Block cubic Newton with greedy selection

Andrea Cristofari*

*Department of Civil Engineering and Computer Science Engineering
University of Rome "Tor Vergata"
Via del Politecnico, 1, 00133 Rome, Italy
E-mail: andrea.cristofari@uniroma2.it

Abstract. A second-order block coordinate descent method is proposed for the unconstrained minimization of an objective function with a Lipschitz continuous Hessian. At each iteration, a block of variables is selected by means of a greedy (Gauss-Southwell) rule which considers the amount of first-order stationarity violation, then an approximate minimizer of a cubic model is computed for the block update. In the proposed scheme, blocks are not required to have a predetermined structure and their size may change during the iterations. For non-convex objective functions, global convergence to stationary points is proved and a worst-case iteration complexity analysis is provided. In particular, given a tolerance ϵ, we show that at most $\mathcal{O}(\epsilon^{-3/2})$ iterations are needed to drive the stationarity violation with respect to a selected block of variables below ϵ, while at most $\mathcal{O}(\epsilon^{-2})$ iterations are needed to drive the stationarity violation with respect to all variables below ϵ. Numerical results are finally given, comparing the proposed approach with other second-order methods and block selection rules.

Keywords. Block coordinate descent. Cubic Newton methods. Second-order methods. Worst-case iteration complexity.

1 Introduction

Many challenging problems require the minimization of an objective function with several variables. In this respect, block coordinate descent methods often represent an advantageous approach, especially when the objective function has a nice structure, since these methods update a block of variables at each iteration and may have a low per-iteration cost. In the literature, block coordinate descent methods have been extensively analyzed in several forms, employing different rules to choose and update the blocks.

In particular, we can recognize three main block selection rules [30, 37]: besides *random* rules which choose blocks randomly, we have *cyclic* (Gauss-Seidel) rules, which choose blocks by requiring that each variable is selected at least once within any window of a pre-specified number of iterations, and *greedy* (Gauss-Southwell) rules, which select the best block in terms of first-order stationarity violation at each iteration. It is well known that greedy rules, compared to cyclic and random rules, on the one hand require an additional effort to compute the whole gradient of the objective function, but on the other hand tend to make

a major progress per iteration [30]. Generally speaking, which block selection rule is the best in practice strongly depends on the features of the problems. For first-order methods, theoretical and numerical comparisons of the different approaches can be found in, e.g., [13, 29, 35], showing that a greedy rule can be more efficient for some classes of problems with a certain structure. In more detail, as discussed in [29] for first-order coordinate descent methods, cyclic and random rules work well when the cost of performing n coordinate updates is similar to the cost of performing one full gradient iteration, while a greedy rule can be efficiently used in other cases, e.g., when the objective function involves the multiplication of a sparse matrix by a vector.

Most block coordinate descent methods use first-order information and gained great popularity as they guarantee high efficiency in several applications. However, when the objective function is twice continuously differentiable, second-order information can be conveniently used as well, in order to speed up the convergence of the algorithm and overcome some drawbacks connected with first-order methods, such as the performance deterioration in ill-conditioned or highly non-separable problems [19]. Of course, second-order information should be used judiciously in a block coordinate descent scheme, so as not to increase the per-iteration cost excessively. A possibility is extending, to a block coordinate descent setting, ideas from *cubic Newton methods* [7, 8, 16, 17, 21, 22, 28], where, at each iteration, the next point is obtained by minimizing a cubic model, that is, a second-order model with cubic regularization (higher order models may also be considered when the objective function is several times continuously differentiable [5, 9]).

In recent years, block coordinate descent versions of cubic Newton methods were proposed in the literature using different block selection rules. In particular, cyclic-type block selection was considered in [1] for high order models which include cubic models as a special case, whereas random block selection was analyzed in [14, 23] and [38] for convex and non-convex objective functions, respectively.

To the best of the author's knowledge, greedy rules have not been investigated yet for block coordinate descent versions of cubic Newton methods. The present paper aims to fill the gap by giving a detailed analysis of this scheme from both a theoretical and a numerical perspective. Let us also highlight that, when using blocks made of q variables, the number of second-order partial derivatives is of the order of q^2, so the cost of computing the full gradient of the objective function may not be dominant. Hence, in a second-order setting, the computational disadvantage of a greedy rule over random and cyclic rules may reduce in some cases. However, as highlighted above, this is strongly related to the structure of the problem.

1.1 Main contributions

For the proposed block cubic Newton method with greedy selection, we provide the following worst-case iteration complexity bounds to minimize a non-convex objective function with Lipschitz continuous Hessian:

- at most $\mathcal{O}(\epsilon^{-3/2})$ iterations are needed to drive the stationarity violation with respect to *a selected block of variables* below ϵ,

Table 1: Worst-case iteration complexity of cyclic and greedy selection when using gradient-type and cubic Newton updates on non-convex objective functions.

Block update	Block selection	Worst-case iteration complexity	Reference
gradient (1st order)	cyclic	$\mathcal{O}(\epsilon^{-2})$	[2]
	greedy	$\mathcal{O}(\epsilon^{-2})$	[29]
cubic Newton (2nd order)	cyclic	$\mathcal{O}(\epsilon^{-3/2})$ for selected block	[1]
		$\mathcal{O}(\epsilon^{-3})$ for all variables	[1]
	greedy	$\mathcal{O}(\epsilon^{-3/2})$ for selected block	this paper
		$\mathcal{O}(\epsilon^{-2})$ for all variables	this paper

- at most $\mathcal{O}(\epsilon^{-2})$ iterations are needed to drive the stationarity violation with respect to *all variables* below ϵ.

Our results are appealing if compared to those given in [1] for cyclic-type block selection when using cubic models. Specifically, the former complexity bound of $\mathcal{O}(\epsilon^{-3/2})$ over a selected block of variables was obtained in [1] as well, but note that the latter complexity bound of $\mathcal{O}(\epsilon^{-2})$ over all variables improves upon the one given in [1], which is $\mathcal{O}(\epsilon^{-3})$.

Note also that first-order block coordinate descent methods need at most $\mathcal{O}(\epsilon^{-2})$ iterations to reduce the stationarity violation over all variables below ϵ using either cyclic [2] or greedy [29] block selection, since they both guarantee an objective decrease proportional to the squared norm of the gradient of the objective function [2, 29].

These results for cyclic and greedy selection in first- and second-order settings are summarized in Table 1.

Hence, when using cubic Newton methods in a block coordinate descent setting, the literature indicates that the worst-case iteration complexity of cyclic selection to reduce the stationarity violation over all variables below ϵ is worse than that of first-order methods, namely $\mathcal{O}(\epsilon^{-3})$ versus $\mathcal{O}(\epsilon^{-2})$. In this fashion, the proposed greedy selection recovers the $\mathcal{O}(\epsilon^{-2})$ complexity of first-order methods, while preserving the same efficiency of the cyclic selection in reducing the stationarity violation over a chosen block below ϵ within $\mathcal{O}(\epsilon^{-3/2})$ iterations.

Let us remark that, for the proposed method, we do not need to know the Lipschitz constant of the Hessian of the objective function. Moreover, we use inexact minimizers of the cubic model whose computation does not require additional evaluations of the objective function or its derivatives. Due to the block structure and the use of inexact information, we name our algorithm *Inexact Block Cubic Newton* (IBCN) method.

We also assess the practical performance of the proposed IBCN method on non-convex and convex problems. In particular, we first demonstrate its efficiency compared with other block updates, and then show that the proposed scheme can outperform cyclic and random variants when their per-iteration cost is similar to that of the greedy selection.

Furthermore, the code is freely available at `https://github.com/acristofari/ibcn`.

The rest of the paper is organized as follows. In Section 2, we introduce the problem and give preliminary results. In Section 3, we describe the proposed method. In Section 4,

we carry out the convergence analysis and give worst-case iteration complexity bounds. In Section 5, we show some numerical results. Finally, we draw some conclusions in Section 6.

2 Preliminaries and notations

We consider the following unconstrained optimization problem:

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f\colon \mathbb{R}^n \to \mathbb{R}$ is a (possibly non-convex) objective function. We assume that the Hessian matrix $\nabla^2 f(x)$ is Lipschitz continuous over $\mathbb{R}^n$ with constant $L > 0$, that is,

$$\|\nabla^2 f(x) - \nabla^2 f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^n,$$

where, here and in the rest of the paper, $\|v\|$ is the Euclidean norm for any vector v, whereas $\|A\|$ is the norm induced by the vector Euclidean norm for any matrix A. The sup-norm of a vector v is indicated by $\|v\|_\infty$.

Given $\mathcal{I} \subseteq \{1, \ldots, n\}$, we denote by $U_\mathcal{I} \in \mathbb{R}^{n\times|\mathcal{I}|}$ the submatrix of the n-dimensional identity matrix obtained by removing all columns with indices not belonging to $\mathcal{I}$. Then, given $x \in \mathbb{R}^n$ and $\mathcal{I} \subseteq \{1, \ldots, n\}$, we use the following notation:

- $x_\mathcal{I} \in \mathbb{R}^{|\mathcal{I}|}$ is the subvector of x with elements in $\mathcal{I}$, that is,
$$x_\mathcal{I} = U_\mathcal{I}^T x;$$
- $\nabla_\mathcal{I} f(x) \in \mathbb{R}^{|\mathcal{I}|}$ is the vector of first-order partial derivatives of f with respect to x_i, $i \in \mathcal{I}$, that is,
$$\nabla_\mathcal{I} f(x) = U_\mathcal{I}^T \nabla f(x); \tag{2}$$
- $\nabla^2_\mathcal{I} f(x) \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ is the matrix of second-order partial derivatives of f with respect to x_i, $i \in \mathcal{I}$, that is,
$$\nabla^2_\mathcal{I} f(x) = U_\mathcal{I}^T \nabla^2 f(x) U_\mathcal{I}. \tag{3}$$

For example, if $n = 5$ and

$$x = \begin{bmatrix} 3 \\ 1 \\ 4 \\ -2 \\ 0 \end{bmatrix}, \quad \nabla f(x) = \begin{bmatrix} 2 \\ -1 \\ 0 \\ -3 \\ 4 \end{bmatrix}, \quad \nabla^2 f(x) = \begin{bmatrix} -2 & 3 & -6 & 0 & -7 \\ 3 & 1 & -5 & 4 & 2 \\ -6 & -5 & 7 & -3 & -1 \\ 0 & 4 & -3 & 5 & -4 \\ -7 & 2 & -1 & -4 & 6 \end{bmatrix},$$

using $\mathcal{I} = \{1, 3, 4\}$ we get

$$U_\mathcal{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad x_\mathcal{I} = \begin{bmatrix} 3 \\ 4 \\ -2 \end{bmatrix}, \quad \nabla_\mathcal{I} f(x) = \begin{bmatrix} 2 \\ 0 \\ -3 \end{bmatrix}, \quad \nabla^2_\mathcal{I} f(x) = \begin{bmatrix} -2 & -6 & 0 \\ -6 & 7 & -3 \\ 0 & -3 & 5 \end{bmatrix}.$$

Note that, for any choice of $\mathcal{I} \subseteq \{1, \dots, n\}$, we have

$$\|U_{\mathcal{I}}\| = 1, \tag{4}$$

$$\|U_{\mathcal{I}} v\| = \|v\| \quad \forall v \in \mathbb{R}^{|\mathcal{I}|}. \tag{5}$$

Moreover, for any choice of $\mathcal{I} \subseteq \{1, \dots, n\}$, we define the block Lipschitz constant $L_{\mathcal{I}}$ such that

$$\|\nabla^2_{\mathcal{I}} f(x + U_{\mathcal{I}} s) - \nabla^2_{\mathcal{I}} f(x)\| \leq L_{\mathcal{I}} \|s\| \quad \forall x \in \mathbb{R}^n, \forall s \in \mathbb{R}^{|\mathcal{I}|}. \tag{6}$$

Note that

$$L_{\mathcal{I}} \in (0, L] \tag{7}$$

since, recalling (3), we have

$$\begin{aligned} \|\nabla^2_{\mathcal{I}} f(x + U_{\mathcal{I}} s) - \nabla^2_{\mathcal{I}} f(x)\| &= \|U_{\mathcal{I}}^T (\nabla^2 f(x + U_{\mathcal{I}} s) - \nabla^2 f(x)) U_{\mathcal{I}}\| \\ &\leq \|\nabla^2 f(x + U_{\mathcal{I}} s) - \nabla^2 f(x)\| \|U_{\mathcal{I}}\|^2 \\ &= \|\nabla^2 f(x + U_{\mathcal{I}} s) - \nabla^2 f(x)\| \leq L \|U_{\mathcal{I}} s\| = L \|s\|, \end{aligned}$$

where (4) has been used in the second equality and (5) has been used in the last equality.

Let us also define

$$L^{\min} = \min_{\mathcal{I} \subseteq \{1, \dots, n\}} L_{\mathcal{I}}.$$

From (7), it follows that

$$0 < L^{\min} \leq L_{\mathcal{I}} \leq L \quad \forall \mathcal{I} \subseteq \{1, \dots, n\}. \tag{8}$$

Extending known results on functions with Lipschitz continuous Hessian [12, 28], we can give the following proposition whose proof is reported in Appendix A.

Proposition 1. *Given a point $x \in \mathbb{R}^n$ and a block of variable indices $\mathcal{I} \subseteq \{1, \dots, n\}$, for all $s \in \mathbb{R}^{|\mathcal{I}|}$ we have that*

$$\|\nabla_{\mathcal{I}} f(x + U_{\mathcal{I}} s) - \nabla_{\mathcal{I}} f(x) - \nabla^2_{\mathcal{I}} f(x) s\| \leq \frac{L_{\mathcal{I}}}{2} \|s\|^2, \tag{9}$$

$$|f(x + U_{\mathcal{I}} s) - f(x) - \nabla_{\mathcal{I}} f(x)^T s - \frac{1}{2} s^T \nabla^2_{\mathcal{I}} f(x) s| \leq \frac{L_{\mathcal{I}}}{6} \|s\|^3. \tag{10}$$

3 The Inexact Block Cubic Newton (IBCN) method

In this section, we describe the proposed algorithm, named *Inexact Block Cubic Newton* (IBCN) method, which is reported in Algorithm 1. As will be shown, it is a natural extension of classical cubic Newton methods to a block coordinate descent scheme.

3.1 Block selection

At the beginning of iteration k, we choose a block of variable indices $\mathcal{I}_k \subseteq \{1,\dots,n\}$ by means of a classical greedy (or Gauss-Southwell) strategy [30], that is, $\mathcal{I}_k$ must include variables providing a sufficiently large amount of first-order stationarity violation. The rule can be stated as follows.

Greedy selection rule: There exists a real number $\theta \in (0,1]$ such that

$$\|\nabla_{\mathcal{I}_k} f(x_k)\| \geq \theta \|\nabla f(x_k)\| \quad \forall k \geq 0. \tag{11}$$

Note that the above greedy selection rule does not require the variables to be a priori partitioned into a fixed number of blocks, so that even the size of blocks may change during the iterations.

In the following two propositions, we describe two simple procedures to satisfy (11). For any iteration k, the first one requires $\mathcal{I}_k$ to include the variable corresponding to the largest component in absolute value of $\nabla f(x_k)$, while the second one, given an arbitrary number of (possibly overlapping) blocks of variables covering $\{1,\dots,n\}$, requires to compute the norm of the subvectors of $\nabla f(x_k)$ with respect to each block in order to choose $\mathcal{I}_k$ as the one yielding the largest norm.

Proposition 2. *For every iteration k, let $\hat{\imath}_k \in \operatorname{Argmax}_{i=1,\dots,n} |\nabla_i f(x_k)|$ and assume that $\hat{\imath}_k \in \mathcal{I}_k$. Then,*

$$\|\nabla_{\mathcal{I}_k} f(x_k)\| \geq (n+1-|\mathcal{I}_k|)^{-1/2} \|\nabla f(x_k)\| \quad \forall k \geq 0.$$

It follows that (11) *is satisfied with*

$$\theta = \left(n+1-\min_{k\geq 0} |\mathcal{I}_k|\right)^{-1/2}.$$

Proof. Fix any iteration k and let

$$\tilde{\mathcal{I}}_k = (\{1,\dots,n\} \setminus \mathcal{I}_k) \cup \{\hat{\imath}_k\}.$$

Recalling the definition of $\hat{\imath}_k$, we have that

$$\|\nabla_{\mathcal{I}_k} f(x_k)\|_\infty = \|\nabla_{\tilde{\mathcal{I}}_k} f(x_k)\|_\infty = \|\nabla f(x_k)\|_\infty = |\nabla_{\hat{\imath}_k} f(x_k)|.$$

Then, we can write

$$\begin{aligned}
\|\nabla_{\mathcal{I}_k} f(x_k)\|^2 &= (\nabla_{\hat{\imath}_k} f(x_k))^2 + \sum_{i \in \mathcal{I}_k \setminus \{\hat{\imath}_k\}} \nabla_i f(x_k)^2 \\
&= \|\nabla_{\tilde{\mathcal{I}}_k} f(x_k)\|_\infty^2 + \sum_{i \in \{1,\dots,n\} \setminus \tilde{\mathcal{I}}_k} \nabla_i f(x_k)^2 \\
&\geq |\tilde{\mathcal{I}}_k|^{-1} \left(\|\nabla_{\tilde{\mathcal{I}}_k} f(x_k)\|^2 + \sum_{i \in \{1,\dots,n\} \setminus \tilde{\mathcal{I}}_k} \nabla_i f(x_k)^2 \right) \\
&= |\tilde{\mathcal{I}}_k|^{-1} \|\nabla f(x_k)\|^2.
\end{aligned}$$

Since $|\tilde{\mathcal{I}}_k| = n+1-|\mathcal{I}_k|$, then the desired result follows. □

Proposition 3. *For every iteration k, let $\mathcal{J}_k^1, \dots, \mathcal{J}_k^{N_k}$ be subsets of $\{1, \dots, n\}$ such that $\bigcup_{j=1}^{N_k} \mathcal{J}_k^j = \{1, \dots, n\}$ and assume that $\mathcal{I}_k \in \operatorname{Argmax}_{\mathcal{I}=\mathcal{J}_k^1,\dots,\mathcal{J}_k^{N_k}} \|\nabla_{\mathcal{I}} f(x_k)\|$. Then,*

$$\|\nabla_{\mathcal{I}_k} f(x_k)\| \geq N_k^{-1/2} \|\nabla f(x_k)\| \quad \forall k \geq 0.$$

It follows that (11) *is satisfied with*

$$\theta = \min_{k \geq 0} N_k^{-1/2}.$$

Proof. Fix any iteration k. Since $\bigcup_{j=1}^{N_k} \mathcal{J}_k^j = \{1, \dots, n\}$, we can write

$$\|\nabla f(x_k)\|^2 \leq \sum_{j=1}^{N_k} \|\nabla_{\mathcal{J}_k^j} f(x_k)\|^2 \leq N_k \|\nabla_{\mathcal{I}_k} f(x_k)\|^2,$$

where the last inequality follows from how $\mathcal{I}_k$ is selected, thus leading to the desired result. □

3.2 Cubic model

At iteration k, in order to update the variables in $\mathcal{I}_k$, we search for a suitable $s_k \in \mathbb{R}^{|\mathcal{I}_k|}$ to move from x_k along $U_{\mathcal{I}_k} s_k$. To this aim, we define the cubic model $m_k(s)$ as follows:

$$m_k(s) = q_k(s) + \frac{\sigma_k}{6} \|s\|^3, \tag{12}$$

where σ_k is a positive scalar, updated during the iterations, which should overestimate $L_{\mathcal{I}_k}$, while $q_k(s)$ is the following quadratic model:

$$q_k(s) = f(x_k) + \nabla_{\mathcal{I}_k} f(x_k)^T s + \frac{1}{2} s^T \nabla^2_{\mathcal{I}_k} f(x_k) s. \tag{13}$$

For the sake of convenience, let us also report the gradient of $m_k(s)$ as follows:

$$\nabla m_k(s) = \nabla_{\mathcal{I}_k} f(x_k) + \nabla^2_{\mathcal{I}_k} f(x_k) s + \frac{\sigma_k}{2} \|s\| s. \tag{14}$$

3.3 Approximate minimizers of the cubic model

At iteration k, we compute s_k as an approximate minimizer of the cubic model (12). In particular, assuming that $\|\nabla_{\mathcal{I}_k} f(x_k)\| \neq 0$, we require s_k to satisfy two conditions. The first one is that the first-order stationarity violation must be sufficiently small compared to $\|s_k\|^2$, that is,

$$\|\nabla m_k(s_k)\| \leq \tau \|s_k\|^2, \tag{15}$$

with a given $\tau \in [0, \infty)$. The second requirement is that $m_k(s_k)$ must be sufficiently low, that is,

$$\begin{aligned} &m_k(s_k) \leq m_k(\hat{s}_k), \quad \text{where} \\ &\hat{s}_k = -\hat{\alpha}_k \nabla_{\mathcal{I}_k} f(x_k) \quad \text{and} \quad \hat{\alpha}_k = \min\left\{ \frac{\beta}{\|\nabla^2_{\mathcal{I}_k} f(x_k)\|}, \sqrt{\frac{3\beta}{\sigma_k \|\nabla_{\mathcal{I}_k} f(x_k)\|}} \right\}, \end{aligned} \tag{16}$$

with a given $\beta \in (0,1)$, letting the first argument within the above minimum to be $+\infty$ when $\|\nabla^2_{\mathcal{I}_k} f(x_k)\| = 0$.

We see that condition (15) is a straightforward adaptation of those used in [1, 5, 9], while condition (16) is inspired by the classical Cauchy condition [7] which requires $m_k(s_k) \leq \min_{\alpha \geq 0} m_k(-\alpha \nabla_{\mathcal{I}_k} f(x_k))$. In our case, $m_k(s_k)$ is compared to $m_k(\hat{s}_k)$, hence (16) is weaker than the Cauchy condition.

In more detail, the role of the two conditions above in the convergence analysis can be outlined as follows:

- Condition (15) allows us to upper bound $\|\nabla_{\mathcal{I}_k} f(x_{k+1})\|$ by a term proportional to $\|s_k\|^2$ at each iteration k (see Proposition 7 below), which is needed for achieving a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-3/2})$ to reduce the stationarity violation over a selected block of variables below ϵ (see Theorem 13 below).

- Condition (16) allows us to lower bound the decrease of both the cubic and quadratic models by a term depending on $\|\nabla_{\mathcal{I}_k} f(x_k)\|^2$ at each iteration k (see Proposition 4 and Lemma 5 below). This implies a lower bound on the objective decrease of the same order at certain iterations (see Proposition 6 below), leading to a worst-case iteration complexity of $\mathcal{O}(\epsilon^{-2})$ to reduce the stationarity violation over all variables below ϵ, thanks to the greedy rule used for the block selection (see Theorem 14 below).

Note that we can compute a vector s_k satisfying (15)–(16) in finite time without the need of additional evaluations of f or its derivatives in other points. In particular, we can apply an algorithm to approximately minimize the cubic model (12) (using, e.g., the methods analyzed in [4, 6, 20, 27]). In our experiments, for the inexact minimization of the cubic model (12), we use a Barzilai-Borwein gradient method [32], which was observed to be effective in practice [4].

Also, observe that (15)–(16) are clearly satisfied if s_k is a global minimizer of the cubic model (12) (details on how to compute global minimizers of a cubic model can be found in [7, 11, 28]).

3.4 Block update

Once s_k has been computed at iteration k, we use standard updating rules inherited from trust-region methods to decide how to set x_{k+1} and σ_{k+1} for the next iteration (see, e.g., [5, 7]). Namely, we compute

$$\rho_k = \frac{f(x_k) - f(x_k + U_{\mathcal{I}_k} s_k)}{q_k(0) - q_k(s_k)} \tag{17}$$

(we will show in Lemma 5 below that the denominator is positive whenever $\nabla_{\mathcal{I}_k} f(x_k) \neq 0$) and check if ρ_k is sufficiently large. In more detail, using $0 < \eta_1 \leq \eta_2 < 1$ and $0 < \gamma_1 \leq 1 < \gamma_2 \leq \gamma_3$, we set

$$x_{k+1} = \begin{cases} x_k + U_{\mathcal{I}_k} s_k & \text{if } \rho_k \geq \eta_1, \\ x_k & \text{otherwise,} \end{cases} \tag{18}$$

and

$$\sigma_{k+1} \in \begin{cases} [\max\{\sigma^{\min}, \gamma_1\sigma_k\}, \sigma_k] & \text{if } \rho_k \geq \eta_2, \\ [\sigma_k, \gamma_2\sigma_k] & \text{if } \rho_k \in [\eta_1, \eta_2), \\ [\gamma_2\sigma_k, \gamma_3\sigma_k] & \text{if } \rho_k < \eta_1, \end{cases} \tag{19}$$

with $\sigma^{\min} \in (0, \infty)$.

We also say that k is a *successful* iteration if $x_{k+1} = x_k + U_{\mathcal{I}_k} s_k$, denoting by $\mathcal{S}$ the set of all successful iterations. Namely, recalling (18), we have

$$\mathcal{S} := \{k \text{ such that } \rho_k \geq \eta_1\}. \tag{20}$$

Algorithm 1 Inexact Block Cubic Newton (IBCN) method

1: Given $x_0 \in \mathbb{R}^n$, $\sigma^{\min} > 0$, $\sigma_0 \in [\sigma^{\min}, \infty)$, $0 < \eta_1 \leq \eta_2 < 1$, $0 < \gamma_1 \leq 1 < \gamma_2 \leq \gamma_3$, $\tau \in [0, \infty)$ and $\beta \in (0, 1)$

2: **while** $\nabla f(x_k) \neq 0$ **do**

3: compute $\mathcal{I}_k \subseteq \{1, \ldots, n\}$ such that $\|\nabla_{\mathcal{I}_k} f(x_k)\| \geq \theta \|\nabla f(x_k)\|$

4: compute s_k such that

$$\|\nabla m_k(s_k)\| \leq \tau \|s_k\|^2 \quad \text{and} \quad m_k(s_k) \leq m_k(\hat{s}_k),$$

where

$$\hat{s}_k = -\hat{\alpha}_k \nabla_{\mathcal{I}_k} f(x_k),$$

$$\hat{\alpha}_k = \min\left\{ \frac{\beta}{\|\nabla^2_{\mathcal{I}_k} f(x_k)\|}, \sqrt{\frac{3\beta}{\sigma_k \|\nabla_{\mathcal{I}_k} f(x_k)\|}} \right\}$$

5: compute $\rho_k = \dfrac{f(x_k) - f(x_k + U_{\mathcal{I}_k} s_k)}{q_k(0) - q_k(s_k)}$

6: set $x_{k+1} = \begin{cases} x_k + U_{\mathcal{I}_k} s_k & \text{if } \rho_k \geq \eta_1 \\ x_k & \text{otherwise} \end{cases}$

7: set $\sigma_{k+1} \in \begin{cases} [\max\{\sigma^{\min}, \gamma_1\sigma_k\}, \sigma_k] & \text{if } \rho_k \geq \eta_2 \\ [\sigma_k, \gamma_2\sigma_k] & \text{if } \rho_k \in [\eta_1, \eta_2) \\ [\gamma_2\sigma_k, \gamma_3\sigma_k] & \text{if } \rho_k < \eta_1 \end{cases}$

8: **end while**

4 Convergence analysis

For the proposed IBCN method, we first establish global convergence to stationary points in Subsection 4.1, and then present worst-case iteration complexity results in Subsection 4.2.

4.1 Global convergence

To begin with, for each iteration we can provide a lower bound on the decrease of the cubic model similarly to when using the Cauchy step [7].

Proposition 4. *For every iteration k, we have*

$$m_k(0) - m_k(s_k) \geq m_k(0) - m_k(\hat{s}_k) \geq (1-\beta)\hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\|^2,$$

where $\hat{s}_k$ is defined as in (16).

Proof. The first inequality of the thesis follows from (16), so we only have to show the second inequality. To this aim, recalling the definitions of m_k and ∇m_k from (12) and (14), respectively, we can write

$$\begin{aligned} m_k(0) - m_k(\hat{s}_k) &= f(x_k) - m_k(-\hat{\alpha}_k \nabla_{\mathcal{I}_k} f(x_k)) \\ &= \hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\|^2 - \frac{\hat{\alpha}_k^2}{2} \nabla_{\mathcal{I}_k} f(x_k)^T \nabla^2_{\mathcal{I}_k} f(x_k) \nabla_{\mathcal{I}_k} f(x_k) + \\ &\quad - \frac{\hat{\alpha}_k^3 \sigma_k}{6} \|\nabla_{\mathcal{I}_k} f(x_k)\|^3 \\ &\geq \hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\|^2 \Big(1 - \frac{\hat{\alpha}_k \|\nabla^2_{\mathcal{I}_k} f(x_k)\|}{2} - \frac{\hat{\alpha}_k^2 \sigma_k \|\nabla_{\mathcal{I}_k} f(x_k)\|}{6}\Big). \end{aligned}$$

From the definition of $\hat{\alpha}_k$ given in (16), it follows that

$$1 - \frac{\hat{\alpha}_k \|\nabla^2_{\mathcal{I}_k} f(x_k)\|}{2} - \frac{\hat{\alpha}_k^2 \sigma_k \|\nabla_{\mathcal{I}_k} f(x_k)\|}{6} \geq 1 - \frac{\beta}{2} - \frac{\beta}{2} = 1 - \beta.$$

Then, the desired result follows. □

Using the above proposition, we can easily lower bound the decrease of the quadratic model at every iteration as follows.

Lemma 5. *For every iteration k, we have*

$$q_k(0) - q_k(s_k) \geq (1-\beta)\hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\|^2 + \frac{\sigma_k}{6}\|s_k\|^3.$$

Proof. For any iteration k, from (13) and (12) it follows that

$$q_k(0) - q_k(s_k) = m_k(0) - m_k(s_k) + \frac{\sigma_k}{6}\|s_k\|^3.$$

Hence, the desired result is obtained by using Proposition 4. □

In the following two propositions we show how, for every successful iteration k, we can lower bound $(f(x_k) - f(x_{k+1}))$ and upper bound $\|\nabla_{\mathcal{I}_k} f(x_{k+1})\|$.

Proposition 6. *For every iteration $k \in \mathcal{S}$, we have*

$$f(x_k) - f(x_{k+1}) \geq \eta_1 \left((1-\beta)\hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\|^2 + \frac{\sigma_k}{6}\|s_k\|^3 \right).$$

Proof. Take any $k \in \mathcal{S}$. From the instructions of the algorithm and the definition of $\mathcal{S}$ given in (20), we have that $\rho_k \geq \eta_1$ and $x_{k+1} = x_k + U_{\mathcal{I}_k} s_k$. Recalling the definition of ρ_k given in (17), then the desired result follows from Lemma 5. □

Remark 1. *According to the instructions of the algorithm and the definition of $\mathcal{S}$ given in (20), we have that $x_{k+1} = x_k$ when $k \notin \mathcal{S}$. Then, using Proposition 6, it follows that the sequence $\{f(x_k)\}$ is monotonically non-increasing.*

Proposition 7. *For every iteration $k \in \mathcal{S}$, we have*

$$\|\nabla_{\mathcal{I}_k} f(x_{k+1})\| \leq \Big(\tau + \frac{\sigma_k + L_{\mathcal{I}_k}}{2}\Big)\|s_k\|^2.$$

Proof. Take any $k \in \mathcal{S}$. First, we can write

$$\|\nabla_{\mathcal{I}_k} f(x_{k+1})\| \leq \|\nabla_{\mathcal{I}_k} f(x_k) + \nabla^2_{\mathcal{I}_k} f(x_k)s_k\| + \|\nabla_{\mathcal{I}_k} f(x_{k+1}) - \nabla_{\mathcal{I}_k} f(x_k) - \nabla^2_{\mathcal{I}_k} f(x_k)s_k\|. \quad (21)$$

Using (14), we can upper bound the first norm in the right-hand side of (21) as follows:

$$\begin{aligned} \|\nabla_{\mathcal{I}_k} f(x_k) + \nabla^2_{\mathcal{I}_k} f(x_k)s_k\| &= \Big\|\nabla m_k(s_k) - \frac{\sigma_k}{2}\|s_k\|s_k\Big\| \\ &\leq \|\nabla m_k(s_k)\| + \frac{\sigma_k}{2}\|s_k\|^2 \\ &\leq \Big(\tau + \frac{\sigma_k}{2}\Big)\|s_k\|^2, \end{aligned} \quad (22)$$

where the last inequality follows from (15). From the instructions of the algorithm and the definition of $\mathcal{S}$ given in (20), we have that $x_{k+1} = x_k + U_{\mathcal{I}_k} s_k$. So, using (9), we can also upper bound the second norm in the right-hand side of (21) as follows:

$$\|\nabla_{\mathcal{I}_k} f(x_{k+1}) - \nabla_{\mathcal{I}_k} f(x_k) - \nabla^2_{\mathcal{I}_k} f(x_k)s_k\| \leq \frac{L_{\mathcal{I}_k}}{2}\|s_k\|^2. \quad (23)$$

Then, the desired result follows from (21), (22) and (23). □

Now, we want to relate the total number of iterations to the successful ones, which will be obtained in Proposition 10 below. To get such a result, we have to pass through a few intermediate steps. First we show that $\sigma_{k+1} \leq \sigma_k$ whenever σ_k is an appropriate overestimate of $L_{\mathcal{I}_k}$.

Proposition 8. *Assume that, for an iteration $k \geq 0$, we have*

$$\sigma_k \geq \frac{L_{\mathcal{I}_k}}{1 - \eta_2}.$$

Then, $\sigma_{k+1} \leq \sigma_k$.

Proof. From (13), we can write

$$-\nabla_{\mathcal{I}_k} f(x_k)^T s_k - \frac{1}{2} s_k^T \nabla^2_{\mathcal{I}_k} f(x_k)s_k = q_k(0) - q_k(s_k). \quad (24)$$

Using Lemma 5, it follows that

$$-\nabla_{\mathcal{I}_k} f(x_k)^T s_k - \frac{1}{2} s_k^T \nabla^2_{\mathcal{I}_k} f(x_k)s_k \geq \frac{\sigma_k}{6}\|s_k\|^3. \quad (25)$$

Using (24) and (10), from the definition of ρ_k given in (17) we obtain

$$1-\rho_k = \frac{-\nabla_{\mathcal{I}_k} f(x_k)^T s_k - \frac{1}{2} s_k^T \nabla^2_{\mathcal{I}_k} f(x_k) s_k - f(x_k) + f(x_k + U_{\mathcal{I}_k} s_k)}{-\nabla_{\mathcal{I}_k} f(x_k)^T s_k - \frac{1}{2} s_k^T \nabla^2_{\mathcal{I}_k} f(x_k) s_k} \leq \frac{L_{\mathcal{I}_k}}{\sigma_k},$$

where, in the last inequality, we have used (10) and (25) to upper bound the numerator by $(L_{\mathcal{I}_k}/6)\|s_k\|^3$ and lower bound the denominator by $(\sigma_k/6)\|s_k\|^3$, respectively. Since $\sigma_k \geq L_{\mathcal{I}_k}/(1-\eta_2)$ by hypothesis, it follows that $1-\rho_k \leq 1-\eta_2$, that is, $\rho_k \geq \eta_2$. Then the desired result follows from the updating rule of σ_k given in (19). □

In the following proposition, we show that σ_k stays bounded during the iterations.

Proposition 9. *For every iteration $k \geq 0$, it holds that*

$$\sigma^{min} \leq \sigma_k \leq \sigma^{max},$$

where $\sigma^{max} := \max\left\{\sigma_0, \dfrac{\gamma_3 L}{1-\eta_2}\right\}$.

Proof. From Proposition 8 and the fact that, by the instructions of the algorithm, σ_{k+1} is increased from σ_k at most by a factor of $\gamma_3 > 1$, it follows that

$$\sigma_{k+1} \leq \max\left\{\sigma_k, \frac{\gamma_3 L_{\mathcal{I}_k}}{1-\eta_2}\right\} \quad \forall k \geq 0.$$

Applying this inequality recursively and upper bounding $L_{\mathcal{I}_k}$ by (8), we get the desired upper bound on σ_k, while the lower bound follows immediately from the instructions of the algorithm. □

We can now upper bound the total number of unsuccessful iterations up to the current iteration.

Proposition 10. *For every iteration $k \geq 0$, let $\mathcal{S}_k$ be the set of successful iterations up to k, that is, $\mathcal{S}_k := \mathcal{S} \cap \{i \leq k\}$. Then,*

$$k+1 \leq \left(1 + \frac{|\log \gamma_1|}{\log \gamma_2}\right)|\mathcal{S}_k| + \frac{1}{\log \gamma_2} \log\left(\frac{\sigma^{max}}{\sigma_0}\right)$$

Proof. It is identical to [5, Lemma 2.4], where the same rule is used to update σ_k during the iterations. □

To give worst-case iteration complexity bounds, we need an assumption on the boundedness of f and $\|\nabla^2 f\|$ over the following level set:

$$\mathcal{L}^0 = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}. \tag{26}$$

Assumption 1. *Two finite constants f^{min} and B exist such that, for all $x \in \mathcal{L}^0$, we have $f(x) \geq f^{min}$ and $\|\nabla^2 f(x)\| \leq B$, where $\mathcal{L}^0$ is defined as in* (26).

We see that Assumption 1 is satisfied if $\mathcal{L}^0$ is compact. Note also that, since $\{f(x_k)\}$ is monotonically non-increasing from Remark 1, then $\{x_k\} \subseteq \mathcal{L}^0$. It follows that, under Assumption 1, we have

$$f(x_k) \geq f^{\min} \quad \forall k \geq 0, \tag{27}$$

$$\|\nabla^2_{\mathcal{I}_k} f(x_k)\| \leq B \quad \forall k \geq 0. \tag{28}$$

Under Assumption 1, we can now relate the objective decrease at successful iterations to the amount of first-order stationarity violation.

Proposition 11. *Given $\epsilon \in [0,1]$, if Assumption 1 holds, then*

$$f(x_k) - f(x_{k+1}) \geq c_1 \epsilon^2 \quad \forall k \in \mathcal{S} \colon \|\nabla f(x_k)\| \geq \epsilon,$$

where

$$c_1 = \theta \eta_1 (1-\beta) \min\left\{ \frac{\theta\beta}{B}, \sqrt{\frac{3\theta\beta}{\sigma^{max}}} \right\}$$

and σ^{max} is given in Proposition 9.

Proof. Take any iteration $k \in \mathcal{S}$ such that $\|\nabla f(x_k)\| \geq \epsilon$, with $\epsilon \in [0,1]$. From the instructions of the algorithm and the definition of $\mathcal{S}$ given in (20), we have that $x_{k+1} = x_k + U_{\mathcal{I}_k} s_k$. Using Proposition 6 and the greedy selection rule (11), we have that

$$f(x_k) - f(x_{k+1}) \geq \eta_1 (1-\beta) \hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\|^2 \geq \theta \eta_1 (1-\beta) \hat{\alpha}_k \|\nabla f(x_k)\| \|\nabla_{\mathcal{I}_k} f(x_k)\|.$$

Since $\|\nabla f(x_k)\| \geq \epsilon$, we obtain

$$f(x_k) - f(x_{k+1}) \geq \theta \eta_1 (1-\beta) \hat{\alpha}_k \epsilon \|\nabla_{\mathcal{I}_k} f(x_k)\|. \tag{29}$$

Now, using the definition of $\hat{\alpha}_k$ given in (16), we can write

$$\hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\| = \min\left\{ \frac{\beta \|\nabla_{\mathcal{I}_k} f(x_k)\|}{\|\nabla^2_{\mathcal{I}_k} f(x_k)\|}, \sqrt{\frac{3\beta \|\nabla_{\mathcal{I}_k} f(x_k)\|}{\sigma_k}} \right\}.$$

Since, from (28) and Proposition 9, respectively, $\|\nabla^2_{\mathcal{I}_k} f(x_k)\| \leq B$ and $\sigma_k \leq \sigma^{\max}$, we get

$$\hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\| \geq \min\left\{ \frac{\beta \|\nabla_{\mathcal{I}_k} f(x_k)\|}{B}, \sqrt{\frac{3\beta \|\nabla_{\mathcal{I}_k} f(x_k)\|}{\sigma^{\max}}} \right\}.$$

So, using the greedy selection rule (11) and the fact that $\|\nabla f(x_k)\| \geq \epsilon$, with $\epsilon \in [0,1]$, we obtain

$$\hat{\alpha}_k \|\nabla_{\mathcal{I}_k} f(x_k)\| \geq \min\left\{ \frac{\theta\beta\epsilon}{B}, \sqrt{\frac{3\theta\beta\epsilon}{\sigma^{\max}}} \right\} \geq \epsilon \min\left\{ \frac{\theta\beta}{B}, \sqrt{\frac{3\theta\beta}{\sigma^{\max}}} \right\}. \tag{30}$$

Then, combining (29) and (30), the desired result follows. □

Using the above result, we can easily show the global convergence of IBCN to stationary points.

Theorem 12. *If Assumption 1 holds, then*

$$\lim_{k\to\infty} \nabla f(x_k) = 0.$$

Proof. Since, in view of the definition of $\mathcal{S}$ given in (20), we have $\nabla f(x_{k+1}) = \nabla f(x_k)$ for all $k \notin \mathcal{S}$, then the result is true if and only if

$$\lim_{\substack{k\to\infty \\ k\in\mathcal{S}}} \nabla f(x_k) = 0.$$

By contradiction, assume that $\epsilon \in (0, 1]$ and an infinite subset $K \subseteq \mathcal{S}$ exist such that $\|\nabla f(x_k)\| \geq \epsilon$ for all $k \in K$. From Proposition 11, it follows that f is unbounded from below, thus contradicting (27). □

Remark 2. *From Theorem 12 and the continuity of ∇f, it follows that every limit point of $\{x_k\}$ is a stationary point.*

4.2 Worst-case iteration complexity

Here, we analyze the worst-case iteration complexity of the proposed IBCN method, providing two main results.

First, in the following theorem, we show that at most $\mathcal{O}(\epsilon^{-3/2})$ iterations are needed to drive $\|\nabla_{\mathcal{I}_k} f(x_{k+1})\|$ below a given threshold $\epsilon > 0$. Note that, in the proof of the following theorem, no role is played by the greedy selection rule (11), that is, the result holds for any arbitrary choice of the blocks.

Theorem 13. *Given $\epsilon > 0$, let*

$$K_\epsilon^b = \{k \geq 0 \colon \|\nabla_{\mathcal{I}_k} f(x_{k+1})\| \geq \epsilon\}.$$

If Assumption 1 holds, then

$$|K_\epsilon^b \cap \mathcal{S}| \leq \left(\frac{f(x_0) - f^{min}}{c_2}\right)\epsilon^{-3/2},$$

where

$$c_2 = \frac{\eta_1 \sigma^{min}}{6}\left(\tau + \frac{\sigma^{max} + L}{2}\right)^{-3/2}.$$

Moreover, let $\bar{k}$ be the first iteration such that $\bar{k} \notin K_\epsilon^b$. Then,

$$\bar{k} \leq \left(1 + \frac{|\log \gamma_1|}{\log \gamma_2}\right)\left(\frac{f(x_0) - f^{min}}{c_2}\right)\epsilon^{-3/2} + \frac{1}{\log \gamma_2}\log\left(\frac{\sigma^{max}}{\sigma_0}\right),$$

where σ^{max} is given in Proposition 9.

Proof. From Proposition 6 and the definition of $\mathcal{S}$ given in (20), we have that $f(x_k) - f(x_{k+1}) \geq 0$ for all $k \geq 0$. So, using the lower bound on $f(x)$ given in (27), we can write

$$f(x_0) - f^{\min} \geq f(x_0) - f(x_{r+1}) \geq \sum_{\substack{h=0 \\ h \in K_\epsilon^{\mathrm{b}} \cap \mathcal{S}}}^{r} (f(x_h) - f(x_{h+1})) \quad \forall r \geq 0.$$

Taking the limit for $r \to \infty$, we get

$$f(x_0) - f^{\min} \geq \sum_{k \in K_\epsilon^{\mathrm{b}} \cap \mathcal{S}} (f(x_k) - f(x_{k+1})). \tag{31}$$

Now we want to lower bound the right-hand side term of (31). First, from Proposition 6, we have that

$$f(x_k) - f(x_{k+1}) \geq \frac{\eta_1 \sigma_k}{6} \|s_k\|^3 \geq \frac{\eta_1 \sigma^{\min}}{6} \|s_k\|^3 \quad \forall k \in \mathcal{S}, \tag{32}$$

where, in the last inequality, we have used Proposition 9 to lower bound σ_k. Moreover, from Proposition 7, we have that

$$\|\nabla_{\mathcal{I}_k} f(x_{k+1})\| \leq \left(\tau + \frac{\sigma_k + L_{\mathcal{I}_k}}{2}\right) \|s_k\|^2 \leq \left(\tau + \frac{\sigma^{\max} + L}{2}\right) \|s_k\|^2 \quad \forall k \in \mathcal{S}, \tag{33}$$

where, in the last inequality, we have used Proposition 9 and (8) to upper bound σ_k and $L_{\mathcal{I}_k}$, respectively. Therefore, from (32) and (33), we obtain

$$f(x_k) - f(x_{k+1}) \geq c_2 \|\nabla_{\mathcal{I}_k} f(x_{k+1})\|^{3/2} \quad \forall k \in \mathcal{S}. \tag{34}$$

It follows that

$$f(x_k) - f(x_{k+1}) \geq c_2 \epsilon^{3/2} \quad \forall k \in \mathcal{S} \text{ such that } \|\nabla_{\mathcal{I}_k} f(x_{k+1})\| \geq \epsilon.$$

Using this inequality in (31), we obtain

$$f(x_0) - f^{\min} \geq |K_\epsilon^{\mathrm{b}} \cap \mathcal{S}| c_2 \epsilon^{3/2},$$

thus leading to the desired upper bound on $|K_\epsilon^{\mathrm{b}} \cap \mathcal{S}|$. To obtain the desired upper bound on $\bar{k}$, it is trivial if $\bar{k} = 0$, otherwise the result follows by using Proposition 10 with $k = \bar{k} - 1$ and noting that $|\mathcal{S}_{\bar{k}-1}| \leq |K_\epsilon^{\mathrm{b}} \cap \mathcal{S}|$. □

From Theorem 13 we see that, to drive the stationarity violation with respect to *a selected block of variables* below ϵ, we need at most $\mathcal{O}(\epsilon^{-3/2})$ iterations, thus matching the complexity bound given in [1] for cyclic-type selection.

Moreover, when $\mathcal{I}_k = \{1, \ldots, n\}$ for all k, we retain the complexity bound of standard cubic Newton methods, that is, at most $\mathcal{O}(\epsilon^{-3/2})$ iterations are needed to obtain $\|\nabla f(x_k)\| < \epsilon$.

In a general case where $|\mathcal{I}_k| < n$, Theorem 13 does not provide information on how many iterations are needed in the worst case to drive the stationarity violation with respect to *all variables* below ϵ. Such a complexity bound is given in the next theorem, ensuring that at most $\mathcal{O}(\epsilon^{-2})$ iterations are needed to get $\|\nabla f(x_k)\| < \epsilon$.

Theorem 14. *Given $\epsilon \in (0,1]$, let*

$$K_\epsilon = \{k \geq 0 \colon \|\nabla f(x_k)\| \geq \epsilon\}.$$

If Assumption 1 holds, then

$$|K_\epsilon \cap \mathcal{S}| \leq \left(\frac{f(x_0) - f^{min}}{c_1}\right)\epsilon^{-2},$$

where c_1 is defined as in Proposition 11. Moreover, let $\hat{k}$ be the first iteration such that $\hat{k} \notin K_\epsilon$. Then,

$$\hat{k} \leq \left(1 + \frac{|\log \gamma_1|}{\log \gamma_2}\right)\left(\frac{f(x_0) - f^{min}}{c_1}\right)\epsilon^{-2} + \frac{1}{\log \gamma_2}\log\left(\frac{\sigma^{max}}{\sigma_0}\right),$$

where σ^{max} is given in Proposition 9.

Proof. From Proposition 6 and the definition of $\mathcal{S}$ given in (20), we have that $f(x_k) - f(x_{k+1}) \geq 0$ for all $k \geq 0$. So, using the lower bound on $f(x)$ given in (27), we can write

$$f(x_0) - f^{\min} \geq f(x_0) - f(x_{r+1}) \geq \sum_{\substack{h=0 \\ h \in K_\epsilon \cap \mathcal{S}}}^{r} (f(x_h) - f(x_{h+1})) \quad \forall r \geq 0.$$

Taking the limit for $r \to \infty$, we get

$$f(x_0) - f^{\min} \geq \sum_{k \in K_\epsilon \cap \mathcal{S}} (f(x_k) - f(x_{k+1})).$$

From Proposition 11, it follows that

$$f(x_0) - f^{\min} \geq |K_\epsilon \cap \mathcal{S}| c_1 \epsilon^2,$$

thus leading to the desired upper bound on $|K_\epsilon \cap \mathcal{S}|$. To obtain the desired upper bound on $\hat{k}$, it is trivial if $\hat{k} = 0$, otherwise the result follows by using Proposition 10 with $k = \hat{k} - 1$ and noting that $|\mathcal{S}_{\hat{k}-1}| \leq |K_\epsilon \cap \mathcal{S}|$. □

Remark 3. *Since $c_1 = \mathcal{O}(\theta^{3/2})$, with $\theta \in (0,1]$, it follows that the larger θ the better the complexity bound of Theorem 14. Values for θ have been derived in Propositions 2–3 when using two simple strategies for the block selection.*

5 Numerical experiments

In this section, we report some numerical results. The experiments were run in Matlab R2024a on an Apple MacBook Pro with an Apple M1 Pro Chip and 16 GB RAM.

Given a set of samples $\{a^1, \dots, a^m\} \subseteq \mathbb{R}^n$ and labels $\{b^1, \dots, b^m\} \subseteq \mathbb{R}$, let $\varphi_x \colon \mathbb{R}^n \to \mathbb{R}$ be a prediction function parameterized by a vector x. We consider optimization problems from regression and classification models where the objective function has the following form:

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} \ell(b^i, \varphi_x(a^i)) + \lambda P(x), \tag{35}$$

with $\ell \colon \mathbb{R} \times \mathbb{R} \to [0, \infty)$ being a loss function, $P \colon \mathbb{R}^n \to [0, \infty)$ being a regularizer and $\lambda \geq 0$ being a regularization parameter.

In what follows, we consider two types of comparison:

- In Subsection 5.1, we compare IBCN with other schemes that use the same greedy block selection but different block updates;
- In Subsection 5.2, we compare IBCN with other schemes that use the same cubic Newton updates but different block selection.

5.1 Comparison with other block updates

We first compare IBCN with two block coordinate descent methods using the same greedy selection rule. Specifically, we consider both a first-order method and a second-order method, referred to as BCD1 and BCD2, respectively. At each iteration k of BCD1 and BCD2, given the current point x_k and a block of variables $\mathcal{I}_k$, we compute a search direction $d_k \in \mathbb{R}^{|\mathcal{I}_k|}$ as follows:

- For BCD1, we use the steepest descent direction, that is,

$$d_k = -\nabla_{\mathcal{I}_k} f(x_k);$$

- For BCD2, we use a diagonally scaled steepest descent direction [3, 34], that is,

$$d_k = -(H_k)^{-1} \nabla_{\mathcal{I}_k} f(x_k),$$

  where $H_k \in \mathbb{R}^{|\mathcal{I}_k| \times |\mathcal{I}_k|}$ is symmetric and positive definite. To compute H_k, we choose a diagonal Hessian approximation as in [34, Subsection 7.2], that is,

$$H_k = \operatorname{diag}(v_k), \quad \text{with} \quad v_k = \left[\min\{\max\{\nabla^2_{\{j\}} f(x_k), 10^{-2}\}, 10^9\}\right]_{j \in \mathcal{I}_k},$$

  where $\operatorname{diag}(v_k)$ denotes the diagonal matrix constructed from the vector v_k.

For both BCD1 and BCD2, once d_k is obtained, we set $x_{k+1} = x_k + \alpha_k U_{\mathcal{I}_k} d_k$, with α_k being computed by means of an Armijo line search, similarly as in [3, 34].

At each iteration k of IBCN, BCD1 and BCD2, a block of variables $\mathcal{I}_k$ is chosen as described in Proposition 2, that is, such that $\|\nabla_{\mathcal{I}_k} f(x_k)\|_\infty = \|\nabla f(x_k)\|_\infty$. In more detail, first we compute the index $\hat{\imath}_k$ corresponding to the largest component in absolute value of $\nabla f(x_k)$, then $\mathcal{I}_k$ is set to include $\hat{\imath}_k$ with the other variable indices being chosen randomly. In our experiments, we use blocks of size $q \in \{1, 5, 10, 20, 50, 100\}$.

In IBCN, we set $\sigma_0 = \sigma^{\min} = 1$, $\eta_1 = \eta_2 = 0.1$, $\gamma_1 = 1$, $\gamma_2 = \gamma_3 = 2$ and $\tau = 1$. To compute s_k at each iteration k of IBCN, we set $s_k = \hat{s}_k$, with $\hat{s}_k$ defined as in (16), if this choice satisfies (15). Otherwise, we run a Barzilai-Borwein gradient method [32] to $m_k(s)$, starting from $\hat{s}_k$, until a point s is produced such that (15) holds with s_k replaced by s.

In all experiments, we run IBCN, BCD1 and BCD2 from the starting point $x_0 = 0$ for 10^4 iterations without using any other stopping condition. Then, considering a sequence $\{x_k\}$ produced by a given algorithm, we analyze the decrease of the objective error $(f(x_k) - f^*)$ and the decrease of the first-order stationarity violation $\|\nabla f(x_k)\|$, with f^* denoting the best objective value found for a given problem over all simulations.

Both non-convex and convex problems are considered in our experiments in Subsubsection 5.1.1 and 5.1.2, respectively.

5.1.1 Sparse least squares

The problem of recovering sparse vectors from linear measurements is central in many applications, such as compressive sensing [15] and variable selection [18]. To obtain sparse solutions, a popular approach is to use least squares with ℓ_1-norm regularization, resulting in a convex formulation known as LASSO [33]. However, in order to overcome the bias connected to the ℓ_1 norm, some non-convex regularizers have also been introduced in the literature [36].

Here we use a non-convex sparsity promoting term considered in, e.g., [25, 31], given by $P(x) = \sum_{i=1}^n (x_i^2 + \omega^2)^{p/2}$, with small $\omega > 0$ and $p \in (0, 1)$. Using the least squares as loss, we hence obtain the following non-convex problem:

$$\min_{x \in \mathbb{R}^n} \frac{1}{m}\|Ax - b\|^2 + \lambda \sum_{i=1}^n (x_i^2 + \omega^2)^{p/2}, \tag{36}$$

where $A = \begin{bmatrix} a^1 & \dots & a^m \end{bmatrix}^T \in \mathbb{R}^{m \times n}$ and $b = \begin{bmatrix} b^1 & \dots & b^m \end{bmatrix}^T \in \mathbb{R}^m$. In our experiments we set $\lambda = 10^{-2}$, $\omega = 10^{-2}$ and $p = 0.5$. After generating the elements of the matrix A randomly from a uniform distribution in $(0, 1)$, with $m = n = 10,000$, a vector $\hat{x} \in \mathbb{R}^n$ was created with all components equal to zero except for 1% of them, which were randomly set to ± 1. Then, we set $b = A\hat{x} + \zeta$, where $\zeta \in \mathbb{R}^m$ is a noise vector with elements drawn from a normal distribution with mean 0 and standard deviation 10^{-3}.

Note that, according to the notation used in (35)–(36), we have

$$\nabla f(x) = \frac{2}{m} A^T (Ax - b) + \lambda \nabla P(x), \quad \text{and} \quad \nabla^2 f(x) = \frac{2}{m} A^T A + \lambda \nabla^2 P(x).$$

In the above formulas, the most computationally expensive operations are those involving the quadratic term, as the regularizer $P(x)$ has a separable structure leading to a negligible cost. In particular, when the problem dimension is large, the computation of $A^T A$ can be prohibitive, as it requires $\mathcal{O}(mn^2)$ arithmetic operations, and should therefore be avoided. Considering such a scenario, here we do not store $A^T A$ in the considered algorithms. Consequently, at every iteration, to obtain the Hessian of f with respect to a block of size q, we compute inner products between selected columns of A, with a cost of $\mathcal{O}(mq^2)$ arithmetic

operations, except when using blocks of dimension $q = 1$, in which case we store the diagonal of $A^T A$ since this only requires an affordable $\mathcal{O}(mn)$ cost in the initialization. Then, to get first-order derivatives at iteration k, we use the residual vector $r_k := Ax_k - b$, so that

$$\nabla f(x_k) = \frac{2}{m} A^T r_k + \lambda \nabla P(x_k). \tag{37}$$

In this way, the computation of $\nabla f(x_k)$ requires $\mathcal{O}(mn)$ arithmetic operations, while the cost to update r_{k+1} from r_k is $\mathcal{O}(mq)$, since $r_{k+1} = r_k + AU_{\mathcal{I}_k} s_k$ with $U_{\mathcal{I}_k} s_k$ having at most q non-zeros. This strategy is commonly adopted when dealing with quadratic functions in a block-coordinate setting (see, e.g., [10, 24, 26]).

Overall, we see that the cost to compute the required first- and second-order derivatives at every iteration is $\mathcal{O}(m \max\{q^2, n\})$ arithmetic operations, which in our case reduces to $\mathcal{O}(mn)$ since $n \geq q^2$.

We run 10 simulations and in Figure 1 we report the average results with respect to both the number of iterations and the CPU time.

We see that, for $q = 1$, all the considered methods perform very similarly, whereas IBCN clearly outperforms both BCD1 and BCD2 as the size of the blocks increases. In particular, while the results of IBCN and BCD2 remain comparable for $q = 5$, we see that, for $q \geq 10$, IBCN yields a much faster decrease in both the objective function and the gradient norm than the competing methods. Within the given limit of 10^4 iterations, we also observe that IBCN is consistently able to achieve a lower objective value with a smaller gradient norm.

5.1.2 Regularized logistic regression

To assess how IBCN works on convex problems, we consider the ℓ_2-regularized logistic regression. In particular, assuming that $b^i \in \{\pm 1\}$, $i = 1, \ldots, m$, the optimization problem can be formulated as follows:

$$\min_{(x,z) \in \mathbb{R}^{n+1}} \frac{1}{m} \sum_{i=1}^{m} \log\Big(1 + e^{-b^i((a^i)^T x + z)}\Big) + \lambda \|x\|^2,$$

We use the following three datasets from `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`:

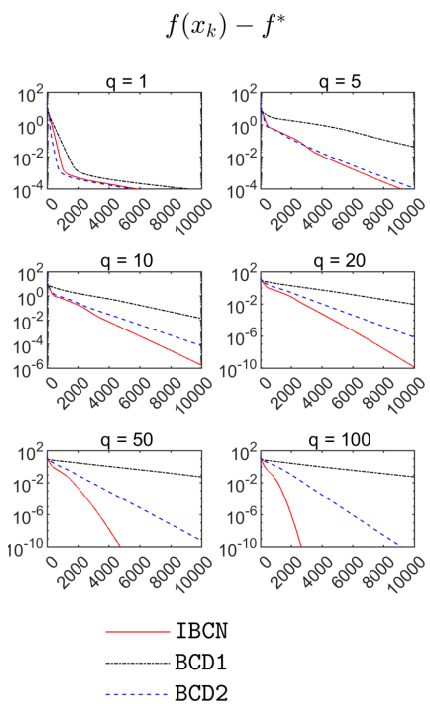
(i) gisette (train), $m = 6000$, $n = 5000$;

(ii) leu (train), $m = 38$, $n = 7129$;

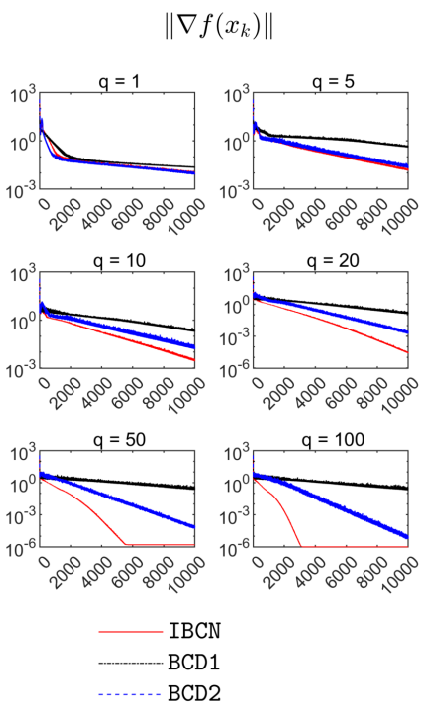
(iii) madelon (train), $m = 2000$, $n = 500$;

scaling all features of the last dataset in $[-1, 1]$, while the other ones had already been scaled or normalized.

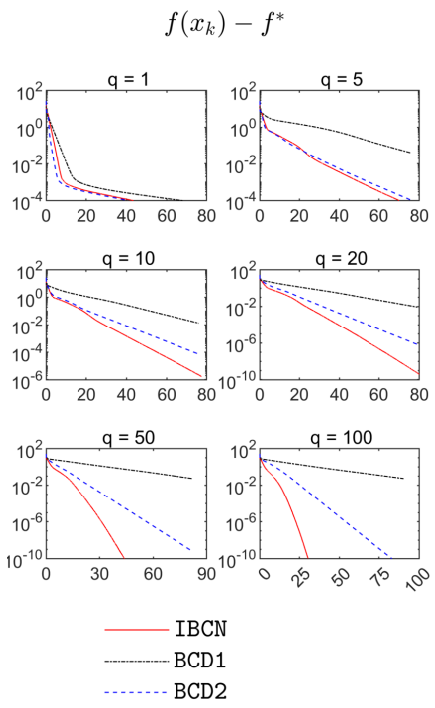
Results with respect to the number of iterations are reported in Figures 2–3. We see that, for $q = 1$, IBCN and BCD2 have similar performance and both of them give better results than BCD1. For larger values of q, IBCN provides a faster objective decrease and is able to produce points with a smaller gradient norm than the two competitive methods. As in the



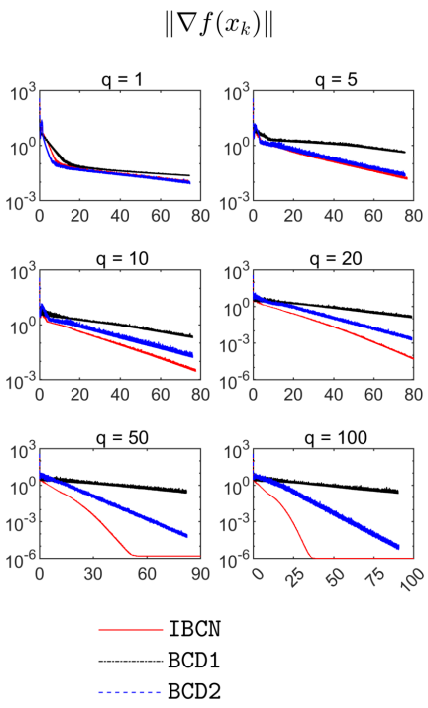
(a) Objective error vs iteration



(b) Stationarity violation vs iteration



(c) Objective error vs CPU time



(d) Stationarity violation vs CPU time

Figure 1: Results on sparse least squares using blocks of size q. In each plot, the y axis is in logarithmic scale.

case of non-convex problems discussed in the previous subsection, the superiority of IBCN becomes more evident as the block size increases.

Finally, results with respect to the CPU time are reported in Figure 4 only for the gisette dataset since, for the other datasets, the methods take a few seconds in most cases. We see that IBCN seems to provide the best results for $q \geq 5$, confirming the above findings.

5.2 Comparison with other block selection rules

As mentioned in Section 1, different block selection rules can be used in a block coordinate descent scheme, with the most efficient choice depending on computational aspects related to the features of the specific problem [13, 29, 30, 35]. Here, we want to compare the proposed greedy selection with two popular choices [30, 37]:

- *cyclic selection*, in which each variable is selected at least once within any window of a pre-specified number of iterations;
- *random selection*, in which blocks are chosen randomly at every iteration.

On the one hand, both cyclic and random selection rules do not require to compute the full gradient of the objective function at each iteration, and can therefore be more computationally efficient in some cases. On the other hand, a greedy selection uses more information, so larger per-iteration progress is expected.

For first-order methods, it is well known that, in practice, the best choice among greedy, cyclic and random selection is related to the relative cost of performing the update of all blocks versus performing a full gradient iteration [29]. Some comparisons can be found in [13, 29, 30, 35].

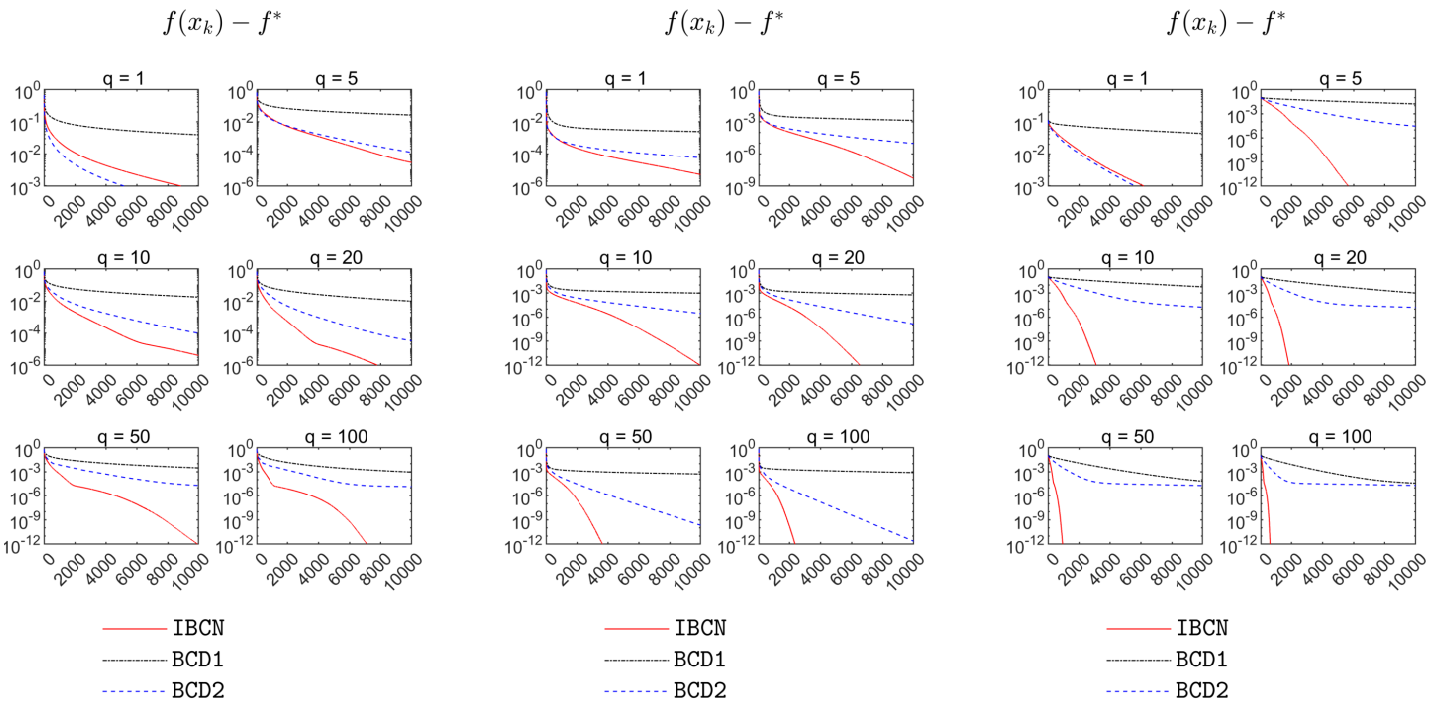
For cubic Newton schemes, an analysis of cyclic selection was carried out in [1], while random selection strategies were studied in [14, 23, 38]. In what follows, we investigate how the performance, measured in terms of running time, of the considered block selection rules is affected by the per-iteration cost. For this analysis, we consider the sparse least squares problem from Subsubsection 5.1.1 and distinguish two scenarios:

(i) the per-iteration cost of greedy selection is higher than that of cyclic and random selection, see Subsubsection 5.2.1;

(ii) the per-iteration cost of greedy selection is similar to that of cyclic and random selection, see Subsubsection 5.2.2.

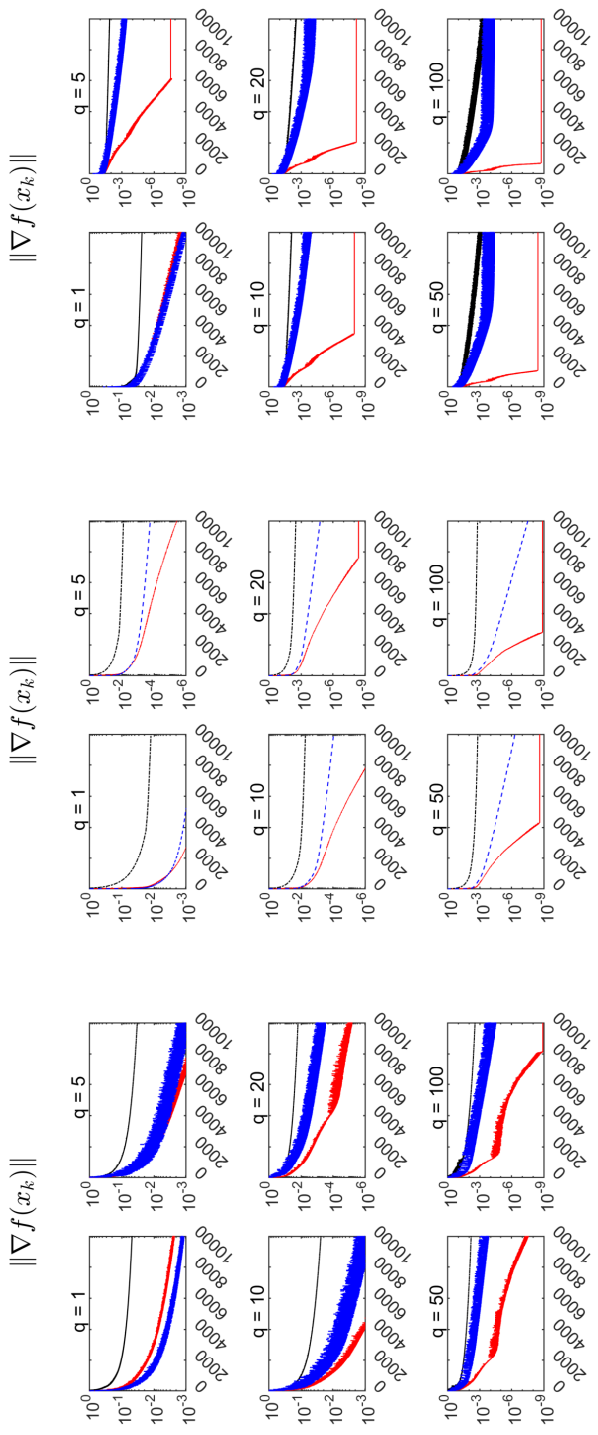
As will be shown, which scenario occurs depends on whether or not the matrix $A^T A$ can be stored. In particular, in our analysis, the per-iteration cost is approximated by the number of arithmetic operations to compute the required first- and second-order derivatives of the quadratic term, which we assume dominates the workload of minimizing the cubic model while the operations on the regularizer $P(x)$ have a negligible cost due to its separable structure.

In the following experiments, cyclic and random selection were implemented using a uniform random sampling strategy without and with replacement, respectively. We left all other algorithmic choices as in IBCN. Moreover, the methods using cyclic and random



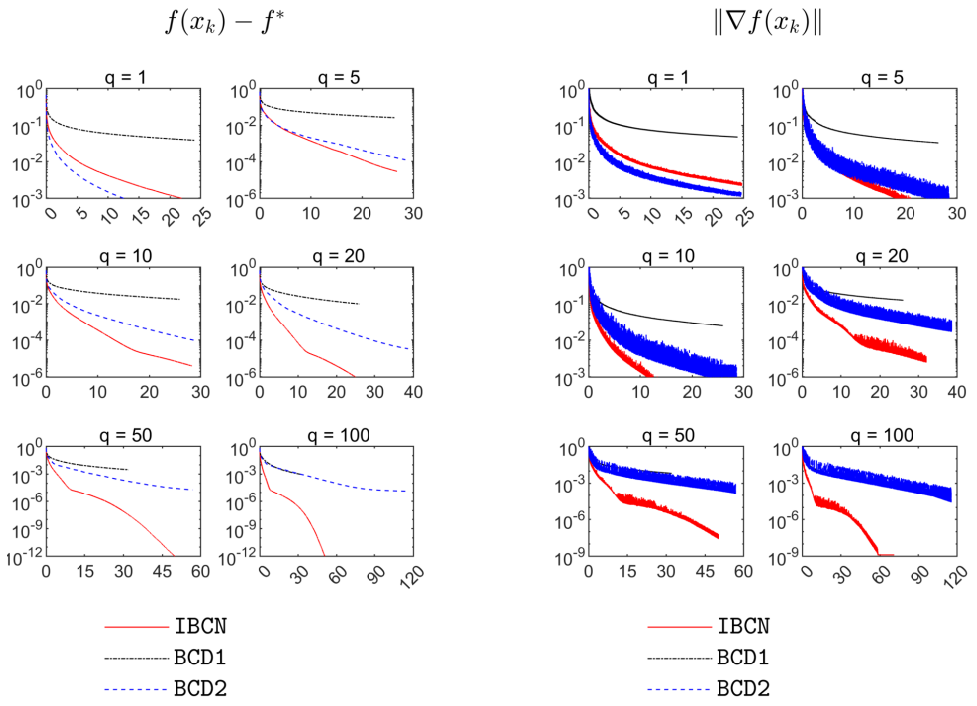
(a) gisette dataset (b) leu dataset (c) madelon dataset

Figure 2: Objective error vs iteration for ℓ_2-regularized logistic regression using blocks of size q. In each plot, the y axis is in logarithmic scale.



(a) gisette dataset (b) leu dataset (c) madelon dataset

Figure 3: Stationarity violation vs iteration for ℓ_2-regularized logistic regression using blocks of size q. In each plot, the y axis is in logarithmic scale.



(a) Objective error vs CPU time

(b) Stationarity violation vs CPU time

Figure 4: Results on ℓ_2-regularized logistic regression with respect to the CPU time using blocks of size q for gisette dataset. In each plot, the y axis is in logarithmic scale.

selection were stopped after the time taken by the method using the greedy selection on the same instance. The results were then averaged over 10 simulations.

5.2.1 Greedy selection: higher per-iteration cost than cyclic and random selection

If the matrix A^TA is not stored, in Subsubsection 5.1.1 we described how, for each iteration k, the Hessian of f with respect to a block of size q requires $\mathcal{O}(mq^2)$ arithmetic operations, while we can use the residual vector r_k to compute $\nabla f(x_k)$ in $\mathcal{O}(mn)$ arithmetic operations by means of (37). Hence, to get the required first- and second-order derivatives for an iteration with greedy selection, we have a total cost of $\mathcal{O}(m \max\{q^2, n\})$ arithmetic operations, which boils down to $\mathcal{O}(mn)$ since $n \geq q^2$ in our experiments.

Employing cyclic or random selection, at each iteration we still need $\mathcal{O}(mq^2)$ arithmetic operations to compute the Hessian of f with respect to a block of size q, while we only have to compute q first-order partial derivatives, the latter having a cost $\mathcal{O}(mq)$ still using (37) for the selected rows of A^T. Hence, the total cost is of $\mathcal{O}(mq^2)$ arithmetic operations for each iteration, that is, computing the Hessian of f with respect to a block is the dominant cost. We conclude that, for the greedy selection, the per-iteration cost is about n/q^2 times higher than for cyclic and random selection.

The results are reported in Figure 5(a). As expected, cyclic and random selection are faster in most cases due to their lower per-iteration cost, even if we observe that the greedy selection performs remarkably better when $q = 1$.

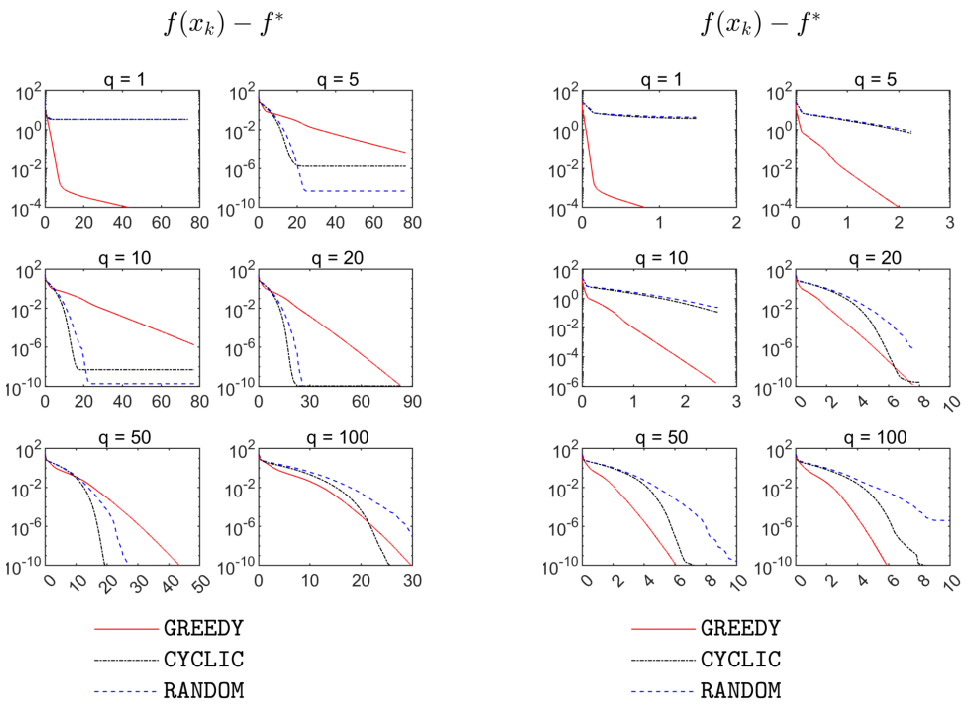
5.2.2 Greedy selection: similar per-iteration cost to cyclic and random selection

Assume now that the matrix A^TA can be stored. First note that the cost to compute second-order derivatives is negligible in this case. To compute first-order derivatives, we can update the gradient of f from iteration k to iteration $k+1$ by

$$\nabla f(x_{k+1}) = \nabla f(x_k) + \frac{2}{m}(A^TA)U_{\mathcal{I}_k}s_k + \lambda(\nabla P(x_{k+1}) - \nabla P(x_k)) \quad \forall k \geq 0.$$

Since $U_{\mathcal{I}_k}s_k$ has at most q non-zeros components, it follows that $\nabla f(x_k)$ can be computed with a cost of $\mathcal{O}(qn)$ arithmetic operations. The same procedure also applies to compute q first-order partial derivatives when using cyclic or random selection, implying that the cost is the same as computing the full gradient, that is, $\mathcal{O}(qn)$. (Note that, since $m = n$ in our experiments, the cost of computing the required first-order derivatives for cyclic and random selection is the same as the one obtained in the previous scenario where A^TA was not stored.)

The results are reported in Figure 5(b). We see that the greedy selection consistently outperforms both the cyclic and the random selection. The reason is that the per-iteration cost is comparable across the considered strategies, while the greedy selection uses more information by employing the full gradient of f to select a block at each iteration. We also observe that computing A^TA takes approximately 3 seconds on average. Therefore, storing that matrix seems the most efficient option in the current configuration (cf. Figure 5(a)).



(a) The Hessian of the quadratic term is not stored. Here greedy selection has a higher per-iteration cost than cyclic and random selection.

(b) The Hessian of the quadratic term is stored. Here greedy selection has a similar per-iteration cost to cyclic and random selection.

Figure 5: Objective error vs CPU time for sparse least squares using blocks of size q. In each plot, the y axis is in logarithmic scale.

6 Conclusions

In this paper, we have considered the unconstrained minimization of an objective function with Lipschitz continuous Hessian. For this problem, we have presented a block coordinate descent version of cubic Newton methods using a greedy (Gauss-Southwell) selection rule, where blocks of variables are chosen by considering the amount of first-order stationarity violation. To update the selected block at each iteration, an inexact minimizer of a cubic model is computed. In practice, such an inexact minimization can be carried out in finite time without the need of additional evaluations of the objective function or its derivatives in other points. In the proposed scheme, blocks are not required to have a predetermined structure and their size may even change during the iterations. Moreover, the knowledge of the Lipschitz constant of the Hessian is not needed.

In a non-convex setting, we have shown global convergence to stationary points and analyzed the worst-case iteration complexity. Specifically, we have shown that at most $\mathcal{O}(\epsilon^{-3/2})$ iterations are needed to drive the stationarity violation with respect to a selected block of variables below ϵ, while at most $\mathcal{O}(\epsilon^{-2})$ iterations are needed to drive the stationarity violation with respect to all variables below ϵ. In particular, the latter result improves over $\mathcal{O}(\epsilon^{-3})$ which was given in [1] for cyclic-type selection.

Then, we have tested the proposed method, named IBCN, on non-convex and convex problems arising in regression and classification models.

Numerical results indicate that the proposed method consistently outperforms other greedy schemes, with the performance gap widening as the block size increases. We have also shown that the proposed approach works particularly well in a regime where the full gradient computation does not significantly affect the per-iteration cost. In such a setting, the proposed greedy selection outperforms cyclic and random schemes.

The code of the proposed IBCN method is freely available at `https://github.com/acristofari/ibcn`.

Finally, further investigation needs to be devoted to analyzing the worst-case iteration complexity in convex and strongly convex problems.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

[1] V. Amaral, R. Andreani, E. Birgin, D. Marcondes, and J. M. Martínez. On complexity and convergence of high-order coordinate descent algorithms for smooth nonconvex box-constrained minimization. *Journal of Global Optimization*, 84(3):527–561, 2022.

[2] A. Beck and L. Tetruashvili. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization*, 23(4):2037–2060, 2013.

[3] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, Belmont, MA, 1999.

[4] T. Bianconcini, G. Liuzzi, B. Morini, and M. Sciandrone. On the use of iterative methods in cubic regularization for unconstrained optimization. *Computational Optimization and Applications*, 60:35–57, 2015.

[5] E. G. Birgin, J. Gardenghi, J. M. Martínez, S. A. Santos, and P. L. Toint. Worst-case evaluation complexity for unconstrained nonlinear optimization using high-order regularized models. *Mathematical Programming*, 163:359–368, 2017.

[6] Y. Carmon and J. Duchi. Gradient Descent Finds the Cubic-Regularized Nonconvex Newton Step. *SIAM Journal on Optimization*, 29(3):2146–2178, 2019.

[7] C. Cartis, N. I. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results. *Mathematical Programming*, 127(2):245–295, 2011.

[8] C. Cartis, N. I. Gould, and P. L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Part II: worst-case function-and derivative-evaluation complexity. *Mathematical programming*, 130(2):295–319, 2011.

[9] C. Cartis, N. I. Gould, and P. L. Toint. Universal Regularization Methods: Varying the Power, the Smoothness and the Accuracy. *SIAM Journal on Optimization*, 29(1): 595–615, 2019.

[10] A. Cristofari. An almost cyclic 2-coordinate descent method for singly linearly constrained problems. *Computational Optimization and Applications*, 73(2):411–452, 2019.

[11] A. Cristofari, T. Dehghan Niri, and S. Lucidi. On global minimizers of quadratic functions with cubic regularization. *Optimization Letters*, 13:1269–1283, 2019.

[12] J. E. Dennis Jr and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, 1996.

[13] I. Dhillon, P. Ravikumar, and A. Tewari. Nearest neighbor based greedy coordinate descent. *Advances in Neural Information Processing Systems*, 24, 2011.

[14] N. Doikov and P. Richtárik. Randomized Block Cubic Newton Method. In *International Conference on Machine Learning*, pages 1290–1298. PMLR, 2018.

[15] D. L. Donoho. Compressed sensing. *IEEE Transactions on information theory*, 52(4): 1289–1306, 2006.

[16] J.-P. Dussault. ARC_q: a new adaptive regularization by cubics. *Optimization Methods and Software*, 33(2):322–335, 2018.

[17] J.-P. Dussault, T. Migot, and D. Orban. Scalable adaptive cubic regularization methods. *Mathematical Programming*, pages 1–35, 2023.

[18] J. Fan and R. Li. Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. *Journal of the American statistical Association*, 96(456):1348–1360, 2001.

[19] K. Fountoulakis and R. Tappenden. A flexible coordinate descent method. *Computa-*

tional Optimization and Applications, 70(2):351–394, 2018.

[20] N. I. Gould and V. Simoncini. Error estimates for iterative algorithms for minimizing regularized quadratic subproblems. *Optimization Methods and Software*, 35(2):304–328, 2020.

[21] G. N. Grapiglia and Y. Nesterov. Regularized Newton methods for minimizing functions with Hölder continuous Hessians. *SIAM Journal on Optimization*, 27(1):478–506, 2017.

[22] A. Griewank. The modification of Newton's method for unconstrained optimization by bounding cubic terms. Technical Report NA/12, 1981.

[23] F. Hanzely, N. Doikov, Y. Nesterov, and P. Richtarik. Stochastic Subspace Cubic Newton method. In *International Conference on Machine Learning*, pages 4027–4038. PMLR, 2020.

[24] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th international conference on Machine learning*, pages 408–415. ACM, 2008.

[25] M.-J. Lai, Y. Xu, and W. Yin. Improved Iteratively Reweighted Least Squares for Unconstrained Smoothed ℓ_q Minimization. *SIAM Journal on Numerical Analysis*, 51 (2):927–957, 2013.

[26] I. Necoara and A. Patrascu. A random coordinate descent algorithm for optimization problems with composite objective function and linear coupled constraints. *Computational Optimization and Applications*, 57(2):307–337, 2014.

[27] Y. Nesterov. Inexact basic tensor methods for some classes of convex optimization problems. *Optimization Methods and Software*, 37(3):878–906, 2022.

[28] Y. Nesterov and B. T. Polyak. Cubic regularization of Newton method and its global performance. *Mathematical Programming*, 108(1):177–205, 2006.

[29] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander, and H. Koepke. Coordinate Descent Converges Faster with the Gauss-Southwell Rule Than Random Selection. In *International Conference on Machine Learning*, pages 1632–1641. PMLR, 2015.

[30] J. Nutini, I. Laradji, and M. Schmidt. Let's Make Block Coordinate Descent Converge Faster: Faster Greedy Rules, Message-Passing, Active-Set Complexity, and Superlinear Convergence. *Journal of Machine Learning Research*, 23(131):1–74, 2022.

[31] J. K. Pant, W.-S. Lu, and A. Antoniou. New Improved Algorithms for Compressive Sensing Based on ℓ_p Norm. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 61(3):198–202, 2014.

[32] M. Raydan. The Barzilai and Borwein Gradient Method for the Large Scale Unconstrained Minimization Problem. *SIAM Journal on Optimization*, 7(1):26–33, 1997.

[33] R. Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.

[34] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117:387–423, 2009.

[35] S. Venturini, A. Cristofari, F. Rinaldi, and F. Tudisco. Laplacian-based semi-Supervised learning in multilayer hypergraphs by coordinate descent. *EURO Journal on Computational Optimization*, 11:100079, 2023.

[36] F. Wen, L. Chu, P. Liu, and R. C. Qiu. A Survey on Nonconvex Regularization-Based Sparse and Low-Rank Recovery in Signal Processing, Statistics, and Machine Learning.

IEEE Access, 6:69883–69906, 2018.

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

[38] J. Zhao, A. Lucchi, and N. Doikov. Cubic regularized subspace Newton for non-convex optimization. *arXiv preprint arXiv:2406.16666*, 2024.

A Properties from Lipschitz continuity

Proof of Proposition 1. Choose $\mathcal{I} \subseteq \{1, \dots, n\}$ and define the function $\psi \colon \mathbb{R}^n \to \mathbb{R}^{|\mathcal{I}|}$, $\psi(x) = U_{\mathcal{I}}^T \nabla f(x)$. Namely, using (2),

$$\psi(x) = \nabla_{\mathcal{I}} f(x) \quad \forall x \in \mathbb{R}^n.$$

Now, take $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^{|\mathcal{I}|}$. Applying the mean value theorem to ψ, we can write

$$\begin{aligned}
\nabla_{\mathcal{I}} f(x + U_{\mathcal{I}} s) - \nabla_{\mathcal{I}} f(x) &= \psi(x + U_{\mathcal{I}} s) - \psi(x) \\
&= \int_0^1 \nabla \psi(x + t U_{\mathcal{I}} s)^T U_{\mathcal{I}} s \, dt \\
&= \int_0^1 U_{\mathcal{I}}^T \nabla^2 f(x + t U_{\mathcal{I}} s) U_{\mathcal{I}} s \, dt \\
&= \int_0^1 \nabla_{\mathcal{I}}^2 f(x + t U_{\mathcal{I}} s) s \, dt,
\end{aligned}$$

where we have used (3) in the last equality. Adding $-\nabla_{\mathcal{I}}^2 f(x) s$ to all terms, we obtain

$$\begin{aligned}
\|\nabla_{\mathcal{I}} f(x + U_{\mathcal{I}} s) - \nabla_{\mathcal{I}} f(x) - \nabla_{\mathcal{I}}^2 f(x) s\| &= \left\| \int_0^1 (\nabla_{\mathcal{I}}^2 f(x + t U_{\mathcal{I}} s) - \nabla_{\mathcal{I}}^2 f(x)) s \, dt \right\| \\
&\leq \int_0^1 \|(\nabla_{\mathcal{I}}^2 f(x + t U_{\mathcal{I}} s) - \nabla_{\mathcal{I}}^2 f(x)) s\| \, dt \\
&\leq \|s\| \int_0^1 \|\nabla_{\mathcal{I}}^2 f(x + t U_{\mathcal{I}} s) - \nabla_{\mathcal{I}}^2 f(x)\| \, dt \qquad (38) \\
&\leq L_{\mathcal{I}} \|s\|^2 \int_0^1 t \, dt \\
&= \frac{L_{\mathcal{I}}}{2} \|s\|^2,
\end{aligned}$$

where the last inequality follows from (6). Thus, (9) holds.

To show (10), by the mean value theorem we can write

$$f(x + U_{\mathcal{I}} s) - f(x) = \int_0^1 \nabla f(x + t U_{\mathcal{I}} s)^T U_{\mathcal{I}} s \, dt = \int_0^1 \nabla_{\mathcal{I}} f(x + t U_{\mathcal{I}} s)^T s \, dt, \qquad (39)$$

where we have used (2) in the last equality. Adding $-\nabla_{\mathcal{I}} f(x)^T s - \frac{1}{2} s^T \nabla^2_{\mathcal{I}} f(x) s$ to all terms, we obtain

$$\begin{aligned}
&\left| f(x + U_{\mathcal{I}} s) - f(x) - \nabla_{\mathcal{I}} f(x)^T s - \frac{1}{2} s^T \nabla^2_{\mathcal{I}} f(x) s \right| = \\
&\left| \int_0^1 (\nabla_{\mathcal{I}} f(x + t U_{\mathcal{I}} s) - \nabla_{\mathcal{I}} f(x) - t \nabla^2_{\mathcal{I}} f(x) s)^T s \, dt \right| \leq \\
&\int_0^1 |(\nabla_{\mathcal{I}} f(x + t U_{\mathcal{I}} s) - \nabla_{\mathcal{I}} f(x) - t \nabla^2_{\mathcal{I}} f(x) s)^T s| dt \leq \\
&\|s\| \int_0^1 \|\nabla_{\mathcal{I}} f(x + t U_{\mathcal{I}} s) - \nabla_{\mathcal{I}} f(x) - t \nabla^2_{\mathcal{I}} f(x) s\| dt \leq \\
&\frac{L_{\mathcal{I}}}{2} \|s\|^3 \int_0^1 t^2 \, dt = \\
&\frac{L_{\mathcal{I}}}{6} \|s\|^3,
\end{aligned}$$

where the last inequality follows from (9). Thus, (10) holds. □