Efficient Greedy Coordinate Descent for Composite Problems

Sai Praneeth Karimireddy* EPFL
Anastasia Koloskova* EPFL
Sebastian U. Stich EPFL
Martin Jaggi EPFL

Abstract

Coordinate descent with random coordinate selection is the current state of the art for many large scale optimization problems. However, greedy selection of the steepest coordinate on smooth problems can yield convergence rates independent of the dimension n, and requiring up to n times fewer iterations.

In this paper, we consider greedy updates that are based on subgradients for a class of non-smooth composite problems, which includes $L1$-regularized problems, SVMs and related applications. For these problems we provide (i) the first linear rates of convergence independent of n, and show that our greedy update rule provides speedups similar to those obtained in the smooth case. This was previously conjectured to be true for a stronger greedy coordinate selection strategy.

Furthermore, we show that (ii) our new selection rule can be mapped to instances of maximum inner product search, allowing to leverage standard nearest neighbor algorithms to speed up convergence. We demonstrate the validity of the approach through extensive numerical experiments.

1 Introduction

In recent years, there has been increased interest in coordinate descent (CD) methods due to their simplicity, low cost per iteration, and efficiency (Wright, 2015). Algorithms based on coordinate descent are the state of the art for many optimization problems (Nesterov, 2012; Shalev-Shwartz and Zhang, 2013b; Lin et al., 2014; Shalev-Shwartz and Zhang, 2013a, 2016; Richtarik and Takac, 2016; Fercoq and Richtárik, 2015; Nesterov and Stich, 2017).Most of the CD methods draw their coordinates from a fixed distribution—for instance from the uniform distribution as in uniform coordinate descent (UCD). However, it is clear that significant improvements can be achieved by choosing more *important* coordinates more frequently (Nesterov, 2012; Nesterov and Stich, 2017; Stich et al., 2017a,b; Perekrestenko et al., 2017). In particular, we could greedily choose the 'best' coordinate at each iteration i.e. the greedy or steepest coordinate descent (GCD).

GCD for composite problems. Consider the smooth quadtratic function $f(\boldsymbol{\alpha}) \overset{\text{def}}{=} \frac{1}{2}\|A\boldsymbol{\alpha} - \boldsymbol{b}\|_2^2$. There are three natural notions of the 'best' coordinate.[1] One could choose (i) `GS-s`: the *steepest* coordinate direction based on (sub)-gradients, (ii) `GS-r`: the coordinate which allows us to take the largest step, and (iii) `GS-q`: the coordinate that allows us to minimize the function value the most. For our example (and in general for smooth functions), the three rules are equivalent. When we add an additional non-smooth function to f, such as $g(\boldsymbol{\alpha}) = \lambda\|\boldsymbol{\alpha}\|_1$, however, the three notions are no more equivalent. The performance of greedy coordinate descent in this composite setting is not well understood, and is the focus of this work.

Iteration complexity of GCD. If the objective f decomposes into n identical separable problems, then clearly GCD is identical to UCD. In all but such extreme cases, Nutini et al. (2015) give a refined analysis of GCD for smooth functions and show that it outperforms UCD. This lead to a renewed interest in greedy methods (e.g. (Karimi et al., 2016; You et al., 2016; Dünner et al., 2017; Song et al., 2017; Nutini et al., 2017; Stich et al., 2017a; Locatello et al., 2018; Lu et al., 2018)). However, for the composite case the analysis in (Nutini et al., 2015) of GCD methods for any of the three rules mentioned earlier falls back to that of UCD. Thus they fail to demonstrate the advantage of greedy methods for the composite case. In fact it is claimed that the rate of the `GS-s` greedy rule may

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

* Equal contribution.

[1] Following standard notation (cf. (Nutini et al., 2015)) we call them the Gauss-Southwell (GS) rules.

even be worse than that of UCD. In this work we provide a refined analysis of GCD for a certain class of composite problems, and show that all three strategies (GS-s, GS-r, and GS-q) converge on composite problems at a rate similar to GCD in the smooth case. Thus for these problems too, greedy coordinate algorithms are provably faster than UCD other than in extreme cases.

Efficiency of GCD. A naïve implementation of GCD would require computing the full gradient at a cost roughly n times more than just computing one coordinate of the gradient as required by UCD. This seems to negate any potential gain of GCD over UCD. The working principle behind *approximate GCD* methods is to trade-off exactness of the greedy direction against the time spent to decide the steepest direction (e.g. (Stich et al., 2017a)). For smooth problems, Dhillon et al. (2011) show that *approximate nearest neighbor search* algorithms can be used to provide in *sublinear time* an approximate steepest descent direction. We build upon these ideas and extend the framework to non-smooth composite problems, thereby capturing a significantly larger class of input problems. In particular we show how to efficiently map the GS-s rule to an instance of maximum inner product search (MIPS).

Contributions. We analyze and advocate the use of the GS-s greedy rule to compute the update direction for composite problems. Our main contributions are:

i) We show that on a class of composite problems, greedy coordinate methods achieve convergence rates which are very similar to those obtained for smooth functions, thereby extending the applicability of GCD. This class of problems covers several important applications such as SVMs (in its dual formulation), Lasso regression, $L1$-regularized logistic regression among others. With this we establish that greedy methods significantly outperform UCD also on composite problems, except in extreme cases (cf. Remark 4).

ii) We show that both the GS-s as well as the GS-r rules achieve convergence rates which are (other than in extreme cases) faster than UCD. This sidesteps the negative results by Nutini et al. (2015) for these methods through a more fine-grained analysis. We also study the effect of approximate greedy directions on the convergence.

iii) Algorithmically, we show how to precisely map the GS-s direction computation as a special instance of a maximum inner product search problem (MIPS). Many standard nearest neighbor algorithms such as e.g. Locality Sensitive Hashing (LSH) can therefore be used to efficiently run GCD on composite optimization problems.

iv) We perform extensive numerical experiments to study the advantages and limitations of our greedy descent combined with a current state-of-the-art MIPS implementation (Boytsov and Naidan, 2013).

Related Literature. Coordinate descent, being one of the earliest known optimization methods, has a rich history (e.g. (Bickley, 1941; Warga, 1963; Bertsekas and Tsitsiklis, 1989, 1991)). A significant renewal in interest followed the works of Nesterov (2012) who provided a simple analysis of the convergence of UCD, and (Shalev-Shwartz and Zhang, 2013b) who apply UCD on the dual problems (called SDCA). In practice, many solvers (e.g. (Ndiaye et al., 2015; Massias et al., 2018)) combine UCD with active set heuristics where attention is restricted to a subset of *active* coordinates. These methods are orthogonal to, and hence can be combined with, the greedy rules studied here.

Greedy coordinate methods can also be viewed as an 'extreme' version of adaptive importance sampling (Stich et al., 2017a; Perekrestenko et al., 2017). For the smooth case, greedy methods choose the coordinate of the gradient with the largest absolute value while importance sampling methods sample coordinates proportional to the absolute value of the gradient coordinate (or its power). Thus, new greedy algorithms can directly be translated into new importance sampling schemes. However unlike greedy methods, even in the smooth case, there are no easily characterized function classes for which the importance sampling schemes or the active set methods are provably faster than UCD. The work closest to ours, other than the already discussed Nutini et al. (2015), would be that of Dhillon et al. (2011). The latter show a sublinear $O(1/t)$ convergence rate for GS-r on composite problems. They also propose a practical variant for $L1$-regularized problems which essentially ignores the regularizer and is hence not guaranteed to converge.

2 Setup

We consider composite optimization problems of the structure

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \Big[F(\boldsymbol{\alpha}) := f(\boldsymbol{\alpha}) + \sum_{i=1}^{n} g_i(\alpha_i)\Big], \tag{1}$$

where n is the number of coordinates, $f\colon \mathbb{R}^n \to \mathbb{R}$ is convex and smooth, and the $g_i\colon \mathbb{R} \to \mathbb{R}$, $i \in [n]$ are convex and possibly non-smooth. In this exposition, we further restrict the function $g(\boldsymbol{\alpha}) := \sum_{i=1}^n g_i(\alpha_i)$ to either enforce a *box constraint* or an $L1$ *regularizer*. This comprises many important problem classes, for instance dual SVM or Lasso regression, see Appendix A.3.

We further assume that the smooth component f is

coordinate-wise L smooth: for any $\boldsymbol{\alpha}$, γ and i,

$$f(\boldsymbol{\alpha}+\gamma\mathbf{e}_i) \leq f(\boldsymbol{\alpha}) + \nabla_i f(\boldsymbol{\alpha})\gamma + \frac{L\gamma^2}{2}\,. \tag{2}$$

Sometimes we will assume that f is in addition also strongly convex with respect to the $\|\cdot\|_p$ norm, $p \in \{1,2\}$, that is,

$$f(\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}) \geq f(\boldsymbol{\alpha}) + \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{\mu_p}{2}\|\Delta\boldsymbol{\alpha}\|_p^2 \tag{3}$$

for any $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}$ in the domain of F. In general it holds $\mu_1 \in [\mu_2/n, \mu_2]$. See Nutini et al. (2015) for a detailed comparison of the two constants.

3 GCD for Non-Smooth Problems

Here we briefly recall the definitions of the `GS-s`, `GS-r` and `GS-q` coordinate selection rules and introduce the approximate `GS-s` rule that we will consider in detail.

$$\texttt{GS-s}(\boldsymbol{\alpha}) := \arg\max_j \left[\min_{s\in\partial g_j} |\nabla_j f(\boldsymbol{\alpha}) + s| \right], \tag{4}$$

$$\texttt{GS-r}(\boldsymbol{\alpha}) := \arg\max_{j\in[n]} |\gamma_j|\,, \tag{5}$$

$$\texttt{GS-q}(\boldsymbol{\alpha}) := \arg\max_{j\in[n]} |\chi_j(\boldsymbol{\alpha})|\,, \tag{6}$$

for an iterate $\boldsymbol{\alpha} \in \mathbb{R}^n$, $\nabla_j f(\boldsymbol{\alpha}) := \langle \nabla f(\boldsymbol{\alpha}), \mathbf{e}_j\rangle$ for standard unit vector $\mathbf{e}_j$. Here $\chi_j(\boldsymbol{\alpha})$ and γ_j are defined as the minimum value and minimizer respectively of

$$\min_\gamma \left[\gamma\nabla_j f(\boldsymbol{\alpha}) + \frac{L\gamma^2}{2} + g_j(\alpha_j+\gamma) - g_j(\alpha_j) \right].$$

We relax the requirement for an exact steepest selection, and define an approximate `GS-s` rule.

Definition 1 (Θ-approximate `GS-s`)**.** *For given $\boldsymbol{\alpha}$, the coordinate j is considered to be a Θ-approximate steepest direction for $\Theta \in (0,1]$ if*

$$\min_{s\in\partial g_j} |\nabla_j f(\boldsymbol{\alpha}) + s| \geq \Theta \max_i \left[\min_{s\in\partial g_i} |\nabla_i f(\boldsymbol{\alpha}) + s| \right].$$

3.1 GCD for $L1$-regularized problems

We now discuss the `GS-s` rule for the concrete example of $L1$ problems, and collect some observations that we will use later to define the mapping to the MIPS instance. A similar discussion is included for box constrained problems in Appendix B.

Consider $L1$-regularized problems of the form

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^n} \left[F(\boldsymbol{\alpha}) := f(\boldsymbol{\alpha}) + \lambda\|\boldsymbol{\alpha}\|_1 \right]. \tag{7}$$

The `GS-s` steepest rule (4) and update rules can be simplified for such functions. Let $\operatorname{sign}(x)$ denote the sign function, and define $S_\lambda(x)$ as the shrinkage operator

$$S_\lambda(x) := \begin{cases} x - \operatorname{sign}(x)\lambda, & \text{if } |x| \geq \lambda \\ 0 & \text{otherwise}\,. \end{cases}$$

Further, for any $\boldsymbol{\alpha}$, let us define $\boldsymbol{s}(\boldsymbol{\alpha})$ as

$$s(\boldsymbol{\alpha})_i := \begin{cases} S_\lambda(\nabla_i f(\boldsymbol{\alpha})), & \text{if } \alpha_i = 0 \\ \nabla_i f(\boldsymbol{\alpha}) + \operatorname{sign}(\alpha_i)\lambda & \text{otherwise}\,. \end{cases} \tag{8}$$

Lemma 1. *For any $\boldsymbol{\alpha}$, the `GS-s` rule is equivalent to*

$$\max_i \left[\min_{s\in\partial g_i} |\nabla_i f(\boldsymbol{\alpha}) + \boldsymbol{s}| \right] \equiv \max_i |s(\boldsymbol{\alpha})_i|\,. \tag{9}$$

Our analysis of `GS-s` rule requires bounding the number of '`bad`' steps (to be detailed in Section 4). For this, we will slightly modify the update of the coordinate descent method. Note that we still always follow the `GS-s` direction, but will sometimes not perform the standard proximal coordinate update along this direction. To update the i_t-th coordinate, we either rely on the standard proximal step on the coordinate,

$$\alpha_{i_t}^+ := S_{\frac{\lambda}{L}}\left(\alpha_{i_t}^{(t)} - \frac{1}{L}\nabla_{i_t} f(\boldsymbol{\alpha}^{(t)}) \right). \tag{10}$$

or we perform line-search

$$\alpha_{i_t}^+ := \arg\min_\gamma F\big(\boldsymbol{\alpha}^{(t)} + (\gamma - \alpha_{i_t}^{(t)})\mathbf{e}_{i_t}\big) \tag{11}$$

Finally, the i_t-th coordinate is updated as

$$\alpha_i^{(t+1)} := \begin{cases} \alpha_i^+, & \text{if } \alpha_i^+\alpha_i^{(t)} \geq 0 \\ 0, & \text{otherwise} \end{cases}. \tag{12}$$

Our modification or 'post-processing' step (12) ensures that the coordinate α_i can never 'cross' the origin. This small change will later on help us bound the precise number of steps needed in our convergence rates (Sec. 4). The details are summarized in Algorithm 1.

Algorithm 1 $L1$ Greedy Coordinate Descent

1: **Initialize:** $\boldsymbol{\alpha}_0 := \mathbf{0} \in \mathbb{R}^n$.
2: **for** $t = 0, 1, \ldots,$ until convergence **do**
3: Select coordinate i_t as in `GS-s`, `GS-r`, or `GS-q`.
4: Find $\alpha_{i_t}^+$ via gradient (10) or line-search (11).
5: Compute $\alpha_{i_t}^{(t+1)}$ as in (12).
6: **end for**

3.2 GCD for Box-Constrained Problems

Using similar ideas, we can also derive the greedy coordinate update for problems with box constraints, such as for the dual SVM. The detailed approach is provided in Appendix B.

4 Convergence Rates

In this section, we present our main convergence results. We illustrate the novelty of our results in the important $L1$-regularized case: For strongly convex functions f, we provide the first linear rates of convergence independent of n for greedy coordinate methods over $L1$-regularized problems, matching the rates in the smooth case. In particular, for `GS-s` this was conjectured to be impossible (Nutini et al., 2015, Section

H.5, H.6) (see Remark 4). We also show the sublinear convergence of the three rules in the non-strongly convex setting. Similar rates also hold for box-constrained problems.

4.1 Linear convergence for strongly convex f

Theorem 1. *Consider an $L1$-regularized optimization problem (7), with f being coordinate-wise L smooth, and μ_1 strongly convex with respect to the $L1$ norm. After t steps of Algorithm 1 where the coordinate i_t is chosen using either the* GS-s *,* GS-r*, or* GS-q *rule,*

$$F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \leq \left(1 - \frac{\mu_1}{L}\right)^{\lceil t/2 \rceil} \left(F(\boldsymbol{\alpha}^{(0)}) - F(\boldsymbol{\alpha}^\star)\right).$$

Remark 2. *The linear convergence rate of Theorem 1 also holds for the Θ-approximate* GS-s *rule as in Definition 1. In this case the μ_1 will be multiplied by Θ^2.*

Remark 3. *All our linear convergence rates can be extended to objective functions which only satisfy the weaker condition of proximal-PL strong convexity (Karimi et al., 2016).*

Remark 4. *The standard analysis (e.g. in Nesterov (2012)) of UCD gives a convergence rate of*

$$\mathbb{E}\left[F(\boldsymbol{\alpha}^{(t)})\right] - F(\boldsymbol{\alpha}^\star) \leq \left(1 - \frac{\mu_2}{nL}\right)^{t} \left(F(\boldsymbol{\alpha}^{(0)}) - F(\boldsymbol{\alpha}^\star)\right).$$

Here μ_2 is the strong convexity constant with respect to the $L2$ norm, which satisfies $\mu_1 \in [\mu_2/n, \mu_2]$. The left boundary $\mu_1 = \mu_2/n$ marks the worst-case for GCD, resulting in convergence slower than UCD. It is shown in Nutini et al. (2015) that this occurs only in extreme examples (e.g. when f consists of n identical separable functions). For all other situations when $\mu_1 \geq 2\mu_2/n$, our result shows that GCD is faster.

Remark 5. *Our analysis in terms of μ_1 works for all three selection rules* GS-s*,* GS-r*, or* GS-q *rules. In (Nutini et al., 2015, Section H5, H6) it was conjectured (but not proven) that this linear convergence rate holds for* GS-q*, but that it cannot hold for* GS-s *or* GS-r*. Example functions were constructed where it was shown that the single step progress of* GS-s *or* GS-r *is much smaller than $1-\mu_2/(nL)$. However these example steps were all* **bad** *steps, as we will define in the following proof sketch, whose number we show can be bounded.*

We state an analogous linear rate for the box-constrained case too, but refer to Appendix B for the detailed algorithm and proof.

Theorem 2. *Suppose that f is coordinate-wise L smooth, and μ_1 strongly convex with respect to the $L1$ norm, for problem (1) with g encoding a box-constraint. After t steps of Algorithm 2 (the box analogon of Algorithm 1) where the coordinate i_t is chosen using the* GS-s *,* GS-r*, or* GS-q *rule, then*

$$f(\boldsymbol{\alpha}^{(t)}) - f(\boldsymbol{\alpha}^\star)$$

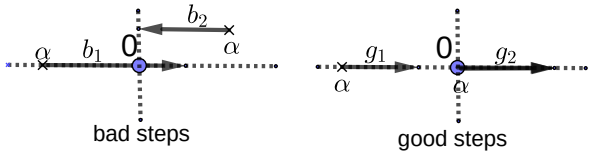

Figure 1: The arrows represent proximal coordinate updates $S_{\frac{\lambda}{L}}(\alpha_i - \frac{1}{L}\nabla_i f(\boldsymbol{\alpha}))$ from different starting points $\boldsymbol{\alpha}$. Updates which 'cross' (b_1) or 'end at' (b_2) the origin are **bad**, whereas the rest (g_1, g_2) are **good**.

$$\leq \left(1 - \max\left(\frac{1}{2n}, \frac{\mu_1}{L}\right)\right)^{t/2} \left(f(\boldsymbol{\alpha}^{(0)}) - f(\boldsymbol{\alpha}^\star)\right).$$

While the proof shares ideas with the $L1$-case, there are significant differences, e.g. the division of the steps into three categories: i) **good** steps which give a $(1-\mu_1/L)$ progress, ii) **bad** steps which may not give much progress but are bounded in number, and a third iii) **cross** steps which give a $(1-1/n)$ progress.

Remark 6. *For the box case, the greedy methods converge faster than UCD if $\mu_1 \geq 2\mu_2/n$, as before, and if $\mu_2/L \leq 1/4$. Typically, μ_2/L is much smaller than 1 and so the second condition is almost always satisfied. Hence we can expect greedy to be much faster in the box case, just as in the unconstrained smooth case. It remains unclear if the $1/n$ term truly affects the rate of convergence. For example, in the separated quadratic case considered in (Nutini et al., 2015, Sec. 4.1), $\mu_1/L \leq 1/n$ and so we can ignore the $1/n$ term in the rate (see Remark 16 in the Appendix).*

Proof sketch. While the full proofs are given in the appendix, we here give a sketch of the convergence of Algorithm 1 for $L1$-regularized problems in the strongly convex case, as in Theorem 1. The key idea is to partition the iterates into two sets: **good** and **bad** steps depending on whether they make (provably) sufficient progress. Then we show that the modification to the update we made in (12) ensures that we do not have too many **bad** steps. Since Algorithm 1 is a descent method, we can focus only on the **good** steps and describe its convergence. The "contradiction" to the convergence of GS-s provided in (Nutini et al., 2015, Section H.5, H.6) are in fact instances of **bad** steps.

The definitions of **good** and **bad** steps are explained in Fig. 1 (and formally in Def. 11). The core technical lemma below shows that in a **good** step, the update along the GS-s direction has an alternative characterization. For the sake of simplicity, let us assume that $\Theta = 1$ and that we use the exact GS-s coordinate.

Lemma 2. *Suppose that iteration t of Algorithm 1 updates coordinate i and that it was a* **good** *step. Then*

$$F(\boldsymbol{\alpha}^{(t+1)}) - F(\boldsymbol{\alpha}^{(t)}) \leq \min_{\boldsymbol{w} \in \mathbb{R}^n} \left\{ \left\langle \nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\rangle \right.$$

$$+ \frac{L}{2}\|\boldsymbol{w}-\boldsymbol{\alpha}^{(t)}\|_1^2 + \lambda(\|\boldsymbol{w}\|_1 - \|\boldsymbol{\alpha}^{(t)}\|_1)\Big\}.$$

Proof sketch. We will only examine the case when $\boldsymbol{\alpha} > 0$ here for the sake of simplicity. Combining this with the assumption that iteration t was a `good` step gives that both $\alpha_i > 0$, $\alpha_i^+ > 0$, and $\alpha_i^+ = \alpha_i - \frac{1}{L}(\nabla_j f(\boldsymbol{\alpha}) + \lambda)$. Further if $\boldsymbol{\alpha} > 0$, the `GS-s` rule simplifies to $\arg\max_{j\in[n]} |\nabla_j f(\boldsymbol{\alpha}) + \lambda|$.

Since f is coordinate-wise smooth (2),

$$\begin{aligned}
&F(\boldsymbol{\alpha}^+) - F(\boldsymbol{\alpha}) \leq \\
&\quad (\nabla_j f(\boldsymbol{\alpha}))(\alpha_i^+ - \alpha_i) + \frac{L}{2}(\alpha_i^+ - \alpha_i)^2 + \lambda(|\alpha_i^+| - |\alpha_i|) \\
&\quad = -\frac{1}{L}(\nabla_j f(\boldsymbol{\alpha}))(\nabla_j f(\boldsymbol{\alpha}) + \lambda) + \\
&\qquad \frac{L}{2L^2}(\nabla_j f(\boldsymbol{\alpha}) + \lambda)^2 - \frac{\lambda}{L}(\nabla_j f(\boldsymbol{\alpha}) + \lambda) \\
&\quad = -\frac{1}{2L}(\nabla_j f(\boldsymbol{\alpha}) + \lambda)^2 .
\end{aligned}$$

But the `GS-s` rule exactly maximizes the last quantity. Thus we can continue:

$$\begin{aligned}
F(\boldsymbol{\alpha}^+) - F(\boldsymbol{\alpha}) &\leq -\frac{1}{2L}\|\nabla f(\boldsymbol{\alpha}) + \lambda \mathbf{1}\|_\infty^2 \\
&= \min_{\boldsymbol{w}\in\mathbb{R}^n} \left\{ \langle \nabla f(\boldsymbol{\alpha}) + \lambda\mathbf{1}, \boldsymbol{w}-\boldsymbol{\alpha}\rangle + \frac{L}{2}\|\boldsymbol{w}-\boldsymbol{\alpha}\|_1^2 \right\} \\
&= \min_{\boldsymbol{w}\in\mathbb{R}^n} \Big\{ \langle \nabla f(\boldsymbol{\alpha}), \boldsymbol{w}-\boldsymbol{\alpha}\rangle + \frac{L}{2}\|\boldsymbol{w}-\boldsymbol{\alpha}\|_1^2 \\
&\qquad + \lambda(\langle \mathbf{1}, \boldsymbol{w}\rangle - \langle \mathbf{1}, \boldsymbol{\alpha}\rangle) \Big\} .
\end{aligned}$$

Recall that $\boldsymbol{\alpha} > 0$ and so $\|\boldsymbol{\alpha}\|_1 = \langle \boldsymbol{\alpha}, \mathbf{1}\rangle$. Further for any $x \in \mathbb{R}$, $|x| \geq x$ and so $\langle \mathbf{1}, \boldsymbol{w}\rangle \leq \|\boldsymbol{w}\|_1$. This means

$$\lambda(\langle \mathbf{1}, \boldsymbol{w}\rangle - \langle \mathbf{1}, \boldsymbol{\alpha}\rangle) \leq \lambda(\|\boldsymbol{w}\|_1 - \|\boldsymbol{\alpha}\|_1) .$$

Plugging this into our previous equation gives us the lemma. See Lemma 8 for the full proof. □

If $\lambda = 0$ (i.e. F is smooth), Lemma 2 reduces to the 'refined analysis' of Nutini et al. (2015). We can now state the rate obtained in the strongly convex case.

Proof sketch for Theorem 1. Notice that if $\alpha_{i_t} = 0$, the step is necessarily `good` by definition (see Fig. 1). Since we start at the origin $\mathbf{0}$, the first time each coordinate is picked is a `good` step. Further, if some step t is `bad`, this implies that $\alpha_{i_t}^+$ 'crosses' the origin. In this case our modified update rule (12) sets the coordinate α_{i_t} to 0. The next time coordinate i_t is picked, the step is sure to be `good`. Thus in t steps, we have at least $\lceil t/2 \rceil$ `good` steps.

As per Lemma 2, every `good` step corresponds to optimizing the upper bound with the $L1$-squared regularizer. We can finish the proof:

$$\begin{aligned}
F(\boldsymbol{\alpha}^{(t+1)}) - F(\boldsymbol{\alpha}^{(t)}) &\leq \min_{\boldsymbol{w}\in\mathbb{R}^n} \Big\{ \Big\langle \nabla f(\boldsymbol{\alpha}^{(t)}, \boldsymbol{w}-\boldsymbol{\alpha}^{(t)} \Big\rangle \\
&\quad + \frac{L}{2}\|\boldsymbol{w}-\boldsymbol{\alpha}^{(t)}\|_1^2 + \lambda(\|\boldsymbol{w}\|_1 - \Big\|\boldsymbol{\alpha}^{(t)}\Big\|_1) \Big\} \\
&\overset{(a)}{\leq} \frac{\mu_1}{L} \min_{\boldsymbol{w}\in\mathbb{R}^n} \Big\{ \Big\langle \nabla f(\boldsymbol{\alpha}^{(t)}, \boldsymbol{w}-\boldsymbol{\alpha}^{(t)} \Big\rangle \\
&\quad + \frac{\mu_1}{2}\|\boldsymbol{w}-\boldsymbol{\alpha}^{(t)}\|_1^2 + \lambda(\|\boldsymbol{w}\|_1 - \Big\|\boldsymbol{\alpha}^{(t)}\Big\|_1) \Big\} \\
&\overset{(b)}{\leq} \frac{\mu_1}{L}\Big(F(\boldsymbol{\alpha}^\star) - F(\boldsymbol{\alpha}^{(t)})\Big) .
\end{aligned}$$

Inequality (a) follows from Karimireddy et al. (2018, Lemma 9), and (b) from strong convexity of f. Rearranging the terms above gives us the required linear rate of convergence. □

4.2 Sublinear convergence for general f

A sublinear convergence rate independent of n for GCD can be obtained when f is not strongly convex.

Theorem 3. *Suppose that F is coordinate-wise L smooth and convex, for g being an $L1$-regularizer or a box-constraint. Also let $\mathcal{Q}^\star$ be the set of minima of F with a minimum value $F^\star$. After t steps of Algorithm 1 or Algorithm 2 respectively, where the coordinate i_t is chosen using the `GS-s`, `GS-r`, or `GS-q` rule,*

$$F(\boldsymbol{\alpha}^{(t)}) - F^\star \leq \mathcal{O}\Big(\frac{LD^2}{t}\Big),$$

where D is the $L1$-diameter of the level set. For the set of minima $\mathcal{Q}^\star$,

$$D = \max_{\boldsymbol{w}\in\mathbb{R}^n} \min_{\boldsymbol{\alpha}^\star\in\mathcal{Q}^\star} \left\{ \|\boldsymbol{w}-\boldsymbol{\alpha}^\star\|_1 \,\middle|\, F(\boldsymbol{w}) \leq F(\boldsymbol{\alpha}^{(0)}) \right\}.$$

While a similar convergence rate was known for the `GS-r` rule (Dhillon et al., 2011), we here establish it for all three rules—even for approximate `GS-s`.

5 Maximum Inner Product Search

We now shift the focus from the theoretical rates to the actual implementation. A very important observation—as pointed out by Dhillon et al. (2011)—is that finding the steepest descent direction is closely related to a geometric problem. As an example consider the function $f(\boldsymbol{\alpha}) \overset{\text{def}}{=} \frac{1}{2}\|A\boldsymbol{\alpha} - \boldsymbol{b}\|^2$ for a data matrix $A \in \mathbb{R}^{d\times n}$. The gradient takes the form $\nabla f(\boldsymbol{\alpha}) = A^\top \boldsymbol{q}$ for $\boldsymbol{q} = (A\boldsymbol{\alpha} - \boldsymbol{b})$ and thus finding steepest coordinate direction is equal to finding the datapoint with the largest (in absolute value) inner product $\langle \boldsymbol{A}_i, \boldsymbol{q}\rangle$ with the query vector $\boldsymbol{q}$, which a priori requires the evaluation of all n scalar products. However, when we have to perform multiple similar queries (such as over the iterations of GCD), it is possible to pre-process the dataset A to speed up the query time. Note that we do not require the columns $\boldsymbol{A}_i$ to be normalized.

For the more general set of problems we consider here, we need the following slightly stronger primitive.

Definition 7 (S-MIPS)**.** *Given a set of m, d-dimensional points $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m \in \mathbb{R}^d$, the* Subset Maxi-

mum Inner Product Search *or* S-MIPS *problem is to pre-process the set* $\mathcal{P}$ *such that for any query vector* $\boldsymbol{q}$ *and any subset of the points* $\mathcal{B} \subseteq [m]$*, the best point* $j \in \mathcal{B}$*,*

$$\text{S-MIPS}_{\mathcal{P}}(\boldsymbol{q}; \mathcal{B}) := \arg\max_{j \in \mathcal{B}} \{\langle \boldsymbol{p}_j, \boldsymbol{q} \rangle\} ,$$

can be computed with $o(m)$ *scalar product evaluations.*

State-of-the-art algorithms relax the exactness assumption and compute an approximate solution in time equivalent to a *sublinear* number of scalar product evaluations, i.e. $o(n)$ (e.g. (Charikar, 2002; Lv et al., 2007; Shrivastava and Li, 2014; Neyshabur and Srebro, 2015; Andoni et al., 2015)). We consciously refrain from stating more precise running times, as these will depend on the actual choice of the algorithm and the parameters chosen by the user. Our approach in this paper is transparent to the actual choice of S-MIPS algorithm, we only show how GCD steps can be *exactly cast* as such instances. By employing an arbitrary solver one thus gets a sublinear time approximate GCD update. An important caveat is that in subsequent queries, we will *adaptively* change the subset $\mathcal{B}$ based on the solution to the previous query. Hence the known theoretical guarantees shown for LSH do not directly apply, though the practical performance does not seem to be affected by this (see Appendix Fig. 13, 17). Practical details of efficiently solving S-MIPS are provided in Section 7.

6 Mapping GS-s to MIPS

We now move to our next contribution and show how the GS-s rule can be *efficiently* implemented. We aim to cast the problem of computing the GS-s update as an instance of MIPS (Maximum Inner Product Search), for which very fast query algorithms exist. In contrast, the GS-r and GS-q rules do not allow such a mapping. In this section, we will only consider objective functions of the following special structure:

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left\{ F(\boldsymbol{\alpha}) \overset{\text{def}}{=} \underbrace{l(A\boldsymbol{\alpha}) + \boldsymbol{c}^\top \boldsymbol{\alpha}}_{f(\boldsymbol{\alpha})} + \sum_{i=1}^{n} g_i(\alpha_i) \right\} . \quad (13)$$

The usual problems such as Lasso, dual SVM, or logistic regression, etc. have such a structure (see Appendix A.3).

Difficulty of the Greedy Rules. This section will serve to strongly motivate our choice of using the GS-s rule over the GS-r or GS-q. Let us pause to examine the three greedy selection rules and compare their relative difficulty. As a warm-up, consider again the smooth function $f(\boldsymbol{\alpha}) \overset{\text{def}}{=} \frac{1}{2} \|A\boldsymbol{\alpha} - \boldsymbol{b}\|^2$ for a data matrix $A \in \mathbb{R}^{d \times n}$ as introduced above in Section 5. We have observed that the steepest coordinate direction is

$$\arg\max_{j \in [n]} |\nabla_j f(\boldsymbol{\alpha})| \equiv \arg\max_{j \in [n]} \max_{s \in \{-1,1\}} \langle s\boldsymbol{A}_j, \boldsymbol{v} \rangle . \quad (14)$$

The formulation on the right is an instance of MIPS over the $2n$ vectors $\pm \boldsymbol{A}_i$. Now consider a non-smooth problem of the form $F(\boldsymbol{\alpha}) \overset{\text{def}}{=} \frac{1}{2} \|A\boldsymbol{\alpha} - \boldsymbol{b}\|^2 + \lambda \|\boldsymbol{\alpha}\|_1$. For simplicity, let us assume $\boldsymbol{\alpha} > 0$ and $L = 1$. In this case, the subgradient is $A^\top \boldsymbol{v} + \lambda \boldsymbol{1}$ and the GS-s rule is

$$\arg\max_{j \in [n]} \min_{s \in \delta |\alpha_j|} |\nabla_i f(\boldsymbol{\alpha}) + s| \equiv \arg\max_{j \in [n]} \max_{s \in \{-1,1\}} \langle s\boldsymbol{A}_j, \boldsymbol{v} \rangle + s\lambda . \quad (15)$$

The rule (15) is clearly not much harder than (14), and can be cast as a MIPS problem with minor modifications (see details in Sec. 6.1).

Let α_j^+ denote the proximal coordinate update along the j-th coordinate. In our case, $\alpha_j^+ = S_\lambda(\alpha_j - \langle \boldsymbol{A}_j, \boldsymbol{v} \rangle)$. The GS-r rule can now be 'simplified' as:

$$\arg\max_{j \in [n]} \begin{Bmatrix} \alpha_j, & \text{if } |\alpha_j - \langle \boldsymbol{A}_j, \boldsymbol{v} \rangle| \leq \lambda \\ \text{sign}(\alpha_j - \langle \boldsymbol{A}_j, \boldsymbol{v} \rangle), & \text{otherwise} . \end{Bmatrix} \quad (16)$$

It does not seem easy to cast (16) as a MIPS instance. It is even less likely that the GS-q rule which reads

$$\arg\min_{j \in [n]} \left\{ \nabla_j f(\boldsymbol{\alpha})(\alpha_j^+ - \alpha_j) + \tfrac{1}{2}(\alpha_j^+ - \alpha_j)^2 + \lambda(|\alpha_j^+| - \alpha_j) \right\}$$

can be mapped as to MIPS. This highlights the simplicity and usefulness of the GS-s rule.

6.1 Mapping $L1$-Regularized Problems

Here we focus on problems of the form (13) where $g(\boldsymbol{\alpha}) = \lambda \|\boldsymbol{\alpha}\|_1$. Again, we have $\nabla f(\boldsymbol{\alpha}) = A^\top \nabla l(\boldsymbol{v}) + \boldsymbol{c}$ where $\boldsymbol{v} = A\boldsymbol{\alpha}$.

For simplicity, let $\boldsymbol{\alpha} \neq 0$. Then the GS-s rule in (9) is

$$\arg\max_{j \in [n]} |s(\boldsymbol{\alpha})_j| = \arg\max_{j \in [n]} |\langle \boldsymbol{A}_j, \nabla l(\boldsymbol{v}) \rangle + c_j + \text{sign}(\alpha_j)\lambda| = \arg\max_{j \in [n]} \max_{s \in \pm 1} s \left[\langle \boldsymbol{A}_j, \nabla l(\boldsymbol{v}) \rangle + c_j + \text{sign}(\alpha_j)\lambda \right] . \quad (17)$$

We want to map the problem of the above form to a S-MIPS instance. Define for some $\beta > 0$, vectors

$$\tilde{\boldsymbol{A}}_j^{\pm} := \begin{pmatrix} \pm\beta, & \beta c_j, & \boldsymbol{A}_j \end{pmatrix}^\top , \quad (18)$$

and form a query vector $\boldsymbol{q}$ as

$$\boldsymbol{q} := \begin{pmatrix} \frac{\lambda}{\beta}, & \frac{1}{\beta}, & \nabla l(\boldsymbol{v}) \end{pmatrix}^\top . \quad (19)$$

A simple computation shows that the problem in (17) is equivalent to

$$\arg\max_{j \in [n]} \max_{s \in \pm 1} \left\langle s\tilde{\boldsymbol{A}}_j^{\text{sign}(\alpha_j)}, \boldsymbol{q} \right\rangle .$$

Thus by searching over a subset of vectors in $\{\pm \tilde{\boldsymbol{A}}_j^{\pm}\}$,

we can compute the GS-s direction. Dealing with the case where $\alpha_j = 0$ goes through similar arguments, and the details are outlined in Appendix E. Here we only state the resulting mapping.

The constant β in (18) and (19) is chosen to ensure that the entry is of the same order of magnitude on average as the rest of the coordinates of $\boldsymbol{A}_i$. The need for β only arises out of the performance concerns about the underlying algorithm to solve the S-MIPS instance. For example, β has no effect if we use exact search.

Formally, define the set $\mathcal{P} := \{\pm\tilde{\boldsymbol{A}}_j^{\pm} : j \in [n]\}$. Then at any iteration t with current iterate $\boldsymbol{\alpha}^{(t)}$, we also define $\mathcal{B}_t$ as $\mathcal{B}_t = \mathcal{B}_t^1 \cup \mathcal{B}_t^2 \cup \mathcal{B}_t^3 \cup \mathcal{B}_t^4$, where

$$\mathcal{B}_t^1 = \left\{\tilde{\boldsymbol{A}}_j^+ : \alpha_j^{(t)} > 0\right\}, \qquad \mathcal{B}_t^2 = \left\{-\tilde{\boldsymbol{A}}_j^+ : \alpha_j^{(t)} \geq 0\right\},$$
$$\mathcal{B}_t^3 = \left\{\tilde{\boldsymbol{A}}_j^- : \alpha_j^{(t)} \leq 0\right\}, \qquad \mathcal{B}_t^4 = \left\{-\tilde{\boldsymbol{A}}_j^- : \alpha_j^{(t)} < 0\right\}. \tag{20}$$

Lemma 3. *At any iteration t, for $\mathcal{P}$ and $\mathcal{B}_t$ as defined in (20), the query vector $\boldsymbol{q}_t$ as in (19), and $s(\boldsymbol{\alpha})$ as in (9) then the following are equivalent for $f(\boldsymbol{\alpha})$ is of the form $l(A\boldsymbol{\alpha}) + \boldsymbol{c}^\top\boldsymbol{\alpha}$:*

$$\text{S-MIPS}_{\mathcal{P}}(\boldsymbol{q}_t; \mathcal{B}_t) = \arg\max_i |s(\boldsymbol{\alpha})_i| \ .$$

The sets $\mathcal{B}_t$ and $\mathcal{B}_{t+1}$ differ in at most four points since $\boldsymbol{\alpha}^{(t)}$ and $\boldsymbol{\alpha}^{(t+1)}$ differ only in a single coordinate. This makes it computationally very efficient to incrementally maintain $\mathcal{B}_{t+1}$ and $\boldsymbol{\alpha}^{(t+1)}$ for $L1$-regularized problems.

6.2 Mapping Box-Constrained Problems

Using similar ideas, we demonstrate how to efficiently map problems of the form (13) where g enforces box constraints, such as for the dual SVM. The detailed approach is provided in Appendix B.1.

7 Experimental Results

Our experiments focus on the standard tasks of Lasso regression, as well as SVM training (on the dual objective). We refer the reader to Appendix A.3 for definitions. Lasso regression is performed on the rcv1 dataset while SVM is performed on w1a and the ijcnn1 datasets. All columns of the dataset (features for Lasso, datapoints for SVM) are normalized to unit length, allowing us to use the standard cosine-similarity algorithms nmslib (Boytsov and Naidan, 2013) to efficiently solve the S-MIPS instances. Note however that our framework is applicable without any normalization, if using a general MIPS solver instead.

We use the hnsw algorithm of the nmslib library with the default hyper-parameter value M and other parameters as in Table 1, selected by grid-search.[2] More details such as the meaning of these parameters can be found in the nmslib manual (Naidan and Boytsov, 2015, pp. 61). We exclude the time required for preprocessing of the datasets since it is amortized over the multiple experiments run on the same dataset (say for hyper-parameter tuning etc.). All our experiments are run on an Intel Xeon CPU E5-2680 v3 (2.50GHz, 30 MB cache) with 48 cores and 256GB RAM.

Table 1: Datasets and hyper-parameters: Lasso is run on rcv1, and SVM on w1a and ijcnn1. (**d**, **n**) is dataset size, the constant β from (18), (19) is set to $50/\sqrt{n}$, nmslib hyper-parameter M is set as a default, $efC = 100$. ρ is the density of the optimal solution.

Dataset	n	d	ρ	efS	post
rcv1, $\lambda = 1$	47,236	15,564	19%	100	2
rcv1, $\lambda = 10$	47,236	15,564	3%	400	2
w1a	2,477	300		100	0
ijcnn1	49,990	22		50	0

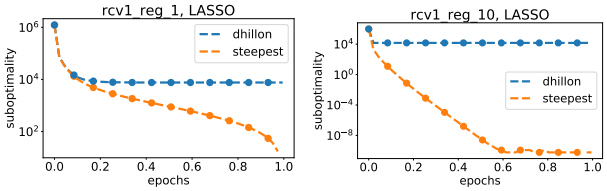

Figure 2: Evaluating dhillon: steepest which is based on the GS-s rule outperforms dhillon which quickly stagnates. Increasing the regularization, it stagnates in even fewer iterations.

First we compare the practical algorithm (dhillon) of Dhillon et al. (2011), which disregards the regularization part in choosing the next coordinate, and Algorithm 1 with GS-s rule (steepest) for Lasso regression. Note that dhillon is not guaranteed to converge. To compare the selection rules without biasing on the choice of the library, we perform exact search to answer the MIPS queries. As seen from Fig. 2, steepest significantly outperforms dhillon. In fact dhillon stagnates (though it does not diverge), once the error $f(\boldsymbol{\alpha})$ becomes small and the $L1$ regularization term starts playing a significant role. Increasing the regularization λ further worsens its performance. This is understandable since the rule used by dhillon ignores the $L1$ regularizer.

Next we compare our steepest strategy (Algorithms 1 and 2 using the GS-s rule), and the corresponding nearest-neighbor-based approximate versions (steepest-nn) against uniform, which picks coordinates uniformly at random. In all these experiments,

[2] A short overview of how to set these hyper-parameters can be found at https://github.com/nmslib/nmslib/blob/master/python_bindings/parameters.md.

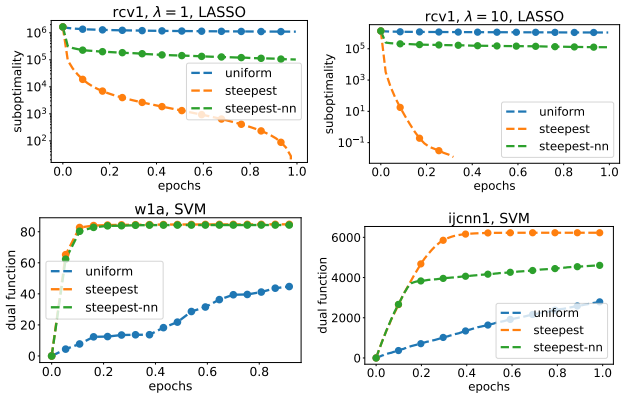

Figure 3: `steepest` as well `steepest-nn` significantly outperform `uniform` in number of iterations.

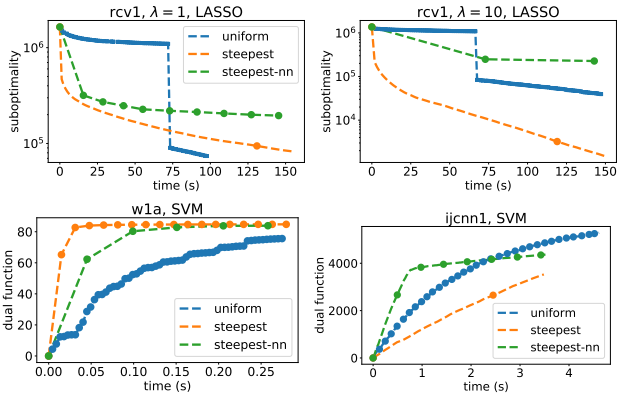

Figure 4: `steepest-nn` is very competitive and sometimes outperforms `uniform` even in terms of wall time especially towards the beginning. However eventually the performance of `uniform` is better than `steepest-nn`. This is because `nmslib` performs poorly as the norm of the gradient becomes small.

$\lambda \in \{1, 10\}$ for Lasso and at $1/n$ for SVM. Fig. 3 shows the clearly superior performance in terms of iterations of the `steepest` strategy as well as `steepest-nn` over `uniform` for both the Lasso as well as SVM problems. However, towards the end of the optimization i.e. in high accuracy regimes, `steepest-nn` fails to find directions substantially better than `uniform`. This is because towards the end, all components of the gradient $\nabla f(\boldsymbol{\alpha})$ become small, meaning that the query vector is nearly orthogonal to all points—a setting in which the employed nearest neighbor library `nmslib` performs poorly (Boytsov and Naidan, 2013).

Fig. 4 compares the wall-time performance of the `steepest`, `steepest-nn` and `uniform` strategies. This includes all the overhead of finding the descent direction. In all instances, the `steepest-nn` algorithm is competitive with `uniform` at the start, compensating for the increased time per iteration by increased progress per iteration. However towards the end `steepest-nn` gets comparable progress per iteration at a significantly larger cost, making its performance worse. With increasing sparsity of the solution (see Table 1 for sparsity levels), exact `steepest` rule starts to outperform `uniform` and `steepest-nn`.

Wall-time experiments (Fig. 4) show that `steepest-nn` always shows a significant performance gain in the important early phase of optimization, but in the later phase loses out to `uniform` due to the query cost and poor performance of `nmslib`. In practice, the recommended implementation is to use `steepest-nn` algorithm in the early optimization regime, and switch to uniform once the iteration cost outweighs the gain. In the Appendix (Fig. 13) we further investigate the poor quality of the solution provided by `nmslib`.

Repeating our experiments with other datasets, or using `FALCONN` (Andoni et al., 2015), another popular library for MIPS, yielded comparable results, provided in Appendix G.

8 Concluding Remarks

In this work we advocate the use of approximate `GS-s` selection rule for coordinate descent, and show its convergence for several important classes of problems for the first time, furthering our understanding of steepest descent on non-smooth problems. Our results demonstrate that significant speedups can be achieved by using the subgradient (4) to efficiently select the most 'important' coordinate at each iteration. Prior to our work, the analysis of greedy coordinate methods for composite functions mostly reverted to that of UCD or made unreasonable assumptions (e.g. Nutini et al. (2015); Zhang and Xiao (2017); Lei et al. (2017)). This is in stark contrast with smooth functions where many algorithms provably have rates better than UCD (e.g. Nutini et al. (2015, 2017); Nesterov and Stich (2017)). We believe our proof techniques can help bridge this gap between the analysis of greedy methods on smooth and non-smooth functions.

Our extensive numerical experiments also showcase the strengths and weaknesses of current state-of-the-art libraries for computing a Θ-approximate `GS-s` direction. As n grows, the cost per iteration for `nmslib` remains comparable to that of UCD, while the progress made per iteration increases. This means that as problem sizes grow, `GS-s` implemented via S-MIPS becomes an increasingly attractive approach. However, when the norm of the gradient becomes small, current state-of-the-art methods struggle to find directions substantially better than uniform. Alleviating this, and leveraging some of the very active development of recent alternatives to LSH as subroutines for our method is a promising direction for future work.

Acknowledgements. We thank Ludwig Schmidt for numerous useful discussions on LSH and using `FALCONN`. We also thank Vyacheslav Alipov for help with `nmslib`, Hadi Daneshmand for algorithmic insights, Mikkel Thorup for discussions on using hashing schemes in practice, and Mathurin Massias for feedback on our writeup. The feedback from many anonymous reviewers has also helped significantly improve the presentation.

References

Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., and Schmidt, L. (2015). Practical and Optimal LSH for Angular Distance. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems 28*, pages 1225–1233. Curran Associates, Inc.

Bertsekas, D. P. and Tsitsiklis, J. N. (1989). *Parallel and distributed computation: numerical methods*, volume 23. Prentice hall Englewood Cliffs, NJ.

Bertsekas, D. P. and Tsitsiklis, J. N. (1991). Some aspects of parallel and distributed iterative algorithms—a survey. *Automatica*, 27(1):3–21.

Bickley, W. (1941). Relaxation methods in engineering science: A treatise on approximate computation.

Boytsov, L. and Naidan, B. (2013). Engineering efficient and effective Non-Metric Space Library. In Brisaboa, N., Pedreira, O., and Zezula, P., editors, *Similarity Search and Applications*, volume 8199 of *Lecture Notes in Computer Science*, pages 280–293. Springer Berlin Heidelberg.

Charikar, M. S. (2002). Similarity Estimation Techniques from Rounding Algorithms. In *Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pages 380–388, New York, NY, USA. ACM.

Dhillon, I. S., Ravikumar, P. K., and Tewari, A. (2011). Nearest Neighbor based Greedy Coordinate Descent. In Shawe-Taylor, J., Zemel, R. S., Bartlett, P. L., Pereira, F., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 24*, pages 2160–2168. Curran Associates, Inc.

Dünner, C., Parnell, T., and Jaggi, M. (2017). Efficient use of limited-memory accelerators for linear learning on heterogeneous systems. In *Advances in Neural Information Processing Systems*, pages 4258–4267.

Fercoq, O. and Richtárik, P. (2015). Accelerated, Parallel, and Proximal Coordinate Descent. *SIAM Journal on Optimization*, 25(4):1997–2023.

Karimi, H., Nutini, J., and Schmidt, M. (2016). Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer.

Karimireddy, S. P. R., Stich, S., and Jaggi, M. (2018). Adaptive balancing of gradient and update computation times using global geometry and approximate subproblems. In *International Conference on Artificial Intelligence and Statistics*, pages 1204–1213.

Lei, Q., Yen, I. E., Wu, C.-y., Dhillon, I. S., and Ravikumar, P. (2017). Doubly greedy primal-dual coordinate descent for sparse empirical risk minimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2034–2042. JMLR. org.

Lin, Q., Lu, Z., and Xiao, L. (2014). An Accelerated Proximal Coordinate Gradient Method. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 3059–3067. Curran Associates, Inc.

Locatello, F., Raj, A., Karimireddy, S. P., Rätsch, G., Schölkopf, B., Stich, S., and Jaggi, M. (2018). On matching pursuit and coordinate descent. In *International Conference on Machine Learning*, pages 3204–3213.

Lu, H., Freund, R. M., and Mirrokni, V. (2018). Accelerating greedy coordinate descent methods. *arXiv preprint arXiv:1806.02476*.

Lv, Q., Josephson, W., Wang, Z., Charikar, M., and Li, K. (2007). Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961. VLDB Endowment.

Massias, M., Gramfort, A., and Salmon, J. (2018). Celer: a fast solver for the lasso with dual extrapolation. In *International Conference on Machine Learning*, pages 3321–3330.

Naidan, B. and Boytsov, L. (2015). Non-metric space library manual. *arXiv preprint arXiv:1508.05470*.

Ndiaye, E., Fercoq, O., Gramfort, A., and Salmon, J. (2015). Gap safe screening rules for sparse multi-task and multi-class models. In *Advances in Neural Information Processing Systems*, pages 811–819.

Nesterov, Y. (2012). Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems. *SIAM Journal on Optimization*, 22(2):341–362.

Nesterov, Y. and Stich, S. (2017). Efficiency of the Accelerated Coordinate Descent Method on Structured Optimization Problems. *SIAM Journal on Optimization*, 27(1):110–123.

Neyshabur, B. and Srebro, N. (2015). On Symmetric and Asymmetric LSHs for Inner Product Search. In

ICML 2015 - Proceedings of the 32th International Conference on Machine Learning, pages 1926–1934.

Nutini, J., Laradji, I., and Schmidt, M. (2017). Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv preprint arXiv:1712.08859*.

Nutini, J., Schmidt, M., Laradji, I. H., Friedlander, M., and Koepke, H. (2015). Coordinate Descent Converges Faster with the Gauss-Southwell Rule Than Random Selection. *arXiv:1506.00552 [cs, math, stat]*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Perekrestenko, D., Cevher, V., and Jaggi, M. (2017). Faster Coordinate Descent via Adaptive Importance Sampling. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 869–877, Fort Lauderdale, FL, USA. PMLR.

Richtarik, P. and Takac, M. (2016). Parallel coordinate descent methods for big data optimization. *Mathematical Programming*, 156(1-2):433–484.

Shalev-Shwartz, S. and Zhang, T. (2013a). Accelerated Mini-batch Stochastic Dual Coordinate Ascent. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 378–385, USA. Curran Associates Inc.

Shalev-Shwartz, S. and Zhang, T. (2013b). Stochastic Dual Coordinate Ascent Methods for Regularized Loss. *J. Mach. Learn. Res.*, 14(1):567–599.

Shalev-Shwartz, S. and Zhang, T. (2016). Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, 155(1-2):105–145.

Shrivastava, A. and Li, P. (2014). Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS). In *NIPS 2014 - Advances in Neural Information Processing Systems 27*, pages 2321–2329.

Song, C., Cui, S., Jiang, Y., and Xia, S.-T. (2017). Accelerated stochastic greedy coordinate descent by soft thresholding projection onto simplex. In *Advances in Neural Information Processing Systems*, pages 4838–4847.

Stich, S. U., Raj, A., and Jaggi, M. (2017a). Approximate Steepest Coordinate Descent. *ICML - Proceedings of the 34th International Conference on Machine Learning*.

Stich, S. U., Raj, A., and Jaggi, M. (2017b). Safe Adaptive Importance Sampling. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 4384–4394. Curran Associates, Inc.

Warga, J. (1963). Minimizing certain convex functions. *Journal of the Society for Industrial and Applied Mathematics*, 11(3):588–593.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34.

You, Y., Lian, X., Liu, J., Yu, H.-F., Dhillon, I. S., Demmel, J., and Hsieh, C.-J. (2016). Asynchronous parallel greedy coordinate descent. In *Advances in Neural Information Processing Systems*, pages 4682–4690.

Zhang, Y. and Xiao, L. (2017). Stochastic primal-dual coordinate method for regularized empirical risk minimization. *The Journal of Machine Learning Research*, 18(1):2939–2980.

Appendix

A Setup and Notation

In this section we go over some of the definitions and notations which we had skipped over previously. We will also describe the class of functions we tackle and applications which fit into this framework.

A.1 Function Classes

Definition 8 (coordinate-wise L-smoothness)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is* coordinate-wise L-smooth *if*

$$f(\boldsymbol{\alpha} + \gamma \mathbf{e}_i) \leq f(\boldsymbol{\alpha}) + \gamma \, \nabla_i f(\boldsymbol{\alpha}) + \frac{L\gamma^2}{2},$$

for any $\boldsymbol{\alpha} \in \mathbb{R}^n$, $\gamma \in \mathbb{R}$, $i \in [n]$, and $\mathbf{e}_i$ is a coordinate basis vector.

We also define strong convexity of the function f.

Definition 9 (μ-strong convexity)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is* μ-strongly convex *with respect to some norm $\|\cdot\|$ if*

$$f(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) \geq f(\boldsymbol{\alpha}) + \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha} \rangle + \frac{\mu}{2} \|\Delta\boldsymbol{\alpha}\|^2$$

for any $\boldsymbol{\alpha}$ and $\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}$ in the domain of f (note that f does not necessarily need to be defined on the entire space $\mathbb{R}^n$).

We will frequently denote by μ_2 the strong convexity constant corresponding to the usual Euclidean norm, and by μ_1 the strong convexity constant corresponding the $L1$ norm. In general it holds : $\mu_1 \in [\mu_2/n, \mu_2]$. See (Nutini et al., 2015) for a detailed comparision of the two constants.

Theorem 4. *Suppose that the function is twice-differentiable and*

$$L \geq \max_{i \in [n]} [\nabla^2 f(\boldsymbol{\alpha})]_{i,i} \, ,$$

for any $\boldsymbol{\alpha}$ in the domain of f i.e. the maximum diagonal element of the Hessian is bounded by L. Then $f(\boldsymbol{\alpha})$ is coordinate-wise L-smooth. Additionally, for any $\Delta\boldsymbol{\alpha} \in \mathbb{R}^n$

$$f(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) \leq f(\boldsymbol{\alpha}) + \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha} \rangle + \frac{L}{2} \|\Delta\boldsymbol{\alpha}\|_1^2 \, .$$

Proof. By Taylor's expansion and the intermediate value theorem, we have that for any $\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha} \in \mathbb{R}^n$, there exists a $\alpha \in [0, 1]$ such that for $\boldsymbol{v} = \boldsymbol{\alpha} + \alpha\Delta\boldsymbol{\alpha}$,

$$f(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) = f(\boldsymbol{\alpha}) + \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha} \rangle + \frac{1}{2} \Delta\boldsymbol{\alpha}^\top \nabla^2 f(\boldsymbol{v}) \Delta\boldsymbol{\alpha} \, . \tag{21}$$

Now if $\Delta\boldsymbol{\alpha} = \gamma \mathbf{e}_i$ for some $\gamma \geq 0$ and coordinate i, the equation (21) becomes

$$f(\boldsymbol{\alpha} + \gamma \mathbf{e}_i) = f(\boldsymbol{\alpha}) + \gamma \, \nabla_i f(\boldsymbol{\alpha}) + \frac{\gamma^2}{2} [\nabla^2 f(\boldsymbol{v})]_{i,i} \, .$$

The first claim now follows since L was defined such that $L \geq \nabla^2 f(\boldsymbol{v})_{i,i}$. For the second claim, consider the following optimization problem over $\widehat{\Delta\boldsymbol{\alpha}}$,

$$\max_{\|\widehat{\Delta\boldsymbol{\alpha}}\|_1 \leq 1} \left\{ \widehat{\Delta\boldsymbol{\alpha}}^\top \nabla^2 f(\boldsymbol{v}) \widehat{\Delta\boldsymbol{\alpha}} \stackrel{\text{def}}{=} \mathcal{Q}(\widehat{\Delta\boldsymbol{\alpha}}) \right\} . \tag{22}$$

We claim that the maximum is achieved for $\widehat{\Delta\boldsymbol{\alpha}} = \mathbf{e}_i$ for some $i \in [n]$. For now assume this is true. Then we would have that

$$\begin{aligned}
\Delta\boldsymbol{\alpha}^\top \nabla^2 f(\boldsymbol{v}) \Delta\boldsymbol{\alpha} &= \|\Delta\boldsymbol{\alpha}\|_1^2 \, \widehat{\Delta\boldsymbol{\alpha}}^\top \nabla^2 f(\boldsymbol{v}) \widehat{\Delta\boldsymbol{\alpha}} \\
&\leq \|\Delta\boldsymbol{\alpha}\|_1^2 \, \mathbf{e}_i^\top \nabla^2 f(\boldsymbol{v}) \mathbf{e}_i \\
&= \|\Delta\boldsymbol{\alpha}\|_1^2 \, [\nabla^2 f(\boldsymbol{v})]_{i,i} \\
&\leq \|\Delta\boldsymbol{\alpha}\|_1^2 \, L \, .
\end{aligned}$$

Using this result in equation (21) would prove our second claim of the theorem. Thus we need to study the optimum of (22). Since f is a convex function, $\mathcal{Q}$ is also a convex function. We can now appeal to Lemma 4 for the convex set $\|\boldsymbol{x}\|_1 \leq 1$. The corners of the set exactly correspond to the unit directional vectors $\mathbf{e}_i$. With this we finish the proof of our theorem. □

Remark 10. *This result states that if we define smoothness with respect to the $L1$ norm, the resulting smoothness constant is same as the coordinate-wise smoothness constant. This is surprising since for a general convex function g, using the update rule*

$$\boldsymbol{\alpha}^{(t+1)} = \arg\min_{\boldsymbol{s}} \left\langle \nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{s} - \boldsymbol{\alpha}^{(t)} \right\rangle + \frac{L}{2} \left\| \boldsymbol{s} - \boldsymbol{\alpha}^{(t)} \right\|_1^2 + g(\boldsymbol{s}) \,,$$

does not necessarily yield a coordinate update. We believe this observation (though not crucial to the current work) was not known before.

Let us prove an elementary lemma about maximizing convex functions over convex sets.

Lemma 4 (Maximum of a constrained convex function)**.** *For any convex function $\mathcal{Q}(\boldsymbol{x})$, the maximum over a compact convex set B is achieved at a 'corner'. Here a 'corner' is defined to be a point $\boldsymbol{x} \in B$ such that there do not exist two points $\boldsymbol{y} \in B$ and $\boldsymbol{z} \in B$, $\boldsymbol{y} \neq \boldsymbol{x}, \boldsymbol{z} \neq \boldsymbol{x}$ such that for some $\gamma \in [0,1]$, $(1-\gamma)\boldsymbol{y} + \gamma \boldsymbol{z} = \boldsymbol{x}$.*

Proof. Suppose that the maximum is achieved at a point $\boldsymbol{x}$ which is not a 'corner'. Then let $\boldsymbol{y}, \boldsymbol{z} \in B$ be two points such that for $\gamma \in [0,1]$, we have $(1-\gamma)\boldsymbol{y} + \gamma \boldsymbol{z} = \boldsymbol{x}$. Since the function is convex,

$$\max(\mathcal{Q}(\boldsymbol{y}), \mathcal{Q}(\boldsymbol{z})) \geq (1-\gamma)\mathcal{Q}(\boldsymbol{y}) + \gamma \mathcal{Q}(\boldsymbol{z}) \geq \mathcal{Q}((1-\gamma)\boldsymbol{y} + \gamma \boldsymbol{z}) = \mathcal{Q}(\boldsymbol{x}) \,. \tag{23}$$

□

We also assume that the proximal term $g(\boldsymbol{\alpha}) = \sum_{i=1}^n g_i(\alpha_i)$ is such that $g_i(\alpha_i)$ is either $|\alpha_i|$ or enforces a box-constraint.

A.2 Proximal Coordinate Descent

As argued in the introduction, coordinate descent is the method of choice for large scale problems of the form (1). We denote the iterates by $\boldsymbol{\alpha}^{(t)} \in \mathbb{R}^n$, and a single coordinate of this vector by a subscript $\alpha_i^{(t)}$, for $i \in [n]$. CD methods only change one coordinate of the iterates in each iteration. That is, when coordinate i is updated at iteration t, we have $\alpha_j^{(t+1)} = \alpha_j^{(t)}$ for $j \neq i$, and

$$\alpha_i^{(t+1)} := \operatorname{prox}_{\frac{1}{L} g_i} \left[\alpha_i^{(t)} - \frac{1}{L} \nabla_i f(\boldsymbol{\alpha}^{(t)}) \right] , \tag{24}$$

$$\text{where } \operatorname{prox}_{\gamma g_i}[y] := \arg\min_{\boldsymbol{x} \in \mathbb{R}} \tfrac{1}{2}(x-y)^2 + \gamma g_i(x).$$

Combining the smoothness condition, and the definition of the proximal update (24), we get the progress made $\chi_j(\boldsymbol{\alpha})$ is

$$\chi_j(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \min_{\gamma} \left[\gamma \nabla_j f(\boldsymbol{\alpha}^{(t)}) + \frac{L\gamma^2}{2} + g_j(\alpha_j^{(t)} + \gamma) - g_j(\alpha_j^{(t)}) \right] \geq F(\boldsymbol{\alpha}) - \min_{\gamma} F(\boldsymbol{\alpha} + \gamma \mathbf{e}_i) \,. \tag{25}$$

A.3 Applications

There is a number of relevant problems in machine learning which are the form

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \left\{ F(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} l(A\boldsymbol{\alpha}) + \boldsymbol{c}^\top \boldsymbol{\alpha} + \sum_{i=1}^n g_i(\alpha_i) \right\} ,$$

where the non-smooth term $g(\boldsymbol{\alpha})$ either enforces a box constraint, or is an $L1$-regularizer. This class covers several important problems such as SVMs, Lasso regression, logistic regression and elastic net regularized problems. We use SVMs and Lasso regression as running examples for our methods.

SVM. The loss function for training SVMs with λ as the regularization parameter can be written as

$$\min_{\boldsymbol{w}} \left[\mathcal{P}(\boldsymbol{w}) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \left[1 - b_i \boldsymbol{w}^\top \boldsymbol{a}_i\right]_+ + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \right], \tag{26}$$

where $\{(\boldsymbol{a}_i, b_i)\}_{i=1}^n$ for $\boldsymbol{a}_i \in \mathbb{R}^d$ and $b_i \in \{\pm 1\}$ the training data. We can define the corresponding *dual* problem for (26) as

$$\max_{\boldsymbol{\alpha} \in [0,1]^n} \left[\mathcal{D}(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2\lambda n^2} \|A\boldsymbol{\alpha}\|_2^2 \right], \tag{27}$$

where $A \in \mathbb{R}^{d \times n}$ for the data matrix of the columns $b_i \boldsymbol{a}_i$ (Shalev-Shwartz and Zhang, 2013b). We can map this to (1) with $l(\boldsymbol{\alpha}) := -\mathcal{D}(\boldsymbol{\alpha})$, with $\boldsymbol{c} := -\frac{1}{n}\vec{\mathbf{1}}$, and $g_i(\alpha_i) := \mathbf{1}_{\{\alpha_i \in [0,1]\}}$ the box indicator function, i.e.

$$F(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \frac{1}{2\lambda n^2} \|A\boldsymbol{\alpha}\|_2^2 - \frac{1}{n} \sum_{i=1}^{n} \alpha_i + \sum_{i=1}^{n} \mathbf{1}_{\{\alpha_i \in [0,1]\}} .$$

It is straight-forward to see that the function f is coordinate-wise L-smooth for $L = \frac{1}{\lambda n^2} \max_{i \in [n]} \|\boldsymbol{A}_i\|^2$.

We map the dual variable $\boldsymbol{\alpha}$ back to the primal variable as $\boldsymbol{w}(\boldsymbol{\alpha}) = \frac{1}{\lambda n^2} A\boldsymbol{\alpha}$ and the duality gap defined as

$$\text{gap}(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \mathcal{P}(\boldsymbol{w}(\boldsymbol{\alpha})) - \mathcal{D}(\boldsymbol{\alpha}) .$$

Logistic regression. Here we consider the $L1$-regularized logistic regression loss which is of the form

$$\min_{\boldsymbol{\alpha}} \left[F(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \sum_{i=1}^{d} \log\left(1 + \exp(-b_i \boldsymbol{\alpha}^\top \boldsymbol{a}_i)\right) + \lambda \|\boldsymbol{\alpha}\|_1 \right], \tag{28}$$

where $\{(\boldsymbol{a}_i, b_i)\}_{i=1}^d$ for $\boldsymbol{a}_i \in \mathbb{R}^n$ and $b_i \in \{\pm 1\}$ is the training data. The data matrix $A \in \mathbb{R}^{d \times n}$ is composed with $\boldsymbol{a}_i$ as the *rows*. Denote $\boldsymbol{A}_i$ to be the ith *column* of A. As in the Lasso regression case, the regularizer $\lambda \|\boldsymbol{\alpha}\|_1$ is $g(\boldsymbol{\alpha})$ and the sigmoid loss corresponds to $l(\boldsymbol{\alpha})$ which is coordinate-wise L-smooth for

$$L = \frac{1}{4} \max_{i \in [n]} \|\boldsymbol{A}_i\|^2 .$$

Lasso regression. The objective function of Lasso regression (i.e. $L1$-regularized least-squares) can directly be mapped to formulation (1) as

$$\min_{\boldsymbol{\alpha}} \left[F(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \frac{1}{2} \|A\boldsymbol{\alpha} - \boldsymbol{b}\|_2^2 + \lambda \|\boldsymbol{\alpha}\|_1 \right] . \tag{29}$$

Here $l(\boldsymbol{\alpha}) = \frac{1}{2} \|A\boldsymbol{\alpha} - \boldsymbol{b}\|_2^2$, $g(\boldsymbol{\alpha}) = \lambda \|\boldsymbol{\alpha}\|_1$ and f is coordinate-wise L-smooth for $L = \max_{i \in [n]} \|\boldsymbol{A}_i\|^2$.

Elastic net regression. The loss function used for Elastic net can similarly be easily mapped to (1) as

$$\min_{\boldsymbol{\alpha}} \left[F(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \frac{1}{2} \|A\boldsymbol{\alpha} - \boldsymbol{b}\|_2^2 + \frac{\lambda_2}{2} \|\boldsymbol{\alpha}\|_2^2 + \lambda_1 \|\boldsymbol{\alpha}\|_1 \right] . \tag{30}$$

Here $l(\boldsymbol{\alpha}) = \frac{1}{2} \|A\boldsymbol{\alpha} - \boldsymbol{b}\|_2^2 + \frac{\lambda_2}{2} \|\boldsymbol{\alpha}\|_2^2$, $g(\boldsymbol{\alpha}) = \lambda \|\boldsymbol{\alpha}\|_1$. The function f is coordinate-wise L-smooth for

$$L = \max_{i \in [n]} \|\boldsymbol{A}_i\|^2 + \lambda_2 ,$$

as well as λ_2-strongly convex. Similarly, $l(\boldsymbol{\alpha})$ could also include an $L2$-regularization term for logistic regression.

Note that in the SVM case n represents the number of data points with d features whereas the roles are reversed in the regression problems. In all the above cases, g was either a box-constraint (in the SVM) case or was an $L1$-regularizer.

B Algorithms for Box-Constrained Problems

In this section, we give an outline of the GS-s strategy as it applies to box constrained problems as well as derive a reduction to the MIPS framework. Assume that the minimization problem at hand is of the form[3]

$$\min_{\boldsymbol{\alpha}\in[0,1]^n} f(\boldsymbol{\alpha})\,. \tag{31}$$

The proximal coordinate update (24) for updating coordinate i_t simplifies to

$$\alpha_{i_t} := \min\left(1, \left[\alpha_{i_t}^{(t)} - \frac{1}{L}\nabla_{i_t} f(\boldsymbol{\alpha}^{(t)})\right]_+\right). \tag{32}$$

At any iteration t, let us define an *active set* of coordinates $\mathcal{A}_t \subseteq [n]$ as follows

$$\mathcal{A}_t := \left\{ i \in [n] \;\text{ s.t. }\; \begin{array}{r} \alpha_i^{(t)} \in (0,1), \text{ or} \\ \alpha_i^{(t)} = 0 \text{ and } \nabla_i f(\boldsymbol{\alpha}^{(t)}) < 0, \text{ or} \\ \alpha_i^{(t)} = 1 \text{ and } \nabla_i f(\boldsymbol{\alpha}^{(t)}) > 0\,. \end{array} \right\} \tag{33}$$

Then, the following lemma shows that we can also simplify the rule (4) for selecting the steepest direction.

Lemma 5. *At any iteration t, the* GS-s *rule is equivalent to*

$$\max_i \left[\min_{s\in\partial g_i} \left|\nabla_i f(\boldsymbol{\alpha}^{(t)}) + s\right|\right] \equiv \max_{i\in\mathcal{A}_t} \left|\nabla_i f(\boldsymbol{\alpha}^{(t)})\right|\,, \tag{34}$$

assuming that the right side evaluates to 0 if $\mathcal{A}_t$ is empty.

While we will see a formal proof later in Section E, let us quickly get some intuition for why the above works. When the ith coordinate is on the border i.e. $\alpha_i \in \{0,1\}$, there is only one valid direction to move—inside the domain. The set $\mathcal{A}_t$ just maintains the coordinates for which the negative gradient direction leads to a valid non-zero step.

We summarize all the details in Algorithm 2. Note that we have not specified *how* to perform steps 3 (coordinate selection) and 7 (updating the active set). This we will do next in Section B.1. Our algorithm also supports performing an optional explicit line search to update the coordinate at step 5.

Algorithm 2 Box Greedy Coordinate Descent (Box-GCD)

1: **Initialize:** $\boldsymbol{\alpha}_0 \leftarrow \mathbf{0} \in \mathbb{R}^n$, and $\mathcal{A}_0 \leftarrow \left\{i \mid \nabla_i f(\boldsymbol{\alpha}^{(t)}) < 0\right\}$.
2: **for** $t = 0, 1, \ldots,$ until convergence **do**
3: Select coordinate i_t as in GS-s, GS-r, or GS-q. (34).
4: Find α_{i_t} according to (32).
5: *(Optional line search:)*
$\alpha_{i_t} \leftarrow \min_{\gamma\in[0,1]} f(\boldsymbol{\alpha}^{(t)} + (\gamma - \alpha_{i_t}^{(t)})\mathbf{e}_{i_t})$.
6: $\alpha_{i_t}^{(t+1)} \leftarrow \alpha_{i_t}$.
7: Update $\mathcal{A}_{t+1}$ according to (33).
8: **end for**

B.1 Mapping Box Constrained Problems

For this part, just as in the $L1$-regularization case in Section 6, we will only look at functions having the special structure of (13), which for $g(\boldsymbol{\alpha})$ being the box indicator function becomes the constrained problem

$$\min_{\boldsymbol{\alpha}\in[0,1]^n} \left\{ f(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} l(A\boldsymbol{\alpha}) + \boldsymbol{c}^\top\boldsymbol{\alpha} = \sum_{i=1}^d l_i(\boldsymbol{A}_i^\top\boldsymbol{\alpha}) + \boldsymbol{c}^\top\boldsymbol{\alpha} \right\}.$$

[3] If the constraints are of the form $\alpha_i \in [a_i, b_i]$ for $i \in [n]$, we simply rescale and shift the coordinates.

For this case the selection rule in equation (34) in Algorithm 2 becomes

$$\max_{i\in\mathcal{A}_t}|\nabla_i f(\boldsymbol{\alpha})| = \max_{i\in\mathcal{A}_t}\left|\boldsymbol{A}_i^\top \nabla l(A\boldsymbol{\alpha}) + c_i\alpha_i\right|$$

We see that the above is nearly in the form suited for MIPS oracle already, which as we recall from Def. 7, is

$$\text{S-MIPS}_{\mathcal{P}}(\boldsymbol{q};\mathcal{B}) := \arg\max_{j\in\mathcal{B}}\{\langle \boldsymbol{p}_j, \boldsymbol{q}\rangle\}\ .$$

The bulk of our work is now to efficiently maintain the active set $\mathcal{A}_t$ as per (33). We will make two changes to the formulation above to make it amenable to the S-MIPS solvers: i) get around the extra $c_i\alpha_i$ term, and ii) get around the $|\cdot|$.

First to tackle the extra $c_i\alpha_i$ term, we modify the columns of the matrix A as follows

$$\tilde{\boldsymbol{A}}_i := \begin{pmatrix}\beta\\ \boldsymbol{A}_i\end{pmatrix}\ . \tag{35}$$

Now define the query vector $\boldsymbol{q}_t$ as

$$\boldsymbol{q}_t := \begin{pmatrix}\frac{c_i}{\beta}\\ \nabla l(A\boldsymbol{\alpha}^{(t)})\end{pmatrix}\ . \tag{36}$$

It is easy to see that

$$\tilde{\boldsymbol{A}}_i^\top \boldsymbol{q}_t = \boldsymbol{A}_i^\top \nabla l(A\boldsymbol{\alpha}) + c_i\alpha_i\ .$$

The constant β is chosen to ensure that the entry is of the same order of magnitude on average as the rest of the coordinates of $\boldsymbol{A}_i$ and can in general be set to $\frac{1}{\sqrt{n}}$. The β affects the distribution of the data points, as well as the input query vector. Depending on the technique used to solve the S-MIPS instance, tuning β might be useful for the practical performance of the algorithm.

Now to get rid of the absolute value, we use the fact that $|x| = \max(x, -x)$. Define the set $\mathcal{P} = \{\pm\tilde{\boldsymbol{A}}_i\}$. Then,

$$\max_{i\in[n]}\left|\tilde{\boldsymbol{A}}_i^\top \boldsymbol{q}_t\right| = \max_{\boldsymbol{a}\in\mathcal{P}} \boldsymbol{a}^\top \boldsymbol{q}_t\ .$$

To remove the coordinate j from the search space for the equation on the left, it is enough to remove the vector $\arg\max_{\boldsymbol{a}\in\pm\tilde{\boldsymbol{A}}_i} \boldsymbol{a}^\top\boldsymbol{q}_t$ from $\mathcal{P}$. In this manner, by controlling the search space $\mathcal{B}_t \subseteq \mathcal{P}$, we can effectively restrict the coordinates over which we search. Formally, let us define

$$\begin{aligned}\mathcal{P} &:= \mathcal{P}^+ \cup \mathcal{P}^-\ \text{, where}\\ \mathcal{P}^\pm &:= \left\{\pm\tilde{\boldsymbol{A}}_1, \ldots, \pm\tilde{\boldsymbol{A}}_n\right\}\ .\end{aligned} \tag{37}$$

The active set $\mathcal{A}$ contains only the 'valid' directions. Hence, if $\alpha_j = 0$ then $j \in \mathcal{A}_t$ only if $\nabla_j f(\boldsymbol{\alpha}_t) = \tilde{\boldsymbol{A}}_j^\top \boldsymbol{q}_t \le 0$. This can be accomplished by removing $\tilde{\boldsymbol{A}}_j$ and keeping only $-\tilde{\boldsymbol{A}}_j$ in $\mathcal{P}$. In general, at any iteration t with current iterate $\boldsymbol{\alpha}^{(t)}$, we define $\mathcal{B}_t := \mathcal{B}_t^+ \cup \mathcal{B}_t^-$ where

$$\begin{aligned}\mathcal{B}_t^+ &:= \left\{\tilde{\boldsymbol{A}}_j : \alpha_j^{(t)} > 0\right\}\ ,\\ \mathcal{B}_t^- &:= \left\{-\tilde{\boldsymbol{A}}_j : \alpha_j^{(t)} < 1\right\}\ .\end{aligned} \tag{38}$$

Lemma 6. *At any iteration t, for $\mathcal{P}$ and $\mathcal{B}_t$ as defined in (37), (38), the query vector $\boldsymbol{q}_t$ as in (36), and $\mathcal{A}_t$ as in (33) then the following are equivalent for $f(\boldsymbol{\alpha})$ is of the form $l(A\boldsymbol{\alpha}) + \boldsymbol{c}^\top\boldsymbol{\alpha}$:*

$$\text{S-MIPS}_{\mathcal{P}}(\boldsymbol{q}_t;\mathcal{B}_t) = \arg\max_{j\in\mathcal{A}_t}\left|\nabla_j f(\boldsymbol{\alpha}^{(t)})\right|\ .$$

Finally note that the since the vector $\boldsymbol{\alpha}^{(t+1)}$ and $\boldsymbol{\alpha}^{(t)}$ differ in a single coordinate (say i_t), the set $\mathcal{B}_{t+1}$ and $\mathcal{B}_t$ differs in at most two points and can be updated in $O(1)$ time.

C Theoretical Analysis for $L1$-regularized Problems

In this section we discuss our main theoretical contributions. We give formal proofs of the rates of convergence of our algorithms and demonstrate the theoretical superiority of the `GS-s` rule over uniform coordinate descent.

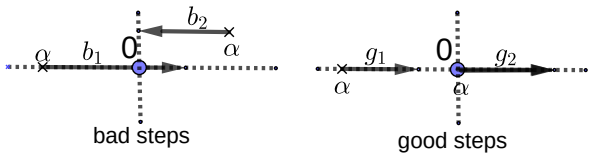

Figure 5: Characterizing steps for the $L1$ case: The arrows represent proximal coordinate updates $S_{\frac{\lambda}{L}}(\alpha_i - \frac{1}{L}\nabla_i f(\boldsymbol{\alpha}))$ from different starting points $\boldsymbol{\alpha}$. Updates which 'cross' (b_1) or 'end at' (b_2) the origin are `bad` whereas the rest (g_1, g_2) are `good`.

Recall the definitions of smoothness and strong convexity of f.

$$f(\boldsymbol{\alpha} + \gamma \mathbf{e}_i) \leq f(\boldsymbol{\alpha}) + \gamma \, \nabla_i f(\boldsymbol{\alpha}) + \frac{L\gamma^2}{2},$$

where $\mathbf{e}_i$ for $i \in [n]$ is a coordinate basis vector and that for any $\Delta\boldsymbol{\alpha} \in \mathbb{R}^n$,

$$f(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) - (f(\boldsymbol{\alpha}) + \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha} \rangle) \geq \frac{\mu_1}{2} \left\| \Delta\boldsymbol{\alpha} \right\|_1^2 .$$

In our proofs we will only focus on the `GS-s` rule, but they are true for the `GS-r` or `GS-q` rules too.

C.1 Good and Bad Steps

The key to the convergence analysis is dividing the steps into `good` steps, which make sufficient progress, and `bad` steps which do not.

Definition 11 (`good` and `bad` steps for $L1$ case)**.** *At any step t, let i_t be the coordinate to be updated. Then if either i) $\alpha_i \cdot \left(S_{\frac{\lambda}{L}}(\alpha_i - \frac{1}{L}\nabla_i f(\boldsymbol{\alpha})) \right) > 0$, or ii) $\alpha_i = 0$, then the step is deemed a **`good`** step (see Figure 5). The steps where $\alpha_i \cdot \left(S_{\frac{\lambda}{L}}(\alpha_i - \frac{1}{L}\nabla_i f(\boldsymbol{\alpha})) \right) \leq 0$ are called **`bad`** steps.*

In general there is no guarantee that all steps are `good`. However our algorithms exploit the structure of the proximal function g to ensure that at least a constant fraction of total steps are `good`. Recall that the algorithm 1 performs the following update step to update the ith coordinate as in (10) and (12):

$$\alpha_i^+ = S_{\frac{\lambda}{L}} \left(\alpha_i^{(t)} - \frac{1}{L} \nabla_i f(\alpha_i^{(t)}) \right) \quad \text{and}$$

$$\alpha_i^{(t+1)} = \begin{cases} \alpha_i^+, & \text{if } \alpha_i^+ \alpha_i^{(t)} \geq 0 \\ 0, & \text{otherwise} \end{cases} .$$

From here on we will stop indicating the iteration number t when obvious.

Lemma 7 (Counting `good` steps)**.** *After t iterations of Algorithm 1, out of the t steps, at least $\lceil t/2 \rceil$ steps are **`good`**.*

Proof. The definition 11 says that the only bad steps are those for which i) $\alpha_i^{(t)} \neq 0$ and ii) $\alpha_i \alpha_i^+ \leq 0$. However by our modification to the update rule (12) ensures that in this case, $\alpha_i^{(t+1)} = 0$. This ensures that the next time coordinate i is picked, we are guaranteed a `good` step. Since we start at the origin, we have our lemma. □

C.2 Progress Made in `good` and `bad` Steps

Below is the core technical lemma in the convergence proof. We show that if the step was `good`, then the update chosen by the `GS-s` rule actually corresponds to optimizing an upper-bound on the function with the usual $L2$-squared regularizer replaced by an $L1$-squared regularizer. We of course also have only an approximate `GS-s` coordinate.

Recall that we had defined

$$\chi_j(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \min_{\gamma} \left[\gamma \nabla_j f(\boldsymbol{\alpha}) + \frac{L\gamma^2}{2} + \lambda(\left\| \alpha_j + \gamma \right\|_1 - \left\| \alpha_j \right\|_1) \right] .$$

Lemma 8 (good steps make a lot of progress). *Suppose that the Θ-approximate GS-s rule (recall Definition 1) chose to update the ith coordinate, and further suppose that it was a good step. Then the following holds:*

$$\chi_i(\boldsymbol{\alpha}) \leq \Theta^2 \min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \left[\langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 + \lambda(|\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}| - |\boldsymbol{\alpha}|)\right] .$$

Proof. Recall from Section 3.1 that the GS-s rule for the $L1$ case was to find a coordinate i such that

$$|s(\boldsymbol{\alpha})_j| \geq \Theta \arg\max_j |s(\boldsymbol{\alpha})_j| \ .$$

The vector $s(\boldsymbol{\alpha})$ (by expanding the definition of the shrinkage operator) can equivalently be defined as

$$s(\boldsymbol{\alpha})_j := \begin{cases} 0, & \text{if } \alpha_j = 0, |\nabla_j f(\boldsymbol{\alpha})| \leq \lambda \\ \nabla_j f(\boldsymbol{\alpha}) - \lambda, & \text{if } \alpha_j = 0, \nabla_j f(\boldsymbol{\alpha}) > \lambda \\ \nabla_j f(\boldsymbol{\alpha}) + \lambda, & \text{if } \alpha_j = 0, \nabla_j f(\boldsymbol{\alpha}) < -\lambda \\ \nabla_j f(\boldsymbol{\alpha}) + \text{sign}(\alpha_j)\lambda & \text{otherwise} . \end{cases}$$

Let us define the folowing vector $\boldsymbol{\zeta} \in [-1,1]^n$ as

$$\zeta_j := (s(\boldsymbol{\alpha})_j - \nabla_j f(\boldsymbol{\alpha}))/\lambda \, .$$

Examining the four cases listed above shows indeed that $\zeta_j \in [-1,1]$. We will use the fact that for any $\beta \in [-1,1]$, $|x| \geq \beta x$. We have that

$$|\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}| \geq \langle \boldsymbol{\zeta}, \boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}\rangle = \langle \boldsymbol{\zeta}, \boldsymbol{\alpha}\rangle + \langle \boldsymbol{\zeta}, \Delta\boldsymbol{\alpha}\rangle \, .$$

This implies that for any vector $\Delta\boldsymbol{\alpha} \in \mathbb{R}^n$,

$$\begin{aligned} \left\{ \mathcal{P}_{||}(\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 + \lambda(\left\|\boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}\right\|_1 - \left\|\boldsymbol{\alpha}\right\|_1) \right\} & \\ \geq \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 + \lambda(\langle \boldsymbol{\zeta}, \boldsymbol{\alpha}+\Delta\boldsymbol{\alpha}\rangle - \left\|\boldsymbol{\alpha}\right\|_1) & \\ = \langle \nabla f(\boldsymbol{\alpha}) + \lambda\boldsymbol{\zeta}, \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 + \lambda(\langle \boldsymbol{\zeta}, \boldsymbol{\alpha}\rangle - \left\|\boldsymbol{\alpha}\right\|_1) & \\ = \langle s(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 + \lambda(\langle \boldsymbol{\zeta}, \boldsymbol{\alpha}\rangle - \left\|\boldsymbol{\alpha}\right\|_1) \, . & \end{aligned}$$

In the last step we used the definition of $\boldsymbol{\zeta}$. Let us examine the expression $(\zeta_j\alpha_j - |\alpha_j|)$. If $\alpha_j = 0$, each of the terms in the expression is 0 and so it evaluates to 0. If $\alpha_j \neq 0$, then by the definition, $\zeta_j = \text{sign}(\alpha_j)$ and $\zeta_j\alpha_j = |\alpha_j|$. In this case too, the expression is 0. Thus,

$$\begin{aligned} \mathcal{P}_{||}(\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha}) &\geq \langle s(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 + \lambda(\langle \boldsymbol{\zeta}, \boldsymbol{\alpha}\rangle - \left\|\boldsymbol{\alpha}\right\|_1) \\ &= \left\{ \langle s(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 \stackrel{\text{def}}{=} \mathcal{P}_{\boldsymbol{\zeta}}(\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha}) \right\} . \end{aligned}$$

Minimizing $\mathcal{P}_{\boldsymbol{\zeta}}(\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha})$ over $\Delta\boldsymbol{\alpha}$ gives us that

$$\min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \langle s(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 = -\frac{1}{2L}\max_{j\in[n]} (s(\boldsymbol{\alpha})_j)^2 \ .$$

This exactly corresponds to the GS-s rule. Since i was a Θ-approximate GS-s direction,

$$\min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \langle s(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha}\rangle + \frac{L}{2}\left\|\Delta\boldsymbol{\alpha}\right\|_1^2 \geq -\frac{1}{2L\Theta^2} (s(\boldsymbol{\alpha})_i)^2$$

Further since this is a good step, by Lemma 9, we can replace the right side of the above equation with $\chi_i(\boldsymbol{\alpha})$. Putting these observations together we have that for any $\boldsymbol{\alpha} \in \mathbb{R}^n$

$$\begin{aligned} \min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \mathcal{P}_{||}(\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha}) &\geq \min_{\Delta\boldsymbol{\alpha}\in\mathbb{R}^n} \mathcal{P}_{\boldsymbol{\zeta}}(\boldsymbol{\alpha}, \Delta\boldsymbol{\alpha}) \\ &\geq \frac{1}{\Theta^2}\chi_i(\boldsymbol{\alpha}) \, . \end{aligned}$$

Rearranging the terms gives the proof of the lemma. □

Lemma 9 (Characterizing a good step). *Suppose that we update the ith coordinate and that it was a good step.*

Then

$$\chi_i(\boldsymbol{\alpha}) = -\frac{1}{2L}(\alpha_i^+ - \alpha_i)^2 = -\frac{1}{2L}(s(\boldsymbol{\alpha})_i)^2\,.$$

Proof. By the definition of the coordinate proximal update,

$$\begin{aligned}\chi_i(\boldsymbol{\alpha}) &= \min_{\gamma\in\mathbb{R}}\left[\gamma\nabla_i f(\boldsymbol{\alpha}) + \frac{L\gamma^2}{2} + \lambda(|\alpha_i+\gamma| - |\alpha_i|)\right] \\ &= (\alpha_i^+ - \alpha_i)\nabla_j f(\boldsymbol{\alpha}) + \frac{L}{2}(\alpha_i^+ - \alpha_i)^2 + \lambda(\left|\alpha_i^+\right| - |\alpha_i|)\,.\end{aligned}$$

Now first suppose that $\alpha_i \neq 0$. Since this is a good step, $\operatorname{sign}(\alpha_i^+) = \operatorname{sign}(\alpha_i)$. Without loss of generality, let us assume that $\alpha_i > 0$. Then $(\alpha_i^+ = \alpha_i - (\nabla_i f(\boldsymbol{\alpha}) + \lambda)/L)$. Using this in the above expression,

$$\begin{aligned}\chi_i(\boldsymbol{\alpha}) &= (\alpha_i^+ - \alpha_i)\nabla_i f(\boldsymbol{\alpha}) + \frac{L}{2}(\alpha_i^+ - \alpha_i)^2 + \lambda(\left|\alpha_i^+\right| - |\alpha_i|) \\ &= -\frac{1}{L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda)(\nabla_i f(\boldsymbol{\alpha}) + \frac{L}{2L^2}\left(\nabla_i f(\boldsymbol{\alpha}) + \lambda\right)^2 - \frac{\lambda}{L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda) \\ &= -\frac{1}{L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda)(\nabla_i f(\boldsymbol{\alpha}) + \lambda) + \frac{1}{2L}\left(\nabla_i f(\boldsymbol{\alpha}) + \lambda\right)^2 \\ &= -\frac{1}{2L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda)^2 \\ &= -\frac{1}{2L}(s(\boldsymbol{\alpha})_i)^2\,.\end{aligned}$$

The proof for when $\alpha_i < 0$ is identical. Now let us see the case when $\alpha_i = 0$. Since this is not a `bad` step, we have that $\alpha_i^+ \neq 0$ meaning that $|\nabla_i f(\boldsymbol{\alpha})| > \lambda$. Without loss of generality, assume that $\alpha_i^+ > 0$—the other case is identical. Then $(\alpha_i^+ = -\frac{1}{L}\left(\nabla_i f(\boldsymbol{\alpha}) + \lambda\right))$ and $(s(\boldsymbol{\alpha})_i = \nabla_i f(\boldsymbol{\alpha}) + \lambda)$. Doing the same computations as before,

$$\begin{aligned}\chi_i(\boldsymbol{\alpha}) &= (\alpha_i^+ - \alpha_i)\nabla_i f(\boldsymbol{\alpha}) + \frac{L}{2}(\alpha_i^+ - \alpha_i)^2 + \lambda(\left|\alpha_i^+\right| - |\alpha_i|) \\ &= -\frac{1}{L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda)(\nabla_i f(\boldsymbol{\alpha}) + \frac{L}{2L^2}\left(\nabla_i f(\boldsymbol{\alpha}) + \lambda\right)^2 - \frac{\lambda}{L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda) \\ &= -\frac{1}{L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda)(\nabla_i f(\boldsymbol{\alpha}) + \lambda) + \frac{1}{2L}\left(\nabla_i f(\boldsymbol{\alpha}) + \lambda\right)^2 \\ &= -\frac{1}{2L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda)^2 \\ &= -\frac{1}{2L}(\alpha_i^+ - \alpha_i)^2 \\ &= -\frac{1}{2L}(s(\boldsymbol{\alpha})_i)^2\,.\end{aligned}$$

Such simple calculations shows that indeed the lemma holds in all cases. □

Remark 12. *Lemma 9 shows that for a* ***good*** *step, the* `GS-s`*,* `GS-r`*, and* `GS-q` *rules coincide. Thus even though we explicitly write the analysis for the* `GS-s` *rule, it also holds for the other updates. Note that while the three rules coincide for a* ***good*** *step, they can be quite different during the* ***bad*** *steps. Further the computations required for the three rules is not the same since we a priori do not know if a step is going to be* ***good*** *or* ***bad****.*

We have characterized the update made in a `good` step and now let us look at the `bad` steps.

Lemma 10 (`bad` steps are not too bad)**.** *In any step of Algorithm 1 including the* ***bad*** *steps, the objective value never increases.*

Proof. If the step was `good`, since the update (10) is just a proximal coordinate update, it is guaranteed to not increase the function value. The modification we make when the step is `bad` (12) makes the lemma slightly less obvious. Without loss of generality, by symmetry of $L1$ about the origin, let us assume that $\alpha_i > 0$. Since the step was `bad`, this implies that $\alpha_i^+ = \alpha_i - \frac{1}{L}(\nabla_i f(\boldsymbol{\alpha}) + \lambda) < 0$ for (12). Then

$$F(\boldsymbol{\alpha}^{t+1}) - F(\boldsymbol{\alpha}^{(t)}) \leq -\nabla_i f(\boldsymbol{\alpha})\alpha_i + \frac{L}{2}\alpha_i^2 - \lambda\,|\alpha_i|$$

$$
\begin{aligned}
&= \alpha_i \left(-\nabla_i f(\boldsymbol{\alpha}) + \frac{L}{2}\alpha_i - \lambda \right) \\
&\leq L\alpha_i \left(-\frac{1}{L}\nabla_i f(\boldsymbol{\alpha}) + \alpha_i/2 - \frac{1}{L}\lambda \right) \\
&= L\alpha_i \left(\left(\alpha_i - \frac{1}{L}[\nabla_i f(\boldsymbol{\alpha}) + \lambda] \right) - \alpha_i/2 \right) \\
&\leq L\alpha_i \left(0 + 0\right) .
\end{aligned}
$$

□

C.3 Convergence in the Strongly Convex Case

Theorem 5. *After t steps of Algorithm 1 where in each step the coordinate was selected using the Θ-approximate* `GS-s` *rule, let $\mathcal{G}_t \subseteq [t]$ indicate the* `good` *steps. Assume that the function was L-coordinate-wise smooth and μ_1 strongly convex with respect to the $L1$ norm. The size $|\mathcal{G}_t| \geq \lceil t/2 \rceil$ and*

$$
F(\boldsymbol{\alpha}^{(t+1)}) - F(\boldsymbol{\alpha}^\star) \leq \left(1 - \frac{\Theta^2 \mu_1}{L}\right)^{|\mathcal{G}_t|} \left(F(\boldsymbol{\alpha}^{(0)}) - F(\boldsymbol{\alpha}^\star) \right) .
$$

We have almost everything we need in place to prove our convergence rates for the strongly convex, and the general convex case. Recall that the function f is μ_1 strongly convex with respect to the $L1$ norm. This implies that

$$
F(\boldsymbol{\alpha}^\star) \geq f(\boldsymbol{\alpha}) + \langle \nabla f(\boldsymbol{\alpha}), \boldsymbol{\alpha}^\star - \boldsymbol{\alpha} \rangle + \frac{\mu_1}{2} \left\| \boldsymbol{\alpha}^\star - \boldsymbol{\alpha} \right\|_1^2 + g(\boldsymbol{\alpha}^\star) .
$$

We will need one additional Lemma from (Karimireddy et al., 2018) to relate the upper bound we minimize in Lemma 8.

Lemma 11 (Relating different regularizing constants (Karimireddy et al., 2018))**.** *For any vectors $\boldsymbol{g}, \boldsymbol{\alpha} \in \mathbb{R}^n$, and constants $L \geq \mu > 0$,*

$$
\min_{\boldsymbol{w} \in \mathbb{R}^n} \left\{ \lambda(\|\boldsymbol{w}\|_1 - \|\boldsymbol{\alpha}\|_1) + \langle \boldsymbol{g}, \boldsymbol{w} - \boldsymbol{\alpha} \rangle + \frac{L}{2} \|\boldsymbol{w} - \boldsymbol{\alpha}\|_1^2 \right\} \leq \frac{\mu}{L} \min_{\boldsymbol{w} \in R^n} \left\{ \lambda(\|\boldsymbol{w}\|_1 - \|\boldsymbol{\alpha}\|_1) + \langle \boldsymbol{g}, \boldsymbol{w} - \boldsymbol{\alpha} \rangle + \frac{\mu}{2} \|\boldsymbol{w} - \boldsymbol{\alpha}\|_1^2 \right\} .
$$

Proof. The function $\lambda(|\boldsymbol{w}| - |\boldsymbol{\alpha}|) + \langle \boldsymbol{g}, \boldsymbol{w} - \boldsymbol{\alpha} \rangle$ is convex and is 0 when $\boldsymbol{w} = \boldsymbol{\alpha}$, and $\|\boldsymbol{w} - \boldsymbol{\alpha}\|_1$ is a convex positive function. Thus we can apply Lemma 9 from (Karimireddy et al., 2018). □

The proof of the theorem now easily follows.

Proof of Theorem 5. First, by Lemma 7, we know that there are at least as many `good` steps as there are `bad` steps. This means that $|\mathcal{G}_t| \geq \lceil t/2 \rceil$ for any t. Now suppose that t was a `good` step and updated the ith coordinate. Now by the progress made in a proximal update (25) and Lemma 8,

$$
\begin{aligned}
F(\boldsymbol{\alpha}^{(t+1)}) &\leq F(\boldsymbol{\alpha}^{(t)}) + \chi_i(\boldsymbol{\alpha}^{(t)}) \\
&= F(\boldsymbol{\alpha}^{(t)}) + \Theta^2 \min_{\boldsymbol{w} \in \mathbb{R}^n} \left\{ \left\langle \nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\rangle + \frac{L}{2} \left\| \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\|_1^2 + \lambda(\|\boldsymbol{w}\|_1 - \left\| \boldsymbol{\alpha}^{(t)} \right\|_1) \right\} \\
&\leq F(\boldsymbol{\alpha}^{(t)}) + \frac{\Theta^2 \mu_1}{L} \min_{\boldsymbol{w} \in \mathbb{R}^n} \left\{ \left\langle \nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\rangle + \frac{\mu_1}{2} \left\| \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\|_1^2 + \lambda(\|\boldsymbol{w}\|_1 - \left\| \boldsymbol{\alpha}^{(t)} \right\|_1) \right\}
\end{aligned}
$$

In the last step we used Lemma 11. We will now use our definition of strong convexity. We have shown that

$$
\begin{aligned}
F(\boldsymbol{\alpha}^{(t+1)}) - F(\boldsymbol{\alpha}^\star) &\leq F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \\
&\quad + \frac{\Theta^2 \mu_1}{L} \min_{\boldsymbol{w} \in \mathbb{R}^n} \left\{ \left\langle \nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\rangle + \frac{\Theta^2 \mu_1}{2} \left\| \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\|_1^2 + \lambda(\|\boldsymbol{w}\|_1 - \left\| \boldsymbol{\alpha}^{(t)} \right\|_1) \right\} \\
&\leq F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) + \\
&\quad \frac{\Theta^2 \mu_1}{L} \left(\left\langle \nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)} \right\rangle + \frac{\Theta^2 \mu_1}{2} \left\| \boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)} \right\|_1^2 + \lambda(\|\boldsymbol{\alpha}^\star\|_1 - \left\| \boldsymbol{\alpha}^{(t)} \right\|_1) \right)
\end{aligned}
$$

$$\leq F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) + \frac{\Theta^2\mu_1}{L}\left(f(\boldsymbol{\alpha}^\star) - f(\boldsymbol{\alpha}^{(t)}) + \lambda(\|\boldsymbol{\alpha}^\star\|_1 - \left\|\boldsymbol{\alpha}^{(t)}\right\|_1)\right)$$
$$= (1 - \frac{\Theta^2\mu_1}{L})\left(F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star)\right).$$

We have shown that we make significant progress every `good` step. Combining this with Lemma 10 which shows that the function value does not increase in a `bad` step finishes the proof. □

Remark 13. *We show that the* `GS-s` *rule has convergence rates indepedent of n for L1-regularized problems. As long as the update is kept the same as in Algorithm 1 (with the modification), our proof also works for the* `GS-r` *and the* `GS-q` *rules. This answers the conjecture posed by (Nutini et al., 2015) in the affirmative, at least for L1-regularized problems.*

C.4 Convergence in the General Convex Case

Theorem 6. *After t steps of Algorithm 1 where in each step the coordinate was selected using the Θ-approximate* `GS-s` *rule, let $\mathcal{G}_t \subseteq [t]$ indicate the* `good` *steps. Assume that the function was L-coordinate-wise smooth. Then the size $|\mathcal{G}_t| \geq \lceil t/2 \rceil$ and*

$$F(\boldsymbol{\alpha}^{(t+1)}) - F(\boldsymbol{\alpha}^\star) \leq \frac{LD^2}{2\Theta^2\,|\mathcal{G}_t|},$$

where D is the L1-diameter of the level set. For the set of minima $\mathcal{Q}^\star$,

$$D = \max_{\boldsymbol{w}\in\mathbb{R}^n} \min_{\boldsymbol{\alpha}^\star\in\mathcal{Q}^\star} \left\{\|\boldsymbol{w} - \boldsymbol{\alpha}^\star\|_1 \,\middle|\, F(\boldsymbol{w}) \leq F(\boldsymbol{\alpha}^{(0)})\right\}.$$

Proof. We start exactly as in the strongly convex case. First, by Lemma 7, we know that there are at least as many `good` steps as there are `bad` steps. This means that $|\mathcal{G}_t| \geq \lceil t/2 \rceil$ for any t. Now suppose that t was a `good` step and updated the ith coordinate. Now by the progress made in a proximal update (25) and Lemma 8,

$$\begin{aligned}
F(\boldsymbol{\alpha}^{(t+1)}) &\leq F(\boldsymbol{\alpha}^{(t)}) + \chi_i(\boldsymbol{\alpha}^{(t)}) \\
&\leq f(\boldsymbol{\alpha}^{(t)}) + \Theta^2 \min_{\boldsymbol{w}\in\mathbb{R}^n}\left\{\left\langle\nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w} - \boldsymbol{\alpha}^{(t)}\right\rangle + \frac{L}{2}\left\|\boldsymbol{w} - \boldsymbol{\alpha}^{(t)}\right\|_1^2 + \lambda\|\boldsymbol{w}\|_1\right\} \\
&\leq f(\boldsymbol{\alpha}^{(t)}) + \Theta^2 \min_{\boldsymbol{w}=(1-\gamma)\boldsymbol{\alpha}^{(t)}+\gamma\boldsymbol{\alpha}^\star}\left\{\left\langle\nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w} - \boldsymbol{\alpha}^{(t)}\right\rangle + \frac{L}{2}\left\|\boldsymbol{w} - \boldsymbol{\alpha}^{(t)}\right\|_1^2 + \lambda\|\boldsymbol{w}\|_1\right\} \\
&= f(\boldsymbol{\alpha}^{(t)}) + \Theta^2 \min_{\gamma\in\mathbb{R}}\left\{\gamma\left\langle\nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)}\right\rangle + \frac{L\gamma^2}{2}\left\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)}\right\|_1^2 + \lambda\left\|(1-\gamma)\boldsymbol{\alpha}^{(t)} + \gamma\boldsymbol{\alpha}^\star\right\|_1\right\} \\
&\leq f(\boldsymbol{\alpha}^{(t)}) + \Theta^2 \min_{\gamma\in\mathbb{R}}\left\{\gamma(f(\boldsymbol{\alpha}^\star) - f(\boldsymbol{\alpha}^{(t)})) + \frac{L\gamma^2}{2}\left\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)}\right\|_1^2 + \lambda(1-\gamma)\left\|\boldsymbol{\alpha}^{(t)}\right\|_1 + \lambda\gamma\|\boldsymbol{\alpha}^\star\|_1\right\}.
\end{aligned}$$

In the last step we used the convexity of f and of $|\cdot|$. Now denote the suboptimality $h_t = F(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star)$. We have shown that

$$\begin{aligned}
h_{t+1} &= F(\boldsymbol{\alpha}^{(t+1)}) - F(\boldsymbol{\alpha}^\star) \\
&\leq f(\boldsymbol{\alpha}^{(t)}) - F(\boldsymbol{\alpha}^\star) \\
&\qquad + \Theta^2 \min_{\gamma\in\mathbb{R}}\left\{\gamma(f(\boldsymbol{\alpha}^\star) - f(\boldsymbol{\alpha}^{(t)})) + \frac{L\gamma^2}{2}\left\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)}\right\|_1^2 + \lambda(1-\gamma)\left\|\boldsymbol{\alpha}^{(t)}\right\|_1 + \lambda\gamma\|\boldsymbol{\alpha}^\star\|_1\right\} \\
&= f(\boldsymbol{\alpha}^{(t)}) + \lambda\left\|\boldsymbol{\alpha}^{(t)}\right\|_1 - F(\boldsymbol{\alpha}^\star) \\
&\qquad + \Theta^2 \min_{\gamma\in\mathbb{R}}\left\{\gamma(f(\boldsymbol{\alpha}^\star) + \lambda\|\boldsymbol{\alpha}^\star\|_1 - f(\boldsymbol{\alpha}^{(t)}) - +\lambda\left\|\boldsymbol{\alpha}^{(t)}\right\|_1) + \frac{L\gamma^2}{2}\left\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)}\right\|_1^2\right\} \\
&= h_t + \Theta^2 \min_{\gamma\in\mathbb{R}} -\gamma h_t + \gamma^2\frac{L}{2}\left\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)}\right\|_1^2 \\
&\leq h_t + \Theta^2 \min_{\gamma\in\mathbb{R}} -\gamma h_t + \gamma^2\frac{LD^2}{2} \\
&= h_t - \frac{2\Theta^2 h_t^2}{LD^2}.
\end{aligned}$$

Now we know that the function value is non-increasing both during the `good` and `bad` steps from Lemma 10. Hence $h_t^2 \geq h_t h_{t+1}$. Using this inequality and dividing the entire equation by $h_t h_{t+1}$ gives

$$\frac{1}{h_t} \leq \begin{cases} \frac{1}{h_{t+1}} - \frac{2\Theta^2}{LD^2} & \text{if } t \in \mathcal{G}_t \\ \frac{1}{h_{t+1}} & \text{o.w.} \end{cases}.$$

Summing this up gives that

$$\frac{1}{h_{t+1}} \geq \frac{1}{h_0} + \frac{|\mathcal{G}_{t+1}|\, 2\Theta^2}{LD^2} \geq \frac{|\mathcal{G}_{t+1}|\, 2\Theta^2}{LD^2}.$$

Inverting the above equation gives the required theorem. □

Remark 14. *We show that the* `GS-s` *rule has convergence rates independent of n for $L1$-regularized problems. As long as the update (12) is kept the same as in Algorithm 1, our proof also works for the* `GS-r` *and the* `GS-q` *rules. Previously only the* `GS-r` *was analyzed for this case by Dhillon et al. (2011). A careful reading of their proof gives the rate*

$$h_t \leq \frac{8LD^2}{t}.$$

Thus we improve the rate of convergence by a factor of 8 even for `GS-r`*, and show a convergence result for the first time for* `GS-s`*.*

D Theoretical Analysis for Box-constrained Problems

In this section we examine the algorithm for the box-constrained case. We give formal proofs of the rates of convergence and demonstrate the theoretical superiority of the `GS-s` rule over uniform coordinate descent.

Recall the definition of coordinate-wise smoothness of f:

$$f(\boldsymbol{\alpha} + \gamma \mathbf{e}_i) \leq f(\boldsymbol{\alpha}) + \gamma\, \nabla_i f(\boldsymbol{\alpha}) + \frac{L\gamma^2}{2},$$

where $\mathbf{e}_i$ for $i \in [n]$ is a coordinate basis vector. Using strong convexity, for any $\Delta\boldsymbol{\alpha} \in \mathbb{R}^n$ gives:

$$f(\boldsymbol{\alpha} + \Delta\boldsymbol{\alpha}) \geq f(\boldsymbol{\alpha}) + \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha} \rangle + \frac{\mu_1}{2} \|\Delta\boldsymbol{\alpha}\|_1^2.$$

D.1 The Good, the Bad, and the Cross Steps

Unlike in the $L1$ case, we need to divide the steps into three kinds. Differentiating between the three kinds is key to our analysis.

Definition 15 (`good`, `bad`, and `cross` steps)**.** *At any step t, let i be the coordinate being updated. Then if $(\alpha_i - \frac{1}{L}\nabla_i f(\boldsymbol{\alpha})) \in (0,1)$ it is called a* ***`good`*** *step. If $(\alpha_i - \frac{1}{L}\nabla_i f(\boldsymbol{\alpha})) \notin (0,1)$, and $\alpha_i \in (0,1)$ the step is considered* ***`bad`****. Finally if $(\alpha_i - \frac{1}{L}\nabla_i f(\boldsymbol{\alpha})) \notin (0,1)$ and $\alpha_i \in \{0,1\}$, we have a* ***`cross`*** *step. See Figure 6 for illustration.*

We would like to bound the number of `bad` steps. This we can do thanks to the structure of the box constraint.

Lemma 12 (Counting `bad` steps)**.** *After t iterations of Algorithm 2, we have atmost $\lfloor t/2 \rfloor$* ***`bad`*** *steps.*

Proof. Suppose we are updating the ith coordinate. As is clear from the definition (and Fig. 6), a `bad` step occurs when we start in the interioir ($\alpha_i \in (0,1)$) and attempt to move outside. But in this case, our update ensures that $\alpha_i^{(t+1)} \in \{0,1\}$. Thus the next time coordinate i is picked, it cannot be a `bad` step. Since we start at the origin $\mathbf{0}^n$, in the first t steps we can have atmost $\lfloor t/2 \rfloor$ `bad` steps. □

D.2 Progress Made in One Step

This section is the core technical part of the proof. The key to our rate is realizing that the three kinds of steps have to be dealt with separately, and proposing a novel analysis of the `cross` step. Recall that we had defined

$$\chi_j(\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \min_{\gamma + \alpha_j \in [0,1]} \left\{ \gamma \nabla_j f(\boldsymbol{\alpha}) + \frac{L\gamma^2}{2} \right\}.$$

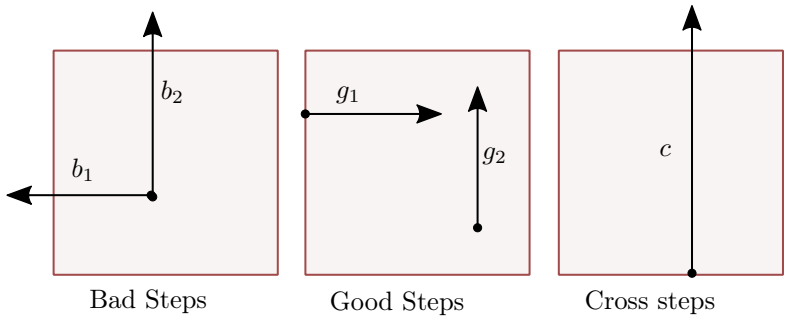

Figure 6: Characterizing steps in box-constrained case (for $n = 2$): The `bad` steps correspond to those which start in the interior and the update $-\frac{1}{L}\nabla_i f(\boldsymbol{\alpha})$ is interrupted by the boundary of the box constraint (as in steps b_1 and b_2). The `good` steps are those which end in the interior of the box (g_1 and g_2), and the `cross` steps such as c are those which both start and end at the boundary.

Let us also recall the update step used in Algorithm 2. The update used is

$$\alpha_i^+ = \min\left(1, \left[\alpha_i - \frac{1}{L}\nabla_i f(\boldsymbol{\alpha})\right]_+\right).$$

The `GS-s` rule used to choose the coordinate $i \in \mathcal{A}$ such that

$$|\nabla_i f(\boldsymbol{\alpha})| \geq \Theta \max_{j \in \mathcal{A}} |\nabla_j f(\boldsymbol{\alpha})| \,,$$

where the active set $\mathcal{A} \subseteq [n]$ consists of the coordinates which have feasible update directions:

$$j \in \mathcal{A} \text{ if } \exists \gamma > 0 \text{ such that } (\alpha_j - \gamma \nabla_j f(\boldsymbol{\alpha})) \in [0,1]\,.$$

Before we begin, we should verify if the algorithm is even feasible—make sure that $\mathcal{A}$ is never empty.

Lemma 13 ($\mathcal{A}$ is not empty)**.** *If $\mathcal{A} = \emptyset$, then $\boldsymbol{\alpha}$ is the optimum.*

Proof. If $\mathcal{A} = \emptyset$, it means that none of the negative gradient directions are feasible i.e. for any $j \in [n]$ and $v \in [0,1]$, $(\nabla_j f(\boldsymbol{\alpha})(v - \alpha_j) \geq 0)$. This means that for any vector $\boldsymbol{v} \in [0,1]^n$,

$$\langle \nabla f(\boldsymbol{\alpha}), \boldsymbol{v} - \boldsymbol{\alpha} \rangle = \sum_{j \in [n]} \nabla_j f(\boldsymbol{\alpha})(v_j - \alpha_j) \geq 0\,.$$

This implies that $\boldsymbol{\alpha}$ is the optimum. □

Now let us first look at the `good` steps.

Lemma 14 (`good` steps make a lot of progress)**.** *Suppose that the Θ-approximate `GS-s` rule (recall Def. 1) chose to update the ith coordinate, and further suppose that it was a `good` step. Then,*

$$\chi_i(\boldsymbol{\alpha}) \leq \Theta^2 \min_{\Delta\boldsymbol{\alpha} \in [0,1]^n - \boldsymbol{\alpha}} \left\{ \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha} \rangle + \frac{L}{2} \|\Delta\boldsymbol{\alpha}\|_1^2 \right\}.$$

Proof. Define the right hand side above to be $\mathcal{P}(\Delta\boldsymbol{\alpha})$ defined for $\Delta\boldsymbol{\alpha} \in \mathbb{R}^n$ such that $\Delta\boldsymbol{\alpha} + \boldsymbol{\alpha} \in [0,1]^n$:

$$\mathcal{P}(\Delta\boldsymbol{\alpha}) \stackrel{\text{def}}{=} \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha} \rangle + \frac{L}{2} \|\Delta\boldsymbol{\alpha}\|_1^2 \,.$$

Suppose we are given some $\boldsymbol{v}$ in the domain. Let us construct a vector $\tilde{\boldsymbol{v}} \in \mathbb{R}^n$ such that

$$\tilde{v}_j = \begin{cases} 0, & \text{if } \alpha_j = 0, \text{ and } \nabla_j f(\boldsymbol{\alpha}) \geq 0 \\ 0, & \text{if } \alpha_j = 1, \text{ and } \nabla_j f(\boldsymbol{\alpha}) \leq 0 \\ v_j, & \text{otherwise}\,. \end{cases}$$

Note that we can instead rewrite $\tilde{\boldsymbol{v}}$ as $\tilde{v}_j = v_j$ if $j \in \mathcal{A}$, or else is 0. Also we have $\|\boldsymbol{v}\|_1 \geq \|\tilde{\boldsymbol{v}}\|_1$.

For a coordinate $j \in [n]$ such that $\alpha_j = 0$, it holds that $v_j \in [0,1]$. Then $\nabla_j f(\boldsymbol{\alpha}) \geq 0$ implies that $\nabla_j f(\boldsymbol{\alpha}) v_j \geq 0$. Similarly if $\alpha_j = 1$ and $\nabla_j f(\boldsymbol{\alpha}) \leq 0$, then it holds that $\nabla_j f(\boldsymbol{\alpha}) v_j \geq 0$. Thus we have

$$\mathcal{P}(\tilde{\boldsymbol{v}}) \leq \mathcal{P}(\boldsymbol{v})\,.$$

This means that if we want to minimize $\mathcal{P}$, we can restrict our search space to coordinates in $([n] \setminus \mathcal{A})$. We will use $\Delta\boldsymbol{\alpha} \in [0,1]^n - \boldsymbol{\alpha}, \Delta\boldsymbol{\alpha}[\mathcal{A}] = 0$ to mean the set $\Delta\boldsymbol{\alpha} \in \mathbb{R}^n$ such that $\Delta\boldsymbol{\alpha} + \boldsymbol{\alpha} \in [0,1]^n$ and for all $j \in \mathcal{A}$, $\Delta\boldsymbol{\alpha}_j = 0$. We then have that

$$\begin{aligned}
\min_{\Delta\boldsymbol{\alpha} \in [0,1]^n - \boldsymbol{\alpha}} \mathcal{P}(\Delta\boldsymbol{\alpha}) &= \min_{\Delta\boldsymbol{\alpha} \in [0,1]^n - \boldsymbol{\alpha}, \Delta\boldsymbol{\alpha}[\mathcal{A}] = 0} \mathcal{P}(\Delta\boldsymbol{\alpha}) \\
&\geq \min_{\Delta\boldsymbol{\alpha} \in \mathbb{R}^n, \Delta\boldsymbol{\alpha}[\mathcal{A}] = 0} \mathcal{P}(\Delta\boldsymbol{\alpha}) \\
&= \min_{\Delta\boldsymbol{\alpha} \in \mathbb{R}^n, \Delta\boldsymbol{\alpha}[\mathcal{A}] = 0} \left\{ \langle \nabla f(\boldsymbol{\alpha}), \Delta\boldsymbol{\alpha} \rangle + \frac{L}{2} \left\| \Delta\boldsymbol{\alpha} \right\|_1^2 \right\} \\
&= -\frac{1}{2L} \max_{j \in \mathcal{A}} |\nabla_j f(\boldsymbol{\alpha})| \\
&\geq -\frac{\Theta^2}{2L} \left| \nabla_i f(\boldsymbol{\alpha}) \right| .
\end{aligned}$$

The last step is because i was defined to be a Θ-approximate `GS-s` direction. Now we also know that the update was a `good` step. This means that $(\alpha_i - \nabla_i f(\boldsymbol{\alpha})/L) \in [0,1]$ which means that

$$\begin{aligned}
\chi_i(\boldsymbol{\alpha}) &= \min_{\gamma + \alpha_i \in [0,1]} \left\{ \gamma \nabla_i f(\boldsymbol{\alpha}) + \frac{L\gamma^2}{2} \right\} \\
&\leq (\alpha_i^+ - \alpha_i) \nabla_i f(\boldsymbol{\alpha}) + \frac{L}{2} (\alpha_i^+ - \alpha_i)^2 \\
&= -\frac{1}{2L} \left| \nabla_i f(\boldsymbol{\alpha}) \right| \\
&\leq \frac{1}{\Theta^2} \min_{\Delta\boldsymbol{\alpha} \in [0,1]^n - \boldsymbol{\alpha}} \mathcal{P}(\Delta\boldsymbol{\alpha})\,.
\end{aligned}$$

This finishes the proof of the lemma. □

We now turn our attention to the `cross` step which crosses from one end to the other.

Lemma 15 (`cross` steps also make a lot of progress)**.** *Suppose that the Θ-approximate `GS-s` rule chose to update the ith coordinate, and further suppose that it was a `cross` step. Then,*

$$\chi_i(\boldsymbol{\alpha}) \leq \frac{\Theta}{2n} \min_{\boldsymbol{v} \in [0,1]^n} \left\{ \langle \nabla f(\boldsymbol{\alpha}), \boldsymbol{v} - \boldsymbol{\alpha} \rangle \right\} .$$

Proof. Since the update was a `cross` step, this means that $(\alpha_i - \nabla_i f(\boldsymbol{\alpha})) \neq [0,1]$ and that $\alpha_i^+ \in \{0,1\}$. In particular this imlpies that when solving for the optimal γ in the below problem, it is greater than 1:

$$\begin{aligned}
1 &< \arg\min_{\gamma \geq 0} \left\{ \gamma (\nabla_i f(\boldsymbol{\alpha}))(\alpha_i^+ - \alpha_i) + \frac{L\gamma^2}{2} (\alpha_j^+ - \alpha_j)^2 \right\} \\
&= \frac{-(\nabla_i f(\boldsymbol{\alpha}))(\alpha_j^+ - \alpha_j)}{L(\alpha_i^+ - \alpha_i)^2} .
\end{aligned}$$

Now using this inequality in $\chi_i(\boldsymbol{\alpha})$ we get

$$\begin{aligned}
\chi_i(\boldsymbol{\alpha}) &= (\nabla_i f(\boldsymbol{\alpha}))(\alpha_i^+ - \alpha_i) + \frac{L}{2} (\alpha_i^+ - \alpha_i)^2 \\
&\leq (\nabla_i f(\boldsymbol{\alpha}))(\alpha_i^+ - \alpha_i) - \frac{1}{2} (\nabla_i f(\boldsymbol{\alpha}))(\alpha_i^+ - \alpha_i) \\
&= \frac{1}{2} (\nabla_i f(\boldsymbol{\alpha}))(\alpha_i^+ - \alpha_i) \\
&= -\frac{1}{2} \left| \nabla_i f(\boldsymbol{\alpha}) \right| .
\end{aligned}$$

The last step is because in a `cross` step, $\left|\alpha_i^+ - \alpha_i\right| = 1$ and is in the opposite direction of the gradient coordinate. Now since i was a Θ-approximate `GS-s` direction,

$$\begin{aligned}\chi_i(\boldsymbol{\alpha}) &\leq -\frac{1}{2}\left|\nabla_i f(\boldsymbol{\alpha})\right| \\ &\leq -\frac{\Theta}{2}\max_{j\in\mathcal{A}}\left|\nabla_j f(\boldsymbol{\alpha})\right| \\ &\leq -\frac{\Theta}{2\left|\mathcal{A}\right|}\max_{\boldsymbol{v}\in[0,1]^n}\left\langle\nabla f(\boldsymbol{\alpha}), \boldsymbol{\alpha}-\boldsymbol{v}\right\rangle \\ &\leq \frac{\Theta}{2n}\min_{\boldsymbol{v}\in[0,1]^n}\left\langle\nabla f(\boldsymbol{\alpha}), \boldsymbol{v}-\boldsymbol{\alpha}\right\rangle.\end{aligned}$$

□

Finally let us check how bad the `bad` steps really are.

Lemma 16 (`bad` steps are not too bad)**.** *In any step of Algorithm 2 including the* `bad` *steps, the objective value never increases.*

Proof. This directly follows from the fact that we always minimize an upper bound on the function $f(\boldsymbol{\alpha})$ at every iteration. □

D.3 Convergence in the Strongly Convex Case

Theorem 7. *After t steps of Algorithm 2 where in each step the coordinate was selected using a Θ-approximate* `GS-s` *rule, let $\mathcal{B}_t \subseteq [t]$ indicate the* `bad` *steps. Assume that the function f is L-coordinate-wise smooth and μ_1 strongly convex with respect to the $L1$ norm. Then the size of $|\mathcal{B}_t| \leq \lfloor t/2 \rfloor$ and*

$$f(\boldsymbol{\alpha}^{(t+1)}) - f(\boldsymbol{\alpha}^\star) \leq \left(1 - \min\left(\frac{\Theta}{2n}, \frac{\Theta^2\mu_1}{L}\right)\right)^{t-|\mathcal{B}_t|}\left(f(\boldsymbol{\alpha}^{(0)}) - f(\boldsymbol{\alpha}^\star)\right).$$

As is standard, $f(\boldsymbol{\alpha}^\star) = \min_{\boldsymbol{\alpha}\in[0,1]^n} f(\boldsymbol{\alpha})$.

We have almost everything we need in place to prove our convergence rates for the strongly convex, and the general convex case. Recall that the function f is μ_1 strongly convex with respect to the $L1$ norm. This implies that

$$f(\boldsymbol{\alpha}^\star) \geq f(\boldsymbol{\alpha}) + \left\langle\nabla f(\boldsymbol{\alpha}), \boldsymbol{\alpha}^\star - \boldsymbol{\alpha}\right\rangle + \frac{\mu_1}{2}\left\|\boldsymbol{\alpha}^\star - \boldsymbol{\alpha}\right\|_1^2.$$

We will need one additional Lemma from (Karimireddy et al., 2018) to relate the upper bound we minimize in Lemma 14.

Lemma 17 (Relating different regularizing constants (Karimireddy et al., 2018))**.** *For any vectors $\boldsymbol{g}, \boldsymbol{w} \in \mathbb{R}^n$, and constants $L \geq \mu > 0$,*

$$\min_{\boldsymbol{w}\in[0,1]^n}\left\{\left\langle\boldsymbol{g}, \boldsymbol{w}-\boldsymbol{\alpha}\right\rangle + \frac{L}{2}\left\|\boldsymbol{w}-\boldsymbol{\alpha}\right\|_1^2\right\} \leq \frac{\mu}{L}\min_{\boldsymbol{w}\in[0,1]^n}\left\{\left\langle\boldsymbol{g}, \boldsymbol{w}-\boldsymbol{\alpha}\right\rangle + \frac{\mu}{2}\left\|\boldsymbol{w}-\boldsymbol{\alpha}\right\|_1^2\right\}.$$

Proof. The function $\langle\boldsymbol{g}, \boldsymbol{w}-\boldsymbol{\alpha}\rangle$ is convex and is 0 when $\boldsymbol{w} = \boldsymbol{\alpha}$, the set $[0,1]^n$ is also convex, and $\left\|\boldsymbol{w}-\boldsymbol{\alpha}\right\|_1$ is a convex positive function. Thus we can apply Lemma 9 from (Karimireddy et al., 2018). □

The proof of the theorem now easily follows.

Proof of Theorem 7. First by Lemma 12, we know that $|\mathcal{B}_t| \leq \lfloor t/2 \rfloor$. Now suppose that t was a `good` step and updated the ith coordinate. The progress made in a proximal update using (25) and Lemma 14,

$$\begin{aligned}f(\boldsymbol{\alpha}^{(t+1)}) &\leq f(\boldsymbol{\alpha}^{(t)}) + \chi_i(\boldsymbol{\alpha}^{(t)}) \\ &\leq f(\boldsymbol{\alpha}^{(t)}) + \Theta^2\min_{\boldsymbol{w}\in[0,1]^n}\left\{\left\langle\nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w}-\boldsymbol{\alpha}^{(t)}\right\rangle + \frac{L}{2}\left\|\boldsymbol{w}-\boldsymbol{\alpha}^{(t)}\right\|_1^2\right\} \\ &\leq f(\boldsymbol{\alpha}^{(t)}) + \frac{\Theta^2\mu_1}{L}\min_{\boldsymbol{w}\in[0,1]^n}\left\{\left\langle\nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w}-\boldsymbol{\alpha}^{(t)}\right\rangle + \frac{\mu_1}{2}\left\|\boldsymbol{w}-\boldsymbol{\alpha}^{(t)}\right\|_1^2\right\}.\end{aligned}$$

In the last step we used Lemma 17. We will now use our definition of strong convexity. We have shown that

$$\begin{aligned} f(\boldsymbol{\alpha}^{(t+1)}) - f(\boldsymbol{\alpha}^\star) &\leq f(\boldsymbol{\alpha}^{(t)}) - f(\boldsymbol{\alpha}^\star) + \frac{\Theta^2\mu_1}{L} \min_{\boldsymbol{w}\in[0,1]^n} \left\{ \left\langle \nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\rangle + \frac{\Theta^2\mu_1}{2} \left\| \boldsymbol{w} - \boldsymbol{\alpha}^{(t)} \right\|_1^2 \right\} \\ &\leq f(\boldsymbol{\alpha}^{(t)}) - f(\boldsymbol{\alpha}^\star) + \frac{\Theta^2\mu_1}{L} \left(\left\langle \nabla f(\boldsymbol{\alpha}^{(t)}), \boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)} \right\rangle + \frac{\Theta^2\mu_1}{2} \left\| \boldsymbol{\alpha}^\star - \boldsymbol{\alpha}^{(t)} \right\|_1^2 \right) \\ &= (1 - \frac{\Theta^2\mu_1}{L}) \left(f(\boldsymbol{\alpha}^{(t)}) - f(\boldsymbol{\alpha}^\star) \right) . \end{aligned}$$

Now instead suppose that the step t was a `cross` step. In that case Lemma 15 tells us that

$$\begin{aligned} f(\boldsymbol{\alpha}^{(t+1)}) &\leq f(\boldsymbol{\alpha}^{(t)}) + \chi_i(\boldsymbol{\alpha}^{(t)}) \\ &\leq f(\boldsymbol{\alpha}^{(t)}) + \frac{\Theta}{2n} \min_{\boldsymbol{w}\in[0,1]^n} \langle \nabla f(\boldsymbol{\alpha}), \boldsymbol{w} - \boldsymbol{\alpha} \rangle \\ &\leq f(\boldsymbol{\alpha}^{(t)}) + \frac{\Theta}{2n} \langle \nabla f(\boldsymbol{\alpha}), \boldsymbol{\alpha}^\star - \boldsymbol{\alpha} \rangle \\ &\leq f(\boldsymbol{\alpha}^{(t)}) + \frac{\Theta}{2n} \left(f(\boldsymbol{\alpha}^\star) - f(\boldsymbol{\alpha}) \right) . \end{aligned}$$

In the last step, we used the convexity of f. Subtracting $f(\boldsymbol{\alpha}^\star)$ and rearranging gives that

$$f(\boldsymbol{\alpha}^{(t+1)}) - f(\boldsymbol{\alpha}^\star) \leq \left(1 - \frac{\Theta}{2n} \right) \left(f(\boldsymbol{\alpha}^{(t)}) - f(\boldsymbol{\alpha}^\star) \right) .$$

Finally if the step was `bad`, we know form Lemma 16 that the function value does not decrease. Putting these three results about the different cases together gives us the theorem. □

Remark 16. *In the box-constrained case, we do not quite get the straightforward linear rate involving the condition number $\frac{L}{\mu_1}$ which we were expecting. We instead get a rate which depends on $\max\left(\frac{L}{\mu_1}, n\right)$. The new n term is because of the `cross` steps. In the case of separable quadratics which a diagonal Hessian:*

$$\nabla^2 f(\alpha) = diag(\lambda_1, \ldots, \lambda_n) .$$

In such a case, Nutini et al. (2015)[Section 4.1] show that the constant μ_1 is

$$\mu_1 = \left(\sum_{j=1}^n \frac{1}{\lambda_j} \right)^{-1} .$$

Since L in our case was defined to be the max diagonal element of the Hessian, we have that

$$\frac{L}{\mu_1} = \left(\max_{j\in[n]} \lambda_j \right) \left(\sum_{j=1}^n \frac{1}{\lambda_j} \right) \geq \left(\sum_{j=1}^n \frac{\lambda_j}{\lambda_j} \right) = n .$$

This means that, at least in the separable quadratic case, $\frac{L}{\mu_1} \geq n$ and the rate of convergence depends only on the condition number.

D.4 Convergence in the General Convex Case

Theorem 8. *After t steps of Algorithm 2 where in each step the coordinate was selected using the Θ-approximate `GS-s` rule, let $\mathcal{B}_t \subseteq [t]$ indicate the `bad` steps. Assume that the function was L-coordinate-wise smooth. Then the size $|\mathcal{B}_t| \leq \lfloor t/2 \rfloor$ and*

$$f(\boldsymbol{\alpha}^{(t+1)}) - f(\boldsymbol{\alpha}^\star) \leq \frac{8LD^2}{\Theta^2(t - |\mathcal{B}_t|)} ,$$

where D is the $L1$ diameter of the level set. For the set of minima $\mathcal{Q}^\star$,

$$D = \max_{\boldsymbol{w}\in[0,1]^n} \min_{\boldsymbol{\alpha}^\star\in\mathcal{Q}^\star} \left\{ \|\boldsymbol{w} - \boldsymbol{\alpha}^\star\|_1 \,|\, f(\boldsymbol{w}) \leq f(\boldsymbol{\alpha}^{(0)}) \right\} .$$

Proof. This essentially follows from the proof of (Dhillon et al., 2011)[Lemma 8]. As we saw before, in `good` steps all three rules `GS-s`, `GS-r` and `GS-q` coincide. For `cross` steps, by definition we take the largest step possible and so

is also a valid GS-r step. Thus we can fall back onto to the analysis of GS-r even for the GS-s rule. One could also derive a rate directly from Lemma 14 and Lemma 15, but we would not achieve a significantly better rate. □

E Proofs of Mapping Between GS-s and MIPS

In this section we will discuss the proof of our claims that our formulation of casting the problem of finding the steepest GS-s direction as an instance of the S-MIPS problem.

Proof of Lemma 5 [Box case]: We want to show that

$$\max_i \left[\min_{s \in \partial g_i} \left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) + \boldsymbol{s} \right| \right] = \max_{i \in \mathcal{A}_t} \left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) \right| ,$$

where the active set $\mathcal{A}_t$ is defined as

$$\mathcal{A}_t := \left\{ i \in [n] \;\text{ s.t. } \begin{array}{l} \alpha_i^{(t)} \in (0,1), \text{ or} \\ \alpha_i^{(t)} = 0 \text{ and } \nabla_i f(\boldsymbol{\alpha}^{(t)}) < 0, \text{ or} \\ \alpha_i^{(t)} = 1 \text{ and } \nabla_i f(\boldsymbol{\alpha}^{(t)}) > 0 . \end{array} \right\}$$

Let us examine the subgradient of the indicator function $\mathbf{1}_{\{[0,1]\}}$. If $\alpha_i^{(t)} \in (0,1)$, the subgradient of the indicator function is 0. If $\alpha_i^{(t)} = 0$, the subgradient is $\partial g_i = (-\infty, 0]$ and if $\alpha_i^{(t)} = 1$, the subgradient is $\partial g_i = [0, \infty)$. Thus $\min_{s \in \partial g_i} \left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) + \boldsymbol{s} \right|$ equals $\left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) \right|$ if $\alpha_i \in (0,1)$, or if $\alpha_i^{(t)} = 0$ and $\nabla_i f(\boldsymbol{\alpha}^{(t)}) < 0$, or $\alpha_i^{(t)} = 1$ and $\nabla_i f(\boldsymbol{\alpha}^{(t)}) > 0$. In all other cases, it is 0. This proves the lemma.

Proof of Lemma 1 [$L1$ case]: We want to show that

$$\max_i \left[\min_{s \in \partial g_i} \left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) + \boldsymbol{s} \right| \right] = \max_i |s(\boldsymbol{\alpha})_i|$$

Here $g_i(\alpha_i^{(t)}) = \lambda \left| \alpha_i^{(t)} \right|$. If $\alpha_i^{(t)} \neq 0$, $\partial g_i = \lambda \operatorname{sign}(\alpha_i^{(t)})$. If $\alpha_i^{(t)} = 0$, $\partial g_i = [-\lambda, \lambda]$. Thus the value of $\min_{s \in \partial g_i} \left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) + \boldsymbol{s} \right|$ is $\left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) + \lambda \operatorname{sign}(\alpha_i^{(t)}) \right|$ if $\alpha_i^{(t)} \neq 0$. If $\alpha_i^{(t)} = 0$, then $\min_{s \in \partial g_i} \left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) + \boldsymbol{s} \right|$ evaluates to $\left| S_\lambda(\nabla_i f(\boldsymbol{\alpha}^{(t)})) \right|$. In short, $\min_{s \in \partial g_i} \left| \nabla_i f(\boldsymbol{\alpha}^{(t)}) + \boldsymbol{s} \right|$ exactly evaluates to $\boldsymbol{s}(\boldsymbol{\alpha}^{(t)})$. This finishes the proof of the lemma.

Proof of Lemma 6 [Box case]: We want to show that

$$\max_{\tilde{\boldsymbol{A}}_j \in \mathcal{B}_t} \left\langle \tilde{\boldsymbol{A}}_j, \boldsymbol{q}_t \right\rangle = \arg\max_{j \in \mathcal{A}_t} \left| \nabla_j f(\boldsymbol{\alpha}^{(t)}) \right| .$$

Given the structure of $f(\boldsymbol{\alpha})$, we can rewrite $\nabla_j f(\boldsymbol{\alpha}^{(t)}) = \left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_j$. Further we can write

$$\arg\max_{j \in \mathcal{A}_t} \left| \nabla_j f(\boldsymbol{\alpha}^{(t)}) \right| = \arg\max_{j \in \mathcal{A}_t} \max \Big(\left\{ \left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_i \right\} , \left\{ \left\langle -\boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle - c_i \right\} \Big) . \tag{39}$$

Let us examine at the cases we need to ignore i.e. the points not in $\mathcal{A}_t$: i) $\alpha_i^{(t)} = 0$ and $\nabla_i f(\boldsymbol{\alpha}^{(t)}) > 0$ or ii) $\alpha_i^{(t)} = 1$ and $\nabla_i f(\boldsymbol{\alpha}^{(t)}) < 0$. Suppose some coordinate j' falls into case (i). Instead of excluding it from $\mathcal{A}_t$, we could instead keep only the $\left\langle -\boldsymbol{A}_{j'}, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle - c_i$ in the right hand side of (39) in place of $\max \left(\left\langle \boldsymbol{A}_{j'}, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_i, \left\langle -\boldsymbol{A}_{j'}, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle \right) - c_i$. This automatically excludes j' from the arg max if $\nabla l(A\boldsymbol{\alpha}^{(t)}) + c_i > 0$ or equivalently if $\nabla_{j'} f(\boldsymbol{\alpha}^{(t)}) > 0$. Similarly if j' falls in case (ii), we will keep only the $\left\langle \boldsymbol{A}_{j'}, \nabla l(A\boldsymbol{\alpha}^{(t)}) + c_i \right\rangle$ term which excludes j' from the arg max if $\nabla l(A\boldsymbol{\alpha}^{(t)}) + c_i < 0$ or equivalently if $\nabla_{j'} f(\boldsymbol{\alpha}^{(t)}) < 0$. This is exactly what the term $\max_{\tilde{\boldsymbol{A}}_j \in \mathcal{B}_t} \left\langle \tilde{\boldsymbol{A}}_j, \boldsymbol{q}_t \right\rangle$ does.

Proof of Lemma 3 [$L1$ case]: We want to show that

$$\max_{\tilde{\boldsymbol{A}}_j \in \mathcal{B}_t} \left\langle \tilde{\boldsymbol{A}}_j, \boldsymbol{q}_t \right\rangle = \arg\max_i |s(\boldsymbol{\alpha})_i| .$$

Given the structure of $f(\boldsymbol{\alpha})$, we can rewrite $\nabla_j f(\boldsymbol{\alpha}^{(t)}) = \left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_j$. Further we can use the equivalence in Lemma 1 to reformulate the required statement as

$$\max_{\tilde{\boldsymbol{A}}_j \in \mathcal{B}_t} \left\langle \tilde{\boldsymbol{A}}_j, \boldsymbol{q}_t \right\rangle = \arg\max_j \left| \min_{s \in \partial \left|\alpha_j^{(t)}\right|} \left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_j + \lambda s \right| . \tag{40}$$

Drawing from the proof of Lemma 1, we look at the following cases:

1. When $\alpha_j > 0$: In this case $\partial \left|\alpha_j^{(t)}\right|$ evaluates to 1 and the right side of (40) becomes $\max\left(\left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_j + \lambda, \left\langle -\boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle - c_j - \lambda\right)$.
2. When $\alpha_j < 0$: In this case $\partial \left|\alpha_j^{(t)}\right|$ evaluates to -1 and the right side of (40) becomes $\max\left(\left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_j - \lambda, \left\langle -\boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle - c_j + \lambda\right)$.
3. When $\alpha_j = 0$ and $\left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_j \geq 0$: The right side of (40) becomes $\max\left(\left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_j - \lambda, 0\right)$.
4. When $\alpha_j = 0$ and $\left\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle + c_j \leq 0$: The right side of (40) becomes $\max\left(\left\langle -\boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha}^{(t)}) \right\rangle - c_j - \lambda, 0\right)$.

Comparing the four cases above to the definition of $\mathcal{B}_t$ shows that the lemma holds.

F Further Extensions of the Steepest and MIPS Framework

In this section we show that the steepest rules `GS-s` can be extended to take into account different smoothness constants along different coordinate directions and that the new rules are also instances of our MIPS framework. Further we show that the steepest rule for $L2$-regularized problems can also be cast as an MIPS instance.

F.1 Coordinate Specific Smoothness Constants

The `GS-s` and the `GS-q` strategies as described in this work reduce to the `GS` rule when the function is smooth and $g = 0$. However if the smoothness constants (Definition 8) highly varies along different coordinate directions, then the `GS-L` rule which accounts for this variation enjoys better theoretical convergence properties. We can similarly modify the `GS-s` and the `GS-q` strategies to account for coordinate specific smoothness constants.

Suppose that we modify Definition 8 as in Nutini et al. (2015) and let L_i be the smoothness constant along coordinate direction i. Then we rescale the ith coordinate by $1/L_i$. This step is equivalent to normalizing the columns of the data matrix A when $f(\boldsymbol{\alpha})$ is of the form $l(A\boldsymbol{\alpha})$. The rescaling will make the smoothness constants of all the coordinates equal (in fact equal to 1). Now we can apply the techniques in this work to obtain steepest rules which implicitly take advantage of the coordinate specific smoothness constants L_i. Of course this would change the regularization parameter λ along the coordinates, and we will have a coordinate-wise regularization constants λ_i. Our algorithms and proofs can easily be extended to this setting.

F.2 Block Coordinate Setting

Just as in the smooth case, it might be possible to also extend the theoretical results here for the block coordinate case using new norms in place of the $L1$ norm in which the analysis is currently done. Define $\boldsymbol{x}_{[:i]}$ to mean a the i largest absolute values of $\boldsymbol{x}$ and $\boldsymbol{x}_{[i:]}$ to mean the $n-i$ smallest absolute values. If we want to pick the top κ coordinates for e.g., we could define a new norm using which the analysis could extend

$$\|\boldsymbol{x}\|_{[\kappa]}^2 \stackrel{\text{def}}{=} \left\|\boldsymbol{x}_{[:n+1-\kappa]}\right\|_1^2 + \left\|\boldsymbol{x}_{[n+1-\kappa:]}\right\|_2^2 .$$

F.3 Solving ℓ_2 Regularized Problems using S-MIPS

Suppose we are given a problem of the form

$$l(A\boldsymbol{\alpha}) + \frac{\lambda}{2}\|\boldsymbol{\alpha}\|^2 .$$

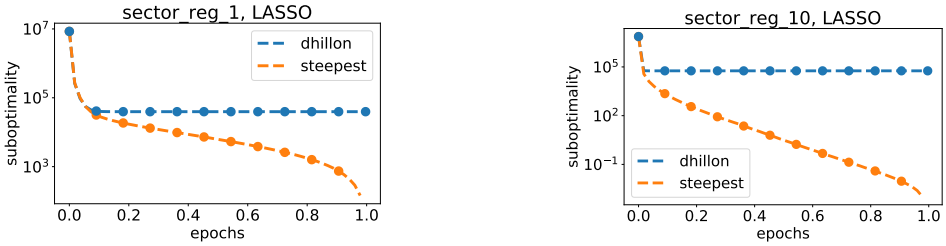

Figure 7: Evaluating `dhillon`: `steepest` which is based on the `GS-s` rule consistently outperforms `dhillon` which quickly stagnates.

In this case, the `GS-s` rule simplifies to

$$\arg\max_{j\in[n]} |\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha})\rangle + \lambda\alpha_j| \ .$$

We can cast this in the MIPS framework. We need to add n new components to the d dimensional vector $\boldsymbol{A}_j$ as follows:

$$\tilde{\boldsymbol{A}}_i := \begin{pmatrix} \beta \mathbf{e}_i \\ \boldsymbol{A}_i \end{pmatrix} . \tag{41}$$

The constant β is tuned to ensure good performance of the underlying MIPS algorithm, and is typically chosen to be $O(1/\sqrt{n})$. Then, define

$$\begin{aligned} \mathcal{P} &:= \mathcal{P}^+ \cup \mathcal{P}^- \text{ where,} \\ \mathcal{P}^\pm &:= \left\{ \pm\tilde{\boldsymbol{A}}_1, \ldots, \pm\tilde{\boldsymbol{A}}_n \right\} . \end{aligned} \tag{42}$$

Finally, construct the query vector $\boldsymbol{q}_t$ as

$$\boldsymbol{q}_t := \begin{pmatrix} \frac{\lambda}{\beta}\boldsymbol{\alpha}^{(t)} \\ \nabla l(A\boldsymbol{\alpha}^{(t)}) \end{pmatrix} . \tag{43}$$

Then it is easy to verify the following equivalence -

$$\text{MIPS}_{\mathcal{P}}(\boldsymbol{q}_t) = \arg\max_{j\in[n]} |\langle \boldsymbol{A}_j, \nabla l(A\boldsymbol{\alpha})\rangle + \lambda\alpha_j| \ .$$

However we are now dealing with vectors of size $d+n$ instead of just d which is usually too expensive ($n \gg d$). Handling $\tilde{\boldsymbol{A}}_i$ is easy since it has at most $d+1$ non-zero entires. To get around the computational cost of hashing $\boldsymbol{q}_t$, we note that the hyperplane hashing only requires computing $\langle \boldsymbol{w}, \boldsymbol{q}_t\rangle$ for some vector $\boldsymbol{w}$. This can be broken down as

$$\langle \boldsymbol{w}, \boldsymbol{q}_t\rangle = \frac{\lambda}{\beta}\left\langle \boldsymbol{w}_1, \boldsymbol{\alpha}^{(t)}\right\rangle + \left\langle \boldsymbol{w}_2, \nabla l(A\boldsymbol{\alpha}^{(t)})\right\rangle .$$

At iteration $t+1$, we will need to recompute the second part in time $O(d)$. However since $\boldsymbol{\alpha}^{(t+1)}$ and $\boldsymbol{\alpha}^{(t)}$ differ in only 1 component (say i_t), updating the first part of the hash computation can be done efficiently as

$$\langle \boldsymbol{w}, \boldsymbol{q}_{t+1}\rangle = \frac{\lambda}{\beta}(\left\langle \boldsymbol{w}_1, \boldsymbol{\alpha}^{(t)}\right\rangle + w_{i_t}(\alpha_{i_{t+1}} - \alpha_{i_t})) + \left\langle \boldsymbol{w}_2, \nabla l(A\boldsymbol{\alpha}^{(t+1)})\right\rangle .$$

This can also be combined with ideas from Section 6 to solve problems which have both an $L2$-regularizer and are box-constrained or $L1$-regularized. This is important for example for solving elastic net regression.

G Additional Experiments

G.1 Consistent poor performance of `dhillon`

Here we run `dhillon` on the `sector` datasets. Figure 7 illustrates consistent poor performance of `dhillon` rule.

Table 2: Parameters for `sector` dataset. (**d**, **n**) is dataset size, β is tunable constant from (18), (19), ρ is sparsity of the solution.

Dataset	n	d	$\sqrt{n}\beta$	efC	efS	post	ρ
`sector`, $\lambda = 1$	55,197	6,412	10	100	100	2	10%
`sector`, $\lambda = 10$	55,197	6,412	10	400	200	0	3%

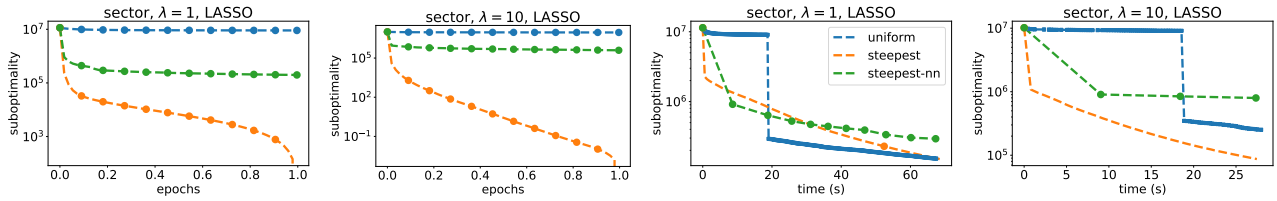

Figure 8: The performance of `steepest-nn` for `sector` dataset is similar to that on `rcv1`.

G.2 `nmslib` on `sector` dataset

We perform additional evaluation of `steepest-nn` algorithm on `sector` dataset for the LASSO. `nmslib` hyper-parameters and sparsity levels could be found in Table 2. Figure 8 shows that `steepest-nn` for `sector` dataset has the same strengths and weaknesses.

G.3 Make regression

We also use the `make-regression` function from Sk-learn (Pedregosa et al., 2011)[4] to create tunable instances for Lasso. Here we keep the number of informative features fixed at 100, fix $d = 1000$, and vary n in $\{10^4, 10^5, 10^6\}$. From Fig. 9, we can see that as n increases, `steepest-nn` starts significantly outperforming `uniform` in terms of wall clock time. This is to be expected since the cost per iteration of `steepest-nn` remains roughly fixed, whereas the gain in progress over `uniform` grows with n.

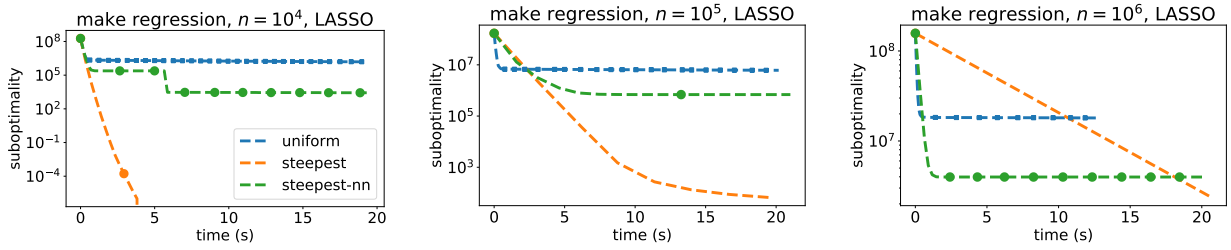

Figure 9: Performance on `make-regression` with $n \in \{10^4, 10^5, 10^6\}$. The advantage of `steepest-nn` over `uniform` and `steepest` increases with increasing problem sizes.

G.4 Extreme Sparse Solution

Here we experiment with LASSO on `sector` dataset in extreme case, with $\lambda = 100$. The solution has only 5 non-zero coordinates. Figure 10 shows comparison of `steepest-nn` (implemented with `nmslib`), `uniform` and `steepest` rules. `steepest` converges to the true optimum in less than 10 iterations while `steepest-nn` and `uniform` strugling to find a good coordinate. As in other experiments, `steepest-nn` shows its usefulness in early stage of optimization.

G.5 Test Accuracy

For SVM experiments we randomly split the datasets to create train (75%) and test (25%) subsets. Figure 11 shows the primal function value, dual function value, duality gap, and accuracy on the test set for `w1a` and `ijcnn1`

[4] http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_regression.html

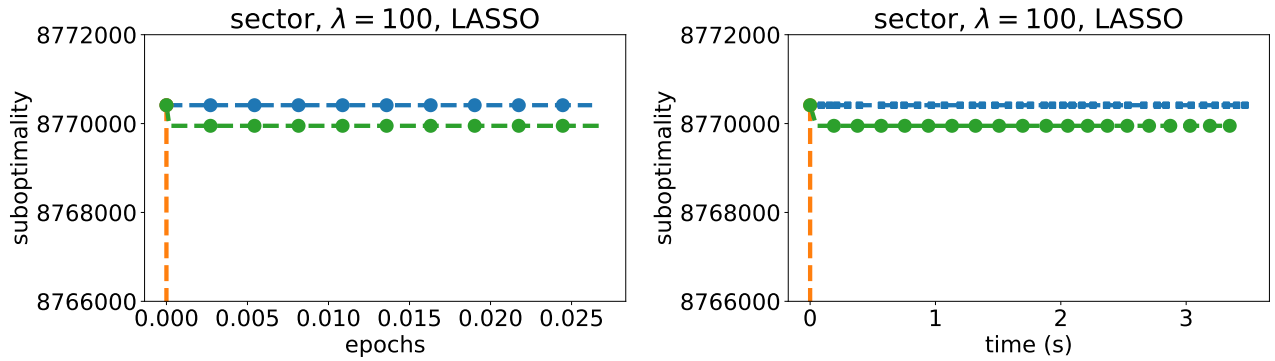

Figure 10: The solution is extremely sparse. `steepest` converges in less than 10 iterations. `steepest-nn` has a small gain over `uniform`, but both of them extremely bad compared to `steepest` even in wall-time.

datasets. Even measured against wall time, `steepest-nn` is very competitive with `uniform` in all the metrics.

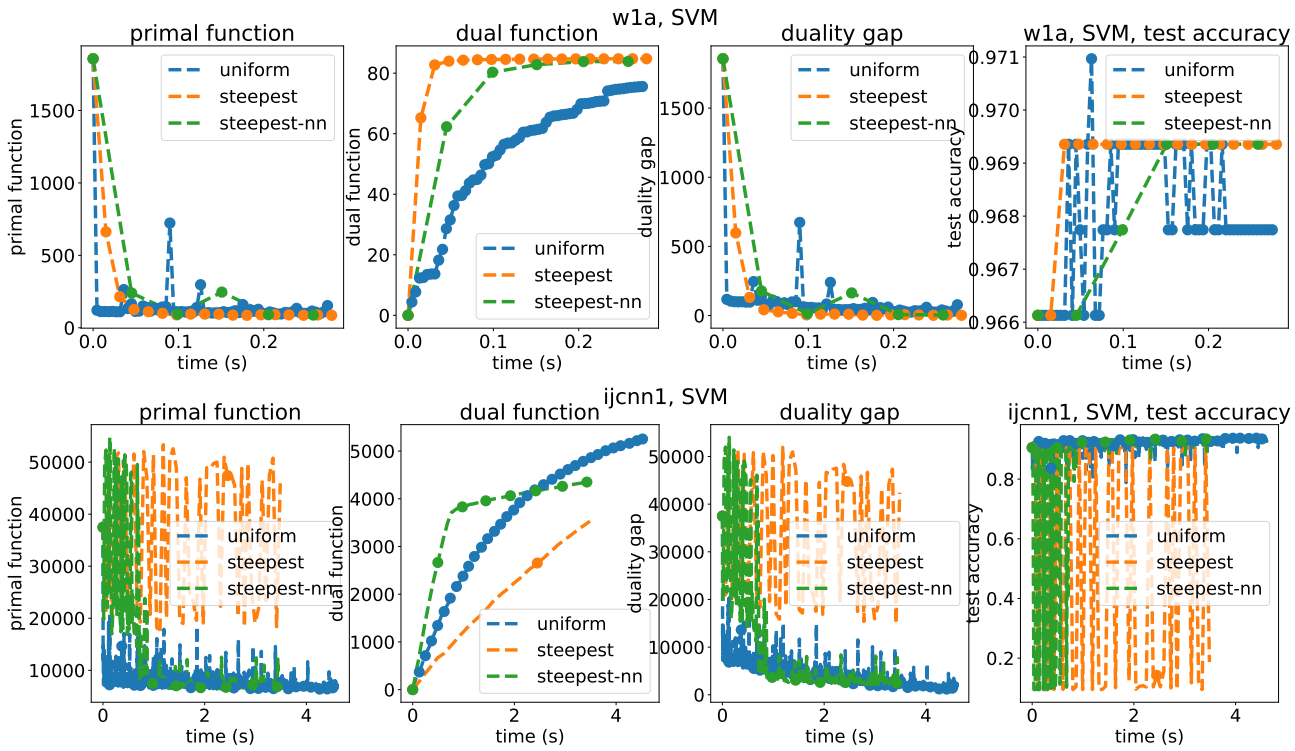

Figure 11: Primal function, dual function, duality gap and accuracy: Even measured against wall time, `steepest-nn` is very competitive with `uniform` in all the metrics.

G.6 Density of the solution

In Figure 12, we plot the number of non-zero coordinates of current solution as a function of time for datasets `sector`, `rcv1` for Lasso and `w1a`, `ijcnn1` for SVM. Recall from Figures 4 and 11 that `steepest-nn` is competitive (and sometimes much better) than `uniform` in terms of optimization error or accuracy especially at the start. Figure 12 shows that the solutions obtained by `steepest-nn` are much sparser at the start. So if we want a *quick*, sparse and accurate solution, then `steepest-nn` could be the algorithm of choice. For SVMs sparsity translates to a small set of support vectors.

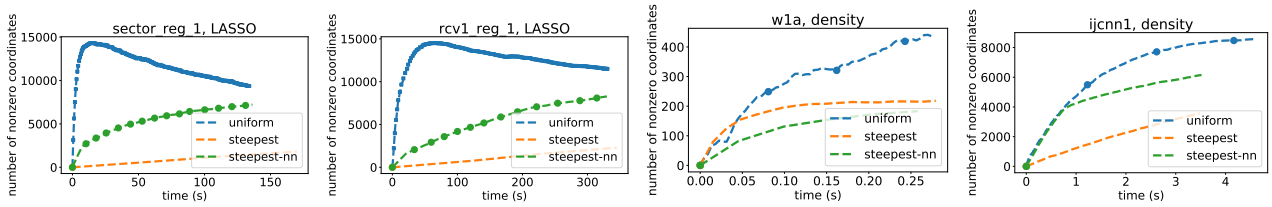

Figure 12: Density: The solutions obtained by `steepest-nn` are much sparser than `uniform`, especially towards the beginning.

G.7 Effect of Adaptivity

Here we investigate whether our adaptive choosing of the subsets has any adverse effect on the performance of `nmslib` to solve the MIPS problem. At each iteration during the course of our algorithm, we compute the value of dot product of the query vector with i) the optimal point over all candidates (MIPS), ii) the optimal point over only the subset of the candidates (S-MIPS), iii) result of running `nmslib` on all candidates (MIPS-nn), and vi) the result of running LSH on the masked subset (S-MIPS-nn). Comparing (i) and (iii) shows the performance of the `nmslib` algorithm, and comparing (ii) and (iv) shows how well the `nmslib` algorithm handles our adaptive queries. The results in Figure 13 show that indeed the influence of adaptivity is negligible - both (MIPS-nn) and (MIPS) are close to (S-MIPS-nn) and (S-MIPS) respectively. What is surprising though is the overall poor performance of the `nmslib` algorithm even after spending significant effort tuning the hyper-parameters as evidenced by the gap between (MIPS-nn) and (MIPS). This strongly suggests that improving the underlying algorithm for solving our S-MIPS instances could lead to significant gains.

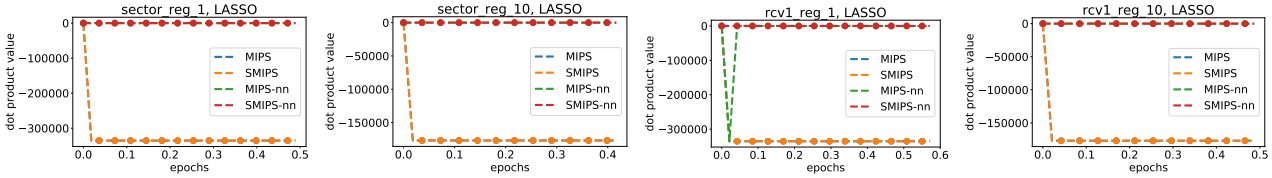

Figure 13: Adaptivity: (MIPS-nn) and (MIPS) are close to (S-MIPS-nn) and (S-MIPS) respectively indicating that choosing subsets adaptively does not affect `nmslib` algorithm substantially. However the gap between (MIPS-nn) and (MIPS) indicates the general poor quality of the solutions returned by `nmslib`.

G.8 GS-s implementation using `FALCONN` library

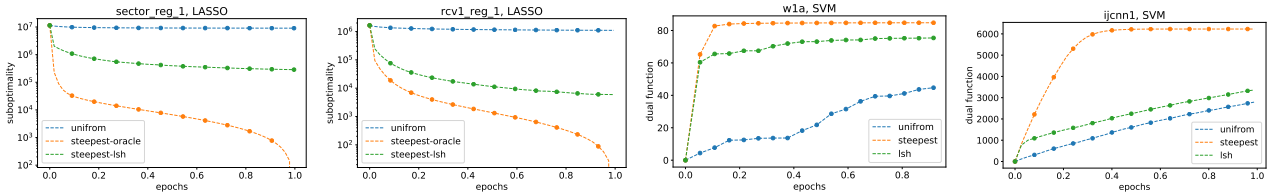

Figure 14: `steepest` as well `steepest-lsh` significantly outperform `uniform` in number of iterations.

We repeat experiments from section 7 using `FALCONN` library, which is an efficient implemenation of LSH. Used `FALCONN` hyper-parameters could be found in the Table 3. Figure 14 shows the superiority of `steepest-lsh` and `steepest` over `uniform` in terms of iterations. Figure 15 shows their wall-time comparison. Qualitatively, the behaviour of `steepest-lsh` is similar to the `nmslib` results, whereas quantitive results are better using `nmslib` library for the SVM and a bit worse for the LASSO.

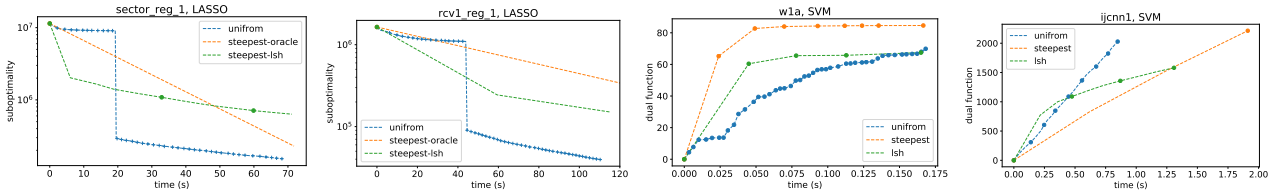

Figure 15: `steepest-lsh` is very competitive and sometimes outperforms `uniform` even in terms of wall time especially towards the beginning. However eventually the performance of `uniform` is better than `steepest-lsh`. This is because as the norm of the gradient becomes small, the *hyperplane* LSH algorithm used performs poorly.

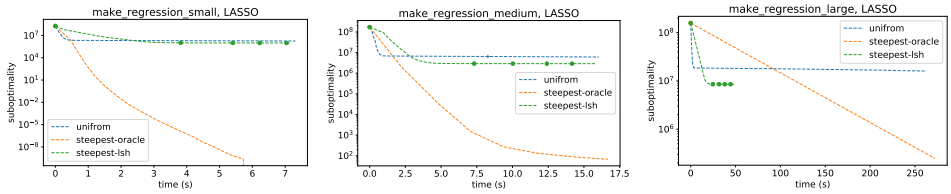

Figure 16: Performance on `make-regression` with $n \in \{10^4, 10^5, 10^6\}$. The advantage of `steepest-lsh` over `uniform` consistently increases with increasing problem sizes, eventually significantly outperforming it.

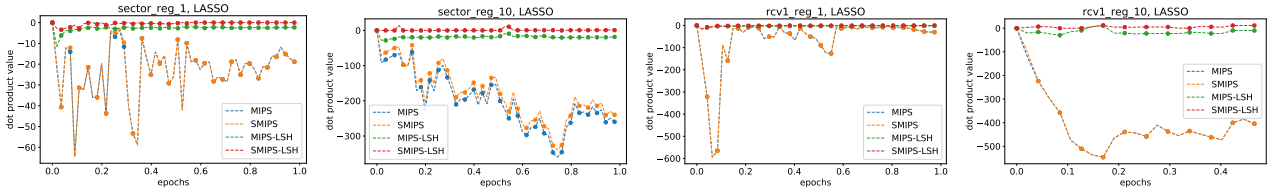

Figure 17: Adaptivity: (MIPS-LSH) and (MIPS) are close to (S-MIPS-LSH) and (S-MIPS) respectively indicating that choosing subsets adaptively does not affect LSH algorithm substantially. However the gap between (MIPS-LSH) and (MIPS) indicates the general poor quality of the solutions returned by LSH.

Table 3: Datasets and `FALCONN` hyper-parameters: Lasso is run on `rcv1` and `sector`, and SVM on `w1a` and `ijcnn1`. L is fixed at 10, and $k = \lfloor \log(n) - h \rfloor$.

Dataset	**n**	**d**	$\sqrt{n}\beta$	**m**	**h**
`sector`	55,197	6,412	3	10	1
`rcv1`	47,236	15,564	1	50	1
`w1a`	2,477	300	10	50	1
`ijcnn1`	49,990	22	3	10	0