Journal of Computational Mathematics
Vol.xx, No.x, 2024, 1–22.
http://www.global-sci.org/jcm
doi:10.4208/jcm.2308-m2023-0044

BINARY LEAST SQUARES: AN ALGORITHM FOR BINARY SPARSE SIGNAL RECOVERY*

Jinming Wen
College of Information Science and Technology, Jinan University, Guangzhou 510632, China
Email: jinming.wen@mail.mcgill.ca

Abstract

A fundamental problem in some applications including group testing and communications is to acquire the support of a K-sparse signal $\boldsymbol{x}$, whose nonzero elements are 1, from an underdetermined noisy linear model. This paper first designs an algorithm called binary least squares (BLS) to reconstruct $\boldsymbol{x}$ and analyzes its complexity. Then, we establish two sufficient conditions for the exact reconstruction of $\boldsymbol{x}$'s support with K iterations of BLS based on the mutual coherence and restricted isometry property of the measurement matrix, respectively. Finally, extensive numerical tests are performed to compare the efficiency and effectiveness of BLS with those of batch orthogonal matching pursuit (Batch-OMP) which to our best knowledge is the fastest implementation of OMP, orthogonal least squares (OLS), compressive sampling matching pursuit (CoSaMP), hard thresholding pursuit (HTP), Newton-step-based iterative hard thresholding (NSIHT), Newton-step-based hard thresholding pursuit (NSHTP), binary matching pursuit (BMP) and ℓ_1-regularized least squares. Test results show that: (1) BLS can be 10-200 times more efficient than Batch-OMP, OLS, CoSaMP, HTP, NSIHT and NSHTP with higher probability of support reconstruction, and the improvement can be 20%-80%; (2) BLS has more than 25% improvement on the support reconstruction probability than the explicit BMP algorithm with a little higher computational complexity; (3) BLS is around 100 times faster than ℓ_1-regularized least squares with lower support reconstruction probability for small K and higher support reconstruction probability for large K. Numerical tests on the generalized space shift keying (GSSK) detection indicate that although BLS is a little slower than BMP, it is more efficient than the other seven tested sparse recovery algorithms, and although it is less effective than ℓ_1-regularized least squares, it is more effective than the other seven algorithms.

Mathematics subject classification: 94A12, 65F22, 65J22.
Key words: Binary sparse signal, Precise support reconstruction, Binary least squares.

1. Introduction

The reconstruction of an unknown K-sparse $\boldsymbol{x}$ (i.e. $\boldsymbol{x}$ has no more than K nonzero entries) from the following underdetermined noisy linear measurements lies at the heart of compressive sensing [8, 12]:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{w}, \tag{1.1}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ is a given measurement vector, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is a given measurement matrix satisfying $m \ll n$, and $\boldsymbol{w} \in \mathbb{R}^m$ is a ℓ_2-bounded noise vector, i.e., $\boldsymbol{w}$ satisfies $\|\boldsymbol{w}\|_2 \leq \epsilon$ for certain small constant ϵ. There are other kinds of noises, for further details, see, e.g. [5, 37].

* Received February 28, 2023 / Revised version received June 1, 2023 / Accepted August 9, 2023 / Published online March 18, 2024 /

Although we assume $\boldsymbol{w}$ is a ℓ_2-bounded noise, the results in this work cannot be hard to be extended to other types of noise by following some techniques in e.g. [5, 36].

The problem of stably acquiring the K-sparse $\boldsymbol{x}$ from (1.1) arises in a massive number of applications, such as magnetic resonance imaging [23] and radar imaging [27]. While in some other applications, such as group testing [17], generalized space shift keying modulation detection [19] and active user detection [10], in addition to be K-sparse, $\boldsymbol{x}$ also fulfills

$$x_i = 1 \quad \text{for} \quad i \in \Omega, \tag{1.2}$$

where $\Omega = \{i | x_i \neq 0\}$ is the support of $\boldsymbol{x}$. Note that K-sparse $\boldsymbol{x}$ satisfying (1.2) is called binary K-sparse signal.

Although there is a large number of algorithms for reconstructing $\boldsymbol{x}$ from (1.1), such as greedy algorithms [11, 25, 26, 32], convex optimization algorithms [8, 12] and thresholding algorithms [2, 4, 30], and there are some studies on the reconstruction of binary sparse signals from (structured) biased measurement matrices [14, 20], there are few algorithms specifically designed for acquiring binary K-sparse $\boldsymbol{x}$ from (1.1) for any measurement matrix $\boldsymbol{A}$, and binary matching pursuit (BMP) [34] is the most recent one.

As explained in [34], designing an efficient and effective algorithm for reconstructing $\boldsymbol{x}$ satisfying (1.2) is of vital importance. Note that the reconstruction of such kind of sparse signals is challenging for orthogonal matching pursuit (OMP) [32] and it has special interest for the comparative study as they represent a particularly challenging case for OMP-type of reconstruction strategies [11]. Furthermore, it is emphasized in [3] that the recovery of sparse signals with equal magnitude nonzero entries is most demanding, and it was conjectured in [38] that the most difficult sparse inverse problem may involve nonzero coefficients with equal magnitudes. [18, Theorem 1] supports the observation, that reconstructing sparse vectors with equal magnitude nonzero coefficients correspond to the most difficult case for many recovery algorithms, by stating that, as long as the satisfaction of mutual coherence conditions for exact recovery is concerned, “flat” vectors (i.e. vectors whose nonzero entries are a constant) correspond to the worst possible case for OMP and OLS.

This work focuses on designing an efficient and effective binary sparse signal reconstruction algorithm and studying its performance. More exactly, we develop an iterative reconstruction algorithm called binary least squares (BLS). In each iteration, as orthogonal least squares (OLS) [9], BLS selects an index such that the residual vector is shortest. But different from OLS which mathematically solves a least squares problem to find the index, it uses (1.2) to find the index and does not solve any least squares problem, hence it is much more efficient than OLS. The new algorithm is a variant of OLS, and since it is designed for acquiring binary sparse signals, we call it BLS.

Although BMP is an iterative algorithm which also uses (1.2) to iteratively find the support of $\boldsymbol{x}$ in each iteration, its selection criterion is different from that of BLS. More exactly, in each iteration, BMP selects an index such that the absolute value of the inner product of the corresponding column of $\boldsymbol{A}$ and the current residual vector is maximized, which is different from that of BLS. Since a natural method to recover $\boldsymbol{x}$ is to minimize $\|\boldsymbol{x}\|_0$ subject to $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \leq \epsilon$, where $\|\boldsymbol{x}\|_0$ denotes the number of nonzero entries of $\boldsymbol{x}$, the selection strategy of BLS is closer to this method than that of BMP, hence BLS is expected to have better recovery performance than BMP. In fact, BLS is much more effective than BMP with a little higher complexity. Further details on this will be presented in Section 4.

To theoretically characterize the reconstruction performance of BLS, we develop two sufficient conditions for exactly acquiring the support of $\boldsymbol{x}$ with BLS based on $\boldsymbol{A}$'s mutual coherence [13] and restricted isometry property (RIP) [8], respectively. Note that the mutual coherence μ of $\boldsymbol{A}$ is defined as [13]

$$\mu = \max_{i \neq j} \frac{\left|\boldsymbol{A}_i^\top \boldsymbol{A}_j\right|}{\|\boldsymbol{A}_i\|_2 \|\boldsymbol{A}_j\|_2}, \tag{1.3}$$

and $\boldsymbol{A}$ satisfies the RIP [8] if for all K-sparse $\boldsymbol{x}$, it holds that

$$(1-\delta)\|\boldsymbol{x}\|_2^2 \leq \|\boldsymbol{A}\boldsymbol{x}\|_2^2 \leq (1+\delta)\|\boldsymbol{x}\|_2^2. \tag{1.4}$$

Moreover, the minimal δ is called the restricted isometry constant (RIC) of $\boldsymbol{A}$ and is denoted as δ_K [8].

The contributions of this article are summarized as follows:

- We develop BLS to acquire K-sparse $\boldsymbol{x}$ satisfying (1.2) from (1.1) and show that its computational complexity is

  $$2mn(N+1) + N(2m+1) - mN^2$$

  flops (i.e., the total numbers of addition, substraction, multiplication and division), where N is the iteration number of BLS, and further prove that the complexity can be reduced to

  $$N\left[2mn + (m+1) - \left((N-1)\left(m - \frac{1}{2}\right) + n\right)\right]$$

  flops if $\boldsymbol{A}$ has equal column norm.

- We prove that if the mutual coherence μ of $\boldsymbol{A}$ and the noise level ϵ respectively satisfy

  $$\mu < \frac{\beta^2}{(2K-1)\alpha^2}, \quad \epsilon < \frac{\beta^2 - (2K-1)\alpha^2\mu}{2\alpha}, \tag{1.5}$$

  where α, β are respectively the largest and smallest Euclidean norm of the columns of $\boldsymbol{A}$, then with certain stopping condition, BLS can accurately acquire $\boldsymbol{x}$'s support with at most K iterations.

- We prove that if $\boldsymbol{A}$ and the noise level ϵ respectively satisfy

  $$\delta_{K+1} < \frac{1}{\sqrt{K+1}+1}, \quad \epsilon < \frac{1 - (\sqrt{K+1}+1)\delta_{K+1}}{2}, \tag{1.6}$$

  then with certain stopping condition, BLS can accurately acquire $\boldsymbol{x}$'s support with at most K iterations. We further show that the two conditions can be respectively relaxed to

  $$\delta_{K+1} < \frac{1}{\sqrt{K+1}}, \quad \epsilon < \frac{1 - \sqrt{K+1}\delta_{K+1}}{2},$$

  if $\boldsymbol{A}$ has equal column norm.

- We performed extensive numerical tests to compare the efficiency and effectiveness of BLS with those of batch orthogonal matching pursuit (Batch-OMP) (which to our best knowledge is the most efficient implementation of OMP [29]), OLS, compressive sampling matching pursuit (CoSaMP) [25], hard thresholding pursuit (HTP) [15], Newton-step-based iterative hard thresholding (NSIHT) [24], Newton-step-based hard thresholding pursuit (NSHTP) [24], BMP [34] and ℓ_1-regularized least squares [21]. The test results show that:

(1) BLS can be 10-200 times faster than Batch-OMP, OLS, CoSaMP, HTP, NSIHT and NSHTP with 20%-80% higher probability of support reconstruction.

(2) BLS has more than 25% higher probability of support recovery than the explicit BMP with a little higher computational complexity.

(3) BLS is around 100 times faster than ℓ_1-regularized least squares with lower support reconstruction probability for small K and higher support reconstruction probability for large K.

Furthermore, numerical tests on the real application of the generalized space shift keying (GSSK) detection also indicate that although BLS is a little slower than BMP, it is more efficient than the other seven tested algorithms, and although it is less effective than ℓ_1-regularized least squares, it is more effective than the other seven tested algorithms.

The remainder of this article is organized as follows. The BLS algorithm and its complexity analysis are presented in Section 2. Section 3 establishes two sufficient conditions of accurate reconstruction of $\boldsymbol{x}$'s support with at most K iterations of BLS. Section 4 conducts extensive numerical tests to explain BLS's efficiency and good reconstruction ability. Finally, we conclude this article and present some future research problems in Section 5.

Notation. Let $\emptyset$ denote the empty set, Ω and $|\Omega|$ respectively denote the support of $\boldsymbol{x}$ and its cardinality. For any set $\Gamma \subseteq \{1,2,\ldots,n\}$, Γ^c represents the complementary set of Γ and $\Omega\setminus\Gamma$ represents the set that is the combination of Ω and Γ^c, $\boldsymbol{x}_\Gamma \in \mathbb{R}^{|\Gamma|}$ denotes the subvector of $\boldsymbol{x}$ with entries indexed by Γ, and $\boldsymbol{A}_\Gamma \in \mathbb{R}^{m\times|\Gamma|}$ denotes the submatrix of $\boldsymbol{A}$ with columns indexed by Γ. Furthermore, denote the transpose of $\boldsymbol{A}_\Gamma$ by $\boldsymbol{A}_\Gamma^\top$ and $\boldsymbol{P}_\Gamma^\perp = \boldsymbol{I} - \boldsymbol{A}_\Gamma(\boldsymbol{A}_\Gamma^\top\boldsymbol{A}_\Gamma)^{-1}\boldsymbol{A}_\Gamma^\top$.

2. Our New Algorithm

In this section, we design BLS to reconstruct K-sparse $\boldsymbol{x}$ satisfying (1.2) from (1.1), study its complexity and compare it with some related sparse recovery algorithms.

2.1. Binary least squares

BLS is an iterative algorithm. In each iteration, it uses (1.2) to select an index such that the residual vector is shortest. Specifically, suppose that BLS performs N iterations, let Γ_{k-1} denote the estimated support and $\boldsymbol{r}^{k-1} := \boldsymbol{y} - \boldsymbol{A}_{\Gamma^{k-1}}$ denote the residual vector at the $(k-1)$-th iteration, where $1 \le k \le N$, $\Gamma_0 = \emptyset$ and $\boldsymbol{r}^0 = \boldsymbol{y}$, then at the k-th iteration, BLS selects $s_k \in \Gamma_{k-1}^c$ such that

$$\begin{aligned} s_k &= \operatorname*{argmin}_{i\in\Gamma_{k-1}^c} \|\boldsymbol{r}^k\|_2^2 = \operatorname*{argmin}_{i\in\Gamma_{k-1}^c} \big\|\boldsymbol{y} - \boldsymbol{A}_{\Gamma^{k-1}\cup\{i\}}\big\|_2^2 \\ &= \operatorname*{argmin}_{i\in\Gamma_{k-1}^c} \|\boldsymbol{y} - \boldsymbol{A}_{\Gamma^{k-1}} - \boldsymbol{A}_i\|_2^2 = \operatorname*{argmin}_{i\in\Gamma_{k-1}^c} \big\|\boldsymbol{r}^{k-1} - \boldsymbol{A}_i\big\|_2^2 \\ &= \operatorname*{argmax}_{i\in\Gamma_{k-1}^c} \boldsymbol{A}_i^\top\big(2\boldsymbol{r}^{k-1} - \boldsymbol{A}_i\big). \end{aligned} \tag{2.1}$$

By some simple calculations, we can see that the cost of finding s_k that satisfies (2.1) is

$$(2m + 2m - 1)(n - k + 1) = (4m - 1)(n - k + 1)$$

flops. Suppose that BLS performs N iterations, then the total cost of finding s_i for $1 \leq i \leq N$ is

$$\sum_{k=1}^{N}(n - k + 1)(4m - 1) = N(4m - 1)\left(n - \frac{N-1}{2}\right)$$

flops, which is high. To reduce the total cost of selection, we define $\boldsymbol{c} \in \mathbb{R}^n$ such that

$$c_i = \frac{1}{2}\|\boldsymbol{A}_i\|_2^2, \quad 1 \leq i \leq n, \tag{2.2}$$

then

$$\boldsymbol{A}_i^\top \left(2\boldsymbol{r}^{k-1} - \boldsymbol{A}_i\right) = 2\left(\boldsymbol{A}_i^\top \boldsymbol{r}^{k-1} - c_i\right).$$

Hence, by (2.1), we find $s_k \in \Gamma_{k-1}^c$ which satisfies

$$s_k = \underset{i \in \Gamma_{k-1}^c}{\operatorname{argmax}} \left(\boldsymbol{A}_i^\top \boldsymbol{r}^{k-1} - c_i\right), \tag{2.3}$$

and the cost is $2m(n - k + 1)$ flops.

After finding s_k, we let

$$\hat{x}_{s_k} = 1, \quad \Gamma_k = \Gamma_{k-1} \cup \{s_k\}, \quad \boldsymbol{r}^k = \boldsymbol{r}^{k-1} - \boldsymbol{A}_{s_k},$$

where $\hat{\boldsymbol{x}} \in \mathbb{R}^n$ is used to estimate the binary K-sparse $\boldsymbol{x}$. According to the above explanation, the BLS algorithm is described in Algorithm 2.1.

Algorithm 2.1: Binary Least Squares.

Input : $\boldsymbol{y}$ and $\boldsymbol{A}$.
Initialize: $k = 0, \boldsymbol{r}^0 = \boldsymbol{y}, \hat{\boldsymbol{x}} = \boldsymbol{0}, \Gamma_0 = \emptyset$ and $\boldsymbol{c} \in \mathbb{R}^n$ with $c_i = \|\boldsymbol{A}_i\|_2^2/2$ for $1 \leq i \leq n$.
1 **while** stopping condition is not satisfied **do**
2 $\quad k = k + 1$.
3 $\quad s_k = \operatorname{argmax}_{i \in \Gamma_{k-1}^c} (\boldsymbol{A}_i^\top \boldsymbol{r}^{k-1} - c_i)$.
4 $\quad \hat{x}_{s_k} = 1$.
5 $\quad \Gamma_k = \Gamma_{k-1} \cup \{s_k\}$.
6 $\quad \boldsymbol{r}^k = \boldsymbol{r}^{k-1} - \boldsymbol{A}_{s_k}$.
7 **end**
Output: Γ_k and $\hat{\boldsymbol{x}}$.

As OMP and BMP, in each iteration, we explicitly form the residual vector $\boldsymbol{r}^k$, which can be used to set a stopping condition for BLS. Alternatively, we may let the algorithm terminate after performing certain iterations. Note that if $\boldsymbol{A}$ has equal column norm, then step 3 of Algorithm 2.1 can be reduced to

$$s_k = \underset{i \in \Gamma_{k-1}^c}{\operatorname{argmax}}\, \boldsymbol{A}_i^\top \boldsymbol{r}^{k-1}, \tag{2.4}$$

which is very similar to the selection strategy of OMP and BMP, that is

$$s_k = \underset{i \in \Gamma_{k-1}^c}{\operatorname{argmax}} \left|\boldsymbol{A}_i^\top \boldsymbol{r}^{k-1}\right|. \tag{2.5}$$

2.2. Complexity analysis of BLS

In the following, we analyze the complexity of Algorithm 2.1. By (2.2), the cost for computing c is $2mn$ flops. Furthermore, if BLS performs $N \le n$ iterations, then step 2 and steps 4-6 of Algorithm 2.1 cost $(m+1)N$ flops in total, and step 3 costs

$$2m\sum_{k=1}^{N}(n-k+1) = mN(2n-N+1)$$

flops. Hence, if BLS performs N iterations, then the total complexity of Algorithm 2.1 is

$$2mn + (m+1)N + mN(2n-N+1) = 2mn(N+1) + N(2m+1) - mN^2$$

flops.

Furthermore, by the above analysis, if $\boldsymbol{A}$ has equal column norm, then we do not need to form $\boldsymbol{c}$ and we can use (2.4) to select the index s_k. Therefore, if BLS performs N iterations, then its complexity is

$$\begin{aligned} &N(m+1) + \sum_{k=1}^{N}[(n-k+1)(2m-1)] \\ =&2mnN + N(m+1) - N\left[(N-1)\left(m-\frac{1}{2}\right)+n\right] \end{aligned} \tag{2.6}$$

flops.

2.3. Comparison with existing sparse recovery algorithms

The most related sparse recovery algorithms to BLS are the explicit BMP algorithm [34, Algorithm 2] and the OLS algorithm (see, e.g. [35, Table 1]), hence, in this subsection, we briefly compare BLS with these two algorithms.

From Algorithm 2.1 and [34, Algorithm 2], one can see that the only difference between BMP and BLS is step 3, i.e. the selection strategy. More exactly, in the k-th iteration, BMP uses (2.5) to select s_k, while BLS uses (2.3) or (2.4) (when $\boldsymbol{A}$ has equal column norm) to select s_k. Since both BMP and BLS use the binary property of $\boldsymbol{x}$, it is not strange that this is their only difference. Note that this improved selection strategy leads to an improved recovery performance, for more details, see the simulation part in Section 4.

To the best of our knowledge, the explicit BMP algorithm, whose complexity is

$$2mnN + N(m+1) - N\left[(N-1)\left(m-\frac{1}{2}\right)+n\right]$$

flops if it runs N iterations [34], is the most efficient binary sparse recovery algorithm. By (2.6), we can see that the complexity of BLS equals to that of BMP if $\boldsymbol{A}$ has equal column norm, and otherwise the complexity of BLS is $n(2m+N)-(N-1)N/2$ flops higher than that of BMP.

It is worth mentioning that, for Bernoulli measurement matrix $\boldsymbol{A}$ with a_{ij} independently and identically randomly chosen from $\{-1/\sqrt{m}, 1/\sqrt{m}\}$ with equal probability, it has equal column norm. Hence, the theoretical computational complexity of BLS is the same as that of BMP. But as will be shown in Section 4, BLS has much better recovery performance than BMP. For other measurement matrices which do not have equal column norm, as will be shown in

Section 4, although BLS is a little bit more expensive than BMP, it has much better recovery performance than BMP.

In the following, we compare BLS with OLS. From Algorithm 2.1 and [35, Table 1], one can see that there are two differences between BLS and OLS: steps 3, and 5-6. Although the main ideas of their selection strategies are the same, i.e. finding s_k such that

$$s_k = \underset{i\in\{1,\dots,n\}}{\operatorname{argmin}} \left\| \boldsymbol{P}^{\perp}_{\Gamma^{k-1}\cup\{i\}} \boldsymbol{y} \right\|_2^2 = \underset{i\in\{1,\dots,n\}}{\operatorname{argmin}} \|\boldsymbol{r}^k\|_2^2,$$

they may choose different index for $k \geq 2$ because OLS compute

$$\hat{x}_{s_{k-1}} = \left(\boldsymbol{A}^{\top}_{\Gamma^{k-1}} \boldsymbol{A}_{\Gamma^{k-1}}\right)^{-1} \boldsymbol{A}^{\top}_{\Gamma^{k-1}} y,$$

which is different from BLS that sets $\hat{x}_{s_{k-1}} = 1$. This leads to the difference between the residual vector $\boldsymbol{r}^k$ computed by OLS and BLS. Furthermore, step 3 of BLS is much more efficient than step 3 of OLS. This difference significantly improves not only efficiency but also recovery performance, for more details, see Section 4.

Since step 4 of BLS uses (1.2), while step 5 of OLS does not use (1.2), this difference also improves both efficiency and recovery performance. From step 6 of BLS and OLS, one can see that BLS is much more efficient than OLS.

3. Sufficient Conditions for Accurate Support Reconstruction with BLS Algorithm

In this section, we respectively use the mutual coherence and RIP of the measurement matrix $\boldsymbol{A}$ to establish two sufficient conditions for the accurate support reconstruction with BLS.

3.1. Sufficient condition for BLS based on the mutual coherence of $\boldsymbol{A}$

Theorem 3.1 below establishes a sufficient condition for the precise recovery of $\boldsymbol{x}$'s support with BLS based on the mutual coherence of $\boldsymbol{A}$.

Theorem 3.1. *Suppose that K-sparse $\boldsymbol{x}$ satisfies* (1.2) *and let*

$$\alpha := \max_{1\leq i\leq n} \|\boldsymbol{A}_i\|_2, \quad \beta := \min_{1\leq i\leq n} \|\boldsymbol{A}_i\|_2. \tag{3.1}$$

If the mutual coherence μ of $\boldsymbol{A}$ satisfies

$$\mu < \frac{\beta^2}{(2K-1)\alpha^2}, \tag{3.2}$$

and $\boldsymbol{w}$ satisfies $\|\boldsymbol{w}\|_2 \leq \epsilon$ with

$$\epsilon < \frac{\beta^2 - (2K-1)\alpha^2\mu}{2\alpha}, \tag{3.3}$$

then with the stopping condition $\|\boldsymbol{r}^k\|_2 \leq \epsilon$, BLS can accurately acquire $\boldsymbol{x}$'s support with at most K iterations.

Proof. As the proof of [37, Theorem 1] and [34, Theorem 1], we prove that BLS selects an element of the support Ω of $\boldsymbol{x}$ in each iteration and performs exactly $|\Omega|$ iterations. We prove the former by induction. By Algorithm 2.1, $\Gamma_0 = \emptyset$. Suppose that $\Gamma_{k-1} \subseteq \Omega$ for certain k with $1 \le k \le |\Omega|$, then by induction and Algorithm 2.1, it is equivalent to prove that $s_k \in \Omega$. By step 3 of Algorithm 2.1, to prove $s_k \in \Omega$, it suffices to prove that

$$\max_{i\in\Omega\setminus\Gamma_{k-1}} \big(\boldsymbol{A}_i^\top \boldsymbol{r}^{k-1} - c_i\big) > \max_{j\in\Omega^c} \big(\boldsymbol{A}_j^\top \boldsymbol{r}^{k-1} - c_j\big). \tag{3.4}$$

By (1.1) and line 6 of Algorithm 2.1, we obtain

$$\boldsymbol{r}^{k-1} = \boldsymbol{y} - \sum_{i\in\Gamma_{k-1}} \boldsymbol{A}_i = \boldsymbol{A}_\Omega \boldsymbol{x}_\Omega + \boldsymbol{w} - \sum_{i\in\Gamma_{k-1}} \boldsymbol{A}_i = \sum_{i\in\Omega\setminus\Gamma_{k-1}} \boldsymbol{A}_i + \boldsymbol{w}, \tag{3.5}$$

where the last equality is due to (1.2). Suppose that

$$i_0 = \operatorname*{argmax}_{i\in\Omega\setminus\Gamma_{k-1}} \big(\boldsymbol{A}_i^\top \boldsymbol{r}^{k-1} - c_i\big), \quad j_0 = \operatorname*{argmax}_{j\in\Omega^c} \big(\boldsymbol{A}_j^\top \boldsymbol{r}^{k-1} - c_j\big). \tag{3.6}$$

Then by (2.2) and (3.5), we get

$$\begin{aligned}
\boldsymbol{A}_{i_0}^\top \boldsymbol{r}^{k-1} - c_{i_0} &= \boldsymbol{A}_{i_0}^\top \sum_{i\in\Omega\setminus\Gamma_{k-1}} \boldsymbol{A}_i + \boldsymbol{A}_{i_0}^\top \boldsymbol{w} - c_{i_0} \\
&\overset{(a)}{=} \boldsymbol{A}_{i_0}^\top \left(\boldsymbol{A}_{i_0} + \sum_{i\in\Omega\setminus(\Gamma_{k-1}\cup i_0)} \boldsymbol{A}_i \right) + \boldsymbol{A}_{i_0}^\top \boldsymbol{w} - \frac{\|\boldsymbol{A}_{i_0}\|_2^2}{2} \\
&\ge \frac{\|\boldsymbol{A}_{i_0}\|_2^2}{2} - \sum_{i\in\Omega\setminus(\Gamma_{k-1}\cup i_0)} \big|\boldsymbol{A}_{i_0}^\top \boldsymbol{A}_i\big| - \big|\boldsymbol{A}_{i_0}^\top \boldsymbol{w}\big| \\
&\overset{(b)}{\ge} \frac{\beta^2}{2} - |\Omega\setminus(\Gamma_{k-1}\cup i_0)|\alpha^2\mu - \alpha\|\boldsymbol{w}\|_2 \\
&\ge \frac{\beta^2}{2} - (K-1)\alpha^2\mu - \alpha\|\boldsymbol{w}\|_2,
\end{aligned} \tag{3.7}$$

where (a) is due to (2.2), and (b) is from (1.3), (3.1) and the Cauchy-Schwarz inequality. Similarly, by (3.5), we have

$$\begin{aligned}
\boldsymbol{A}_{j_0}^\top \boldsymbol{r}^{k-1} - c_{j_0} &= \boldsymbol{A}_{j_0}^\top \sum_{i\in\Omega\setminus\Gamma_{k-1}} \boldsymbol{A}_i + \boldsymbol{A}_{j_0}^\top \boldsymbol{w} - c_{j_0} \\
&\le \sum_{i\in\Omega\setminus\Gamma_{k-1}} \big|\boldsymbol{A}_{j_0}^\top \boldsymbol{A}_i\big| + \big|\boldsymbol{A}_{j_0}^\top \boldsymbol{w}\big| - c_{j_0} \\
&\le |\Omega\setminus\Gamma_{k-1}|\alpha^2\mu + \left(\alpha\|\boldsymbol{w}\|_2 - \frac{\beta^2}{2}\right) \\
&\le K\alpha^2\mu - \frac{\beta^2}{2} + \alpha\|\boldsymbol{w}\|_2,
\end{aligned} \tag{3.8}$$

where the second to the last inequality is from (2.2) and (3.1).

Then, by (3.7) and (3.8), to show (3.4), it suffices to show that

$$\beta^2 - (2K-1)\alpha^2\mu \ge 2\alpha\|\boldsymbol{w}\|_2. \tag{3.9}$$

Since $\|\boldsymbol{w}\|_2 \le \epsilon$, by (3.2) and (3.3), (3.9) holds. Therefore, $s_k \in \Omega$ for $1 \le k \le |\Omega|$.

In the following, we show that with the stopping condition $\|\boldsymbol{r}^k\|_2 \leq \epsilon$, BLS runs precisely $|\Omega|$ iterations. Let $\sigma_{\min}$ denote the minimum singular value of $\boldsymbol{A}^\top_{\Omega\setminus\Gamma_{k-1}}\boldsymbol{A}_{\Omega\setminus\Gamma_{k-1}}$, then, by the Gerschgorin theorem, we obtain

$$\begin{aligned}\sigma_{\min} &\geq \min_{i\in\Omega\setminus\Gamma_{k-1}}\left(\|\boldsymbol{A}_i\|_2^2 - \sum_{j\in\Omega\setminus(\Gamma_{k-1}\cup i)}|\boldsymbol{A}_i^\top\boldsymbol{A}_j|\right)\\ &\geq \beta^2-(K-1)\alpha^2\mu, \end{aligned}\tag{3.10}$$

where the second inequality is from (1.3) and (3.1). Hence, we have

$$\begin{aligned}\|\boldsymbol{r}^{k-1}\|_2 &\overset{(a)}{=} \|\boldsymbol{A}_{\Omega\setminus\Gamma_{k-1}}\boldsymbol{x}_{\Omega\setminus\Gamma_{k-1}}+\boldsymbol{w}\|_2\\ &\overset{(b)}{\geq} \|\boldsymbol{A}_{\Omega\setminus\Gamma_{k-1}}\boldsymbol{x}_{\Omega\setminus\Gamma_{k-1}}\|_2-\epsilon \overset{(c)}{\geq} \sqrt{\sigma_{\min}}-\epsilon\\ &\overset{(d)}{\geq} \sqrt{\beta^2-(K-1)\alpha^2\mu}-\epsilon = \alpha\sqrt{\frac{1}{\alpha^2}\big(\beta^2-(K-1)\alpha^2\mu\big)}-\epsilon\\ &\overset{(e)}{\geq} \alpha\frac{1}{\alpha^2}\big(\beta^2-(K-1)\alpha^2\mu\big)-\epsilon > \epsilon, \end{aligned}\tag{3.11}$$

where (a) is from (1.2) and (3.5), (b) is because $\|\boldsymbol{w}\|_2 \leq \epsilon$, (c) follows from (1.2) and the fact that $\sigma_{\min}$ denotes the minimum singular value of $\boldsymbol{A}^\top_{\Omega\setminus\Gamma_{k-1}}\boldsymbol{A}_{\Omega\setminus\Gamma_{k-1}}$, and (d) is due to (3.10), (e) is since

$$\frac{1}{\alpha^2}\big(\beta^2-(K-1)\alpha^2\mu\big) \leq 1,$$

(see (3.1)) and the last inequality follows from (3.3).

By (3.11), we can see that under the stopping condition $\|\boldsymbol{r}^k\|_2 \leq \epsilon$, BLS performs at least $|\Omega|$ iterations. Furthermore, by (3.5), we have

$$\|\boldsymbol{r}^{|\Omega|}\|_2 = \left\|\sum_{i\in\Omega\setminus\Omega}\boldsymbol{A}_i+\boldsymbol{w}\right\|_2 = \|\boldsymbol{w}\|_2 \leq \epsilon. \tag{3.12}$$

Therefore, BLS terminates after performing $|\Omega|$ iterations. Hence, BLS performs exactly $|\Omega|$ iterations. The proof is complete. □

From the above proof, we can see that, as the proof of the sufficient conditions of OMP and OLS, in each iteration, we need to prove (3.4). But different from OMP and BLS which mathematically solve a least squares problem to update $\boldsymbol{x}$, BLS just sets the new element of $\boldsymbol{x}$ as 1 in each iteration. Furthermore, the selection strategy of BLS is different from that of BMP. Hence, the main techniques of the above proof are different from those of OMP, OLS and BMP.

In many references, like [5, 6, 31, 34, 35], for simplicity, it is assumed that $\boldsymbol{A}$ is column normalized. Under this condition, $\alpha = \beta = 1$ (see (3.1)). Hence, the following corollary can be obtained from Theorem 3.1.

Corollary 3.1. *Suppose that K-sparse $\boldsymbol{x}$ satisfies* (1.2) *and the mutual coherence of column normalized $\boldsymbol{A}$ satisfies*

$$\mu < \frac{1}{2K-1}. \tag{3.13}$$

If $\boldsymbol{w}$ satisfies $\|\boldsymbol{w}\|_2 \leq \epsilon$ with

$$\epsilon < \frac{1-(2K-1)\mu}{2},$$

then BLS can accurately acquire $\boldsymbol{x}$'s support with at most K iterations under the stopping condition $\|\boldsymbol{r}^k\|_2 \leq \epsilon$.

By [6, Theorem 3.1], there is a K-sparse $\boldsymbol{x}$ satisfies (1.2) and a column normalized measurement matrix $\boldsymbol{A}$ whose mutual coherence satisfies $\mu = 1/(2K-1)$ such that there is no algorithm can exactly recover $\boldsymbol{x}$. Hence, (3.13) is a necessary condition for any sparse recovery algorithm to reconstruct all K-sparse signals. In other words, there is no better sufficient condition than (3.13) for any sparse recovery algorithm that can reconstruct all K-sparse signals. Since (3.2) reduces to (3.13) when $\boldsymbol{A}$ is column normalized, (3.2) is a tight sufficient condition for column normalized $\boldsymbol{A}$. Note that, (3.13) is also a sufficient condition for ℓ_1-minimization method [6], OMP [37] and BMP [34]. Whether sufficient condition (3.2) can be further improved requires further investigation.

3.2. Sufficient condition for BLS based on the RIP of $\boldsymbol{A}$

In this subsection, we study the performance of BLS with the RIP of $\boldsymbol{A}$. We begin with introducing the following lemma.

Lemma 3.1 ([25, Proposition 3.2]). *Suppose that $\boldsymbol{A}$ satisfies the RIP of order k, Γ_1 and Γ_2 are two disjoint sets with $|\Gamma_1 \cup \Gamma_2| \le k$. Then,*

$$\left\|\boldsymbol{A}_{\Gamma_1}^\top \boldsymbol{A}_{\Gamma_2}\right\|_2 \le \delta_k.$$

Theorem 3.2 below establishes a sufficient condition for the precise recovery of $\boldsymbol{x}$'s support with BLS based on the RIP of $\boldsymbol{A}$.

Theorem 3.2. *Suppose that K-sparse $\boldsymbol{x}$ satisfies* (1.2) *and $\boldsymbol{A}$ satisfies*

$$\delta_{K+1} < \frac{1}{\sqrt{K+1}+1}. \tag{3.14}$$

If $\boldsymbol{w}$ satisfies $\|\boldsymbol{w}\|_2 \le \epsilon$ with

$$\epsilon < \frac{1-(\sqrt{K+1}+1)\delta_{K+1}}{2}, \tag{3.15}$$

then BLS can accurately acquire $\boldsymbol{x}$'s support with at most K iterations under the stopping condition $\|\boldsymbol{r}^k\|_2 \le \epsilon$.

Proof. As the proof of Theorem 3.1, to prove Theorem 3.2, we only need to show that $s_k \in \Omega$ for $1 \le k \le |\Omega|$ and BLS performs $|\Omega|$ iterations. We first prove the former. According to the proof of Theorem 3.1, it suffices to show (3.4).

Let $k_0 \in \Omega \backslash \Gamma_{k-1}$ such that

$$k_0 = \underset{i \in \Omega\backslash\Gamma_{k-1}}{\operatorname{argmax}} \left|\boldsymbol{A}_i^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right|. \tag{3.16}$$

Then

$$\begin{aligned}
\boldsymbol{A}_{k_0}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}} &= \boldsymbol{A}_{k_0}^\top \left(\boldsymbol{A}_{k_0} + \boldsymbol{A}_{\Omega\backslash(\Gamma_{k-1}\cup k_0)} \boldsymbol{x}_{\Omega\backslash(\Gamma_{k-1}\cup k_0)}\right) \\
&\ge \|\boldsymbol{A}_{k_0}\|_2^2 - \left|\boldsymbol{A}_{k_0}^\top \boldsymbol{A}_{\Omega\backslash(\Gamma_{k-1}\cup k_0)} \boldsymbol{x}_{\Omega\backslash(\Gamma_{k-1}\cup k_0)}\right| \\
&\overset{(a)}{\ge} \|\boldsymbol{A}_{k_0}\|_2^2 - \left\|\boldsymbol{A}_{k_0}^\top \boldsymbol{A}_{\Omega\backslash(\Gamma_{k-1}\cup k_0)}\right\|_2 \left\|\boldsymbol{x}_{\Omega\backslash(\Gamma_{k-1}\cup k_0)}\right\|_2 \\
&\overset{(b)}{\ge} 1 - \delta_{K+1} - \sqrt{K-1}\,\delta_{K+1} > 0,
\end{aligned}$$

where (a) is due to the Cauchy-Schwarz inequality, (b) is according to (1.2), (1.4) and Lemma 3.1, and the last inequality is from (3.14). Therefore, by (3.16), we have

$$\begin{aligned} \boldsymbol{A}_{k_0}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}} &= \left|\boldsymbol{A}_{k_0}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right| \\ &= \left\|\boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty. \end{aligned} \tag{3.17}$$

Then, we obtain

$$\begin{aligned} \max_{i\in\Omega\backslash\Gamma_{k-1}} \left(\boldsymbol{A}_i^\top \boldsymbol{r}^{k-1} - c_i\right) &\overset{(a)}{=} \max_{i\in\Omega\backslash\Gamma_{k-1}} \left[\boldsymbol{A}_i^\top \left(\sum_{j\in\Omega\backslash\Gamma_{k-1}} \boldsymbol{A}_j + \boldsymbol{w}\right) - c_i\right] \\ &\overset{(b)}{\geq} \boldsymbol{A}_{k_0}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}} + \boldsymbol{A}_{k_0}^\top \boldsymbol{w} - c_{k_0} \\ &\overset{(c)}{\geq} \left\|\boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty - \left|\boldsymbol{A}_{k_0}^\top \boldsymbol{w}\right| - \frac{\|\boldsymbol{A}_{i_0}\|_2^2}{2} \\ &\geq \left\|\boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty - \frac{1+\delta_{K+1}}{2} - \left|\boldsymbol{A}_{k_0}^\top \boldsymbol{w}\right|, \end{aligned} \tag{3.18}$$

where (a) follows from (3.5), (b) is due to (1.2) and $k_0 \in \Omega\backslash\Gamma_{k-1}$ (see (3.16)), (c) is from (2.2) and (3.17), and the last inequality follows from (1.4). Let j_0 be defined as in (3.6), then by (1.2), (2.2) and (3.5), we have

$$\begin{aligned} \max_{j\in\Omega^c} \left(\boldsymbol{A}_i^\top \boldsymbol{r}^{k-1} - c_j\right) &\leq \left|\boldsymbol{A}_{j_0}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right| + \left|\boldsymbol{A}_{j_0}^\top \boldsymbol{w}\right| - \frac{\|\boldsymbol{A}_{j_0}\|_2^2}{2} \\ &\leq \left\|\boldsymbol{A}_{\Omega^c}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty - \frac{1-\delta_{K+1}}{2} + \left|\boldsymbol{A}_{j_0}^\top \boldsymbol{w}\right|, \end{aligned} \tag{3.19}$$

where the second inequality is due to (1.4).

By (3.18) and (3.19), to prove (3.4), we only need to prove that

$$\left\|\boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty - \left\|\boldsymbol{A}_{\Omega^c}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty - \delta_{K+1} \geq \left|\boldsymbol{A}_{i_0}^\top \boldsymbol{w}\right| + \left|\boldsymbol{A}_{j_0}^\top \boldsymbol{w}\right|. \tag{3.20}$$

By [34, Eq. (D2)], we obtain

$$\left\|\boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty - \left\|\boldsymbol{A}_{\Omega^c}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty \geq 1 - \sqrt{K+1}\,\delta_{K+1}. \tag{3.21}$$

Furthermore, by [34, Eq. (B6)], we have

$$\left|\boldsymbol{A}_{i_0}^\top \boldsymbol{w}\right| + \left|\boldsymbol{A}_{j_0}^\top \boldsymbol{w}\right| \leq 2\epsilon. \tag{3.22}$$

By (3.14), (3.15), (3.21) and (3.22), one can see that (3.20) holds.

By (3.12), $\|\boldsymbol{r}^{|\Omega|}\|_2 \leq \epsilon$, and according to the proof of [34, Theorem 3], $\|\boldsymbol{r}^k\|_2 \geq \epsilon$ for $1 \leq k < |\Omega|$. Hence, under the stopping condition $\|\boldsymbol{r}^k\|_2 \leq \epsilon$, BLS performs exactly $|\Omega|$ iterations. The proof is complete. □

If $\boldsymbol{A}$ has equal column norm, then by the second last inequalities of (3.18) and (3.19), (3.20) can be reduced to

$$\left\|\boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty - \left\|\boldsymbol{A}_{\Omega^c}^\top \boldsymbol{A}_{\Omega\backslash\Gamma_{k-1}} \boldsymbol{x}_{\Omega\backslash\Gamma_{k-1}}\right\|_\infty \geq \left|\boldsymbol{A}_{i_0}^\top \boldsymbol{w}\right| + \left|\boldsymbol{A}_{j_0}^\top \boldsymbol{w}\right|.$$

Then, by (3.21) and (3.22), we have the following corollary.

Corollary 3.2. *Suppose that K-sparse $\boldsymbol{x}$ satisfies* (1.2), $\boldsymbol{A}$ *has equal column norm and satisfies the RIP with*

$$\delta_{K+1} < \frac{1}{\sqrt{K+1}}. \tag{3.23}$$

If $\boldsymbol{w}$ satisfies $\|\boldsymbol{w}\|_2 \leq \epsilon$ with

$$\epsilon < \frac{1-\sqrt{K+1}\delta_{K+1}}{2},$$

then BLS can accurately acquire $\boldsymbol{x}$'s support with at most K iterations under the stopping condition $\|\boldsymbol{r}^k\|_2 \leq \epsilon$.

It has been shown in [22] that OLS can accurately recover the support of any K-sparse $\boldsymbol{x}$ if $\boldsymbol{A}$ is column normalized and satisfies

$$\delta_{K+1} \leq \begin{cases} \frac{1}{\sqrt{K}}, & \text{if} \quad K=1, \\ \frac{1}{\sqrt{K+1/4}}, & \text{if} \quad K=2, \\ \frac{1}{\sqrt{K+1/16}}, & \text{if} \quad K=3, \\ \frac{1}{\sqrt{K}}, & \text{if} \quad K \geq 4, \end{cases}$$

and further show that it is an optimal sufficient condition. The above sufficient condition for OLS is weaker than the sufficient condition (3.23) for BLS. Hence, we think by following the techniques in [22], one can improve (3.23) to obtain an optimal sufficient condition, and we leave it as a future research problem.

3.3. Comparison with existing sufficient conditions

Although for measurement matrix $\boldsymbol{A}$ which is not column normalized, the mutual coherence based sufficient condition (3.2) of BLS is not as sharp as the sufficient condition (3.13) of OMP and BMP, and the RIP based sufficient condition (3.14) of BLS is not as sharp as the sufficient condition (3.23) of OMP and BMP, numerical tests in Section 4 will show that BLS is much more effective than both OMP and BMP in recovering the support of $\boldsymbol{x}$. Furthermore, there exist $\boldsymbol{A}$ and $\boldsymbol{x}$ for which neither OMP nor BMP can acquire $\boldsymbol{x}$ based on $\boldsymbol{y}$ and $\boldsymbol{A}$, but BLS can recover it. Example 3.1 below shows this.

Example 3.1. For any integer $K \geq 2$ and

$$1-\sqrt{1-\frac{2K}{(K+1)^2}} < t < \frac{1}{K+1}, \tag{3.24}$$

let $\boldsymbol{w} = \boldsymbol{0}$, and

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{e} \\ 0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} \left(\frac{K}{K+1} - t\right)\boldsymbol{I} & \frac{\boldsymbol{e}}{K+1} \\ \frac{\boldsymbol{e}^\top}{K+1} & \frac{K+2}{K+1} - t \end{bmatrix}, \tag{3.25}$$

where $\boldsymbol{e} \in \mathbb{R}^K$ is an all 1 column vector and $\boldsymbol{I}$ is a $K \times K$ identity matrix. Then $\boldsymbol{x}$ is K-sparse which satisfies (1.2), and its support Ω is $\{1, \dots, K\}$. Furthermore, by some elementary calculations, one can show that the eigenvalues of $\boldsymbol{B}$ are

$$\lambda_1 = \cdots = \lambda_{K-1} = \frac{K}{K+1} - t, \quad \lambda_K = 1 - \frac{1}{\sqrt{K+1}} - t, \quad \lambda_{K+1} = 1 + \frac{1}{\sqrt{K+1}} - t. \quad (3.26)$$

Hence, $\boldsymbol{B}$ is a symmetric positive definite matrix. Suppose that the Cholesky decomposition of $\boldsymbol{B}$ is $\boldsymbol{B} = \boldsymbol{A}^\top \boldsymbol{A}$. By (1.4) and (3.26), the δ_{K+1} of $\boldsymbol{A}$ is

$$\delta_{K+1} = \frac{1}{\sqrt{K+1}} + t.$$

Therefore, by (3.24), neither (3.14) nor (3.23) holds. Furthermore, by (1.3), (3.24) and (3.25), the mutual coherence of $\boldsymbol{A}$ is

$$\mu = \frac{1}{K+1} \Big/ \sqrt{\left(\frac{K}{K+1} - t\right)\left(\frac{K+2}{K+1} - t\right)} \geq \frac{1}{K+1} \Big/ \sqrt{\frac{K}{K+1}\frac{K+2}{K+1}}$$
$$= \frac{1}{\sqrt{K(K+2)}} \geq \frac{1}{K+1} \geq \frac{1}{2K-1}.$$

Hence, $\boldsymbol{A}$ does not satisfy (3.13).

However, we will show that for the above $\boldsymbol{A}, \boldsymbol{x}$ and $\boldsymbol{w}$, neither BMP nor OLS can utilize $\boldsymbol{y}$ (i.e. $\boldsymbol{Ax}$) and $\boldsymbol{A}$ to reconstruct $\boldsymbol{x}$ in K iterations, but BLS can recover $\boldsymbol{x}$ in K iterations. For more details, see Appendix A.

4. Simulation

This section utilizes MATLAB 2017b to conduct numerical tests on a desktop computer with Intel(R) Core(TM) i5-7500 CPU @3.40 GHz to show the efficiency and effectiveness of BLS by comparing them with those of Batch-OMP [29], OLS [9], the explicit BMP (denote as BMP) algorithm [34], CoSaMP [25], HTP [15], NSIHT and NSHTP [24], and ℓ_1-regularized least squares [21] with $\lambda = 0.1$ (denoted as L_1). Note that, since numerical tests in [15] indicate that 0.71 is the optimal step size for HTP, the step size of HTP is set as 0.71 in this section.

4.1. Simulation for Gaussian measurement matrix

This subsection compares the efficiency and recovery performance of BLS with those of Batch-OMP, OLS, BMP, CoSaMP, HTP, NSIHT and NSHTP, and ℓ_1-regularized least squares. As in [3,7,8,10,30,32,33], we assume $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a random matrix whose entries independently and identically follow the Gaussian distribution $\mathcal{N}(0, 1/m)$. Note that by using the method in [24], the two parameters of both NSIHT and NSHTP are respectively set as 5 and 15.

Our numerical results are based on 100 independent runs with $m = 128, n = 1024$ and $K = 5 : 5 : 50$. For each fixed K and for each run, we first randomly generate a measurement matrix $\boldsymbol{A}$, a K element set Ω whose elements are chosen uniformly at random from $\{1, 2, \dots, n\}$ and then generate an $\boldsymbol{x} \in \mathbb{R}^n$ which satisfies (1.2). Then, we randomly generate a vector $\boldsymbol{u} \in \mathbb{R}^m$ with $\boldsymbol{u} \sim \mathcal{N}(0, 0.01\boldsymbol{I})$, and then set $\boldsymbol{w} = 0.1\boldsymbol{u}/\|\boldsymbol{u}\|_2$, so that $\boldsymbol{w}$ is a ℓ_2-bound noise with $\epsilon = 0.1$. Then we obtain $\boldsymbol{y}$ by setting $\boldsymbol{y} = \boldsymbol{Ax} + \boldsymbol{w}$. Finally, based on $\boldsymbol{A}$ and $\boldsymbol{y}$, we respectively reconstruct $\boldsymbol{x}$ with the 9 sparse recovery algorithms.

Figs. 4.1 and 4.2 respectively show the average support recovery probability and CPU time of the 9 sparse recovery algorithms versus K over 100 independent runs. It is worth mentioning that the probability of support recovery is the cardinality, of the intersection of the support of the true $\boldsymbol{x}$ and the support of $\boldsymbol{x}$ returned by the algorithms, divided by K.

From Figs. 4.1-4.2, we can observe that

1. BLS is more than 10 times and 100 times faster than Batch-OMP and OLS, respectively. Furthermore, BLS has higher probability of support recovery than Batch-OMP and OLS, whose support recovery probabilities are more or less the same, and the improvement is more than 80% when $K \geq 30$.

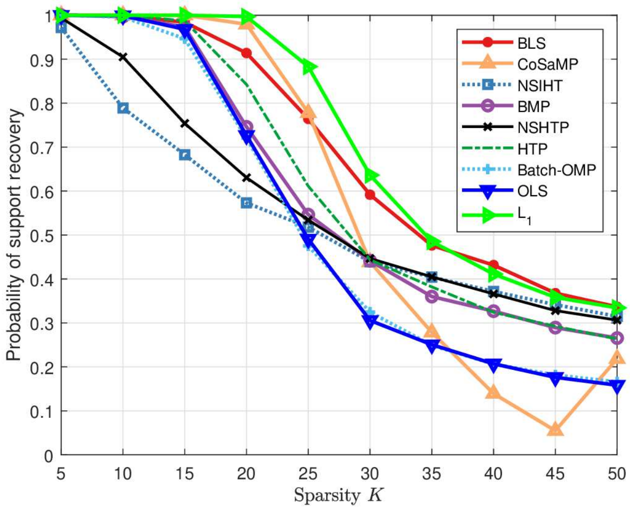

Fig. 4.1. Average support recovery probability versus K for Gaussian measurement matrix with m=128 and $n = 1024$ over 100 independent runs.

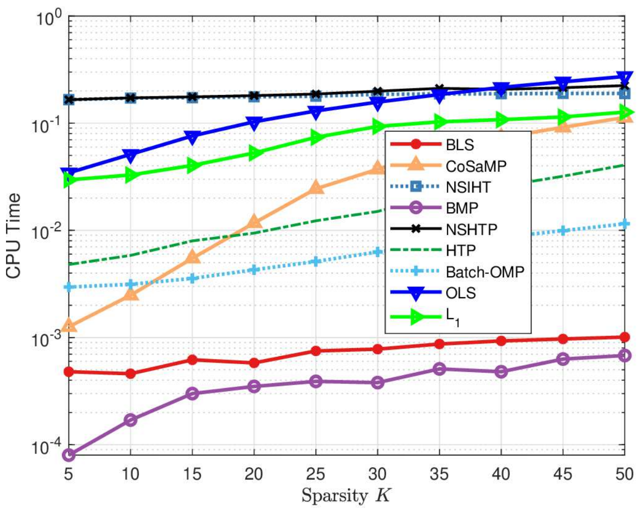

Fig. 4.2. Average CPU time versus sparsity K for Gaussian measurement matrix with $m = 128$ and $n = 1024$ over 100 independent runs.

2. BLS is more than 10 times faster than HTP with higher probability of support recovery, and the improvement is more than 30% when $K \geq 25$.

3. BLS is more than 100 times faster than both NSIHT and NSHTP with higher probability of support recovery, and the improvement is more than 40% when $K \leq 25$.

4. BLS is always and can be 100 times more efficient than CoSaMP with higher probability of support recovery, and the improvement is more than 35% when $K \geq 30$.

5. Although BMP is a little more efficient than BLS, BLS has higher probability of support recovery than BMP and the improvement is more than 30% when $K \geq 25$.

6. Although ℓ_1-regularized least squares has higher probability of support recovery than BLS for $K \leq 35$, BLS has higher probability of support recovery than ℓ_1-regularized least squares for $K > 35$. Furthermore, BLS is around 100 times faster than ℓ_1-regularized least squares.

4.2. Simulation for Bernoulli measurement matrix

This subsection compares the efficiency and recovery performance of BLS with Batch-OMP, OLS, BMP, CoSaMP, HTP, NSIHT and NSHTP, and ℓ_1-regularized least squares for Bernoulli measurement matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, whose entries independently and identically randomly chosen from $\{-1/\sqrt{m}, 1/\sqrt{m}\}$ with equal probability. Note that such class of measurement matrices have also been used to do simulations in [7, 32], and the two parameters of both NSIHT and NSHTP are respectively set as 5 and 15. Since $\boldsymbol{A}$ from this class of measurement matrices has equal column norm, as analyzed in Section 2, the selection strategy (i.e. step 3) of Algorithm 2.1 is replaced by (2.4), and it is not needed to compute $\boldsymbol{c}$ (see (2.2)). Hence, the theoretical computational cost of BLS is equal to that of BMP. As in Section 4.1, the numerical results in this subsection are based on 100 independent runs with $K = 5 : 5 : 50$ and $\boldsymbol{w}$ is a ℓ_2-bound noise with $\epsilon = 0.1$.

Figs. 4.3-4.4 respectively display the average support recovery probability and CPU time of the 9 sparse recovery algorithms versus K over 100 independent runs.

As Figs. 4.1-4.2, we can observe from Figs. 4.3-4.4 that:

1. BLS is more than 10 times and 100 times faster than Batch-OMP and OLS, respectively. Furthermore, BLS has higher probability of support recovery than Batch-OMP and OLS, whose support recovery probabilities are more or less the same, and the improvement is more than 80% when $K \geq 30$.

2. BLS is not only more than 30 times faster than HTP, but also has much better recovery performance than HTP, and the improvement on the support recovery probability is more than 25% when $K \geq 25$.

3. BLS is not only more than 200 times faster, but also has much higher probability of support recovery, than both NSIHT and NSHTP, and the improvement on the support recovery probability is more than 20% when $K \leq 30$.

4. BLS is around 100 times faster than CoSaMP with more than 40% improvement on the probability of support recovery when $K \geq 30$.

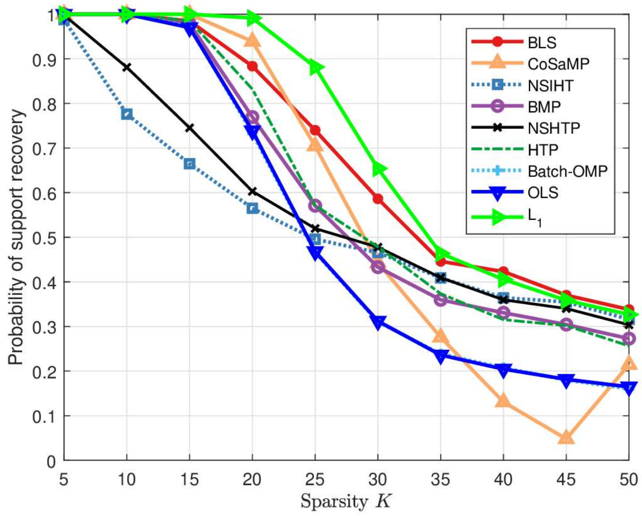

Fig. 4.3. Average support recovery probability versus K for Bernoulli measurement matrix with $m = 128$ and $n = 1024$ over 100 independent runs.

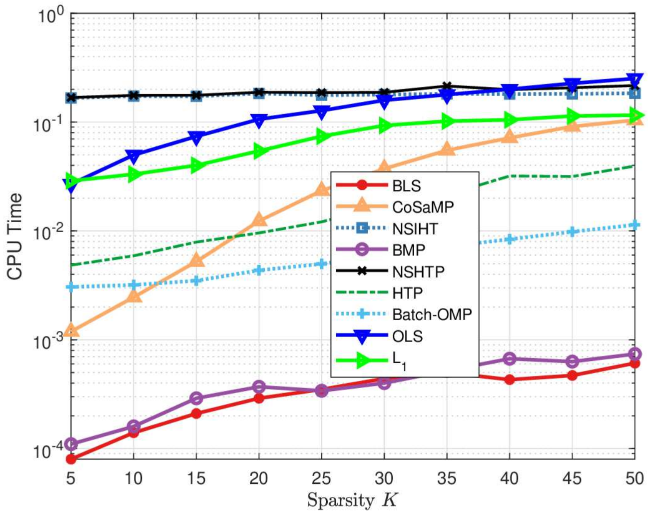

Fig. 4.4. Average CPU time versus sparsity K for Bernoulli measurement matrix with $m = 128$ and $n = 1024$ over 100 independent runs.

5. BMP and BLS are more or less the same efficient (actually as analyzed at the beginning of this subsection, their theoretical computational costs are the same), but BLS has higher probability of support recovery than BMP, and the improvement is more than 25% when $K \geq 25$.

6. Although ℓ_1-regularized least squares has higher probability of support recovery than BLS for $K \leq 35$, BLS has higher probability of support recovery than ℓ_1-regularized least squares for $K > 35$. Furthermore, BLS is more than 100 times more efficient than ℓ_1-regularized least squares.

4.3. Simulation on GSSK applications

As mentioned in Section 1, recovering a binary sparse signal arises in many applications including the generalized space shift keying (GSSK) modulation detection. Hence, in the following, we compare the recovery performance of Batch-OMP, OLS, BLS, BMP, CoSaMP, HTP, NSIHT, NSHTP, and ℓ_1-regularized least squares for this real application.

We first briefly introduce the background of GSSK modulation detection. GSSK is a famous wireless communication modulation technique [19], and the main challenge of GSSK detection is to quickly find the active antennas. Note that there are usually 2 or 3 active antennas. Assume that there are n transmit antennas and m receive antennas, then the signal model of GSSK detection is formulated as

$$\bar{\boldsymbol{y}} = \bar{\boldsymbol{A}}\boldsymbol{x} + \bar{\boldsymbol{w}}, \tag{4.1}$$

where $\bar{\boldsymbol{y}} \in \mathbb{R}^m$ is the received signal, $\bar{\boldsymbol{A}} \in \mathbb{R}^{m\times n}$ is a channel matrix with $\bar{a}_{ij} \sim \mathcal{CN}(0, 1/\sqrt{2})$, $\bar{\boldsymbol{w}} \in \mathbb{R}^m$ is a noise vector which satisfies $\bar{\boldsymbol{w}} \sim \mathcal{CN}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$ for some σ and $\boldsymbol{x} \in \mathbb{R}^n$, which satisfies (1.2), is the transmitted signal [19].

Assume that K antennas are active, then $\boldsymbol{x}$ has K nonzero entries which are 1. Hence, to detect the active antennas, is equivalent to acquire $\boldsymbol{x}$'s support. Denote

$$\boldsymbol{y} = \begin{bmatrix} \Re(\bar{\boldsymbol{y}}) \\ \Im(\bar{\boldsymbol{y}}) \end{bmatrix}, \quad \boldsymbol{A} = \begin{bmatrix} \Re(\bar{\boldsymbol{A}}) \\ \Im(\bar{\boldsymbol{A}}) \end{bmatrix}, \quad \boldsymbol{w} = \begin{bmatrix} \Re(\bar{\boldsymbol{w}}) \\ \Im(\bar{\boldsymbol{w}}) \end{bmatrix},$$

where $\Re(\cdot)$ and $\Im(\cdot)$ represent the real and imaginary part of a given vector or matrix. Then, (4.1) can be transformed to (1.1), hence, we can use Batch-OMP, OLS, BLS, BMP, HTP, CoSaMP, NSIHT, NSHTP, and ℓ_1-regularized least squares to recover $\boldsymbol{x}$'s support.

Figs. 4.5-4.6 respectively display the average probability of correct active antennas detection and CPU time of the nine sparse recovery algorithms versus number of antennas m for $K = 2$ and $n = 64$ over 1000 independent runs with $\sigma = 0.1$. From Figs. 4.5-4.6, we can observe that although BLS is a little slower than BMP, it is more efficient than other seven sparse recovery

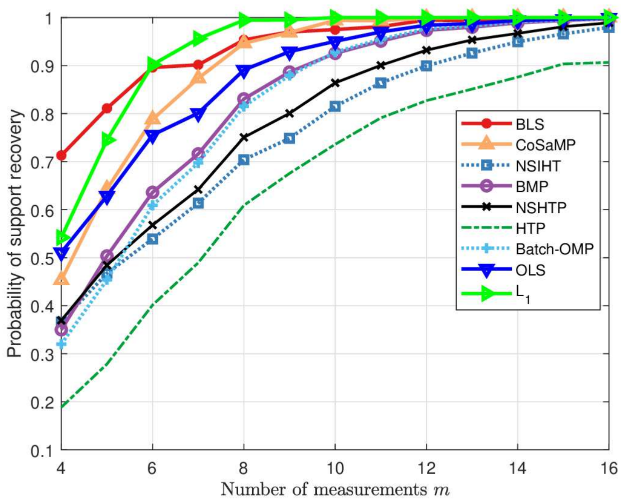

Fig. 4.5. Average probability of correct active antennas detection versus total number of antennas m for $K = 2$ and $n = 64$ over 1000 independent runs.

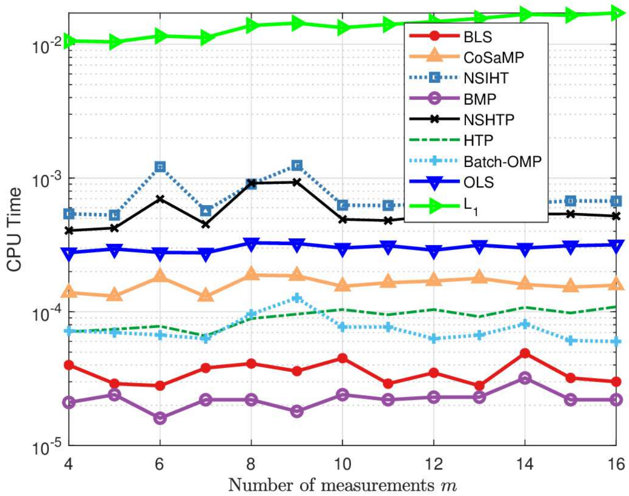

Fig. 4.6. Average CPU time versus total number of antennas m for $K = 2$ and $n = 64$ over 1000 independent runs.

algorithms. Furthermore, although ℓ_1-regularized least squares is more effective than BLS, BLS has (much) higher probability of correct detection than other seven sparse recovery algorithms.

5. Conclusion and Future Research Problem

In this paper, we develop BLS to precisely recover a binary K-sparse signal $\boldsymbol{x}$ from an underdetermined noisy linear model. To show the effectiveness of BLS, in Theorems 3.1 and 3.2, we develop two sufficient conditions for the precise reconstruction of the support of $\boldsymbol{x}$ with at most K iterations based on the restricted isometry property and mutual coherence, respectively. If the measurement matrix is column normalized, the proposed mutual coherence based condition becomes $\mu < 1/(2K-1)$, which is the optimal mutual coherence based condition of recovering any K-sparsity signal. The experimental results show that BLS can be more than 10-200 times faster than BMP, Batch-OMP, OLS, HTP, CoSaMP, NSIHT, NSHTP and ℓ_1-regularized least squares with 20%-80% improvement on the probability of support reconstruction. Numerical tests on the real application of GSSK detection also indicate that although BLS is a little slower than BMP, it is more efficient than the other seven tested sparse recovery algorithms, and although it is less effective than ℓ_1-regularized least squares, it is more effective than the other seven algorithms.

In the following, we list some future research problems:

- As in [33], BLS can be extended to a new binary sparse signal recovery algorithm which selects multiple indices per iteration.
- Global optimality conditions for quadratic optimization problems with binary constraints have been established in [1], whether such techniques can be used to design a fast and effective binary sparse recovery algorithm will be investigated in the future.
- By invoking the Welch bound $\mu \geq \sqrt{(n-m)/(m(n-1))}$ (see, e.g. [16, Theorem 5.7]), one can show that, to ensure $\boldsymbol{A}$ to satisfy (1.5), m and n should satisfy $m \geq \mathcal{O}(K^2 \log n)$.

Furthermore, according to [8], to ensure random matrix $\boldsymbol{A}$ whose entries independently and identically follow the normal distribution $\mathcal{N}(0,1/m)$ to satisfy (1.6), m and n should also satisfy $m \geq \mathcal{O}(K^2 \log n)$. Hence, how to design a greedy binary recovery algorithm which works for $m \geq \mathcal{O}(K \log n)$ is desired to be investigated in the future.

Appendix A. Proof of Example 3.1

In this appendix, we will show that for $\boldsymbol{A}, \boldsymbol{x}$ and $\boldsymbol{w}$ defined in Example 3.1, neither BMP nor OLS can utilize $\boldsymbol{y}$ (i.e. $\boldsymbol{Ax}$) and $\boldsymbol{A}$ to reconstruct $\boldsymbol{x}$ in K iterations, but BLS can recover $\boldsymbol{x}$.

Since $\boldsymbol{A}^\top \boldsymbol{A} = \boldsymbol{B}$, one can easily show that

$$\boldsymbol{A}_i^\top \boldsymbol{r}^0 = \boldsymbol{A}_i^\top \boldsymbol{y} = \boldsymbol{e}_i^\top \boldsymbol{B}\boldsymbol{x} = \frac{K}{K+1} - t, \quad 1 \leq i \leq K, \tag{A.1}$$

$$\boldsymbol{A}_{K+1}^\top \boldsymbol{r}^0 = \boldsymbol{A}_{K+1}^\top \boldsymbol{y} = \boldsymbol{e}_{K+1}^\top \boldsymbol{B}\boldsymbol{x} = \frac{K}{K+1}. \tag{A.2}$$

Then, by (3.24), we have

$$\max_{1\leq i\leq K} \left|\boldsymbol{A}_i^\top \boldsymbol{r}^0\right| < \left|\boldsymbol{A}_{K+1}^\top \boldsymbol{r}^0\right|.$$

Since the support Ω of $\boldsymbol{x}$ is $\{1,\ldots,K\}$, BMP makes a wrong selection in the first iteration. Therefore, it is unable to reconstruct $\boldsymbol{x}$ in K iterations.

In the sequel, we show that OLS cannot reconstruct $\boldsymbol{x}$ in K iterations, either. Since $\boldsymbol{B} = \boldsymbol{A}^\top \boldsymbol{A}$, by (3.25), we have

$$\|\boldsymbol{A}_i\|_2 = \sqrt{\frac{K}{K+1} - t}, \quad 1 \leq i \leq K, \quad \|\boldsymbol{A}_{K+1}\|_2 = \sqrt{\frac{K+2}{K+1} - t}. \tag{A.3}$$

By [28] (see also [35, Eq. (8)]), OLS selects

$$s_k = \operatorname*{argmax}_{i\in\{1,\ldots,n\}} \left|\left\langle \frac{\boldsymbol{P}^{\perp}_{\Gamma^{k-1}}\boldsymbol{A}_i}{\left\|\boldsymbol{P}^{\perp}_{\Gamma^{k-1}}\boldsymbol{A}_i\right\|_2}, \boldsymbol{r}^{k-1}\right\rangle\right|.$$

Hence, in the first iteration, OLS selects s_1 such that

$$s_1 = \operatorname*{argmax}_{i\in\{1,\ldots,n\}} \left|\left\langle \frac{\boldsymbol{A}_i}{\|\boldsymbol{A}_i\|_2}, \boldsymbol{y}\right\rangle\right| = \operatorname*{argmax}_{i\in\{1,\ldots,n\}} \frac{\left|\boldsymbol{A}_i^\top \boldsymbol{y}\right|}{\|\boldsymbol{A}_i\|_2}.$$

By (A.2) and (A.3), we have

$$\max_{1\leq i\leq K} \frac{\left|\boldsymbol{A}_i^\top \boldsymbol{y}\right|}{\|\boldsymbol{A}_i\|_2} = \sqrt{\frac{K}{K+1} - t}, \quad \frac{\left|\boldsymbol{A}_{K+1}^\top \boldsymbol{y}\right|}{\|\boldsymbol{A}_{K+1}\|_2} = \frac{K}{K+1} \Big/ \sqrt{\frac{K+2}{K+1} - t}.$$

Then, by (3.24), it is not hard to show that

$$\max_{1\leq i\leq K} \frac{\left|\boldsymbol{A}_i^\top \boldsymbol{y}\right|}{\|\boldsymbol{A}_i\|_2} < \frac{\left|\boldsymbol{A}_{K+1}^\top \boldsymbol{y}\right|}{\|\boldsymbol{A}_{K+1}\|_2}.$$

Therefore, OLS will select a wrong index in the first iteration, and hence it is unable to acquire $\boldsymbol{x}$ in K iterations.

In the following, we prove that BLS can reconstruct $\boldsymbol{x}$ based on $\boldsymbol{y}$ and $\boldsymbol{A}$. By (2.2) and (A.3), we have

$$c_i = \frac{1}{2}\left(\frac{K}{K+1} - t\right), \quad 1 \leq i \leq K, \quad c_{K+1} = \frac{1}{2}\left(\frac{K+2}{K+1} - t\right). \tag{A.4}$$

Therefore, by (A.1) and (A.2), we obtain

$$\max_{1\le i\le K}\left|\boldsymbol{A}_i^\top\boldsymbol{y}\right|-c_i=\frac{1}{2}\left(\frac{K}{K+1}-t\right),\quad \left|\boldsymbol{A}_{K+1}^\top\boldsymbol{y}\right|-c_{K+1}=\frac{1}{2}\left(\frac{K-2}{K+1}+t\right).$$

Then, by (3.24), we have

$$\max_{1\le i\le K}\left|\boldsymbol{A}_i^\top\boldsymbol{y}\right|-c_i>\left|\boldsymbol{A}_{K+1}^\top\boldsymbol{y}\right|-c_{K+1}.$$

Hence, by line 3 of Algorithm 2.1, BLS will make a correct selection in the first iteration. Suppose that $i_0\in\Omega$ is chosen in the first iteration, we then show that BLS will chose an index in Ω in the second iteration.

By line 6 of Algorithm 2.1,

$$\boldsymbol{r}^1=\boldsymbol{r}^0-\boldsymbol{A}_{i_0}=\boldsymbol{y}-\boldsymbol{A}_{i_0}=\boldsymbol{A}\boldsymbol{x}-\boldsymbol{A}_{i_0}.$$

Hence, by (3.25), (A.1), (A.2), we obtain

$$\begin{aligned}\boldsymbol{A}_i^\top\boldsymbol{r}^1&=\boldsymbol{A}_i^\top\boldsymbol{y}-\boldsymbol{A}_i^\top\boldsymbol{A}_{i_0}=\frac{K}{K+1}-t,\quad 1\le i\ne i_0\le K,\\ \boldsymbol{A}_{K+1}^\top\boldsymbol{r}^1&=\boldsymbol{A}_{K+1}^\top\boldsymbol{y}-\boldsymbol{A}_{K+1}^\top\boldsymbol{A}_{i_0}=\frac{K-1}{K+1}.\end{aligned}$$

Then, by (A.4), we have

$$\max_{1\le i\ne i_0\le K}\left|\boldsymbol{A}_i^\top\boldsymbol{r}^1\right|-c_i=\frac{1}{2}\left(\frac{K}{K+1}-t\right),\quad \left|\boldsymbol{A}_{K+1}^\top\boldsymbol{y}\right|-c_{K+1}=\frac{1}{2}\left(\frac{K-4}{K+1}+t\right).$$

Then, according to (3.24),

$$\max_{1\le i\ne i_0\le K}\left|\boldsymbol{A}_i^\top\boldsymbol{r}^1\right|-c_i>\left|\boldsymbol{A}_{K+1}^\top\boldsymbol{r}^1\right|-c_{K+1}.$$

Therefore, BLS will choose an index in Ω in the second iteration. By induction, it is not hard to show that BLS will make correct selections in all of the first K iterations. Therefore, BLS can successfully reconstruct $\boldsymbol{x}$ in K iterations.

Acknowledgments. This work was partially supported by the NSFC (Grant Nos. 12271215, 12326378 and 11871248), by the Guangdong Basic and Applied Basic Research Foundation (Grant No. 2022A1515010029), by the Guangzhou Science and Technology Foundation Committee (Grant No. 202201011126).

References

[1] A. Beck and M. Teboulle, Global optimality conditions for quadratic optimization problems with binary constraints, *SIAM J. Optim.*, **11**:1 (2000), 179–188.

[2] A. Beck and M. Teboulle, A fast iterative shrinkage-thresholding algorithm for linear inverse problems, *SIAM J. Imaging Sci.*, **2**:1 (2009), 183–202.

[3] J.D. Blanchard, J. Tanner, and K. Wei, Conjugate gradient iterative hard thresholding: Observed noise stability for compressed sensing, *IEEE Trans. Signal Process.*, **63**:2 (2015), 528–537.

[4] T. Blumensath and M.E. Davies, Iterative hard thresholding for compressed sensing, *Appl. Comput. Harmon. Anal.*, **27**:3 (2009), 265–274.

[5] T. Cai and L. Wang, Orthogonal matching pursuit for sparse signal recovery with noise, *IEEE Trans. Inf. Theory*, **57**:7 (2011), 4680–4688.

[6] T. Cai, L. Wang, and G. Xu, Stable recovery of sparse signals and an oracle inequality, *IEEE Trans. Inf. Theory*, **56**:7 (2020), 3516–3522.

[7] E. Candes, M. Rudelson, T. Tao, and R. Vershynin, Error correction via linear programming, in: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, (2005), 668–681.

[8] E.J. Candés and T. Tao, Decoding by linear programming, *IEEE Trans. Inf. Theory*, **51**:12 (2005), 4203–4215.

[9] S. Chen, S.A. Billings, and W. Luo, Orthogonal least squares methods and their application to non-linear system identification, *Int. J. Control*, **50**:5 (1989), 1873–1896.

[10] Z. Chen, F. Sohrabi, and W. Yu, Sparse activity detection for massive connectivity, *IEEE Trans. Signal Process.*, **66**:7 (2018), 1890–1904.

[11] W. Dai and O. Milenkovic, Subspace pursuit for compressive sensing signal reconstruction, *IEEE Trans. Inf. Theory*, **55**:5 (2009), 2230–2249.

[12] D.L. Donoho, Compressed sensing, *IEEE Trans. Inf. Theory*, **52**:4 (2006), 1289–1306.

[13] D.L. Donoho and X. Huo, Uncertainty principles and ideal atomic decomposition, *IEEE Trans. Inf. Theory*, **47**:7 (2001), 2845–2862.

[14] A. Flinth and S. Keiper, Recovery of binary sparse signals with biased measurement matrices, *IEEE Trans. Inf. Theory*, **65**:12 (2019), 8084–8094.

[15] S. Foucart, Hard thresholding pursuit: An algorithm for compressive sensing, *SIAM J. Numer. Anal.*, **49**:6 (2011), 2543–2563.

[16] S. Foucart and H. Rauhut, *A Mathematical Introduction to Compressive Sensing*, Springer-Verlag, 2013.

[17] A.C. Gilbert, M.A. Iwen, and M.J. Strauss, Group testing and sparse signal recovery, in: *Proceedings of 42nd Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, (2008), 1059–1063.

[18] C. Herzet, A. Drémeau, and C. Soussen, Relaxed recovery conditions for OMP/OLS by exploiting both coherence and decay, *IEEE Trans. Inf. Theory*, **62**:1 (2016), 459–470.

[19] J. Jeganathan, A. Ghrayeb, and L. Szczecinski, Generalized space shift keying modulation for MIMO channels, in: *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, (2008), 1–5.

[20] S. Keiper, Recovery of binary sparse signals from structured biased measurements, *arXiv: 2006.14835*, 2020.

[21] S.J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, An interior-point method for large-scale ℓ_1-regularized least squares, *IEEE J. Sel. Top. Signal. Process.*, **1**:4 (2007), 606–617.

[22] J. Kim, J. Wang, and B. Shim, Optimal restricted isometry condition of normalized sampling matrices for exact sparse recovery with orthogonal least squares, *IEEE Trans. Signal Process.*, **69** (2021), 1521–1536.

[23] M. Lustig, D.L. Donoho, J.M. Santos, and J.M. Pauly, Compressed sensing MRI, *IEEE Signal Proces. Mag.*, **25**:2 (2008), 72–82.

[24] N. Meng and Y. Zhao, Newton-step-based hard thresholding algorithms for sparse signal recovery, *IEEE Trans. Signal Process.*, **68** (2020), 6594–6606.

[25] D. Needel and J.A. Tropp, CoSaMP: Iterative signal recovery from incomplete and inaccurate samples, *Appl. Comput. Harmon. Anal.*, **26**:3 (2009), 301–321.

[26] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition, in: *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, **1** (1993), 40–44.

[27] L.C. Potter, E. Ertin, J.T. Parker, and M. Cetin, Sparsity and compressed sensing in radar imaging, *Proceedings of the IEEE*, **98**:6 (2010), 1006–1020.

[28] L. Rebollo-Neira and D. Lowe, Optimized orthogonal matching pursuit approach, *IEEE Signal*

Process. Lett., **9**:4 (2002), 137–140.

[29] R. Rubinstein, M. Zibulevsky, and M. Elad, *Efficient Implementation of the K-SVD Algorithm Using Batch Orthogonal Matching Pursuit*, Israel Institute of Technology, 2008.

[30] J. Shen and P. Li, A tight bound of hard thresholding, *J. Mach. Learn. Res.*, **18** (2018), 1–42.

[31] J.A. Tropp, Greed is good: Algorithmic results for sparse approximation, *IEEE Trans. Inf. Theory*, **50**:10 (2004), 2231–2242.

[32] J.A. Tropp and A.C. Gilbert, Signal recovery from random measurements via orthogonal matching pursuit, *IEEE Trans. Inf. Theory*, **53**:12 (2007), 4655–4666.

[33] J. Wang and P. Li, Recovery of sparse signals using multiple orthogonal least squares, *IEEE Trans. Signal Process.*, **65**:8 (2017), 2049–2062.

[34] J. Wen and H. Li, Binary sparse signal recovery with binary matching pursuit, *Inverse Problems*, **37**:6 (2021), 065014.

[35] J. Wen, J. Wang, and Q. Zhang, Nearly optimal bounds for orthogonal least squares, *IEEE Trans. Signal Process.*, **65**:20 (2017), 5347–5356.

[36] J. Wen, J. Weng, C. Tong, C. Ren, and Z. Zhou, Sparse signal recovery with minimization of 1-norm minus 2-norm, *IEEE Trans. Veh. Technol.*, **68**:7 (2019), 6847–6854.

[37] J. Wen, Z. Zhou, J. Wang, X. Tang, and Q. Mo, A sharp condition for exact support recovery with orthogonal matching pursuit, *IEEE Trans. Signal Process.*,**65**:6 (2017), 1370–1382.

[38] D.P. Wipf, B.D. Rao, and S. Nagarajan, Latent variable bayesian models for promoting sparsity, *IEEE Trans. Inf. Theory*, **57**:9 (2011), 6236–6255.