IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 12, 2020, accepted January 19, 2020, date of publication January 22, 2020, date of current version January 31, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2968716

Sparse Group Feature Selection by Weighted Thresholding Homotopy Method

JINGLAN WU[1], HUATING HUANG[2], AND WENXING ZHU[2]
[1]College of Computer and Control Engineering, Minjiang University, Fuzhou 350116, China
[2]Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350116, China

Corresponding author: Wenxing Zhu (wxzhu@fzu.edu.cn)

This work was supported in part by the Natural Science Foundation of Fujian Province under Grant 2017J01768, and in part by the National Natural Science Foundation of China under Grant 61672005.

ABSTRACT In this paper, we investigate the sparse group feature selection problem, in which covariates posses a grouping structure sparsity at the level of both features and groups simultaneously. We reformulate the feature sparsity constraint as an equivalent weighted l_1-norm constraint in the sparse group optimization problem. To solve the reformulated problem, we first propose a weighted thresholding method based on a dynamic programming algorithm. Then we improve the method to a weighted thresholding homotopy algorithm using homotopy technique. We prove that the algorithm converges to an L-stationary point of the original problem. Computational experiments on synthetic data show that the proposed algorithm is competitive with some state-of-the-art algorithms.

INDEX TERMS Homotopy technique, weighted thresholding method, sparse group feature selection.

I. INTRODUCTION

In this paper, we are interested in sparse group feature selection, which simultaneously selects important groups as well as important individual variables. More precisely, the sparse group feature selection problem can be written as [1]:

$$\begin{aligned} &\min f(x) \\ &s.t.\ \|x\|_0 \le s_f \\ &\quad \|x\|_G \le s_g \\ &\quad l \le x \le u. \end{aligned} \tag{1}$$

In the problem, $-\infty \le l \le 0 \le u \le +\infty$, $x = (x_1, \ldots, x_n)^T$ is partitioned into $|G|$ non-overlapping groups $\{x_{G_i}, i = 1, 2, \cdots, |G|\}$, and $\|x\|_0$ is the number of nonzero elements of x. Let $r = |G|$. Without loss of generality, let $x = (x_{G_1}^T, \ldots, x_{G_r}^T)^T$ and $\|x\|_G = \sum_{i=1}^{r} I(\|x_{G_i}\|_2 \neq 0)$, where $I(\cdot)$ is the indicator function. (s_f, s_g) are the number of features and the number of groups respectively. $f(x)$ is a continuously differentiable function, for which the gradient is Lipschitz-continuous with Lipschitz constant $L_f > 0$.

Problem (1) has been applied in structured genome-wide association studies [2], semantic concept detection for high-level human interpretation of video contents [3], and structure-aware spectrum estimation of frequency-hopping signals [4], etc. However, it is NP-hard and not approximable within polynomial time, since sparse optimization with L_0-norm is hard to approximate unless P = NP [5].

During the last decade, many works have been devoted to group sparse optimization, such as group Lasso [6], [7], which uses an l_2-regularization for each group. This kind of methods is incapable of variable selection at the individual level. To address problem (1) which selects important groups and individual variables simultaneously, group Lasso has been extended carefully for the problem. In [8], by extending group Lasso the authors proposed a group bridge method, which is the first penalized regularization method. This was further improved in [9] to a general framework which is capable of two-level selection. Based on a new class of group penalties, the group exponential Lasso proposed in [10] allows the penalty to decay exponentially for selecting individual features.

Another approach to problem (1) is the iterative sparse group hard thresholding algorithm proposed in [1], in which $f(x)$ is approximated by a proximal function. And then the function is minimized over the constraints of problem (1), which can be solved optimally by a dynamic programming algorithm. However according to [11], it is restrictive to minimize a proximal function over individual variable sparsity constraint directly. Hence it is interesting to improve the iterative sparse group hard thresholding algorithm [1] for problem (1) using the reformulation technique in [11].

The associate editor coordinating the review of this manuscript and approving it for publication was Md Asaduzzaman.

To achieve this purpose, we reformulate the s_f-sparse individual constraint in (1) by a weighted l_1-norm strategy and get a new problem, which is equivalent to problem (1). This reformulation is similar to that in [11], which does not consider the group sparsity constraint. To solve the reformulated problem, we penalize the side constraint and propose a weighted thresholding method, which is based on a dynamic programming algorithm. Moreover, to improve the performance of the weighted thresholding method, we develop a weighted thresholding homotopy method for the problem based on the homotopy technique. Computational experiments on synthetic data show that the proposed algorithm is competitive with some state-of-the-art algorithms.

The rest of this paper is organized as follows. In Section 2, we first give some notations. Then we develop a weighted thresholding method for the sparse group feature selection problem. In Section 3, we further improve the weighted thresholding method with homotopy technique. Moreover, we introduce the corresponding algorithm and give convergence analysis. In Section 4, we conduct computational experiments on synthetic data to show that the proposed method is competitive with some state-of-the-art algorithms. Finally, conclusions are presented in Section 5.

II. WEIGHTED THRESHOLDING METHOD

A. NOTATIONS

Unless otherwise stated, $|\cdot|$ denotes module and $\|\cdot\|$ represents the Euclidean norm. The transpose of a vector $x \in \mathbb{R}^n$ is denoted by x^T. For an index set $S \subseteq \{1, \ldots, n\}$, x_S is the subvector of components of x indexed by S. Given a weight w, $w \circ x = (w_1x_1, w_2x_2, \cdots, w_nx_n)^T$. The box constraint is represented by $B = \{x \in \mathbb{R}^n : l \le x \le u\}$. Let C be the set of all s-sparse vectors, i.e.,

$$C = \{x \in \mathbb{R}^n : \|x\|_0 \le s_f, \|x\|_G \le s_g\}.$$

Let $\Pi_C(y)$ be the projection of $y \in \mathbb{R}^n$ onto the set C, i.e.,

$$\Pi_C(y) = arg\,min\,\{\|x - y\|^2 : x \in C\}.$$

B. EQUIVALENT FORMULATION

First, we introduce an equivalent formulation of the sparsity constraint $x \in C_f$, where $C_f = \{x \in \mathbb{R}^n : \|x\|_0 \le s_f\}$.

Lemma 1 [11]: $x \in C_f$ is equivalent to that there exists $w \in \{0, 1\}^n$ such that

$$\begin{cases} \|w \circ x\|_1 = 0 \\ \|1 - w\|_0 \le s_f. \end{cases} \tag{2}$$

By Lemma 1, problem (1) is equivalent to

$$\begin{aligned} &\min\ f(x) \\ &s.t.\ \|w \circ x\|_1 = 0 \\ &\quad \|x\|_G \le s_g \\ &\quad \|1 - w\|_0 \le s_f \\ &\quad w \in \{0, 1\}^n \\ &\quad l \le x \le u. \end{aligned} \tag{3}$$

For the convenience of description, let $\Omega = \{(x, w) : \|x\|_G \le s_g, \|1 - w\|_0 \le s_f, w \in \{0, 1\}^n, l \le x \le u\}$.

By penalizing the constraint $\|w \circ x\|_1 = 0$ to the objective function, we have the following problem:

$$\min_{(x,w)\in\Omega} F_\lambda(x, w) = f(x) + \lambda\|w \circ x\|_1, \tag{4}$$

where $\lambda \ge 0$. To tackle the above problem, we approximate $f(x)$ at the current solution x^k by the second order Taylor expansion, and get

$$\begin{aligned} &H(x, w, x^k) \\ &= f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{L}{2}\|x - x^k\|^2 + \lambda\|w \circ x\|_1. \end{aligned}$$

Then minimizing $H(x, w, x^k)$ over $(x, w) \in \Omega$ is equivalent to

$$\min_{(x,w)\in\Omega} \phi(x, w) = \tfrac{L}{2}\|x - y^k\|^2 + \lambda\|w \circ x\|_1, \tag{5}$$

where $y^k = x^k - \frac{1}{L}\nabla f(x^k)$.

In problem (5), $\phi(x, w)$ is a separable function which can be written as $\phi(x, w) = \sum_{i=1}^n \phi_i(x_i, w_i)$, where

$$\phi_i(x_i, w_i) = \frac{L}{2}|x_i - y_i^k|^2 + \lambda|w_i x_i|. \tag{6}$$

Observing that the constraints in (5) contain group and individual sparsity constraints, we will develop a dynamic programming algorithm to solve problem (5) in the next subsection.

C. DYNAMIC PROGRAMMING ALGORITHM

In this subsection, we propose a dynamic programming algorithm to find an optimal solution of problem (5). Let $S = \cup_{l=1}^i G_l$. We consider a general form of problem (5):

$$\begin{aligned} &\min\ \frac{L}{2}\|(x - y)_S\|^2 + \lambda\|(w \circ x)_S\|_1 \\ &s.t.\ \|(1 - w)_S\|_0 \le m \\ &\quad w_S \in \{0, 1\}^{|S|} \\ &\quad \|x\|_S \le j \\ &\quad l_S \le x_S \le u_S, \end{aligned} \tag{7}$$

where $y = x^k - \frac{1}{L}\nabla f(x^k)$, $m \in \{1, 2, \ldots, s_f\}$, $j \in \{1, 2, \ldots, s_g\}$. Denote the problem by $P(i, j, m)$, and let $V(i, j, m)$ be its minimum value.

This problem can be solved by considering the following two cases:

1) If group G_i is not selected, then $V(i, j, m) = V(i - 1, j, m) + \frac{L}{2}\|y_{G_i}\|^2$, and the corresponding solution $(x_{G_i}, w_{G_i}) = (0, 1)$.

2) If group G_i is selected, then problem (7) can be divided into two subproblems: $P(i-1, j-1, m-t)$ and the following problems

$$\begin{aligned} &\min\ \frac{L}{2}\|(x - y)_{G_i}\|^2 + \lambda\|(w \circ x)_{G_i}\|_1 \\ &s.t.\ \|(1 - w)_{G_i}\|_0 \le t \\ &\quad w \in \{0, 1\}^{|G_i|} \\ &\quad l_{G_i} \le x_{G_i} \le u_{G_i}, \end{aligned} \tag{8}$$

$t = 1, 2, \cdots, \min\{m, |G_i|\}$. Then

$$V(i, j, m) = \min_{1 \le t \le \min\{m, |G_i|\}} V(i-1, j-1, m-t) + SH(i, t),$$

where $SH(i, t)$ denotes the minimum value of problem (8).

Problem (8) has a closed form solution presented in Lemma 2. For the convenience of description, we assume without loss of generality that $G_i = \{1, 2, \ldots, n\}$ in this lemma.

Lemma 2 [11]: Problem (8) *has the closed form solution* (x^*, w^*)*:*

$$(x_i^*, w_i^*) = \begin{cases} ([\Pi_B(y)]_i, 0), & \text{if } i \in A; \\ ([\Pi_B(soft(y))]_i, 1), & \text{otherwise,} \end{cases} \quad (9)$$

$i = 1, \cdots, n$. *In* (9),

$$soft(y) = sign(y) \circ \max\{|y| - \lambda/L, 0\}$$

is the soft thresholding operator [12], and $A \subseteq \{1, 2, \ldots, n\}$ *is the index set corresponding to the* t *largest values of* $\{v_i\}_{i=1}^n$*, where* $v_i = \phi_i([\Pi_B(soft(y))]_i, 1) - \phi_i([\Pi_B(y)]_i, 0)$*, and* $\phi_i(x_i, w_i)$ *is defined in equation* (6)*.*

Thus, $V(i, j, k)$ can be written in the following recursive form:

$$V(i, j, m) = \min\{V(i-1, j, m) + \frac{L}{2}\|y_{G_i}\|^2, \min_{1 \le t \le \min\{m, |G_i|\}} V(i-1, j-1, m-t) + SH(i, t)\}. \quad (10)$$

The boundary condition for the above dynamic programming equation is

$$V(i, j, m) = 0, \text{ if } i = 0, \text{ or } j = 0, \text{ or } m = 0. \quad (11)$$

Based on equations (10) and (11), we design a dynamic programming algorithm (Algorithm 1) for problem (5) via the standard bottom-up approach. In the algorithm, lines 2-7 precompute the values of $IH(i)$ and $SH(i, t)$. Lines 8-21 are for calculating the values of $V(i, j, m)$ based on equation (10). Here, V is a three dimensional array $(0..|G|, 0..s_g, 0..s_f)$. And $ID(0..|G|, 0..s_g, 0..s_f)$ is a three dimensional array to record the number of selected elements in each group, which is for Algorithm 2 to construct the optimal solution of problem (5). More specifically, $ID(i, j, m) = 0$ if group G_i is not selected in computing $V(i, j, m)$; otherwise

$$ID(i, j, m) = \arg \min_{1 \le t \le \min\{m, |G_i|\}} V(i-1, j-1, m-t) + SH(i, t).$$

After computing the ID in Algorithm 1, Algorithm 2 is a linear backtracking algorithm for computing the number of selected elements in group G_i of the optimal solution of problem (7), i.e., the t in problem (8). Then the optimal solution can be reconstructed by the solution of problem (8), which is in Lemma 2.

Next, we analyze the time complexity of the dynamic programming algorithm and the backtracking algorithm. For Algorithm 1, by Lemma 2, computing each $SH(i, t)$ needs time $O(|G_i| \log(|G_i|))$. Hence the time complexity of lines 2-7 in Algorithm 1 is

$$O(\sum_{i=1}^{r} |G_i| \log(|G_i|)) = O(n \log n).$$

For lines 9-20 in Algorithm 1, the time complexity is $O(s_f s_g |G_i|)$. Hence the time complexity of lines 9-20 is

$$O(\sum_{i=1}^{r} s_f s_g |G_i|) = O(s_f s_g n).$$

Thus, the time complexity of Algorithm 1 is $O(s_f s_g n + n \log n)$.

Algorithm 1 Dynamic Programming Algorithm for Problem (8)

Input: G, s_f, s_g, x^k. // $L > L_f$;
Output: V, ID.

```
1: initialization V ← 0, ID ← 0, y^k = x^k − (1/L)∇f(x^k), r = |G|;
2: for i = 1 : r do
3:    IH(i) = (L/2)‖y^k_{G_i}‖^2;
4:    for t = 1 : |G_i| do
5:       calculate SH(i, t);
6:    end for
7: end for
8: for i = 1 : r do
9:    for j = 1 : s_g do
10:      for m = 1 : s_f do
11:         V(i, j, m) = V(i − 1, j, m) + IH(i);
12:         for t = 1 : min{m, |G_i|} do
13:            vp = V(i − 1, j − 1, m − t) + SH(i, t);
14:            if vp < V(i, j, m) then
15:               V(i, j, m) = vp;
16:               ID(i, j, m) = t;
17:            end if
18:         end for
19:      end for
20:   end for
21: end for
```

Algorithm 2 Backtracking Algorithm for Computing the Number of Selected Elements in Each Group of the Optimal Solution

Input: ID, s_f, s_g;
Output: cp.

```
1: initialization j = s_g, m = s_f;
2: for i = |G| : 1 do
3:    cp(i) = ID(i, j, m);
4:    if cp(i) > 0 then
5:       j = j − 1;
6:       m = m − cp(i);
7:    end if
8: end for
```

Moreover, it is obvious that Algorithm 2 needs time $O(|G|)$ to calculate optimal t for each group. Further, if the solution of problem (8) for each t is stored while computing $SH(i,t)$ in lines 2-7 of Algorithm 1, then Algorithm 2 still needs time $O(|G|)$ to get the optimal solution of problem (5).

Based on the dynamic programming algorithm, we present a weighted thresholding method for problem (1) in the next subsection.

D. WEIGHTED THRESHOLDING METHOD

Motivated by the weighted thresholding method for individual variable sparsity constrained optimization [11] and the ISTA with sparse group hard thresholding method [1], we adopt the weighted thresholding framework and present the following Algorithm 3 for problem (4). For the convenience of description, we denote the solution in Algorithm 1 by $WT_{\lambda,L}(x^k)$, then we propose the weighted thresholding method for problem (4) in Algorithm 3.

Algorithm 3 Weighted Thresholding Method

Input: $s_f, s_g, \lambda, L, x^0$; //$L > L_f$
Output: $\hat{x}, \hat{w}$.
1: initialization $k \leftarrow 0$;
2: **repeat**
3: $(x^{k+1}, w^{k+1}) \leftarrow WT_{\lambda,L}(x^k)$;
4: $k \leftarrow k+1$;
5: **until** termination condition is reached
6: $\hat{x} \leftarrow x^k, \hat{w} \leftarrow w^k$.

In Algorithm 3, the Lipschitz constant L_f is generally unknown, hence it might be hard to calculate directly a fixed value of L in practice. We will use a line search strategy to explore a feasible value of L in subsection IV-A in detail.

Similar to the proof in [11], Algorithm 3 has the following convergence property.

Theorem 1: Let the sequence $\{x^k\}$ *be generated by Algorithm 3,* $\eta = L - L_f > 0$. *Then*

(i) the sequence $\{F_\lambda(x^k, w^k)\}$ *is nonincreasing and satisfies that*

$$F_\lambda(x^k, w^k) - F_\lambda(x^{k+1}, w^{k+1}) \geq \frac{\eta}{2}\|x^{k+1} - x^k\|^2; \quad (12)$$

(ii) the sequence $\{x^k\}$ *converges.*

It must be remarked that, the proposed weighted thresholding method for problem (4) has a penalty parameter λ. Similar to the regularization methods [13]–[15], it is difficult to choose a proper value for λ. Fortunately, since our approach is based on the penalty function method, we may choose a sufficiently large value of λ during implementation. However, if the value of λ is too large, then the solution given by Algorithm 3 may be poor. In the next section, we give a homotopy technique to improve the performance of Algorithm 3.

III. WEIGHTED THRESHOLDING HOMOTOPY METHOD

A. WEIGHTED THRESHOLDING HOMOTOPY METHOD

Homotopy technique has been widely used for compressed sensing in the l_1-regularized least-squares problems [13], [15]–[17]. By noting that problem (4) is a penalized version of problem (3), the value of penalty parameter λ should be increased sequentially to a target value. This could be viewed as a kind of homotopy technique. Moreover, we also use the homotopy idea on the sparsity level s_f, and thus develop the weighted thresholding homotopy method named as WTH.

To solve problem (1), the key idea of WTH is tracing the path of solutions of problem (4) with varying (s_f, λ), where (s_f, λ) starts from $(0, \lambda_0)$ and is gradually increased simultaneously, until target values of sparsity level s_f and parameter λ are reached. For each fixed $s_f = s_k$ and $\lambda = \lambda_k$, we use the weighted thresholding method (Algorithm 1) to solve problem (4). The WTH algorithm is outlined in Algorithm 4.

Algorithm 4 Weighted Thresholding Homotopy Method (WTH)

Input: $L, s_f, s_g, x^0, \lambda_0, N$ is a positive integer; // $L > L_f$
Output: $\hat{x}, \hat{w}$.
1: initialize $k \leftarrow 0$, $s_0 \leftarrow 0$, $\rho \geq 1$, $\epsilon > 0$, $\gamma > 0$, $\Delta \leftarrow \lceil s_f/N \rceil$;
2: **for** $k = 1 : N$ **do**
3: $i \leftarrow 0$;
4: $x^{k,i} \leftarrow x^{k-1}$;
5: $s_k \leftarrow \min\{s_{k-1} + \Delta, s_f\}$;
6: $\lambda_k \leftarrow \rho\lambda_{k-1}$;
7: **repeat**
8: $(x^{k,i+1}, w^{k,i+1}) \leftarrow WT_{\lambda_k,L}(x^{k,i})$;
9: $i \leftarrow i+1$;
10: **until** $\|x^{k,i} - x^{k,i+1}\| / \max\{\|x^{k,i}\|, 10^{-6}\} < \epsilon$
11: $x^k \leftarrow x^{k,i}, w^k \leftarrow w^{k,i}$;
12: **end for**
13: $\hat{x} \leftarrow x^N, \hat{w} \leftarrow w^N$.

Next, we prove convergence of Algorithm 4. By generalizing the definition of L-stationarity condition for sparsity constrained problems [18], we get the L-stationarity condition for problem (1) as follows.

Definition 1 (L-Stationarity): Suppose that $L > 0$. *A vector* $x \in C \cap B$ *is called an L-stationary point of problem* (1) *if*

$$x \in \Pi_{C\cap B}\Big(x - \frac{1}{L}\nabla f(x)\Big). \quad (13)$$

Theorem 2: Let $\{x^{k,i}\}$ *be the sequence generated by Algorithm 4. Then:*

(i) the sequence $\{F_{\lambda_k}(x^{k,i}, w^{k,i})\}$ *is nonincreasing and* $\{x^{k,i}\}$ *converges for any fixed* k.

(ii) let $\lim_{i\to\infty} x^{N,i} = \hat{x}$. *If* $\|\hat{w} \circ \hat{x}\|_1 = 0$, *then* $\hat{x}$ *is an L-stationary point of problem* (1).

Proof: (i) Since the inner loop of Algorithm 4 calls the weighted thresholding method (Algorithm 1), by Theorem 1

the sequence $\{F_{\lambda_k}(x^{k,i}, w^{k,i})\}$ is nonincreasing and $\{x^{k,i}\}$ converges for each fixed k.

(ii) If $\|\hat{w} \circ \hat{x}\|_1 = 0$, then by line 8 of Algorithm 4, $\hat{w} \in \{0,1\}^n$ and $\|1 - \hat{w}\|_0 \leq s_f$. Further, by Lemma 1, $\|\hat{x}\|_0 \leq s_f$. Thus, combining $l \leq \hat{x} \leq u$ with $\|\hat{x}\|_G \leq s_g$, $\hat{x}$ is a feasible solution of problem (1). Since $\lim_{i\to\infty} x^{N,i} = \hat{x}$, by line 8 of Algorithm 4 and Lemma 2, it is obvious that

$$\hat{x} \in \underset{(x,w)\in\Omega}{arg\,min}\{\frac{L}{2}\|x - \hat{y}\|^2 + \lambda_N \|w \circ x\|_1\}, \tag{14}$$

where $\hat{y} = \hat{x} - \frac{1}{L}\nabla f(\hat{x})$ and $\Omega = \{(x, w) : \|x\|_G \leq s_g, \|1 - w\|_0 \leq s_f, w \in \{0,1\}^n, l \leq x \leq u\}$.

Next, we prove that $\hat{x}$ is an L-stationary point of problem (1). By Definition 1, we need to prove that

$$\hat{x} \in \Pi_{C\cap B}\Big(\hat{x} - \frac{1}{L}\nabla f(\hat{x})\Big) = \Pi_{C\cap B}(\hat{y}),$$

where $\Pi_{C\cap B}(\hat{y}) = arg\,min\{\|x - \hat{y}\|^2 : x \in C \cap B\}$. That is, $\hat{x}$ is the solution of problem

$$\min\{\|x - \hat{y}\|^2 : \|x\|_G \leq s_g, \|x\|_0 \leq s_f, l \leq x \leq u\}.$$

Suppose by contradiction that there exists $\tilde{x}$ such that $\|\tilde{x} - \hat{y}\|^2 < \|\hat{x} - \hat{y}\|^2$, where $\|\tilde{x}\|_0 \leq s_f$ and $l \leq \tilde{x} \leq u$. Since $\|\tilde{x}\|_0 \leq s_f$, we can assign the value of $\tilde{w}$ as

$$\begin{cases} \tilde{w}_i = 0, & \text{if } \tilde{x}_i = 1; \\ \tilde{w}_i = 1, & \text{if } \tilde{x}_i \neq 0. \end{cases}$$

Then $\tilde{w} \in \{0,1\}^n$ and $\|1 - \tilde{w}\|_0 \leq s_f$. So $(\tilde{x}, \tilde{w}) \in \Omega$, and

$$\begin{aligned} \frac{L}{2}\|\tilde{x} - \hat{y}\|^2 + \lambda_N\|\tilde{w} \circ \tilde{x}\|_1 &= \frac{L}{2}\|\tilde{x} - \hat{y}\|^2 + 0 \\ &< \frac{L}{2}\|\hat{x} - \hat{y}\|^2 + 0 \leq \frac{L}{2}\|\hat{x} - \hat{y}\|^2 + \lambda_N\|\hat{w} \circ \hat{x}\|_1, \end{aligned}$$

which contradicts Equation (14). Hence $\hat{x}$ is an L-stationary point of problem (1). ■

B. THE CHOICE OF λ_0

If $s_f = 0$, then $w_i = 1, i = 1, 2, \ldots, n$ meet the constraints of problem (4), and problem (4) becomes an l_1-norm regularization problem with a box constraint. That is, the problem

$$\begin{cases} \min f(x) + \lambda\|w \circ x\|_1 \\ s.t.\ \ \|1 - w\|_0 \leq 0 \\ \quad w \in \{0,1\}^n \\ \quad \|x\|_G \leq s_g \\ \quad l \leq x \leq u \end{cases}$$

is reduced to

$$\begin{cases} \min\ \psi(x) = f(x) + \lambda\|x\|_1 \\ s.t.\ \ \|x\|_G \leq s_g \\ \quad l \leq x \leq u. \end{cases} \tag{15}$$

Similar to the proof in [11], we can prove that, if $f(x)$ is a differentiable convex function, then for $\lambda \geq \|\nabla f(0)\|_\infty$, $x = 0$ is the optimal solution of problem (15). Thus, $\lambda = \|\nabla f(0)\|_\infty$ is big enough such that $(x, w) = (0, 1)$ is the optimal solution of problem (4) with $s_f = 0$. So we set $\lambda_0 = \|\nabla f(0)\|_\infty$ in our computational experiments.

IV. COMPUTATIONAL EXPERIMENTS

In this section, we compare our algorithm with some state-of-the-art algorithms by performing them on the same set of test instances. All experiments were conducted on a personal computer with an Intel(R) Core(TM) i3-6100 CPU(3.70GHz) and 4.00GB memory. We set $l = -\infty$, $u = +\infty$. Unless otherwise stated, all parameters were set as default for the compared methods in the experiments. First, we propose a practical algorithm of Algorithm 4 in the following subsection.

A. PRACTICAL ALGORITHM

In Algorithm 4, a fixed Lipschitz constant L is used throughout all iterations. However, it is hard to calculate the value for a general function [19]. Here, we adopt the line search technique in [20] to update the value of L dynamically. The practical weighted thresholding method (PWTH) is given as Algorithm 5, in which the line search technique for updating L is between lines 9 and 12.

Algorithm 5 $\{\hat{x}, \hat{w}, \hat{L}\} \leftarrow PWTH(L_1, \lambda_0, x^0)$

Input: $L_1, s_f, s_g, x^0, \lambda_0, N$ is a positive integer;
// $L_1 \in [L_{min}, L_{max}]$
Output: $\hat{x}, \hat{w}$.

1: initialize $k \leftarrow 0$, $s_0 \leftarrow 0$, $\Delta \leftarrow \lceil s_f/N \rceil$, $\rho \geq 1$, $\epsilon > 0$, $\gamma > 1$;
2: **for** $k = 1 : N$ **do**
3: $\quad i \leftarrow 0$;
4: $\quad x^{k,i} \leftarrow x^{k-1}$;
5: $\quad s_k \leftarrow \min\{s_{k-1} + \Delta, s_f\}$;
6: $\quad \lambda_k \leftarrow \rho\lambda_{k-1}$;
7: $\quad$ **repeat**
8: $\quad\quad (x^{k,i+1}, w^{k,i+1}) \leftarrow T_{\lambda_k, L_k}(x^{k,i})$;
9: $\quad\quad L_{k,i} \leftarrow L_k$;
10: $\quad\quad$ **while** $f(x^{k,i+1}) > f(x^{k,i}) + \nabla f(x^{k,i})^T(x^{k,i+1} - x^{k,i}) + \frac{L_{k,i}}{2}\|x^{k,i+1} - x^{k,i}\|^2$ **do**
11: $\quad\quad\quad L_{k,i} \leftarrow \min\{\gamma L_{k,i}, L_{max}\}$; $(x^{k,i+1}, w^{k,i+1}) \leftarrow T_{\lambda_k, L_{k,i}}(x^{k,i})$;
12: $\quad\quad$ **end while**
13: $\quad\quad L_{k,i+1} \leftarrow L_{k,i}$;
14: $\quad\quad i \leftarrow i + 1$;
15: $\quad$ **until** $\|x^{k,i} - x^{k,i+1}\| / \max\{\|x^{k,i}\|, 10^{-6}\} < \epsilon$
16: $\quad x^k \leftarrow x^{k,i}$, $w^k \leftarrow w^{k,i}$, $L_{k+1} \leftarrow L_{k,i}$;
17: **end for**
18: $\hat{x} \leftarrow x^N$, $\hat{w} \leftarrow w^N$, $\hat{L} \leftarrow L_{N+1}$.

B. SYNTHETIC DATA

In this subsection, we compare the performance of PWTH with several existing algorithms for problem (1). We make comparisons with HT-ISTAL [1], grLasso [6], cMCP and

TABLE 1. Indexes for comparisons.

index	description
No	the number of selected nonzero elements
NoG	the number of selected nonzero groups
FN	the number of truly nonzero variables that were not selected
FNG	the number of truly nonzero groups that were not selected
FP	the number of false positive variables
FPG	the number of false positive groups
CPU	cpu time in seconds

gBridge [9], gel [10], and SGL [21]. Our aim is to find an optimal solution of the following linear regression problem:

$$\min_{x\in\mathbb{R}^n}\{\frac{1}{2}\|Ax-b\|_2^2 : \|x\|_0 \le s_f, \|x\|_G \le s_g\}.$$

1) PARAMETER SETTINGS

Similar to that in [13], we set

$$L_1 = \max_{1\le j\le n} \|A_j\|^2,$$

and $\gamma = 2$. In our algorithm, we initialize s as $s_0 = 0$, and set

$$s_k = \min\{s_{k-1} + \Delta, s\},$$

where $\Delta = \lceil s/N \rceil$ and $k = 1, 2, \ldots, N$. Thus, it will reach the target sparsity level s when $k = N$.

Since 0 is the optimal solution of problem (4) with $\lambda = \|\nabla f(0)\|_\infty$ and $s = 0$, we set $x^0 = 0$. Further, we use

$$\frac{\|x^{k+1} - x^k\|}{\max\{\|x^k\|, 10^{-6}\}} \le \epsilon,$$

as the termination condition of Algorithm 5, where ϵ is a small constant.

Then, we use some indexes in [9] for comparing the performance of all the methods. These indexes are listed in Table 1.

To see the effect of the values of parameters ρ and N on the performance of our PWTH method, we generated 10 instances and test the performance of our PWTH. In the experiments, A (with size 300×1000) is the Gaussian matrix, whose components obey the Gaussian distribution. The 1000 features (columns) are partitioned into 100 groups with equal size. The truth vector x^* (with size $n = 1000$) is drawn identically from the Gaussian distribution and possesses 30 nonzero groups. In addition, only 4 elements in each nonzero group are nonzero. Then the observations can be obtained by $b = Ax^* + z$, where z follows the normal distribution $\mathcal{N}(0, 0.5^2)$. Here, we set $\epsilon = 10^{-5}$.

The average values of the evaluation indexes are listed in Table 2. From Table 2, we can observe that ρ has almost no effect on the quality of the results including running times. By observing Table 2, we can also find that the algorithm has good performance when $N = 10$. Hence we just set $N = 10$ and $\rho = 2$ in the next experiments. Moreover, in order to save running time, ϵ is allowed to be some bigger value in the first $N - 1$ iterations of outer loop. More specifically, we set $\epsilon = 10^{-5} * 10^{\lfloor (N-k)/2 \rfloor}$ $(k = 1, 2, \ldots, N)$ in the experiments.

2) COMPARISON WITH OTHER METHODS

We compare our PWTH with HT-ISTAL[1] [1]. This experiment was implemented in MATLAB R2016b. We generated a $p \times 1000$ matrix A, whose components obey the Gaussian distribution $\mathcal{N}(0, 1)$. The number of samples $p \in \{500, 400, 350, 300, 250, 200\}$. The 1000 features (columns) are partitioned into 100 groups with equal size. The truth vector x^* possesses 30 nonzero groups, and every nonzero group has 10 nonzero elements which follow the

[1]Code for HT-ISTAL: https://github.com/coderxiang/MachLearnScripts.

TABLE 2. Results of PWTH with different ascent speeds (ρ) and outer loop numbers (N).

N	1							2						
ρ	CPU	No	FN	FP	NoG	FPG	FNG	CPU	No	FN	FP	NoG	FPG	FNG
1	0.93	120	52.8	52.8	30	7.1	7.1	1.08	120	31.9	31.9	30	3.2	3.2
2	0.92	120	52.8	52.8	30	7.1	7.1	1.08	120	31.9	31.9	30	3.2	3.2
5	0.93	120	52.8	52.8	30	7.1	7.1	1.07	120	31.9	31.9	30	3.2	3.2
10	0.93	120	52.8	52.8	30	7.1	7.1	1.08	120	31.9	31.9	30	3.2	3.2
	3							5						
1	1.44	120	18.6	18.6	30	1.7	1.7	1.57	120	23.8	23.8	30	2.3	2.3
2	1.43	120	18.6	18.6	30	1.7	1.7	1.59	120	23.8	23.8	30	2.3	2.3
5	1.43	120	18.6	18.6	30	1.7	1.7	1.59	120	23.8	23.8	30	2.3	2.3
10	1.45	120	18.6	18.6	30	1.7	1.7	1.64	120	23.8	23.8	30	2.3	2.3
	10							15						
1	2.34	120	16.1	16.1	30	1.5	1.5	3.08	120	17.8	17.8	30	1.7	1.7
2	2.31	120	16.1	16.1	30	1.5	1.5	3.03	120	17.8	17.8	30	1.7	1.7
5	2.28	120	16.1	16.1	30	1.5	1.5	3.06	120	17.8	17.8	30	1.7	1.7
10	2.27	120	16.1	16.1	30	1.5	1.5	3.06	120	17.8	17.8	30	1.7	1.7
	20							25						
1	4.27	120	26	26	30	2.6	2.6	5.18	120	31.9	31.9	30	3.5	3.5
2	4.29	120	26	26	30	2.6	2.6	5.23	120	31.9	31.9	30	3.5	3.5
5	4.23	120	26	26	30	2.6	2.6	5.27	120	31.9	31.9	30	3.5	3.5
10	4.27	120	26	26	30	2.6	2.6	5.24	120	31.9	31.9	30	3.5	3.5
	30							35						
1	5.47	120	21.2	21.2	30	2.3	2.3	5.58	120	21.2	21.2	30	2.3	2.3
2	5.43	120	21.2	21.2	30	2.3	2.3	5.54	120	21.2	21.2	30	2.3	2.3
5	5.53	120	21.2	21.2	30	2.3	2.3	5.58	120	21.2	21.2	30	2.3	2.3
10	5.62	120	21.2	21.2	30	2.3	2.3	5.55	120	21.2	21.2	30	2.3	2.3

TABLE 3. Average results of PWTH and HT-ISTAL.

SAMPLE	method	time	No	FN	FP	NoG	FPG	FNG
700	PWTH	0.72	120	2.68	2.68	30	0	0
	HT-ISTAL	1.36	120	4.77	4.77	30	0.23	0.23
650	PWTH	0.77	120	2.58	2.58	30	0	0
	HT-ISTAL	1.75	120	6.7	6.7	30	0.53	0.53
600	PWTH	0.81	120	2.72	2.72	30	0	0
	HT-ISTAL	2.05	120	10.32	10.32	30	1.04	1.04
550	PWTH	0.83	120	3.09	3.09	30	0	0
	HT-ISTAL	2.42	120	17.98	17.98	30	2.14	2.14
500	PWTH	0.86	120	2.9	2.9	30	0	0
	HT-ISTAL	2.11	120	25.8	25.8	30	3	3
450	PWTH	1.01	120	3.7	3.7	30	0	0
	HT-ISTAL	3.74	120	33.2	33.2	30	4	4
400	PWTH	1.05	120	3.2	3.2	30	0	0
	HT-ISTAL	2.53	120	48.6	48.6	30	7.1	7.1
350	PWTH	1.28	120	3.8	3.8	30	0	0
	HT-ISTAL	2.73	120	49.8	49.8	30	6.8	6.8
300	PWTH	1.76	120	15.4	15.4	30	1.6	1.6
	HT-ISTAL	3.3	120	62.2	62.2	30	9.5	9.5
250	PWTH	2.59	120	63.3	63.3	30	8.54	8.54
	HT-ISTAL	2.68	120	59.1	59.1	30	6.9	6.9
200	PWTH	4.04	120	79.8	79.8	30	12.8	12.8
	HT-ISTAL	7.51	120	76.2	76.2	30	12.2	12.2

TABLE 4. Average results of compared algorithms with respect to different group sparsities.

K2	method	time	No	FN	FP	NoG	FPG	FNG
3	cMCP	0.11	28.57	0.95	2.38	3.46	0.46	0
	gBridge	0.09	40.71	11.71	1	3.06	0.08	0.02
	grLasso	0.14	100.8	70.8	0	5.04	2.04	0
	gel	0.14	60.16	30.16	0	3.13	0.13	0
	PWTH	0.10	30	1.54	1.54	3	0	0
6	cMCP	0.19	55.44	0.99	5.55	6.17	0.17	0
	gBridge	0.26	87.06	31.41	4.35	6.71	0.93	0.22
	grLasso	0.25	238	178	0	11.9	5.9	0
	gel	0.39	120	60	0	6	0	0
	PWTH	0.13	60	3.25	3.25	6	0	0
9	cMCP	0.35	81.16	2.55	11.39	9.52	0.52	0
	gBridge	0.72	141.25	62.93	11.68	10.55	2.22	0.67
	grLasso	0.71	532.8	442.8	0	26.64	17.64	0
	gel	0.82	173.25	84.86	1.61	8.98	0	0.02
	PWTH	0.17	90	8.84	8.84	9	0.22	0.22
12	cMCP	0.43	84.67	9.93	45.26	17.48	5.48	0
	gBridge	1.37	189.95	92.64	22.69	13.72	3.11	1.39
	grLasso	1.06	690.4	570.4	0	34.52	22.52	0
	gel	0.85	189.74	93.91	24.17	10.24	0.07	1.83
	PWTH	0.25	120	34.62	34.62	12	1.47	1.47
15	cMCP	0.39	73.31	16.43	93.12	24.72	9.86	0.14
	gBridge	2.04	225.79	121.18	45.39	16.74	4.43	2.69
	grLasso	1.34	764.6	614.6	0	38.23	23.23	0
	gel	0.86	190.2	96.02	55.82	10.86	0.53	4.67
	PWTH	0.36	150	65.01	65.01	15	3.13	3.13

normal distribution $\mathcal{N}(0, 1)$. Then the observations can be obtained by $b = Ax^* + z$, where z follows the distribution $\mathcal{N}(0, 0.5^2)$.

Table 3 shows the average results of the 100 instances. From Table 3, we can see that when the number of samples $p \geq 350$, our PWTH can correctly identify the underlying groups and features. However, it is not the case until $n = 650$ on which HT-ISTAL gets a good solution. When $p \leq 250$, the results of indexes of PWTH are worse than those of HT-ISTAL, but it is apparent that both the two algorithms cannot identify the groups and features. Overall, PWTH outperforms HT-ISTAL both in solution quality and running time.

We also conducted experiments to compare the performance of our PWTH with some state-of-the-art algorithms, including gel [10], cMCP and gBridge [9], and grLasso [6]. The algorithms were implemented in R language.[2]

[2]Related code: https://cran.r-project.org/web/packages/grpreg/index.html.

TABLE 5. Average results of compared algorithms with respect to different numbers of nonzero elements within a group.

K1	method	time	No	FN	FP	NoG	FPG	FNG
2	cMCP	0.09	18.25	0	1.75	9.91	0	0.09
	gBridge	0.21	49.46	33.97	4.51	8.21	0.17	1.96
	grLasso	0.51	474.6	454.9	0.3	23.73	13.88	0.15
	gel	0.39	141.63	122.85	1.22	9.65	0	0.35
	PWTH	0.11	20	0.27	0.27	10	0	0
4	cMCP	0.13	35.98	0	4.02	10	0	0
	gBridge	0.34	84.41	50.37	5.96	9.63	0.68	1.05
	grLasso	0.74	565.6	525.6	0	28.28	18.28	0
	gel	0.77	174.39	135.43	1.04	9.88	0	0.12
	PWTH	0.12	40	0.66	0.66	10	0	0
6	cMCP	0.19	53.09	0.05	6.96	10	0	0
	gBridge	0.45	104.31	51.82	7.51	10.08	0.83	0.75
	grLasso	0.75	555.8	495.8	0	27.79	17.79	0
	gel	0.77	161.33	106.49	5.16	9.62	0	0.38
	PWTH	0.14	60	1.3	1.3	10	0	0
8	cMCP	0.29	70.74	0.38	9.64	10.09	0.09	0
	gBridge	0.83	142.27	75.18	12.91	11.69	2.74	1.05
	grLasso	0.86	582.6	502.6	0	29.13	19.13	0
	gel	0.82	167.9	94.34	6.44	9.62	0	0.38
	PWTH	0.18	80	2.9	2.9	10	0.03	0.03
10	cMCP	0.44	87.41	2.32	14.91	10.56	0.56	0
	gBridge	0.89	155.41	70.84	15.43	11.5	2.45	0.95
	grLasso	0.90	589.8	489.8	0	29.49	19.49	0
	gel	0.87	179.36	85.46	6.1	9.78	0	0.22
	PWTH	0.22	100	12.44	12.44	10	0.46	0.46

Tuning parameters were set as the default values. We generated data with $p = 300$ and $n = 1000$ for all the experiments, with varying numbers of group sparsity and nonzero elements within a group.

Table 4 lists the average results with respect to a range of group sparsities $K2 \in \{3, 6, 9, 12, 15\}$. Here, every nonzero group has 10 nonzero elements. From Table 4, we can see that grLasso and gel method select too many elements, while gel is able to select groups. Contrarily, gBridge, cMCP and PWTH all work well in the selection of features and groups. In fact, Nos of PWTH are the same as true numbers of features and groups of the tested instances. This is due to that PWTH makes a full use of information about feature sparsity and group sparsity. However, gBridge selects a little more variables than the true number leading to high FN. Moreover, besides PWTH, cMCP gives the best solution in terms of solution quality and running time. So we compare PWTH with cMCP in detail in the sequel.

When $K2 = 3$, the indexes (No, FN, FP) of cMCP are (28.57,0.95,2.38), from which we can see that cMCP misses two or three components on average. While the indexes (No, FN, FP) of PWTH are (30,1.54,1.54), from which we can see that PWTH misses one or two components on average. So PWTH outperforms cMCP slightly in solution quality when $K1 = 2$. Similarly, for other $K2$, we can draw the same conclusion. Moreover, PWTH is the fastest among all the compared methods. To sum up, PWTH outperforms other algorithms in terms of solution quality and running time.

Table 5 lists the average results with respect to a range number of nonzero elements within a group $K1 \in \{2, 4, 6, 8, 10\}$. Here, we set $p = 300$, $n = 1000$ and $s_g = 10$. Similar to the above analysis of Table 4, we can argue that PWTH is the best among the compared algorithms for any $K1 \in \{2, 4, 6, 8, 10\}$.

V. CONCLUSION

By reformulating the l_0-norm sparsity constraint as an equivalent weighted l_1-norm constraint, we have proposed a weighted thresholding method for the sparse group feature selection problem, which is based on a dynamic programming algorithm. The weighted method was further improved using the homotopy technique. Computational experiments on synthetic data show that the proposed method is competitive with some state-of-the-art algorithms for the sparse group feature selection problem.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable suggestions and comments, which helped improving the quality of this manuscript greatly.

REFERENCES

[1] S. Xiang, X. Shen, and J. Ye, "Efficient nonconvex sparse group feature selection via continuous and discrete optimization," *Artif. Intell.*, vol. 224, pp. 28–50, Jul. 2015.
[2] Y. Zhao, H. Zhu, Z. Lu, R. C. Knickmeyer, and F. Zou, "Structured genome-wide association studies with Bayesian hierarchical variable selection," *Genetics*, vol. 212, no. 2, pp. 397–415, Jun. 2019.
[3] V. K. Vivekraj, D. Sen, and B. Raman, "Video skimming: Taxonomy and comprehensive survey," *ACM Comput. Surv. (CSUR)*, vol. 52, no. 5, pp. 1–38, Sep. 2019.
[4] S. Liu, Y. D. Zhang, T. Shan, and R. Tao, "Structure-aware Bayesian compressive sensing for frequency-hopping spectrum estimation with missing observations," *IEEE Trans. Signal Process.*, vol. 66, no. 8, pp. 2153–2166, Apr. 2018.
[5] Y. C. Chen and M. D. Wang, "Worst-case hardness of approximation for sparse optimization with l_0 norm," Tech. Rep., 2016. [Online]. Available: http://www.optimization-online.org/DB HTML/2016/02/5334.html
[6] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Roy. Stat. Soc. B*, vol. 68, no. 1, pp. 49–67, Feb. 2006.
[7] L. Meier, S. Van De Geer, and P. Bühlmann, "The group lasso for logistic regression," *J. Roy. Stat. Soc. B (Stat. Methodol.)*, vol. 70, no. 1, pp. 53–71, Aug. 2010.
[8] J. Huang, S. Ma, H. Xie, and C.-H. Zhang, "A group bridge approach for variable selection," *Biometrika*, vol. 96, no. 2, pp. 339–355, Jun. 2009.
[9] P. Breheny and J. Huang, "Penalized methods for bi-level variable selection," *Statist. Interface*, vol. 2, no. 3, pp. 369–380, 2009.
[10] P. Breheny, "The group exponential lasso for bi-level variable selection," *Biometrics*, vol. 71, no. 3, pp. 731–740, Sep. 2015.
[11] W. Zhu, H. Huang, L. Jiang, and J. Chen, "Weighted thresholding homotopy method for sparsity constrained optimization," Tech. Rep., 2018. [Online]. Available: https://www.optimization-online.org/DB_FILE/2018/12/6986.html
[12] Z. Zhang, Y. Xu, J. Yang, X. Li, and D. Zhang, "A survey of sparse representation: Algorithms and applications," *IEEE Access*, vol. 3, pp. 490–530, 2015.
[13] L. Xiao and T. Zhang, "A proximal-gradient homotopy method for the sparse least-squares problem," *SIAM J. Optim.*, vol. 23, no. 2, pp. 1062–1091, Jan. 2013.
[14] Z. Xu, X. Chang, F. Xu, and H. Zhang, "$L_{1/2}$ regularization: A thresholding representation theory and a fast solver," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1013–1027, Jul. 2012.
[15] C. Soussen, J. Idier, J. Duan, and D. Brie, "Homotopy based algorithms for l_0-regularized least-squares," *IEEE Trans. Signal Process.*, vol. 63, no. 13, pp. 3301–3316, Jul. 2015.
[16] D. L. Donoho and Y. Tsaig, "Fast solution of ℓ_1-norm minimization problems when the solution may be sparse," *IEEE Trans. Inf. Theory*, vol. 54, no. 11, pp. 4789–4812, Nov. 2008.
[17] Y. Jiao, B. Jin, and X. Lu, "A primal dual active set with continuation algorithm for the l_0-regularized optimization problem," *Appl. Comput. Harmon. Anal.*, vol. 39, no. 3, pp. 400–426, Nov. 2015.
[18] A. Beck and N. Hallak, "On the minimization over sparse symmetric sets: Projections, optimality conditions, and algorithms," *Math. Oper. Res.*, vol. 41, no. 1, pp. 196–223, Feb. 2016.
[19] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009.
[20] Z. Lu, "Iterative hard thresholding methods for l_0 regularized convex cone programming," *Math. Program.*, vol. 147, nos. 1–2, pp. 125–154, Oct. 2014.
[21] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso," *J. Comput. Graph. Statist.*, vol. 22, no. 2, pp. 231–245, May 2012.

JINGLAN WU received the B.Sc. degree in applied mathematics from East China Normal University, in 1991, and the M.Sc. degree in computer science from Fuzhou University, in 2004. Since 1991, she has been with Minjiang University, where she has been an Associate Professor, since 2007. Her research interests include sparse optimization and machine learning.

HUATING HUANG received the B.Sc. degree in information and computing science and the M.Sc. degree in operations research from Fuzhou University, Fuzhou, China, in 2015 and 2018, respectively. Her research interest is sparse optimization and applications.

WENXING ZHU received the B.Sc. degree in applied mathematics and the M.Sc. and Ph.D. degrees in operations research from Shanghai University, Shanghai, China, in 1989, 1992, and 1996, respectively. Since 1996, he has been with Fuzhou University, where he has been a Professor with the Center for Discrete Mathematics and Theoretical Computer Science, since 2004. He has published more than 65 articles in refereed journals, including the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CAD, IEEE TRANSACTIONS ON SMCC, IEEE TRANSACTIONS ON NNLS, IEEE TRANSACTIONS ON MC, IEEE TRANSACTIONS ON EC, *INFORMS Journal on Computing*, and *SIAM Journal on Discrete Mathematics*. His research interests include optimization theory and algorithms, algorithms for VLSI design automation, and algorithms for sparse optimization. He received the Best Paper Award at Design Automation Conference (DAC), in 2017, the Best Paper Award nominee at International Conference on Computer-Aided Design (ICCAD), in 2018, the First Place in ICCAD 2017 Contest, and the First Place in ICCAD 2018 Contest.