th} iterate and the optimal solution, where $\|\cdot\|_F$ stands for the Frobenius norm. Figure 7 shows recovered images for both the modified MIT logo when $p = 1300$ and PolyU logo when $p = 2000$.



(a) (b)

Figure 5: Convergence behavior for recovering the modified MIT logo using the ordinary SGM and our sampling SGM when $p = 1500$. (a) shows the convergence property in objective values and (b) shows the convergence property in iterates.



(a) (b)

Figure 6: Convergence behavior for recovering the modified PolyU logo using the ordinary SGM and our sampling SGM when $p = 2200$. (a) shows the convergence property in objective values and (b) shows the convergence property in iterates.

Funding. The paper was partially supported by a grant with grant no: PolyU 5295/12E from the Research Grants Council of Hong Kong, the National Science Foundation of China

(a) MIT
(b) PolyU

Figure 7: Recovered images for the modified MIT logo when $p = 1300$ and PolyU logo when $p = 2000$.

(11431004) and the Natural Science Foundation of SZU (201544).

References

[1] A. Beck and M. Teboulle (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 31(3):167–175.

[2] D. P. Bertsekas (1999). *Nonlinear Programming.* Athena Scientific, Cambridge.

[3] D. P. Bertsekas, A. Nedić, and A. Ozdaglar (2003). *Convex Analysis and Optimization.* Athena Scientific, Cambridge.

[4] D. B. Brown and J. E. Smith (2014). Information relaxations, duality, and convex stochastic dynamic programs. *Oper. Res.*, 62:1394–1415.

[5] R. S. Burachik, R. N. Gasimov, N. A. Ismayilova, and C. Y. Kaya (2006). On a modified subgradient algorithm for dual problems via sharp augmented Lagrangian. *J. Global Optim.*, 34:55–78.

[6] J. V. Burke, A. S. Lewis, and M. L. Overton (2002). Approximating subdifferentials by random sampling of gradients. *Math. Oper. Res.*, 27:567–584.

[7] J. V. Burke, A. S. Lewis, and M. L. Overton (2002). Two numerical methods for optimizing matrix stability. *Linear Algebra Appl.*, 351–352:117–145.

[8] J. V. Burke, A. S. Lewis, and M. L. Overton (2005). A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. *SIAM J. Optim.*, 15(3):751–779.

[9] Y. M. Ermoliev (1966). Methods of solution of nonlinear extremal problems. *Cybern. Syst. Anal.*, 2:1–14.

[10] R. N. Gasimov (2002). Augmented Lagrangian duality and nondifferentiable optimization methods in nonconvex programming. *J. Global Optim.*, 24:187–203.

[11] J.-L. Goffin and K. C. Kiwiel (1999). Convergence of a simple subgradient level method. *Math. Program.*, 85:207–211.

[12] J.-B. Hiriart-Urruty and C. Lemaréchal (1996). *Convex Analysis and Minimization Algorithms*. Springer-Verlag, Berlin.

[13] J.-B. Hiriart-Urruty and C. Lemaréchal (2001). *Fundamentals of Convex Analysis*. Springer-Verlag, Berlin.

[14] Y. H. Hu, X. Q. Yang, and C.-K. Sim (2015). Inexact subgradient methods for quasi-convex optimization problems. *Eur. J. Oper. Res.*, 240(2):315–327.

[15] S. Kim and H. Ahn (1991). Convergence of a generalized subgradient method for nondifferentiable convex optimization. *Math. Program.*, 50:75–80.

[16] K. C. Kiwiel (1983). An aggregate subgradient method for nonsmooth convex minimization. *Math. Program.*, 27:320–341.

[17] K. C. Kiwiel (1996). The efficiency of subgradient projection methods for convex optimization, part I: General level methods. *SIAM J. Control Optim.*, 34(2):660–676.

[18] K. C. Kiwiel (1996). The efficiency of subgradient projection methods for convex optimization, part II: Implementations and extensions. *SIAM J. Control Optim.*, 34(2):677–697.

[19] K. C. Kiwiel (2004). Convergence of approximate and incremental subgradient methods for convex optimization. *SIAM J. Optim.*, 14(3):807–840.

[20] K. C. Kiwiel, T. Larsson, and P. O. Lindberg (1999). The efficiency of ballstep subgradient level methods for convex optimization. *Math. Oper. Res.*, 24(1):237–254.

[21] K. C. Kiwiel, T. Larsson, and P. O. Lindberg (2007). Lagrangian relaxation via ballstep subgradient methods. *Math. Oper. Res.*, 32(3):669–686.

[22] T. Larsson, M. Patriksson, and A.-B. Strömberg (1998). Ergodic convergence in subgradient optimization. *Optim. Method. Softw.*, 9(1-3):93–120.

[23] T. Larsson, M. Patriksson, and A.-B. Strömberg (2003). On the convergence of conditional epsilon-subgradient methods for convex programs and convex-concave saddle-point problems. *Eur. J. Oper. Res.*, 151(3):461–473.

[24] A. Nedić and D. P. Bertsekas (2001). Incremental subgradient methods for nondifferentiable optimization. *SIAM J. Optim.*, 12(1):109–138.

[25] A. Nedić and A. Ozdaglar (2009). Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM J. Optim.*, 19(4):1757–1780.

[26] Y. Nesterov (2009). Primal-dual subgradient methods for convex problems. *Math. Program.*, 120:221–259.

[27] B. T. Polyak (1967). A general method for solving extremum problems. *Soviet Mathematics Doklady*, 8:593–597.

[28] B. T. Polyak (1987). *Introduction to Optimization.* Optimization Software, New York.

[29] B. Recht, M. Fazel, and P. A. Parrilo (2010). Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Rev.*, 52(3):471–501.

[30] R. T. Rockafellar and R. J.-B. Wets (1998). *Variational Analysis.* Springer-Verlag, Berlin.

[31] A. Ruszczyński (2008). A merit function approach to the subgradient method with averaging. *Optim. Method. Softw.*, 23:161–172.

[32] N. Z. Shor (1979). *Minimization Methods for Non-differentiable Functions.* Naukova Dumka, Kiev [English translation: Springer, Berlin, 1985].