A Forward-Backward Splitting Method with Component-wise Lazy Evaluation for Online Structured Convex Optimization*

Yukihiro Togari and Nobuo Yamashita†

March 28, 2016

Abstract: We consider large-scale optimization problems whose objective functions are expressed as sums of convex functions. Such problems appear in various fields, such as statistics and machine learning. One method for solving such problems is the online gradient method, which exploits the gradients of a few of the functions in the objective function at each iteration.

In this paper, we focus on the sparsity of two vectors, namely a solution and a gradient of each function in the objective function. In order to obtain a sparse solution, we usually add the L1 regularization term to the objective function. Then, because the L1 regularization term consists of all decision variables, the usual online gradient methods update all variables even if each gradient is sparse. To reduce the number of computations required for all variables at each iteration, we can employ a lazy evaluation. This only updates the variables corresponding to nonzero components of the gradient, by ignoring the L1 regularization terms, and evaluates the ignored terms later. Because a lazy evaluation does not exploit the information related to all terms at every iteration, the online algorithm using lazy evaluation may converge slowly.

In this paper, we describe a forward-backward splitting type method, which exploits the L1 terms corresponding to the nonzero components of the gradient at each iteration. We show that its regret bound is $\mathcal{O}(\sqrt{T})$, where T is the number of functions in the objective function. Moreover, we present some numerical experiments using the proposed method, which demonstrate that the proposed method gives promising results.

Keywords: online optimization; regret analysis; text classification

1 Introduction

In this paper, we consider the following problem:

$$\min \quad \sum_{t=1}^{T} \big\{ f_t(x) + r(x) \big\}, \tag{1}$$

*Technical Report 2016-001, March 27, 2016.

†Graduate School of Informatics, Kyoto University, Yoshida-honmachi, Kyoto, 606-8501, Japan. E-mail: nobuo@i.kyoto-u.ac.jp

where $f_t : \mathbb{R}^n \to \mathbb{R}(t = 1, \ldots, T)$ are differentiable convex functions, and $r : \mathbb{R}^n \to \mathbb{R}$ is a continuous convex function that is not necessarily differentiable. Typical examples of this problem appear in machine learning and signal processing. In these examples, f_t is regarded as a loss function, such as a logistic loss function $f_t(x) = \log(1 + \exp(-b_t\langle a_t, x\rangle))$ [2, 7] or a hinge loss function $f_t(x) = [-b_t\langle a_t, x\rangle + 1]_+$ [4]. Here, $(a_t, b_t) \in \mathbb{R}^{n+1}$ is a pair from the tth sample data. Furthermore, r is a regularization function to avoid over-fitting, such as the L1 regularization term $r(x) = \lambda \sum_{i=1}^n |x^{(i)}|$ or the L2 regularization term $r(x) = \frac{\lambda}{2}\|x\|^2$. For some applications, this may be an indicator function of the feasible set Ω, which is defined by

$$r_\Omega(x) = \begin{cases} 0 & if \quad x \in \Omega \\ +\infty & otherwise. \end{cases}$$

In this paper, we consider problem (1) under the following circumstances:

(i) Either the upper limit of the summation T is very large, or the functions f_t are given one by one at the tth iteration; that is, the number of functions f_t increases with t.

(ii) The gradient $\nabla f_t(x)$ is sparse; that is, $I_t \ll n$, where $I_t = \left\{ i \middle| \frac{\partial f_t(x)}{\partial x^{(i)}} \neq 0 \right\}$.

(iii) The function r is decomposable.

In machine learning and statistics, T corresponds to the number of training samples, and f_t is a loss function for the tth sample. Therefore, if we have large number of samples, T becomes large. When f_t is the logistic function given above and some sample data a_t in f_t is sparse, then the gradient of f_t is also sparse. Further, if r is the L1 or L2 regularization term, then r can be decomposed as

$$r(x) = \sum_{i=1}^n r^i(x^{(i)}), \tag{2}$$

where $r^i : \mathbb{R} \to \mathbb{R}$ and $x^{(i)}$ denotes the ith component of the decision variable x. Throughout this paper, we assume that r is expressed as (2). We can easily extend the subsequent discussions to the case where r is decomposable with respect to blocks of variables, such as the group lasso.

For problems satisfying (i)-(iii), the online gradient method is useful. This generates a sequence of reasonable solutions using only the gradient of f_t at the tth iteration [15, 16]. Note that this is essentially the same as the incremental gradient method [1] and the stochastic gradient method [5, 6]. In this paper, we adopt the online gradient method for the problem (1) satisfying (i)-(iii).

We usually evaluate the online gradient method using a regret, which is defined by

$$R(T) = \sum_{t=1}^T \big\{ f_t(x_t) + r(x_t) \big\} - \min_x \sum_{t=1}^T \big\{ f_t(x) + r(x) \big\}, \tag{3}$$

where $\{x_t\}$ is a sequence generated by the online gradient method. The regret $R(T)$ is the residual between the optimum value of (1) and the sum of the values of f_t at x_t. When $R(T)$ satisfies $\lim_{T\to\infty} \frac{R(T)}{T} = 0$, we conclude that the online gradient method converges to an optimal solution.

Various online gradient methods have been proposed for problem (1) [14]. When the objective function consists of non-differentiable functions, such as the L1 regularization term, the online subgradient method for problem (1) updates a sequence as follows [5]:

$$x_{t+1} = x_t - \alpha_t\big(\nabla f_t(x_t) + \eta_t\big),$$

where $\eta_t \in \partial r(x_t)$ and α_t is the stepsize at the tth iteration. If f_t is convex, then the regret of the online subgradient method is $\mathcal{O}(\sqrt{T})$. Furthermore, if f_t is strongly convex, then it reduces to $\mathcal{O}(\log T)$ [14]. However, this algorithm does not fully exploit the structure of r, and it may converge slowly.

The forward-backward splitting (FOBOS) method [8, 9] uses the structure of r. This consists of the gradient method for f_t and the proximal point method for r; that is, a sequence is generated as

$$x_{t+1} = \mathrm{argmin}_x\Big\{\langle \nabla f_t(x_t), x\rangle + \frac{1}{2\alpha_t}\|x - x_t\|^2 + r(x)\Big\}. \tag{4}$$

For given $\nabla f_t(x)$, we can compute (4) in $\mathcal{O}(|I_t|)$ when $r^i(x) = 0$ for all i. Therefore, when $\nabla f_t(x)$ is sparse as described in (ii), such that $|I_t| \ll n$, we can obtain x_{t+1} quickly. However, when $r^i(x) \neq 0$ for all i we require a computation time of $\mathcal{O}(n)$ for (4), even if $|I_t| \ll n$.

In order to overcome this drawback, Langford [10] proposed the lazy evaluation method, which defers the evaluation of r in (4) to a later round at most iterations, and exploits the information relating to each deferred r all together after a number of iterations. This technique enables us to obtain x_{t+1} using (4) with only a few computations when $\nabla f_t(x)$ is sparse. However, this method does not fully exploit the information of r at all iterations, and hence its behavior is inferior. Furthermore, even if we employ the L1 regularization term as r, x_t cannot be sparse except at the iteration where the deferred evaluations of r are performed.

In this paper, we propose a novel algorithm that conducts the lazy evaluation for r^i with respect to i such that $\nabla f_t(x_t) \neq 0$ at each iteration. The usual lazy evaluation process ignores r for almost all iterations, and evaluates r at a few iterations while updating all components in x_t. In contrast, the proposed method conducts the lazy evaluation for the updated variables $x_{t+1}^{(i)}$ $(i \in I_t)$ at each iteration. Thus, the proposed method can exploit the full information given by r^i $(i \in I_t)$ at each iteration, and hence the FOBOS method is expected to converge faster with this technique than with the usual lazy evaluation [10].

The remainder of this paper is organized as follows. In Section 2, we introduce some online gradient methods and the existing lazy evaluation procedure. In Section 3, we propose a FOBOS method with component-wise lazy evaluation. In Section 4, we prove that the regret of the proposed method is $\mathcal{O}(\sqrt{T})$. In Section 5, we present some numerical experiments using the proposed method. Finally, we provide some concluding remarks in Section 6.

Throughout this paper, we employ the following notations. Note that $\|x\|$ and $\|x\|_p$ denote the Euclidean and p-norm of x, respectively. For a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, $\nabla^{(i)} f(x)$ denotes the ith component of $\nabla f(x)$. Furthermore, ∂f denotes the subgradient of f. The function $[\cdot]_+$ denotes the max function; that is, $[a]_+ = \max\{a, 0\}$.

We are aware of that Lipton and Elkan [11] quite recently presented similar idea for the problem. The idea of the proposed method in this paper were presented in the bachelor thesis of one of the authors of this paper [12]. Further, the regret analysis and numerical experiments are presented at the meeting of Operations Research Society of Japan [13], which has been held in March, 2015.

2 Preliminaries

In this section, we introduce the existing online gradient methods, which are the basis of the method that will be proposed in Section 3.

2.1 Forward-Backward Splitting Method

In this subsection, we introduce some well-known online gradient methods for solving problem (1). The most fundamental algorithm for (1) is the online subgradient method [5], given as follows.

Algorithm 1 Online Subgradient Method

Step 0: Let $x_1 \in \mathbb{R}^n$ and $t = 1$.
Step 1: Compute the subgradient $s_t \in \partial\{f_t(x_t) + r(x_t)\}$.
Step 2: Update the sequence by

$$x_{t+1} = \mathrm{argmin}_x\{\langle s_t, x\rangle + \frac{1}{2\alpha_t}\|x - x_t\|^2\}.$$

Step 3: Update the stepsize. Set $t = t + 1$, and return to **Step 1**.

When all of the loss functions f_t are convex and $\alpha_t = \frac{c}{\sqrt{t}}$, the regret (3) of the online subgradient method is $\mathcal{O}(\sqrt{T})$ [16], where c is a positive constant. Moreover, when the functions f_t are strongly convex, the regret reduces to $\mathcal{O}(\log T)$ [14]. However, this algorithm is inefficient when r is a non-differentiable function, such as the L1 regularization term.

The forward-backward splitting (FOBOS) method [8] is an efficient algorithm for problems where r has a special structure.

Algorithm 2 Forward-Backward Splitting Method (FOBOS)

Step 0: Let $x_1 \in \mathbb{R}^n$ and $t = 1$.
Step 1: Set $I_t = \{i|\nabla^{(i)} f_t(x_t) \neq 0\}$ and update the sequence using the gradient method.

$$\hat{x}_{t+1}^{(i)} = \begin{cases} \mathrm{argmin}_y\Big\{\alpha_t \ \nabla^{(i)} f_t(x_t)y + \frac{1}{2}(y - x_t^{(i)})^2\Big\} & if \quad i \in I_t \\ x_t^{(i)} & otherwise. \end{cases} \tag{5}$$

Step 2: Evaluate r^i by the proximal point method.

$$x_{t+1}^{(i)} = \mathrm{argmin}_y\Big\{r^i(y) + \frac{1}{2\alpha_t}(y - \hat{x}_{t+1}^{(i)})^2\Big\} \quad for \quad i = 1, \ldots, n. \tag{6}$$

Set $t = t + 1$, and return to **Step 1**.

When $r^i(x^{(i)}) = \lambda|x^{(i)}|$ for a positive constant λ, (6) is rewritten as

$$x_{t+1}^{(i)} = \mathrm{sgn}\big(x_t^{(i)} - \alpha_t\nabla^{(i)} f_t(x_t)\big)\max\big\{|x_t^{(i)} - \alpha_t\nabla^{(i)} f_t(x_t)| - \lambda\alpha_t, 0\big\} \quad for \ \ i = 1, \ldots, n,$$

where sgn(·) denotes the signature function; that is,

$$\mathrm{sgn}(a) = \begin{cases} +1 & if \quad a > 0 \\ 0 & if \quad a = 0 \\ -1 & otherwise\ . \end{cases}$$

Moreover, when $r^i(x^{(i)}) = \frac{\lambda}{2}{x^{(i)}}^2$, (6) is given by

$$x_{t+1}^{(i)} = \frac{1}{1+\lambda\alpha_t} x_t^{(i)} - \frac{\alpha_t}{1+\lambda\alpha_t} \nabla^{(i)} f_t(x_t) \quad for \quad i = 1, ..., n.$$

The regret obtained by FOBOS has been demonstrated to be $\mathcal{O}(\sqrt{T})$. We will assume that condition (ii) from the introduction holds. Then, we can note that $x_{t+1}^i = x_t^i$ holds in Step 1 for i such that $\nabla f_t(x_t) \neq 0$. Therefore, Step 1 only updates $x^{(i)}$ for $i \in I_t$. However, even if $\nabla f_t(x_t)$ is sparse, Step 2 has to update all components when $r^i(x^{(i)}) \neq 0$ for all i.

2.2 Lazy Evaluations

In this subsection, we introduce the lazy evaluation technique [10], which ignores (6) in FOBOS for some consecutive iterations. We carry out the ignored updates (6) all together after every K iterations.

Algorithm 3 FOBOS with Lazy evaluation (L-FOBOS)

Step 0: Let K be a natural number, $x_1 \in \mathbb{R}^n$, and $t = 1$.

Step 1: Set $I_t = \{i | \nabla^{(i)} f_t(x_t) \neq 0\}$ and update using the gradient method.

$$x_{t+1}^{(i)} \quad = \quad \begin{cases} \mathrm{argmin}_x \left\{ \alpha_t \nabla^{(i)} f_t(x_t) x + \frac{1}{2}(x - x_t^{(i)})^2 \right\} & if \quad i \in I_t \\ x_t^{(i)} & otherwise. \end{cases}$$

If t mod K $\neq 0$, then set $t = t + 1$ and return to **Step 1**.
Otherwise, go to **Step 2**.

Step 2: Conduct the lazy evaluation for each r^i. Let $y_1 = x_t$ and $j = 1$.

$$y_{j+1}^{(i)} = \mathrm{argmin}_y \{ r^i(y) + \frac{1}{2\alpha_{t-K+j}} (y - y_j^{(i)})^2 \} \quad for \quad i = 1, \ldots, n. \tag{7}$$

If $j = j + 1$ and $j \leq K$, then return to **Step 2**.
Otherwise, set $t = t + 1$ and $x_t = y_j$ and go to **Step 1**.

In this paper, we refer to the above algorithm as L-FOBOS. Step 1 of L-FOBOS updates only x_{t+1}^i for $i \in I_t$ using the gradient method. After K consecutive iterations of Step 1, Step 2 evaluates the deferred r. Here, Step 2 employs the same stepsizes as the most recent K iterations in Step 1.

If Step 2 directly calculates (7) K times, then $\mathcal{O}(Kn)$ computations are required. However, if r satisfies the following property, then the number of computations required in Step 2 can be shown to be $\mathcal{O}(n)$.

Property 1. *Let $x_0 \in \mathbb{R}^n$, and let $\{x_t\}$ be the sequence obtained by the following formula:*

$$x_{t+1} = \mathrm{argmin}_x \left\{ \frac{1}{2}\|x - x_t\|^2 + \alpha_t r(x) \right\}.$$

Moreover, let x^* be the optimal solution of the following optimization problem:

$$x^* = \mathrm{argmin}_x \left\{ \frac{1}{2}\|x - x_0\|^2 + \left(\sum_{t=1}^{K} \alpha_t \right) r(x) \right\}.$$

Then, $x_K = x^*$ holds for any x_0.

Note that the L1 and L2 regularization terms and the indicator function for boxed constraints satisfy the above property [8, Proposition 11].

When r satisfies Property 1, we can aggregate all of the computations needed for K calculations (7) into the following $\mathcal{O}(n)$ formula:

$$y_K^{(i)} = \mathrm{argmin}_y \{ r^i(y) + \frac{1}{2\tilde{\alpha}_t}(y - x_t^{(i)})^2 \} \quad (i = 1, \ldots, n),$$

where $\tilde{\alpha}_t$ is the sum of the stepsizes of the last K iterations at the tth iteration; that is, $\tilde{\alpha}^t = \sum_{j=t-K+1}^{t} \alpha_j$, with $\tilde{\alpha}_0 = 0$.

By using L-FOBOS, we can update the sequence effectively with respect to the computational complexity. However, the convergence of L-FOBOS might be slower than FOBOS for many problems, because we cannot sufficiently exploit the information given by r. Moreover, we cannot obtain a sparse sequence, even if we set r as the L1 regularization term. Hence, in the next section we will present a novel lazy evaluation technique that can exploit the information given by r.

3 FOBOS with Component-wise Lazy Evaluation

As discussed in Section 2, applying the proximal point method to $i \notin I_t$ is inefficient for a problem with sparse gradients. It may not be useful to update the variables without r, as in the L-FOBOS method. Therefore, we propose the following method, which exploits both the sparsity of gradients and the decomposability of r. Note that the proposed algorithm adopts the lazy evaluation of r^i for $x_t^{(i)}$ $(i \in I_t)$ after the gradient method has been used to update $x_t^{(i)}$.

Algorithm 4 is hereafter referred to as CL-FOBOS, where $\xi^{(i)}$ denotes the number of iterations to elapse since the iteration where $x^{(i)}$ was most recently updated. Note that L-FOBOS only updates $x^{(i)}$ once every K iterations using the information of r [10], while CL-FOBOS exploits $r^i (i \in I_t)$ at each iteration.

Step 2 of CL-FOBOS requires $\xi^{(i)}$ iterations for $i \in I_t$. However, we can aggregate all of the necessary computations for Step 2 if r satisfies Property 1 from Section 2. From this point on, we assume that r satisfies the following assumption:

Assumption 1. *(i) The function r is decomposable with respect to each component of the decision valuable x.*

Algorithm 4 FOBOS with Component-wise Lazy Evaluation (CL-FOBOS)

Step 0: Let $x_1 \in \mathbb{R}^n$ and $t = 1$.

Step 1: Set $I_t = \{i | \nabla^{(i)} f_t(x_t) \neq 0\}$ and perform the gradient method.

$$\hat{x}_{t+1}^{(i)} = \begin{cases} \mathrm{argmin}_y \left\{ \alpha_t \ \nabla^{(i)} f_t(x_t) y + \frac{1}{2}(y - x_t^{(i)})^2 \right\} & if \quad i \in I_t \\ x_t^{(i)} & otherwise. \end{cases} \tag{8}$$

Step 2: Conduct the lazy evaluation of r^i for $i \in I_t$. Let $y_1^{(i)} = \hat{x}_t^{(i)}$ and $j^{(i)} = 1$ for $i \in I_t$.

$$y_{j^{(i)}+1}^{(i)} = \mathrm{argmin}_y \{ r^i(y) + \frac{1}{2\alpha_{t-\xi^{(i)}+j^{(i)}}} (y - y_{j^{(i)}}^{(i)})^2 \} \quad for \quad i \in I_t.$$

Set $j^{(i)} = j^{(i)} + 1$. If $j^{(i)} \leq \xi^{(i)}$, then return to **Step 2**. If $j^{(i)} > \xi^{(i)}$, then set $i \notin I_t$. Furthermore, if $I_t = \emptyset$, then set $t = t + 1$ and $x_t^{(i)} = y_{j^{(i)}}^{(i)}$, and go to **Step 1**. Otherwise, return to **Step 2**.

(ii) The function r satisfies Property 1.

Under Assumption 1, Step 2 can be rewritten as

$$\begin{aligned} x_{t+1}^{(i)} &= \begin{cases} \mathrm{argmin}_y \left\{ \frac{1}{2}(y - \hat{x}_t^{(i)})^2 + (\check{\alpha}_t - \alpha_\xi^{(i)}) r^i(y) \right\} & if \quad i \in I_t \\ \hat{x}_{t+1}^{(i)} & otherwise, \end{cases} \\ \check{\alpha}_{t+1} &= \check{\alpha}_t + \alpha_t, \\ \alpha_\xi^{(i)} &= \check{\alpha}_{t+1} \qquad for \quad i \in I_t, \end{aligned} \tag{9}$$

where $\check{\alpha}_t$ is the sum of the stepsizes from the first iteration to the tth iteration, with $\check{\alpha}_0 = 0$. Then, $\check{\alpha}_t - \alpha_\xi^{(i)}$ denotes the sum of the stepsizes from the most recent iteration where x_i has been updated to the tth iteration.

We see that the computation of $\alpha_\xi^{(i)}$ only requires $\mathcal{O}(|I_t|)$ computations at each iteration. After we calculate $\nabla f_t(x_t)$, we update x_t in $\mathcal{O}(|I_t|)$.

4 Regret Bound of CL-FOBOS

In this section, we derive the regret bound of CL-FOBOS.

We can rewrite (8) and (9) as

$$\begin{aligned} \hat{x}_{t+1}^{(i)} &= \begin{cases} \mathrm{argmin}_{x^{(i)}} \left\{ \alpha_t \nabla^{(i)} f_t(x_t) x^{(i)} + \frac{1}{2}(x^{(i)} - x_t^{(i)})^2 \right\} & if \quad i \in I_t \\ x_t^{(i)} & otherwise, \end{cases} \\ x_{t+1}^{(i)} &= \begin{cases} \mathrm{argmin}_{y \in \mathbb{R}} \left\{ \frac{1}{2}(y - \hat{x}_{t+1}^{(i)})^2 + \left(\sum_{j=t_\xi^{(i)}+1}^{t} \alpha_j \right) r^i(y) \right\} & if \quad i \in I_t \\ \hat{x}_{t+1}^{(i)} & otherwise, \end{cases} \end{aligned} \tag{10}$$

where $t_\xi^{(i)}$ denotes the most recent iteration where $x^{(i)}$ has been updated prior to the tth

iteration. The regret bound of CL-FOBOS, $R_{\rm cl}(T)$, can then written as

$$R_{\rm cl}(T) = \sum_{t=1}^{T}\Big\{f_t(x_t) + \sum_{i=1}^{n} r^i(x_t^{(i)})\Big\} - \sum_{t=1}^{T}\Big\{f_t(x_*) + \sum_{i=1}^{n} r^i(x_*^{(i)})\Big\}, \tag{11}$$

where x_* denotes the optimal solution of problem (1). We will analyze the upper bound of $R_{\rm cl}(T)$, and will show that $\lim_{T\to\infty} \frac{R_{\rm cl}(T)}{T} = 0$ holds under the following assumption.

Assumption 2. *(i) $t - t_{\xi}^{(i)} \leq K$ holds, where K is a positive constant.*

(ii) There exists a positive constant D such that $|x_t - x^*| \leq D$ for $t = 1, \ldots, T$.

(iii) There exists a positive constant G such that $\sup_{v_f \in \partial f_t(x_t)} \|v_f\| \leq G$ and $\sup_{v_r \in \partial r(x_t)} \|v_r\| \leq G$ for $t = 1, \ldots, T$.

(iv) The stepsize α_t is given as $\alpha_t = \frac{\alpha_0}{\sqrt{t}}$, where α_0 is a positive constant.

Assumption 2 (i) ensures that all of the components of $\{x_t\}$ are updated at least once in every K iterations. The conditions (ii), (iii), and (iv) of Assumption 2 are assumed in (5) and (6) of the regret analysis of FOBOS. In particular, Assumption 2 (ii) implies the boundedness of $\{x_t\}$, and thus $\big\{|r^i(x_t^{(i)}) - r^i(x_*^{(i)})|\big\}$ and $\big\{|r^i(x_t^{(i)})|\big\}$ are also bounded. Hence, there exist positive constants R and S such that

$$\sum_{i=1}^{n} |r^i(x_t^{(i)}) - r^i(x_*^{(i)})| \leq R \quad for \quad t = 1, \ldots, T. \tag{12}$$

$$|r^i(x_t^{(i)})| \leq S \quad for \quad i = 1, \ldots, n, \text{and } t = 1, \ldots, T. \tag{13}$$

Finally, the stepsize given in Assumption 2 (iv) is often employed for online gradient methods. Under Assumptions 1 and 2, we now provide a regret bound for CL-FOBOS.

Theorem 1. *Let $\{x_t\}$ be a sequence generated by CL-FOBOS, and suppose that Assumptions 1 and 2 hold. Then, the regret bound of CL-FOBOS is given by*

$$\begin{aligned} R_{\rm cl}(T) &= \sum_{t=1}^{T}\Big\{f_t(x_t) + \sum_{i=1}^{n} r^i(x_t^{(i)})\Big\} - \sum_{t=1}^{T}\Big\{f_t(x_*) + \sum_{i=1}^{n} r^i(x_*^{(i)})\Big\} \\ &= \mathcal{O}(\log T) + \mathcal{O}(\sqrt{T}). \end{aligned}$$

To prove Theorem 1, we use the existing result on the regret bound of FOBOS. Specifically, we consider the sequence $\{x_t\}$ given by CL-FOBOS (10) as that generated by FOBOS for a certain convex optimization problem.

We first see that (10) is equivalent to

$$x_{t+1}^{(i)} = \mathrm{argmin}_{x^{(i)}}\Big\{\alpha_t \nabla^{(i)} f_t(x_t) x^{(i)} + \alpha_t \sum_{i=1}^{n} r_t^i(x^{(i)}) + \frac{1}{2}(x^{(i)} - x_t^{(i)})^2\Big\}, \tag{14}$$

where $r_t^i : \mathbb{R} \to \mathbb{R}$ is defined as

$$\begin{aligned} r_t^i(y) &= c_t^{(i)} r^i(y), \\ c_t^{(i)} &= \begin{cases} \frac{\sum_{j=t_\xi^{(i)}+1}^{t} \alpha_j}{\alpha_t} & if \quad i \in I_t \\ 0 & otherwise. \end{cases} \end{aligned}$$

Note that (14) generates the same sequence as (10).

The update (14) constitutes the FOBOS procedure for the following problem, with f_t and $\sum_{i=1}^n r_t^i$:

$$\min \quad \sum_{t=1}^{T} \Big\{ f_t(x) + \sum_{i=1}^{n} r_t^i(x) \Big\}. \tag{15}$$

Note that r_t^i in problem (15) depends on t, whereas $r(x)$ of problem (1) does not. Hereafter, we refer to (14) as Algorithm γ. Because Algorithm γ constitutes the FOBOS procedure for problem (15), [8] demonstrates that its regret (16) is given by

$$\begin{aligned} R_\gamma(T) &= \sum_{t=1}^{T} \Big\{ \Big(f_t(x_t) + \sum_{i=1}^{n} r_t^i(x_t^{(i)}) \Big) - \Big(f_t(x_*) + \sum_{i=1}^{n} r_t^i(x_*^{(i)}) \Big) \Big\} \\ &= \sum_{t=1}^{T} \Big\{ \Big(f_t(x_t) + \sum_{i=1}^{n} c_t^{(i)} r^i(x_t^{(i)}) \Big) - \Big(f_t(x_*) + \sum_{i=1}^{n} c_t^{(i)} r^i(x_*^{(i)}) \Big) \Big\} \\ &\leq U_\gamma(T), \end{aligned} \tag{16}$$

where $U_\gamma(T) = 2GD + \Big(\frac{D^2}{2\alpha_0} + 7G^2\alpha_0 \Big) \sqrt{T}$. Note that $R_{\rm cl}(T)$ is different from $R_\gamma(T)$, whereas $\{x_t\}$ in $R_{\rm cl}(T)$ is same as that in $R_\gamma(T)$. In fact, $c_t^{(i)}$ in r_t^i influences $R_\gamma(T)$. We will first evaluate the upper bound of $R_{\rm cl}(T)$ by using $R_\gamma(T)$.

From a simple rearranging, we obtain the upper bound of $R_{\rm cl}(T)$ as follows:

$$\begin{aligned} R_{\rm cl}(T) &= \sum_{t=1}^{T} \Big\{ f_t(x_t) + \sum_{i=1}^{n} r^i(x_t^{(i)}) \Big\} - \sum_{t=1}^{T} \Big\{ f_t(x_*) + \sum_{i=1}^{n} r^i(x_*^{(i)}) \Big\} \\ &\leq \sum_{t=1}^{T} \Big\{ \Big(f_t(x_t) + \sum_{i \in I_t} k_t^{(i)} r^i(x_t^{(i)}) \Big) - \Big(f_t(x_*) + \sum_{i=1}^{n} r^i(x_*^{(i)}) \Big) \Big\} \\ &\quad + \Big| \sum_{t=1}^{T} \sum_{i=1}^{n} r^i(x_t^{(i)}) - \sum_{t=1}^{T} \sum_{i \in I_t} k_t^{(i)} r^i(x_t^{(i)}) \Big|, \end{aligned} \tag{17}$$

where $k_t^{(i)}$ are constants defined by

$$k_t^{(i)} = t - t_\xi^{(i)}. \tag{18}$$

Throughout the subsequent discussions, let $\tilde{c}_t^{(i)}$ be given by

$$\tilde{c}_t^{(i)} = \frac{c_t^{(i)}}{k_t^{(i)}} \ . \tag{19}$$

Using these notations, we have the following lemma.

Lemma 1. *Let $\{x_t\}$ be a sequence generated by CL-FOBOS, and suppose that Assumptions 1 and 2 hold. Then, the following inequality holds:*

$$\begin{aligned} R_{\mathrm{cl}}(T) &\leq U_\gamma(T) + \sum_{t=1}^{T}\sum_{i\in I_t} k_t^{(i)}\Big|1-\tilde{c}_t^{(i)}\Big|\Big|r^i(x_t^{(i)}) - r^i(x_*^{(i)})\Big| \\ &\quad + \Big|\sum_{t=1}^{T}\sum_{i=1}^{n} r^i(x_t^{(i)}) - \sum_{t=1}^{T}\sum_{i\in I_t} k_t^{(i)} r^i(x_t^{(i)})\Big| + nKS. \end{aligned}$$

Proof. To prove this lemma, we evaluate the first term of the right-hand side in (17); that is,

$$\sum_{t=1}^{T}\Big\{\Big(f_t(x_t) + \sum_{i\in I_t} k_t^{(i)} r^i(x_t^{(i)})\Big) - \Big(f_t(x_*) + \sum_{i=1}^{n} r^i(x_*^{(i)})\Big)\Big\}. \tag{20}$$

First, we obtain that

$$\begin{aligned} \sum_{t=1}^{T}\sum_{i=1}^{n} r^i(x_*^{(i)}) &= \sum_{i=1}^{n}\sum_{t=1}^{T} r^i(x_*^{(i)}) \\ &= \sum_{i=1}^{n}\sum_{t=1}^{T_\xi^{(i)}} r^i(x_*^{(i)}) + \sum_{i=1}^{n}\sum_{t=T_\xi^{(i)}+1}^{T} r^i(x_*^{(i)}) \\ &= \sum_{t=1}^{T}\sum_{i\in I_t} k_t^{(i)} r^i(x_*^{(i)}) + \sum_{i=1}^{n}\sum_{t=T_\xi^{(i)}+1}^{T} r^i(x_*^{(i)}) \ , \end{aligned} \tag{21}$$

where the last equation follows from (18) and the property that $r^i(x_*)$ is independent of t. From the conditions (i), (13), and (18) of Assumption 2, we can evaluate the second term in (21) as

$$\begin{aligned} \sum_{i=1}^{n}\sum_{t=T_\xi^{(i)}+1}^{T} r^i(x_*^{(i)}) &= \sum_{i=1}^{n} k_T^{(i)} r^i(x_*^{(i)}) \\ &\geq -K\sum_{i=1}^{n} |r^i(x_*^{(i)})| \\ &\geq -nKS. \end{aligned}$$

Moreover, because $c_t^{(i)} = 0$ for $i \notin I_t$, we obtain that

$$\sum_{i=1}^{n} c_t^{(i)} r^i(x_t^{(i)}) = \sum_{i\in I_t} c_t^{(i)} r^i(x_t^{(i)}). \tag{22}$$

By substituting (21) and (22) into (20), we have that

$$\begin{aligned}
&\sum_{t=1}^{T}\Big\{\Big(f_t(x_t)+\sum_{i\in I_t}k_t^{(i)}r^i(x_t^{(i)})\Big)-\Big(f_t(x_*)+\sum_{i=1}^{n}r^i(x_*^{(i)})\Big)\Big\}\\
&=\sum_{t=1}^{T}\Big\{\Big(f_t(x_t)+\sum_{i\in I_t}k_t^{(i)}r^i(x_t^{(i)})\Big)-\Big(f_t(x_*)+\sum_{i\in I_t}k_t^{(i)}r^i(x_*^{(i)})\Big)\Big\}-\sum_{i=1}^{n}\sum_{t=T_\xi^{(i)}+1}^{T}r^i(x_*^{(i)})\\
&\le\sum_{t=1}^{T}\Big\{\Big(f_t(x_t)+\sum_{i=1}^{n}c_t^{(i)}r^i(x_t^{(i)})\Big)-\Big(f_t(x_*)+\sum_{i=1}^{n}c_t^{(i)}r^i(x_*^{(i)})\Big)\Big\}+\Big|\sum_{t=1}^{T}\sum_{i\in I_t}k_t^{(i)}r^i(x_t^{(i)})\\
&\quad-\sum_{t=1}^{T}\sum_{i=1}^{n}c_t^{(i)}r^i(x_t^{(i)})+\sum_{t=1}^{T}\sum_{i=1}^{n}c_t^{(i)}r^i(x_*^{(i)})-\sum_{t=1}^{T}\sum_{i\in I_t}k_t^{(i)}r^i(x_*^{(i)})\Big|+nKS\\
&\le U_\gamma(T)+\Big|\sum_{t=1}^{T}\sum_{i\in I_t}(k_t^{(i)}-c_t^{(i)})\big(r^i(x_t^{(i)})-r^i(x_*^{(i)})\big)\Big|+nKS, \qquad (23)
\end{aligned}$$

where the last inequality of (23) follows from (16). Finally, by replacing $c_t^{(i)}$ in (23) with $\tilde{c}_t^{(i)}$ from (19), we obtain

$$\begin{aligned}
&\sum_{t=1}^{T}\Big\{\Big(f_t(x_t)+\sum_{i\in I_t}k_t^{(i)}r^i(x_t^{(i)})\Big)-\Big(f_t(x_*)+\sum_{i=1}^{n}r^i(x_*^{(i)})\Big)\Big\}\\
&\le U_\gamma(T)+\sum_{t=1}^{T}\sum_{i\in I_t}k_t^{(i)}\Big|1-\tilde{c}_t^{(i)}\Big|\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big|+nKS.
\end{aligned}$$

□

From Lemma 1, we can now obtain the upper bound of the regret $R_{\mathrm{cl}}(T)$ as

$$R_{\mathrm{cl}}(T)\le U_\gamma(T)+P(T)+Q(T)+nKS, \qquad (24)$$

where $P(T)$ and $Q(T)$ are defined by

$$P(T) = \sum_{t=1}^{T}\sum_{i\in I_t}k_t^{(i)}\Big|1-\tilde{c}_t^{(i)}\Big|\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big|. \qquad (25)$$

$$Q(T) = \Big|\sum_{t=1}^{T}\sum_{i=1}^{n}r^i(x_t^{(i)})-\sum_{t=1}^{T}\sum_{i\in I_t}k_t^{(i)}r^i(x_t^{(i)})\Big|, \qquad (26)$$

respectively. We will now evaluate the upper bounds of $P(T)$ and $Q(T)$. For this purpose, let us first derive an upper bound for the term $|1-\tilde{c}_t^{(i)}|$.

Lemma 2. *Suppose that Assumption 2 holds. Then, we have that*

$$|1-\tilde{c}_t^{(i)}|\le\delta(t)$$

for $i \in I_t$, where $\delta(t)$ is given by

$$\delta(t) \;=\; \begin{cases} \sqrt{t}-1 & if \quad t \le K-1 \\ \frac{K-1}{2(t-K+1)} & otherwise. \end{cases}$$

Proof. From the definitions of $\tilde{c}_t^{(i)}$ and $c_t^{(i)}$, we have for each $i \in I_t$,

$$\begin{aligned} \left|1-\tilde{c}_t^{(i)}\right| &= \left|1-\frac{c_t^{(i)}}{k_t^{(i)}}\right| \\ &= \left|1-\frac{\sum_{j=t_\xi^{(i)}+1}^{t}\alpha_j}{\alpha_t k_t^{(i)}}\right|. \end{aligned} \tag{27}$$

Note that $\alpha_j \ge \alpha_t$ holds for $j \le t$, because it follows from Assumption 2 (iv) that $\alpha_t = \frac{\alpha_0}{\sqrt{t}}$. Then, it follows from (18) that

$$\begin{aligned} \frac{\sum_{j=t_\xi^{(i)}+1}^{t}\alpha_j}{\alpha_t k_t^{(i)}} &\ge \frac{\sum_{j=t_\xi^{(i)}+1}^{t}\alpha_t}{\alpha_t k_t^{(i)}} \\ &= \frac{(t-t_\xi^{(i)})\alpha_t}{\alpha_t k_t^{(i)}} \\ &= 1. \end{aligned} \tag{28}$$

Moreover, from (27) and (28), we obtain that

$$\begin{aligned} \left|1-\frac{\sum_{j=t_\xi^{(i)}+1}^{t}\alpha_j}{\alpha_t k_t^{(i)}}\right| &= \frac{\sum_{j=t_\xi^{(i)}+1}^{t}\alpha_j}{\alpha_t k_t^{(i)}} - 1 \\ &= \frac{\sum_{j=t_\xi^{(i)}+1}^{t}(\alpha_j-\alpha_t)}{\alpha_t k_t^{(i)}}. \end{aligned} \tag{29}$$

Next, let us consider the upper bound of α_j in (29). When $t > K-1$, we have that $0 < t-K+1 \le t_\xi^{(i)}+1$, from Assumption 2 (i). Then, we have that $\alpha_j \le \alpha_{t-K+1}$ holds for

$j = t_\xi^{(i)} + 1, \ldots, t$, which follows from Assumption 2 (iv). Hence, (29) can be rewritten as

$$\begin{aligned}
\left|1 - \frac{\sum_{j=t_\xi^{(i)}+1}^{t} \alpha_j}{\alpha_t k_t^{(i)}}\right| &= \frac{\sum_{j=t_\xi^{(i)}+1}^{t} (\alpha_j - \alpha_t)}{\alpha_t k_t^{(i)}} \\
&\leq \frac{\sum_{j=t_\xi^{(i)}+1}^{t} (\alpha_{t-K+1} - \alpha_t)}{\alpha_t k_t^{(i)}} \\
&= \frac{k_t^{(i)} (\alpha_{t-K+1} - \alpha_t)}{\alpha_t k_t^{(i)}} \\
&= \frac{(\alpha_{t-K+1} - \alpha_t)}{\alpha_t} \\
&= \frac{\left(\frac{\alpha_0}{\sqrt{t-K+1}} - \frac{\alpha_0}{\sqrt{t}}\right)}{\frac{\alpha_0}{\sqrt{t}}} \\
&= \frac{\sqrt{t} - \sqrt{t-K+1}}{\sqrt{t-K+1}} \\
&= \frac{K-1}{\sqrt{t-K+1}(\sqrt{t} + \sqrt{t-K+1})} \\
&\leq \frac{K-1}{\sqrt{t-K+1}(2\sqrt{t-K+1})} \\
&= \frac{K-1}{2(t-K+1)}. \qquad (30)
\end{aligned}$$

On the other hand, when $t \leq K-1$, $\alpha_j \leq \alpha_0$ holds for any j. Then, it then follows from (18) that (29) can be evaluated as

$$\begin{aligned}
\left|1 - \frac{\sum_{j=t_\xi^{(i)}+1}^{t} \alpha_j}{\alpha_t k_t^{(i)}}\right| &= \frac{\sum_{j=t_\xi^{(i)}+1}^{t} (\alpha_j - \alpha_t)}{\alpha_t k_t^{(i)}} \\
&\leq \frac{\sum_{j=t_\xi^{(i)}+1}^{t} (\alpha_0 - \alpha_t)}{\alpha_t k_t^{(i)}} \\
&= \frac{(\alpha_0 - \alpha_t) k_t^{(i)}}{\alpha_t k_t^{(i)}} \\
&= \sqrt{t} - 1\ . \qquad (31)
\end{aligned}$$

Consequently, we have from (27), (30), and (31) that $\left|1 - \tilde{c}_t^{(i)}\right| \leq \delta(t)$. □

Using Lemma 2, we can give an upper bound for $P(T)$ as follows.

Lemma 3. *Suppose that Assumptions 1 and 2 hold. Then, the following inequality holds:*

$$P(T) \leq K(K\sqrt{K} - K + 1)R + \frac{K(K-1)R}{2}\big(1 + \log(T - K + 1)\big).$$

Proof. By applying Lemma 2 to the definition (25) of $P(T)$, we get that

$$\begin{aligned}
P(T) &= \sum_{t=1}^{T}\sum_{i\in I_t} k_t^{(i)}\Big|1-\tilde{c}_t^{(i)}\Big|\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big| \\
&\leq K\sum_{t=1}^{T}\sum_{i\in I_t}\Big|1-\tilde{c}_t^{(i)}\Big|\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big| \\
&\leq K\sum_{t=1}^{T}\sum_{i\in I_t}\delta(t)\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big| \\
&\leq K\sum_{t=1}^{T}\sum_{i=1}^{n}\delta(t)\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big| \\
&= K\sum_{t=1}^{K-1}\Big\{(\sqrt{t}-1)\sum_{i=1}^{n}\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big|\Big\}+K\sum_{t=K}^{T}\Big\{\frac{K-1}{2(t-K+1)}\sum_{i=1}^{n}\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big|\Big\},
\end{aligned}$$

where the first inequality follows from Assumption 2 (i). Moreover, by using (12) we have that

$$\begin{aligned}
P(T) &\leq K\sum_{t=1}^{K-1}\Big\{(\sqrt{t}-1)\sum_{i=1}^{n}\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big|\Big\}+K\sum_{t=K}^{T}\Big\{\frac{K-1}{2(t-K+1)}\sum_{i=1}^{n}\Big|r^i(x_t^{(i)})-r^i(x_*^{(i)})\Big|\Big\} \\
&\leq KR\sum_{t=1}^{K-1}(\sqrt{t}-1)+\frac{K(K-1)R}{2}\sum_{t=K}^{T}\frac{1}{t-K+1}. \qquad (32)
\end{aligned}$$

The first term of the right-hand side in (32) can be evaluated as

$$\sum_{t=1}^{K-1}(\sqrt{t}-1)\leq\int_1^K(\sqrt{t}-1)dt \;<\; K\sqrt{K}-K+1.$$

Moreover, the second term of the right-hand side in (32) is bounded above, as

$$\sum_{t=K}^{T}\frac{1}{t-K+1}=\sum_{i=1}^{T-K+1}\frac{1}{i}\;\leq\;1+\log(T-K+1).$$

By substituting these two inequalities into (32), we obtain the desired inequality. □

Finally, we now derive the upper bound of $Q(T)$ in (24).

Lemma 4. *Let $\{x_t\}$ be a sequence generated by CL-FOBOS, and suppose that Assumptions 1 and 2 hold. Then, it holds that $Q(T)\leq nKS$.*

Proof. To simplify, let $\underline{Q^{(i)}}(T)$ and $\overline{Q^{(i)}}(T)$ be given by

$$\begin{aligned}
\underline{Q^{(i)}}(T) &= \Big|\sum_{t=1}^{T_\xi^{(i)}} r^i(x_t^{(i)})-\sum_{t=1}^{T_\xi^{(i)}}\kappa_t^{(i)}r^i(x_t^{(i)})\Big|, \\
\overline{Q^{(i)}}(T) &= \Big|\sum_{t=T_\xi^{(i)}+1}^{T} r^i(x_t^{(i)})\Big|,
\end{aligned}$$

respectively, where $T_\xi^{(i)}$ denotes the most recent iteration where $x^{(i)}$ was updated prior to the Tth iteration, and each $\kappa_t^{(i)}$ is given by

$$\kappa_t^{(i)} = \begin{cases} k_t^{(i)} & if \quad i \in I_t \\ 0 & otherwise. \end{cases}$$

First, we first show that

$$\sum_{i=1}^{n} \Big\{ \underline{Q^{(i)}}(T) + \overline{Q^{(i)}}(T) \Big\} \geq Q(T). \tag{33}$$

From the triangle inequality, we have that

$$\begin{aligned} \sum_{i=1}^{n} \Big\{ \underline{Q^{(i)}}(T) + \overline{Q^{(i)}}(T) \Big\} &= \sum_{i=1}^{n} \Big\{ \Big| \sum_{t=1}^{T_\xi^{(i)}} r^i(x_t^{(i)}) - \sum_{t=1}^{T_\xi^{(i)}} \kappa_t^{(i)} r^i(x_t^{(i)}) \Big| + \Big| \sum_{t=T_\xi^{(i)}+1}^{T} r^i(x_t^{(i)}) \Big| \Big\} \\ &\geq \Big| \sum_{i=1}^{n} \sum_{t=1}^{T_\xi^{(i)}} r^i(x_t^{(i)}) - \sum_{i=1}^{n} \sum_{t=1}^{T_\xi^{(i)}} \kappa_t^{(i)} r^i(x_t^{(i)}) + \sum_{i=1}^{n} \sum_{t=T_\xi^{(i)}+1}^{T} r^i(x_t^{(i)}) \Big| \\ &= \Big| \sum_{i=1}^{n} \sum_{t=1}^{T} r^i(x_t^{(i)}) - \sum_{i=1}^{n} \sum_{t=1}^{T_\xi^{(i)}} \kappa_t^{(i)} r^i(x_t^{(i)}) \Big|. \end{aligned}$$

It follows from the definition of $\kappa_t^{(i)}$ and (18) that

$$\begin{aligned} \sum_{i=1}^{n} \Big\{ \underline{Q^{(i)}}(T) + \overline{Q^{(i)}}(T) \Big\} &\geq \Big| \sum_{i=1}^{n} \sum_{t=1}^{T} r^i(x_t^{(i)}) - \sum_{i=1}^{n} \sum_{t=1}^{T_\xi^{(i)}} \kappa_t^{(i)} r^i(x_t^{(i)}) \Big| \\ &= \Big| \sum_{t=1}^{T} \sum_{i=1}^{n} r^i(x_t^{(i)}) - \sum_{i=1}^{n} \sum_{t=1}^{T} \kappa_t^{(i)} r^i(x_t^{(i)}) \Big| \\ &= \Big| \sum_{t=1}^{T} \sum_{i=1}^{n} r^i(x_t^{(i)}) - \sum_{t=1}^{T} \sum_{i=1}^{n} \kappa_t^{(i)} r^i(x_t^{(i)}) \Big| \\ &= \Big| \sum_{t=1}^{T} \sum_{i=1}^{n} r^i(x_t^{(i)}) - \sum_{t=1}^{T} \sum_{i \in I_t} k_t^{(i)} r^i(x_t^{(i)}) \Big|. \end{aligned}$$

We see that the last term is $Q(T)$, and hence we have that $\sum_{i=1}^{n} \Big\{ \underline{Q^{(i)}}(T) + \overline{Q^{(i)}}(T) \Big\} \geq Q(T)$. From this inequality, we may evaluate both $\underline{Q^{(i)}}(T)$ and $\overline{Q^{(i)}}(T)$ to obtain the upper bound of $Q(T)$.

First, we first define some notations to be employed in these evaluations. Let $l(i,m)$ be the iteration at which $x^{(i)}$ has been updated m times with $l(i,0) = 0$. Furthermore, let $M_T^{(i)}$ denote the number times that $x^{(i)}$ has been updated by the Tth iteration. Note that $M_T^{(i)}$ satisfies

$$l(i, M_T^{(i)}) = T_\xi^{(i)}, \tag{34}$$

because the most recent iteration at which $x^{(i)}$ has been updated before the Tth iteration is the same as that at which $x^{(i)}$ has been updated $M_T^{(i)}$ times. We consider the properties of the sequence $\{x_t^{(i)}\}$ with respect to $l(i,m)$ and $M_T^{(i)}$. Because $\{x_t^{(i)}\}$ is the sequence generated by (10), $x_t^{(i)}$ is only updated when $i \in I_t$. Therefore we have that

$$i \in I_t \quad \Leftrightarrow \quad t \in \left\{ l(i,m) \;\middle|\; m = 1, \ldots, M_T^{(i)} \right\}, \tag{35}$$

$$i \notin I_t \quad \Leftrightarrow \quad t \in \left\{ l(i,m)+1, \ldots, l(i,m+1)-1 \;\middle|\; m = 1, \ldots, M_T^{(i)} \right\}. \tag{36}$$

In particular, we have that $x_{t+1}^{(i)} = x_t^{(i)}$ holds at each iteration such that $i \notin I_t$. Then, it follows from (36) that

$$x_t^{(i)} = x_{l(i,m+1)}^{(i)} \quad (t = l(i,m)+1, \ldots, l(i,m+1)). \tag{37}$$

Now, using these notations we evaluate will $\underline{Q^{(i)}}(T)$ and $\overline{Q^{(i)}}(T)$. First, we can use (35) to rewrite $\underline{Q^{(i)}}(T)$ as

$$\begin{aligned} \underline{Q^{(i)}}(T) &= \left| \sum_{t=1}^{T_\xi^{(i)}} r^i(x_t^{(i)}) - \sum_{t=1}^{T_\xi^{(i)}} \sum_{i \in I_t} k_t^{(i)} r^i(x_t^{(i)}) \right| \\ &= \left| \sum_{m=0}^{M_T^{(i)}-1} \sum_{t=l(i,m)+1}^{l(i,m+1)} r^i(x_t^{(i)}) - \sum_{m=0}^{M_T^{(i)}-1} k_{l(i,m+1)}^{(i)} r^i(x_{l(i,m+1)}^{(i)}) \right|. \end{aligned}$$

Here, note that $k_{l(i,m+1)}^{(i)} r^i(x_{l(i,m+1)}^{(i)}) = \sum_{t=l(i,m)+1}^{l(i,m+1)} r^i(x_{l(i,m+1)}^{(i)})$, because $k_{l(i,m+1)}^{(i)} = l(i,m+1) - l(i,m)$. Therefore, we have that

$$\begin{aligned} \underline{Q^{(i)}}(T) &= \left| \sum_{m=0}^{M_T^{(i)}-1} \sum_{t=l(i,m)+1}^{l(i,m+1)} r^i(x_t^{(i)}) - \sum_{m=0}^{M_T^{(i)}-1} \sum_{t=l(i,m)+1}^{l(i,m+1)} r^i(x_{l(i,m+1)}^{(i)}) \right| \\ &= 0, \end{aligned} \tag{38}$$

where the last equation follows from (37).

On the other hand, from conditions (i) and (13) of Assumption 2 we have that

$$\begin{aligned} \overline{Q^{(i)}}(T) &= \left| \sum_{t=T_\xi^{(i)}+1}^{T} r^i(x_t^{(i)}) \right| \\ &\leq \left| \sum_{t=T-K+1}^{T} r^i(x_t^{(i)}) \right| \\ &\leq KS. \end{aligned} \tag{39}$$

Furthermore, from (33), (38), and (39), we obtain that $Q(T) \leq nKS$. □

Proof of Theorem 1. It is obvious that Theorem 1 follows from Lemma 3, Lemma 4, (16), and (24). □

Theorem 1 yields that $\lim_{T\to\infty} \frac{R_{\mathrm{cl}}(T)}{T} = 0$.

We can regard CL-FOBOS as not only an online gradient method, but also as a stochastic gradient method for the following stochastic programming problem:

$$\min \quad \mathrm{E}\big(\theta(x,z)\big), \tag{40}$$

where z is a random variable, and θ is the loss function with respect to z and a parameter x. Problem (40) is approximately equivalent to the following problem:

$$\min \quad \phi(x) \equiv \frac{1}{T}\sum_{t=1}^{T} \theta(x, z_t), \tag{41}$$

where z_t represents a random variable for each sample. Now, we obtain the following theorem in a way similar to [15].

Theorem 2. *Let $\{x_t\}$ be the sequence generated by CL-FOBOS for problem (41), and suppose that Assumptions 1 and 2 hold. Then, we have that*

$$\lim_{T\to\infty} \mathrm{E}\Big[\phi\Big(\frac{1}{T}\sum_{t=1}^{T} x_t\Big)\Big] - \phi^* = 0,$$

where ϕ^* is the optimal value of problem (40).

5 Numerical Experiments

In this section, we present some numerical results using CL-FOBOS in order to verify its efficiency. The experiments are carried out on a machine with a 1.7 GHz Intel Core i7 CPU and 8 GB memory, and we implement all codes in C++. We set the initial point x_1 to 0 in all experiments.

We compare the performance of CL-FOBOS with L-FOBOS [10] for binary classification problems. We use the logistic function $f_t(x) = \log\big(1 + \exp(-b_t\langle a_t, x\rangle)\big)$ and the L1 regularization term $r(x) = \lambda\sum_{i=1}^{n} |x^{(i)}|$, where λ is a positive constant that controls the sparsity of solutions, and (a_t, b_t) denotes the tth sample data.

We use an Amazon review dataset, called the Multi-Domain Sentiment Dataset [3], for the experiments. This dataset contains many merchandise reviews from Amazon.com. We attempt to divide the reviews into two classes, those expressing positive impressions of the merchandises and those that are negative.

The data for the tth review is composed of a vector $a_t \in \mathbb{R}^n$ and a label $b_t \in \{+1, -1\}$. The vector $a_t \in \mathbb{R}^n$ is constructed by the so-called "Bag-of-Words" process, and $a_t^{(i)}$ represents the number of times that the word represented by i appears in the tth sentence of the review. Hence, a_t is regarded as a feature vector of the tth sentence of the review. Meanwhile, the label b_t indicates the evaluation of the merchandise given by the author of the tth review, where "$+1$" represents a positive impression and "-1" represents a negative one.

In the experiments, we only use the "Book" domain from the dataset. The book domain is comprised of 4,465 samples, and the total number of words in all samples is 332,440. Each feature vector $a_t (t = 1, \ldots, M)$ contains at least 11 (at most 5,412) nonzero elements, and the average number of nonzero elements is 228.

The number of samples, 4,465, seems small for comparing online gradient methods. Therefore, we constructed a large dataset using the $4,465$ samples. That is, we randomly choose $100,000$ samples from the original 4,465, and we employed this as the dataset for the experiments. We compare CL-FOBOS with L-FOBOS in the following four aspects: the average of the total loss defined in (42), the rate of correct estimations, the number of nonzero components in $\{x_t\}$, and the run time.

$$\bar{R}(T) = \frac{\sum_{t=1}^{T}\left\{f_t(x_t) + \sum_{i=1}^{n} r^{(i)}(x_t^i)\right\}}{T}. \tag{42}$$

Note that $\bar{R}(T)$ represents a kind of the regret $R(T)$. The rate of correct estimations is defined as follows. We estimate the class of a sample at each iteration using $\langle a_t, x_t\rangle$, and then count the number of correct estimations; that is, the number of samples t such that $b_t\langle a_t, x_t\rangle > 0$. The rate of correct estimations is then the number of correct estimations divided by the number of samples that have been observed.

We tested both methods with $\alpha_t = \frac{\alpha_0}{\sqrt{t}}$, where α_0 denotes the initial stepsize, and is set between 0.005 and 1.0 with width of 0.005. We present the results achieved by each method with the best choice of α_0; that is, we choose α_0 such that smallest $\bar{R}(T)$ is obtained.

Tables 1-6 show the run time, $\bar{R}(T)$, and the rate of correct estimations for both methods. All of the numbers in the tables indicate an average taken over 10 trials for 10 randomly generated datasets.

We can see from Tables 1-6 that the run time of L-FOBOS is shortened as K becomes larger. This is because for large K L-FOBOS rarely carries out Step 2, which requires $\mathcal{O}(n)$ computations. We also observe that the rate of correct estimations for CL-FOBOS is equal to or better than that of L-FOBOS when $\lambda \leq 10^{-2}$. This shows that CL-FOBOS effectively exploits the information given by r^i when λ is sufficiently large. However, CL-FOBOS achieved a remarkably low accuracy rate when λ is large. The main reason for this is that most of the values $x_t^{(i)}$ become close zero at each iteration in CL-FOBOS, whereas L-FOBOS only considers the loss functions other than at iterations such that t mod K $= 0$. Thus, the result does not imply that CL-FOBOS is inferior to L-FOBOS.

Next, we consider the relation between $\bar{R}(T)$ and the run time. Figure 1 presents a summary of Tables 1-6 in terms of $\bar{R}(T)$ and the run time. From this figure, we can confirm the tradeoff of L-FOBOS as we change K when $\lambda \geq 5 \times 10^{-3}$. Furthermore, there exists no parameter K that enables L-FOBOS to outperform CL-FOBOS in terms of both the run time and $\bar{R}(T)$. These results indicate that, to a certain degree, CL-FOBOS is more efficient when λ is large. In addition, CL-FOBOS does not require the tuning up of the parameter K, which is an advantage of the proposed method.

On the other hand, when λ is smaller than 5×10^{-4}, such a tradeoff does not occur, because the influence of the L1 regularization term is very small.

Finally, we note the sparsity of solutions $\{x_t\}$ generated by both methods. Figure 2 illustrates the percentage of nonzero components in $\{x_t\}$ when $\lambda = 5 \times 10^{-3}$ and $\lambda = 10^{-5}$. The red lines indicate the performance of CL-FOBOS, and the green lines indicate that of L-FOBOS. When $\lambda = 5 \times 10^{-3}$, we employ $K = 400$ for L-FOBOS with an initial stepsize of $\alpha_0 = 0.04$. Further, when $\lambda = 10^{-5}$, we choose $K = 750$ for this method, with $\alpha_0 = 0.595$. These candidates for K and α are chosen because L-FOBOS achieves the same run time with them as CL-FOBOS. In this figure, we observe that the green lines are jagged. This is because

Table 1: Results for each algorithm, $\lambda = 10^{-1}$

Algorithm Name	α_0	Run Time	$\bar{R}(T)$	Rate of Right Estimations
L-FOBOS ($K = 100$)	0.005	2.00	0.708	0.533
L-FOBOS ($K = 500$)	0.005	1.0	0.759	0.548
L-FOBOS ($K = 1000$)	0.005	0.9	0.818	0.560
CL-FOBOS	0.005	0.95	0.694	0.526

Table 2: Results for each algorithm, $\lambda = 10^{-2}$

Algorithm Name	α_0	Run Time	$\bar{R}(T)$	Rate of Right Estimations
L-FOBOS ($K = 100$)	0.03	2.15	0.646	0.737
L-FOBOS ($K = 500$)	0.015	1.05	0.662	0.726
L-FOBOS ($K = 1000$)	0.01	0.95	0.673	0.719
CL-FOBOS	0.03	1.2	0.645	0.738

Table 3: Results for each algorithm, $\lambda = 5 \times 10^{-3}$

Algorithm Name	α_0	Run Time	$\bar{R}(T)$	Rate of Right Estimations
L-FOBOS ($K = 100$)	0.06	2.15	0.599	0.788
L-FOBOS ($K = 500$)	0.035	0.9	0.612	0.780
L-FOBOS ($K = 1000$)	0.025	0.8	0.625	0.774
CL-FOBOS	0.055	1.05	0.601	0.787

Table 4: Results for each algorithms, $\lambda = 5 \times 10^{-4}$

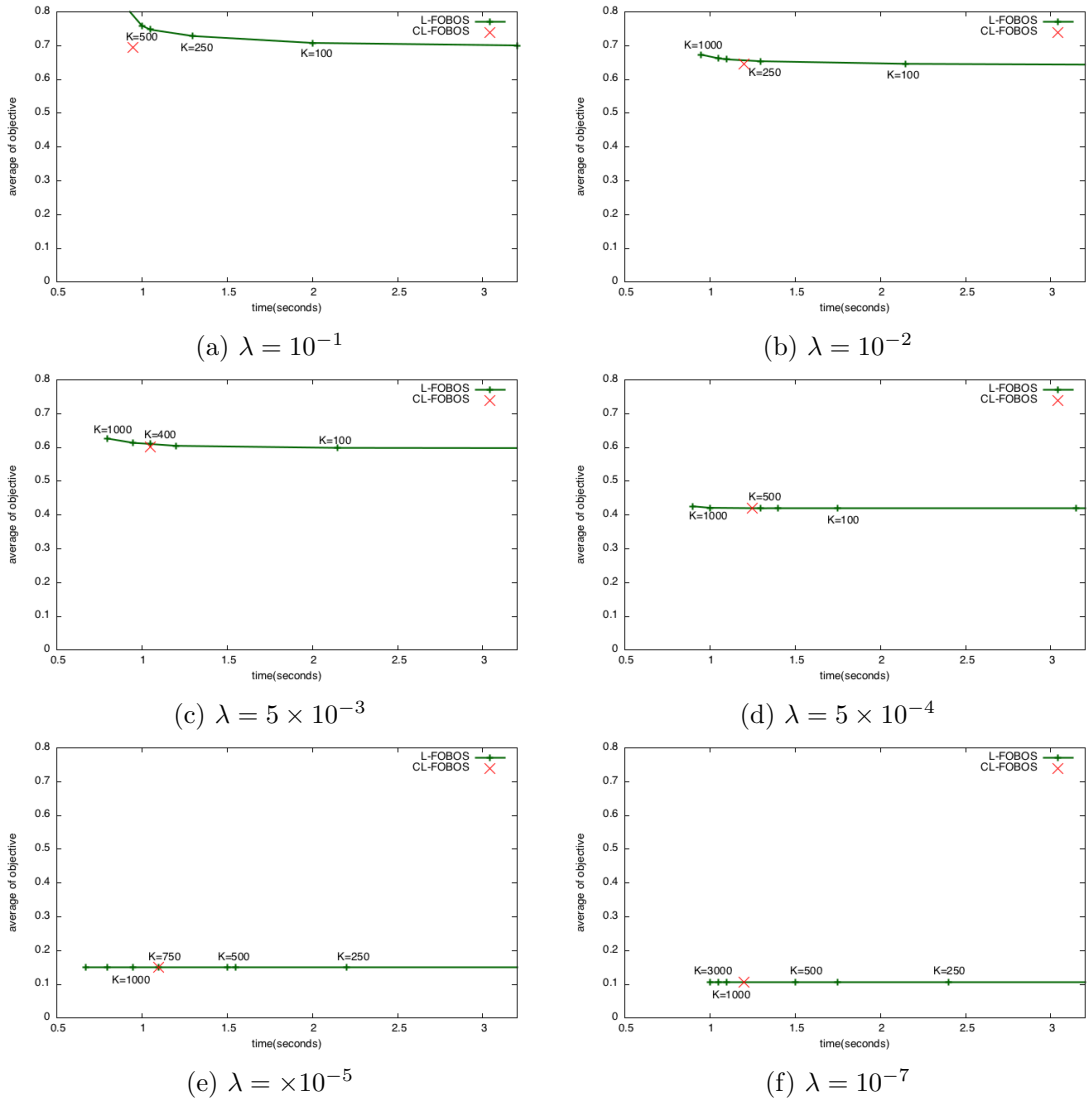
Algorithm Name	α_0	Run Time	$\bar{R}(T)$	Rate of Right Estimations
L-FOBOS ($K = 100$)	0.155	3.15	0.4199	0.906
L-FOBOS ($K = 500$)	0.155	1.3	0.4199	0.906
L-FOBOS ($K = 1000$)	0.155	1.0	0.421	0.905
CL-FOBOS	0.155	1.25	0.421	0.905

Table 5: Results for each algorithm, $\lambda = 10^{-5}$

Algorithm Name	α_0	Run Time	$\bar{R}(T)$	Rate of Right Estimations
L-FOBOS ($K = 100$)	0.595	5.00	0.1514	0.981
L-FOBOS ($K = 500$)	0.595	1.45	0.1514	0.981
L-FOBOS ($K = 1000$)	0.595	1.00	0.1514	0.9809
CL-FOBOS	0.595	1.00	0.152	0.9808

Table 6: Results for each algorithm, $\lambda = 10^{-7}$

Algorithm Name	α_0	Run Time	$\bar{R}(T)$	Rate of Right Estimations
L-FOBOS ($K = 100$)	0.87	4.9	0.1061	0.979
L-FOBOS ($K = 500$)	0.87	1.5	0.1061	0.979
L-FOBOS ($K = 1000$)	0.87	1.10	0.1061	0.979
CL-FOBOS	0.87	1.20	0.1061	0.979

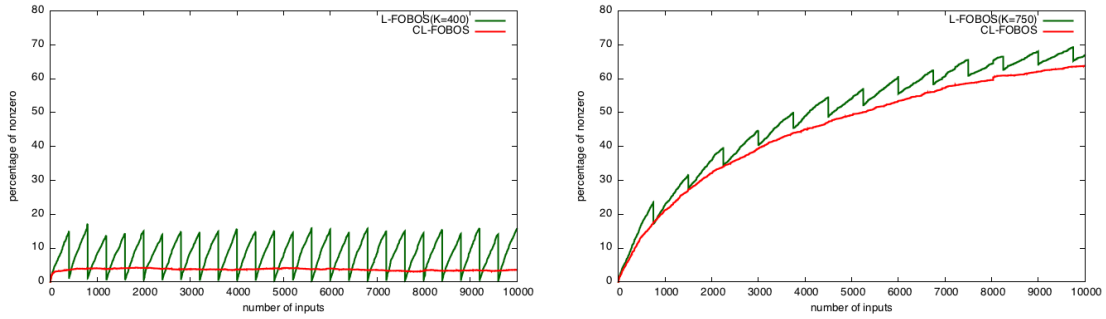


(a) $\lambda = 10^{-1}$ (b) $\lambda = 10^{-2}$

(c) $\lambda = 5 \times 10^{-3}$ (d) $\lambda = 5 \times 10^{-4}$

(e) $\lambda = \times 10^{-5}$ (f) $\lambda = 10^{-7}$

Figure 1: Relation between $\bar{R}(T)$ and the run time



(a) $\lambda = 5 \times 10^{-3}$ (b) $\lambda = 10^{-5}$

Figure 2: The number of nonzero components in $\{x_t\}$

x_t becomes radically sparse when t mod K $= 0$, on account of the lazy evaluation.

In particular, when $\lambda = 5 \times 10^{-3}$ the solution x_t obtained by L-FOBOS becomes approximately zero at such iterations. Meanwhile, when we set $\lambda = 10^{-5}$, we see that both methods generate more dense solutions, because the L1 regularization is almost ignored.

In addition, we can observe the results of classification for the original $4,465$ samples using several solutions, such as x_{399}, x_{400}, x_{9999}, x_{10000}, x_{99999}, and x_{100000}. The rate and number of correct estimations for each solution are summarized in Table 7. Moreover, Figure 3 illustrates the results.

Table 7: Rate of correct estimations, $\lambda = 5 \times 10^{-3}$. The numbers in parenthesis denote the number of correct estimations.

Solution Number	x_{399}	x_{400}	x_{9999}	x_{10000}	x_{99999}	x_{100000}
L-FOBOS ($K = 400$)	68.7% (3,066)	57.3% (2,953)	78.4% (3,501)	77.9% (3,476)	79.2% (3,537)	79.0% (3,529)
CL-FOBOS	67.5% (3,015)	67.4% (3,020)	78.5% (3,504)	78.5% (3,506)	79.1% (3,533)	79.0% (3,530)

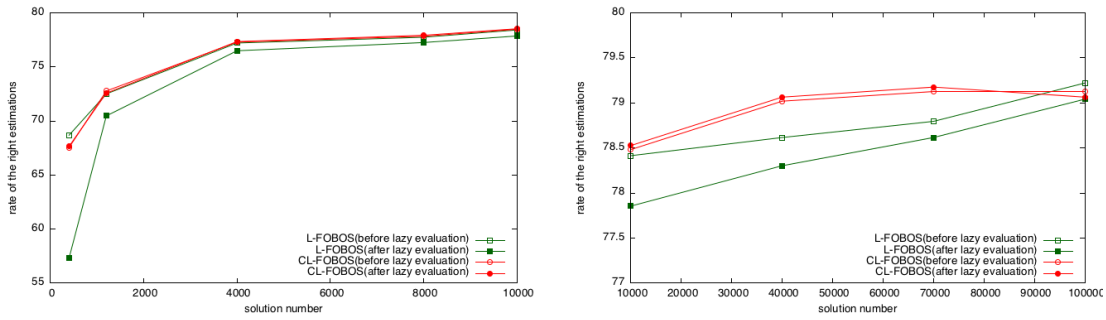


Figure 3: Number of correct estimations

We observe from Table 7 and Figure 3 that the rate of correct estimations decreases in L-FOBOS following its lazy evaluation. This means that L-FOBOS cannot achieve the two

aims of generating sparse solutions and minimizing the total loss simultaneously. On the other hand, CL-FOBOS is well-balanced. That is, it always creates sparse solutions, and its accuracy is improved as it observes samples.

In conclusion, we confirm that the proposed method, CL-FOBOS, is more efficient for problems that are strongly influenced by the function r.

6 Conclusion

In this paper, we have proposed a novel algorithm, CL-FOBOS, for large-scale problems with sparse gradients. Furthermore, we have derived a bound on the regret of CL-FOBOS. Finally, we confirmed the efficiency of the proposed method using numerical experiments.

As a direction for future work, it would be worth investigating a reasonable stepsize rule to be determined before CL-FOBOS starts.

References

[1] Bertsekas, P, D. : *Incremental Gradient, Subgradient, and Proximal Methods for Convex Optimization: A Survey*, Lab. for information and decision systems report LIDSP-2848, MIT, pp.1-38, 2010.

[2] Bishop, C. : *Pattern Recognition and Machine Learning*, Springer-Verlag New York, 2006.

[3] Blitzer, J., Dredze, M., Pereira, F. : *Domain Adaptation for Sentiment Classification*, In proceedings of the 45th annual meeting of the association of computational linguistics, pp. 440-447, 2007.

[4] Blondel, M., Seki, K., Uehara, K. : *Block Coordinate Descent Algorithms for Large-scale Sparse Multiclass Classification*, The journal of machine learning research volume 93 , pp. 31-52, 2013.

[5] Bottou, L. : *Online Algorithms and Stochastic Approximations*, Online learning and neural networks, pp. 9-42, 1998.

[6] Cesa-Bianchi, N. : *Analysis of Two Gradient-based Algorithms for On-line Regression*, Journal of computer and system sciences volume 59, pp. 392-411, 1999.

[7] Dredze, M., Crammer, K., Pereira. : *Confidence-weighted Linear Classification*, Proceedings of international conference on machine learning, pp. 264-271, 2008.

[8] Duchi, J., Singer, Y. : *Efficient Online and Batch Learning Using Forward Backward Splitting*, The journal of machine learning research volume 10 , pp. 2899-2934, 2009.

[9] Duchi, J., Shalev-Shwartz, S., Singer, Y., Tewari, A. : *Composite Objective Mirror Descent*, Conference on learning theory (COLT), pp. 14-26, 2010.

[10] Langford, J., Li, L., Zhang, T. : *Sparse Online Learning via Truncated Gradient*, The journal of machine learning research volume 10 , pp. 777-801, 2009.

[11] Lipton, Z., Elkan, C. : *Efficient Elastic Net Regularization for Sparse Linear Models*, arXiv:1505.06449, 24 May 2015.

[12] Togari, Y., Yamashita, N. : *Regret Analysis of Online Optimization Algorithm that performs Lazy Evaluation*, Bachelor thesis, Kyoto University, March 2014, `http://www-optima.amp.i.kyoto-u.ac.jp/papers/bachelor/2014_bachelor_togari.pdf` [publish in Japanese].

[13] Togari, Y., Yamashita, N. : *Regret Analysis of Forward-Backward Splitting Method with Component-wise Lazy Evaluation*, In Proceedings of the Spring Meeting of the Operations Research Society of Japan, 2015 [publish in Japanese].

[14] Shalev-Shwartz, S. : *Online Learning and Online Convex Optimization*, Foundations and trends in machine learning volume 4, no 2 , pp. 107-194, 2011.

[15] Xiao, L. : *Dual Averaging Methods for Regularized Stochastic Learning and Online Optimization*, The journal of machine learning research volume 11, pp. 2543-2596, 2010.

[16] Zinkevich, M. : *Online Convex Programming and Generalized Infinitesimal Gradient Ascent*, Proceedings of the twentieth international conference on machine learning (ICML) pp. 928-936, 2003.