Design of sparse filters by a discrete filled function technique

Zhi Guo Feng · ✉ Ka Fai Cedric Yiu · Soon Yi Wu ·

Abstract In this paper, we consider the sparse filter design problem where some of the coefficients can be reduced to zeroes in order to lower implementation complexity. The objective is to choose the fewest number of nonzero filter coefficients to meet a given performance requirement. We formulate a discrete optimization problem to minimize the number of nonzero terms and develop a discrete search method to find the minimal nonzero terms. In each step, we need to consider a subproblem to design the filter coefficients with a given set of nonzero terms. We formulate this subproblem as a linear programming problem and apply an exchange algorithm to find the optimal coefficients. For illustration, we compare the proposed algorithm with existing methods and show that the proposed method gives better results in all our test cases.

Keywords Sparse filter design · filled function · discrete search method

1 Introduction

Digital finite-impulse response (FIR) filters design have been widely studied in the literature. There are many optimization methods available to deal with this class of problems. In particular, Parks-McClellan method is recognized to

Zhi Guo Feng
Faculty of Mathematics and Computer Science, Guangdong Ocean University, Zhanjiang, Guangdong, PR China, E-mail: 18281102@qq.com

✉ Ka Fai Cedric Yiu
Department of Applied Mathematics, The Hong Kong Polytechnic University, Hunghom, Kowloon, Hong Kong, PR China, Tel.: +852-34008981, E-mail: macyiu@polyu.edu.hk

Soon Yi Wu
Department of Mathematics and National Center for Theoretical Sciences, National Cheng Kung University, Tainan, Taiwan, E-mail: soonyi@mail.ncku.edu.tw

be the most efficient method to design infinite wordlength filter coefficients. For hardware implementation, it's desirable to reduce the complexity of the designed filters. One way is to use the space of powers-of-two for the filter coefficients so that the wordlength becomes finite. Several methods have been proposed in the literature, such as L_1 norm method [13], mixed integer linear programming (MILP) [12] and other discrete search methods [2,3,18]. In [11], a frequency response masking technique is proposed to design sharp FIR filters with very few nonzero filter coefficients. An important insight in reducing implementation complexity is the sparseness of nonzero elements so that arithmetic operations can be omitted if the corresponding coefficient is zero. In view of this, the nonzero coefficients should be determined such that the number of nonzero terms is minimum, subject to a given performance requirement. This problem can find many applications including beamformers [**?**, 4, 5, 8, 16] and other digital processing devices. The cost of implementation can be reduced by using maximally sparse microphone array elements.

The main difficulty of sparse filter design is the combinatorial nature in dealing with the possible number of nonzero terms. Because of this difficulty, very few papers in the literature have attempted this problem. In [1], an approximate method was used where the number of nonzero terms are expressed by the norm $||\cdot||_0$. The cost function of this problem can then be approximated by $||\cdot||_p$ with a small p. In [9], an approximate method was applied with small coefficients being forced to zero. Some other methods can be found in [7, 15], where discrete optimization problems were formulated but the complexity of the algorithms are high. In short, there are very few discrete optimization methods that can handle this NP-complete problem efficiently. In this paper, we propose a discrete filled function approach to tackle this problem. The concept of filled function was first introduced in [6] for global optimization with continuous variables. It searches for a global minimizer among the local minimizers by means of a function, which is called a filled function. A discrete filled function method was developed in [14] for solving a discrete global optimization problem. It searches for a local minimizer by a local search method. Then, a discrete filled function is constructed, from which a better local minimizer, if it exists, is obtained.

During the iterative process in finding the right number of nonzero elements, a subproblem is incurred where the set of nonzero terms are prescribed and a sparse FIR filter is designed. As some of the harmonics are forced to carry zero weights, the determination of the filter coefficients is no longer straightforward even for infinite wordlengths because the Parks-McClellan method cannot be applied without the complete set of harmonics. However, this subproblem can still be formulated into a linear semi-infinite programming problem. After discretization, it reduces to solving a single linear programming problem. If the grid is sufficiently dense in order to maintain accuracy, the dimension of the linear programming problem will be high, which increases significantly the complexity of the overall algorithm. The linear programming method can be applied after a sufficiently dense grid points are used to dis-

cretize the infinite constraints. In order to avoid this problem, a dedicated exchange algorithm is developed for solving this subproblem efficiently.

The rest of the paper is organized as follows. First, we formulate the sparse filter design problem in Section 2 and transform it into a bi-level optimization problem in Section 3. In Section 4, we develop a dedicated exchange algorithm to tackle the subproblem. In Section 5, we develop a discrete filled function algorithm to determine the sparseness of the filter. For illustration, we show how the proposed algorithm performs better than the other algorithms in the literature in Section 6.

2 Problem formulation

Ignoring the linear phase term, the frequency response of a linear phase FIR filter is given by

$$H(\omega) = \boldsymbol{h}^{\intercal}\boldsymbol{C}(\omega), \tag{2.1}$$

where $\boldsymbol{h} = (h_1, \cdots, h_m)^{\intercal}$, and m is defined by $m = (L+1)/2$ if the filter length L is odd and $m = L/2$ if L is even. $\boldsymbol{C}(\omega) = (C_1(\omega), C_2(\omega), \ldots, C_m(\omega))^{\intercal}$ is an appropriate cosine function vector given by

$$C_k(\omega) = \begin{cases} \cos(k-1)\omega, & \text{if } L \text{ is odd,} \\ \cos(k-0.5)\omega, & \text{otherwise,} \end{cases} \quad k = 1, \ldots, m.$$

A general filter design problem is to find the coefficient vector $\boldsymbol{h}$ such that the frequency response $H(\omega)$ given in (2.1) fits a given desired frequency response function $H_d(\omega)$, that is,

$$\begin{aligned} &\min_{z,\boldsymbol{h}} \ z \\ &s.t. \ g\left(\boldsymbol{h}, \omega\right) \le z, \quad \forall \omega \in \mathcal{F} \subseteq [0, \pi], \end{aligned} \tag{2.2}$$

where $\mathcal{F}$ is the specified region and the function g is given by

$$g\left(\boldsymbol{h}, \omega\right) = W(\omega)|H(\omega) - H_d(\omega)|,$$

where $W(\omega)$ is a positive weighting function which measure the importance of passband and stopband.

We denote the optimal value of (2.2) by z^*. However, the implementation of the filter coefficients may be expensive to achieve the performance value z^*, since there may exist many nonzero coefficients. For this, we should design the filter to achieve a better performance value, and at the same time that the implementation complexity of the filter coefficients can also be reduced. That is, we suppose that the performance requirement is $\delta > z^* > 0$, then the filter should be designed to satisfy the following constraint

$$g\left(\boldsymbol{h}, \omega\right) \le \delta.$$

For the implementation complexity of the filter coefficients, it can be expressed as the total number of nonzero filter coefficients. Then, we can reduce

the implementation complexity by minimizing the total number of nonzero filter coefficients. For this, we use the zero norm function as follows.

$$||x||_0 = \begin{cases} 1, & \text{if } x \neq 0 \\ 0, & \text{if } x = 0 \end{cases}. \tag{2.3}$$

Then, for each k, the value $||h_k||_0$ indicates that the filter coefficient h_k is used or not. The total number of nonzero filter coefficients can be expressed by

$$f(\boldsymbol{h}) = \begin{cases} ||h_1||_0 + 2\sum\limits_{k=2}^{m} ||h_k||_0, & \text{if } L \text{ is odd,} \\ 2\sum\limits_{k=1}^{m} ||h_k||_0, & \text{if } L \text{ is even.} \end{cases} \tag{2.4}$$

Then, the sparse filter design problem is formulated into an optimization problem as

$$\begin{aligned} &\min_{\boldsymbol{h}} f(\boldsymbol{h}) \\ &s.t.\ g(\boldsymbol{h}, \omega) \leq \delta, \quad \forall \omega \in \mathcal{F} \subseteq [0, \pi]. \end{aligned} \tag{2.5}$$

Note that the zero norm function is discontinuous, general optimization method can not be applied for solving this problem. It's required to develop efficient and effective method to solve this problem.

Remark 1 The filter length L is an important factor for the problem (2.5). If L is too small, the constraint in (2.5) may not be satisfied and there is no solution for (2.5). However, if L is sufficiently large, the constraint in (2.5) can always be satisfied. Hence, we always choose a sufficiently large value for L such that the constraint can be satisfied.

3 Problem Transformation

For the problem (2.5), we decompose it into two subproblems as follows.

The coefficient vector $\boldsymbol{h}$ can be treated as two kinds of variables. One is the set of chosen base, the other is the coefficient vector of the corresponding chosen base. For the chosen base, it's from the basis set $\{C_k(\omega) : k = 1, \ldots, m\}$. Assume that we choose m_Λ basis functions from this set, where the index is $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{m_\Lambda}\}$, which is a subset of the index set $\{1, \ldots, m\}$. That is, for a given Λ, we only use the following base from $\{C_{\lambda_k}(\omega) : k = 1, \ldots, m_\Lambda\}$. Denote a vector of these chosen base by

$$\bar{\boldsymbol{C}}(\omega, \Lambda) = \left(C_{\lambda_1}(\omega), C_{\lambda_2}(\omega), \ldots, C_{\lambda_{m_\Lambda}}(\omega)\right)^{\mathsf{T}},$$

and the corresponding coefficient vector is denoted by

$$\bar{\boldsymbol{h}}(\Lambda) = \left(h_{\lambda_1}, h_{\lambda_2}, \ldots, h_{\lambda_{m_\Lambda}}\right)^{\mathsf{T}}.$$

For the index which is not in Λ, the corresponding coefficient is zero, that is,

$$h_k = 0, \quad \forall k \in \{1, \ldots, m\} \setminus \Lambda.$$

Hence, we can solve the problem (2.5) by finding the index set of chosen base Λ and the corresponding coefficient vector $\bar{\boldsymbol{h}}(\Lambda)$. Note that $\bar{\boldsymbol{h}}(\Lambda)$ is a continuous vector, we can find the optimal $\bar{\boldsymbol{h}}(\Lambda)$ for any given Λ first.

For any chosen index set Λ, the filter response (2.1) can be rewritten as

$$H(\omega, \Lambda) = \sum_{k=1}^{m_\Lambda} h_{\lambda_k} C_{\lambda_k}(\omega) = \bar{\boldsymbol{h}}^{\mathsf{T}}(\Lambda)\bar{\boldsymbol{C}}(\omega, \Lambda). \tag{3.1}$$

The general filter design problem (2.2) is transformed into

$$\begin{aligned} &\min_{z(\Lambda), \bar{\boldsymbol{h}}} \ z(\Lambda) \\ &\ s.t.\ g\left(\bar{\boldsymbol{h}}(\Lambda), \omega\right) \le z(\Lambda), \quad \forall \omega \in \mathcal{F} \subseteq [0, \pi], \end{aligned} \tag{3.2}$$

where

$$g\left(\bar{\boldsymbol{h}}(\Lambda), \omega\right) = W(\omega)|H(\omega, \Lambda) - H_d(\omega)|.$$

Hence, if Λ is given, we can find the corresponding optimal coefficient vector $\bar{\boldsymbol{h}}^*(\Lambda)$ and optimal value $z^*(\Lambda)$ by solving the problem (3.2). Next, we should check whether the constraint in (2.5) is satisfied. That is, if $z^*(\Lambda) \le \delta$, then $(\Lambda, \bar{\boldsymbol{h}}^*(\Lambda))$ is a feasible solution in the problem (2.5), else there is no feasible solution for this problem.

For any set Λ, the cost function (2.4) becomes

$$F(\Lambda) = \begin{cases} 2m_\Lambda - 1, & \text{if } L \text{ is odd and } 1 \in \Lambda, \\ 2m_\Lambda, & \text{otherwise.} \end{cases} \tag{3.3}$$

Then, the problem (2.5) is transformed into

$$\begin{aligned} &\min_{\Lambda} \ F(\Lambda) \\ &\ s.t.\ z^*(\Lambda) \le \delta, \end{aligned} \tag{3.4}$$

where $z^*(\Lambda)$ is obtained by solving (3.2).

Thus, we have formulated two subproblems (3.2) and (3.4) for solving the problem (2.5). Note that the problem (3.2) is a semi-infinite programming (SIP) optimization problem and the problem (3.4) is a discrete optimization problem. It's required to develop the corresponding methods for solving these two problems.

4 Exchange Algorithm

For the problem (3.2), it can be solved very efficiently by Remez exchange algorithm in the case that $\Lambda = \{1, \ldots, k\}$ is consecutive. However, for the general sparse case that $\Lambda = \{1, \ldots, k\}$ is not consecutive, the optimal value can not be calculated directly in each iteration if the Remez exchange algorithm is applied. Hence, we need to develop a modified exchange algorithm for solving this problem.

The constraint in Problem (3.2) can be rewritten as

$$|W(\omega)H(\omega, \Lambda) - W(\omega)H_d(\omega)| \leq z(\Lambda),$$

which is equivalent to a linear form as

$$\begin{aligned} W(\omega)H(\omega, \Lambda) - W(\omega)H_d(\omega) - z(\Lambda) &\leq 0, \\ -(W(\omega)H(\omega, \Lambda) - W(\omega)H_d(\omega)) - z(\Lambda) &\leq 0. \end{aligned}$$

Substituting (3.1) and reorganizing it in the form of vector, Problem (3.2) is equivalent to

$$\begin{aligned} \min_{z(\Lambda), \bar{\boldsymbol{h}}(\Lambda)} \quad & z(\Lambda) \\ s.t. \ & \boldsymbol{G}^+(\omega) \begin{bmatrix} z(\Lambda) \\ \bar{\boldsymbol{h}}(\Lambda) \end{bmatrix} - R^+(\omega) \geq 0, \quad \forall \omega \in \mathcal{F} \subseteq [0, \pi], \\ & \boldsymbol{G}^-(\omega) \begin{bmatrix} z(\Lambda) \\ \bar{\boldsymbol{h}}(\Lambda) \end{bmatrix} - R^-(\omega) \geq 0, \quad \forall \omega \in \mathcal{F} \subseteq [0, \pi], \end{aligned} \tag{4.1}$$

where

$$\boldsymbol{G}^+(\omega) = \begin{bmatrix} 1 & -W(\omega)\bar{\boldsymbol{C}}(\omega, \Lambda) \end{bmatrix}, \tag{4.2a}$$

$$\boldsymbol{G}^-(\omega) = \begin{bmatrix} 1 & W(\omega)\bar{\boldsymbol{C}}(\omega, \Lambda) \end{bmatrix}, \tag{4.2b}$$

$$R^+(\omega) = -W(\omega)H_d(\omega), \tag{4.3a}$$

$$R^-(\omega) = W(\omega)H_d(\omega). \tag{4.3b}$$

We denote the problem (4.1) as $P(\mathcal{F}_+, \mathcal{F}_-)$, where $\mathcal{F}_+$ denotes the indices set $\mathcal{F}$ in the first constraint and $\mathcal{F}_-$ denotes the indices set $\mathcal{F}$ in the second constraint.

Basically, after the infinite set $\mathcal{F}$ is replaced by its sufficiently dense discrete set, Problem (4.1) is transformed into a linear programming problem and can be solved by any linear programming method. In general, if the number of constraints is not large, the linear programming problem can be solved efficiently by the simplex method. However, if the number of constraints is large enough, the interior point method is better than the simplex method for solving this problem.

For the semi-infinite programming problem (4.1), the number of constraints is large in general after discretization, if the precision of performance is required. Then, the computation is expensive even if the interior point method is used. For this, we develop an algorithm for this problem as follows.

For the infinite set $\mathcal{F}$, we choose any finite sets $\Omega_+ = \{\omega_1, \ldots, \omega_{l_1}\} \in \mathcal{F}$ and $\Omega_- = \{\omega_{l_1+1}, \ldots, \omega_{l_2}\} \in \mathcal{F}$. Define a subproblem of (4.1) as

$$P(\Omega_+, \Omega_-):$$

$$\begin{aligned} \min_{\bar{\boldsymbol{h}}(\Lambda), z(\Lambda)} \quad & z(\Lambda) \\ s.t.\ & \boldsymbol{G}^+(\Omega_+) \begin{bmatrix} z(\Lambda) \\ \bar{\boldsymbol{h}}(\Lambda) \end{bmatrix} - \boldsymbol{R}^+(\Omega_+) \geq 0 \\ & \boldsymbol{G}^-(\Omega_-) \begin{bmatrix} z(\Lambda) \\ \bar{\boldsymbol{h}}(\Lambda) \end{bmatrix} - \boldsymbol{R}^-(\Omega_-) \geq 0, \end{aligned} \tag{4.4}$$

where

$$\boldsymbol{G}^+(\Omega_+) = \begin{bmatrix} 1 & -W(\omega_1)\bar{\boldsymbol{C}}(\omega_1, \Lambda) \\ \vdots & \vdots \\ 1 & -W(\omega_{l_1})\bar{\boldsymbol{C}}(\omega_{l_1}, \Lambda) \end{bmatrix}, \boldsymbol{G}^-(\Omega_-) = \begin{bmatrix} 1 & W(\omega_{l_1+1})\bar{\boldsymbol{C}}(\omega_{l_1+1}, \Lambda) \\ \vdots & \vdots \\ 1 & W(\omega_{l_2})\bar{\boldsymbol{C}}(\omega_{l_2}, \Lambda) \end{bmatrix}$$

and

$$\boldsymbol{R}^+(\Omega_+) = \begin{bmatrix} -W(\omega_1)H_d(\omega_1, \Lambda) \\ \vdots \\ -W(\omega_{l_1})H_d(\omega_{l_1}, \Lambda) \end{bmatrix}, \boldsymbol{R}^-(\Omega_-) = \begin{bmatrix} W(\omega_{l_1+1})H_d(\omega_{l_1+1}, \Lambda) \\ \vdots \\ W(\omega_{l_2})H_d(\omega_{l_2}, \Lambda) \end{bmatrix}.$$

We introduce a modified exchange algorithm as follows:

Algorithm 1

Step 0. (Initialization)
Choose initial finite sets $\Omega_+^{(0)}, \Omega_-^{(0)} \subset \mathcal{F}$. Set $\bar{\Omega}^{(0)} = (\Omega_+^{(0)}, \Omega_-^{(0)})$. Solve the problem $P\left(\bar{\Omega}^{(0)}\right)$ to obtain the optimal solution $(z^{(0)}, \bar{\boldsymbol{h}}^{(0)})$. Set the relative error $\varepsilon > 0$. $k = 0$.

Step 1. (Stop test)
Calculate the constraint values in (4.1) for all $\omega \in \mathcal{F}$, by using the coefficient vector $(z^{(k)}, \bar{\boldsymbol{h}}^{(k)})$. If all the constraint values are greater than $-z^{(k)}\varepsilon$, $\forall \omega \in \mathcal{F}$, then return $\bar{\boldsymbol{h}}^{(k)}$ as the optimal solution and stop, else goto Step 2.

Step 2. (Obtain the sets of all the minimizers)
Find all the minimizers of the first constraint in (4.1) which are less than 0. Denote the set of all these points by $\left\{\omega_{new}^{(k+)}\right\}$. Find all the minimizers of the second constraint in (4.1) which are less than 0. Denote the set of all these points by $\left\{\omega_{new}^{(k-)}\right\}$.

Step 3. (Obtain the active sets)
Check the constraint values of the problem $P\left(\bar{\Omega}^{(k)}\right)$ and find all the points where the corresponding constraint values greater than 0. Denote the set $\left\{\omega_{ina}^{(k+)}\right\}$ of all these points in $\Omega_+^{(k)}$ and denote the set $\left\{\omega_{ina}^{(k-)}\right\}$ of all these points in $\Omega_-^{(k)}$. Then, the active sets are

$$\left\{\omega_{act}^{(k+)}\right\} = \Omega_+^{(k)} \setminus \left\{\omega_{ina}^{(k+)}\right\}, \quad \left\{\omega_{act}^{(k-)}\right\} = \Omega_-^{(k)} \setminus \left\{\omega_{ina}^{(k-)}\right\}.$$

Step 4. (Exchange rule)
Exchange the finite set as

$$\begin{aligned}\Omega_+^{(k+1)} &= \left\{\omega_{act}^{(k+)}\right\} \cup \left\{\omega_{new}^{(k+)}\right\} = \Omega_+^{(k)} \cup \left\{\omega_{new}^{(k+)}\right\} \setminus \left\{\omega_{ina}^{(k+)}\right\}, \\ \Omega_-^{(k+1)} &= \left\{\omega_{act}^{(k-)}\right\} \cup \left\{\omega_{new}^{(k-)}\right\} = \Omega_-^{(k)} \cup \left\{\omega_{new}^{(k-)}\right\} \setminus \left\{\omega_{ina}^{(k-)}\right\}. \end{aligned} \tag{4.5}$$

Set $\bar{\Omega}^{(k+1)} = (\Omega_+^{(k+1)}, \Omega_-^{(k+1)})$.
Step 5. (Calculate the solution)
Solve the problem $P\left(\bar{\Omega}^{(k+1)}\right)$ *to obtain the optimal solution* $(z^{(k+1)}, \bar{\boldsymbol{h}}^{(k+1)})$.
Set $k = k + 1$. *Goto Step 1.*

The convergence results of this exchange algorithm can be found in [17]. Then, we can apply Algorithm 1 to solve Problem (4.1).

Remark 2 The merit of the exchange algorithm is that its computational complexity is almost independent of the number of grid size. This can be seen from the fact that in each iteration, a linear programming subproblem (4.4) which is independent of grid size is solved. After solving every subproblem, the computation of finding the maximum points is related to the grid size. However, the number of iterations is always very small, and this can be ignored when compared to that of solving the subproblem.

5 Filled Function Discrete Search Method

Since the set Λ is a discrete variable, Problem (3.2) is a discrete optimization problem. There are several hurdles we need to overcome. First, Λ is a set and is not convenient to be applied in discrete space search. To handle this, we introduce an equivalent definition as

$$\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)^\intercal, \quad \sigma_i \in \{0, 1\}. \tag{5.1}$$

It means that if $\sigma_i = 1$ (or 0), the basis function $C_i(\omega)$ is used (or not used). That is, $\boldsymbol{\sigma}$ and Λ are one to one correspondence. We can denote the index which corresponds to $\boldsymbol{\sigma}$ by $\Lambda(\boldsymbol{\sigma})$.

Note that for each element σ_i, it has two choices 0 and 1. Hence, the space of all feasible $\boldsymbol{\sigma}$ is $\{0, 1\}^m$ and the total number of this space is

$$2 \times 2 \cdots \times 2 = 2^m.$$

Remark 3 If $\boldsymbol{\sigma} = (0, \ldots, 0)^\intercal$, that is, the corresponding Λ is an empty set and there is no $C_i(\omega)$ used, then its cost function value cannot be evaluated. For convenience, we can penalize its cost function value by a sufficiently large value so that it will not influence the solution process.

Then, Problem (3.2) is equivalent to

$$\begin{aligned} \min_{\boldsymbol{\sigma}} \quad & \tilde{F}(\boldsymbol{\sigma}) \\ s.t. \quad & \tilde{z}^*(\boldsymbol{\sigma}) \leq \delta, \end{aligned} \tag{5.2}$$

where $\tilde{F}(\boldsymbol{\sigma})$ and $\tilde{z}^*(\boldsymbol{\sigma})$ are given by

$$\begin{aligned} \tilde{F}(\boldsymbol{\sigma}) &= F(\Lambda(\boldsymbol{\sigma})), \\ \tilde{z}^*(\boldsymbol{\sigma}) &= z^*(\Lambda(\boldsymbol{\sigma})). \end{aligned}$$

It can be seen that Problem (5.2) is a constrained optimization problem. Penalty function method can be applied and the problem is transformed into

$$\min_{\boldsymbol{\sigma}} \quad \bar{F}(\boldsymbol{\sigma}), \tag{5.3}$$

where

$$\bar{F}(\boldsymbol{\sigma}) = \begin{cases} \tilde{F}(\boldsymbol{\sigma}) + \tilde{z}^*(\boldsymbol{\sigma}), & \text{if } \tilde{z}^*(\boldsymbol{\sigma}) \leq \delta, \\ \tilde{F}(\boldsymbol{\sigma}) + \tilde{z}^*(\boldsymbol{\sigma}) + N, & \text{if } \tilde{z}^*(\boldsymbol{\sigma}) > \delta, \end{cases}$$

where N is a sufficiently large positive value.

Problem (5.3) is a discrete optimization problem. Note that for the case of continuous optimization problem, the gradient information is used in many methods, that is, for each iteration, the method calculate the gradient of current iterate and then generate the next iterate. But for the discrete optimization problem, we can't have any gradient information. However, we can imitate this search idea in continuous case by mimicking the function of a gradient. To achieve this, we introduce the definition of neighborhood as

Definition 1 For any $\boldsymbol{\sigma} \in \{0,1\}^m$, the neighborhood of $\boldsymbol{\sigma}$ is defined by

$$\mathcal{N}(\boldsymbol{\sigma}) = \{\boldsymbol{\sigma}' : ||\boldsymbol{\sigma}' - \boldsymbol{\sigma}|| \leq 1\},$$

where the norm $||\cdot||$ is given by

$$||\boldsymbol{\sigma}' - \boldsymbol{\sigma}|| = \sum_{i=1}^{m} |\sigma_i' - \sigma_i|.$$

From Definition 1, we can see that the neighborhood of any $\boldsymbol{\sigma}$ consists of itself, and all the points with $m-1$ elements same as itself and 1 element different to itself. Then, the number of points in any neighborhood is $m+1$.

Then, similar to the steepest descent algorithm in continuous case, we propose a discrete version of steepest descent algorithm as follows, where in each iteration, the next iterate point is obtained by searching over the current neighborhood and selecting the point which produces the largest reduction in the cost function value.

Algorithm 2

Step 0. Choose an initial point $\boldsymbol{\sigma}^{(0)}$*. Compute its cost function value* $\bar{F}\left(\boldsymbol{\sigma}^{(0)}\right)$*. Set* $k = 0$*.*

Step 1. For each point in the neighborhood $\boldsymbol{\sigma} \in \mathcal{N}\left(\boldsymbol{\sigma}^{(k)}\right) \setminus \left\{\boldsymbol{\sigma}^{(k)}\right\}$*, where* $A_1 \setminus A_2 = \{\boldsymbol{\sigma} \in A_1 : \boldsymbol{\sigma} \notin A_2\}$*, compute the corresponding cost function value* $\bar{F}(\boldsymbol{\sigma})$*. Choose the point* $\boldsymbol{\sigma}^{min}$ *which* $\bar{F}(\boldsymbol{\sigma}^{min})$ *is the minimum. If* $\bar{F}(\boldsymbol{\sigma}^{min}) \geq \bar{F}\left(\boldsymbol{\sigma}^{(k)}\right)$*, then return* $\boldsymbol{\sigma}^{(k)}$ *as the solution and stop, else goto Step 2.*

Step 2. Set $\boldsymbol{\sigma}^{(k+1)} = \boldsymbol{\sigma}^{min}$ *and* $k = k + 1$*. Goto Step 1.*

After applying Algorithm 2, we can find a point which minimizes the cost function $\bar{F}(\boldsymbol{\sigma})$ over its neighborhood. We call this point as a local minimizer, similar to that of continuous case. The definition of local minimizer is given by

Definition 2 A point $\boldsymbol{\sigma}^*$ is called a local minimizer of $\bar{F}$ over $\{0,1\}^m$ if $\bar{F}(\boldsymbol{\sigma}^*) \leq \bar{F}(\boldsymbol{\sigma})$, $\forall \boldsymbol{\sigma} \in \mathcal{N}(\boldsymbol{\sigma}^*)$.

A minimizer is the best point only in its neighborhood. For the region far away from the point, there may exists a better point. In other words, there may exist many local minimizers for this problem and it's required to choose the best minimizer. To achieve this, we should jump out of the neighborhood of current minimizer and at the same time keep the corresponding cost function value relatively small. This will help to find a possible better point. A method to implement this idea is the discrete filled function, which is constructed in [14]. The discrete filled function is given by

$$\hat{F}_{\mu,\rho}(\boldsymbol{\sigma};\boldsymbol{\sigma}^*) = \mu[\bar{F}(\boldsymbol{\sigma}) - \bar{F}(\boldsymbol{\sigma}^*)]^2 - \rho||\boldsymbol{\sigma} - \boldsymbol{\sigma}^*||^2, \quad \text{if } \bar{F}(\boldsymbol{\sigma}) \geq \bar{F}(\boldsymbol{\sigma}^*), \tag{5.4}$$

where $\rho > 0$, $0 < \mu < \rho/K$ (K is a sufficiently large real number).

Then, after a local minimizer is obtained, we can start from this point and search for a possible better local minimizer by minimizing the discrete filled function with Algorithm 2 again. Since the maximum distance between any two points in the space $\{0,1\}^m$ is m, we can set the maximal number of searching steps as m. If the number of searching steps is greater than m, we can't find the improved solution and then stop. The main algorithm to solve Problem (5.3) is summarized below.

Algorithm 3

Step 0. Choose an initial point $\boldsymbol{\sigma}^{(0)}$*. Compute its cost function value* $\bar{F}\left(\boldsymbol{\sigma}^{(0)}\right)$*. Set the optimal solution and optimal value as* $\boldsymbol{\sigma}^* = \boldsymbol{\sigma}^{(0)}$ *and* $v^* = \bar{F}\left(\boldsymbol{\sigma}^{(0)}\right)$*.* $k = 0$*.*

Step 1. Apply Algorithm 2 to obtain a local minimizer of the cost function $\bar{F}$ *with the starting point* $\boldsymbol{\sigma}^{(k)}$*. Let the local minimizer be denoted as* $\boldsymbol{\sigma}^{(k+1)}$*. Set the optimal solution and optimal value as* $\boldsymbol{\sigma}^* = \boldsymbol{\sigma}^{(k+1)}$ *and* $v^* = \bar{F}\left(\boldsymbol{\sigma}^{(k+1)}\right)$*.* $k = k + 1$*.*

Step 2. Set $\boldsymbol{\sigma}^*$ as the starting point, and apply Algorithm 2 to search for a point better than the current local minimizer, with the discrete filled function $\hat{F}$ in (5.4) used as the cost function. Denote l by the number of searching steps.

Step 3. If a point $\boldsymbol{\sigma}'$ is found such that $\bar{F}(\boldsymbol{\sigma}') < \bar{F}(\boldsymbol{\sigma}^*)$ when $l \leq m$ during Step 2, then stop searching. Set $\boldsymbol{\sigma}^{(k)} = \boldsymbol{\sigma}'$ and goto Step 1. Else, stop searching and return the optimal solution $\boldsymbol{\sigma}^*$ and optimal value v^*.

To apply Algorithm 3, a good initial point is often required. In general, it can be chosen as $\boldsymbol{\sigma}^{(0)} = (1, 1, \ldots, 1)^\intercal$, where its cost function value of (2.2) is the best. If this value does not satisfy the performance requirement in (3.2), it means that there is no feasible solution and it's not necessary to apply Algorithm 3.

6 Illustrative Examples

In this section, we implemented the proposed method and compared the numerical results with other methods, where the computation was performed in Matlab.

6.1 Example of exchange algorithm

First, we investigate the numerical results of the proposed exchange algorithm for solving subproblem 4.1. For the convenience of comparison with interior point method, which is the most efficient method for solving large scale linear optimization problems, the discretization of $\mathcal{F}$ for the exchange algorithm and interior algorithm is chosen as the same. In Matlab, the interior point solver LIPSOL is implemented in the function linprog with the option 'LargeScale=on'.

The test example we considered is Problem (2.2) with odd, symmetric filter coefficients and the desired frequency response function given by

$$H_d(\omega) = \begin{cases} 1, \text{ if } \omega \in [0, \omega_p], \\ 0, \text{ if } \omega \in [\omega_s, \pi], \end{cases} \tag{6.1}$$

where $\omega_p = 0.3\pi$ and $\omega_s = 0.5\pi$. The weighting function is given by $W(\omega) = 1$ and the number of independent filter coefficients is $m = 30$. The discrete grid points is taken every interval from $\pi/400$, $\pi/500$ to $\pi/2000$. We execute the program 100 times by starting from a different randomly selected $\boldsymbol{\sigma}$ using the function $rand(\boldsymbol{\sigma} > 0.5)$; that is, if the random number of σ_i is greater than 0.5, then set $\sigma_i = 1$, else set $\sigma_i = 0$. Then, we implement the exchange method and the interior point method respectively, and the mean running times are depicted in Figure 1.

From Figure 1, we can see that the complexity of the exchange method is almost independent of the number of grid points. It's better than the interior point method, especially when the number of grid points increases.

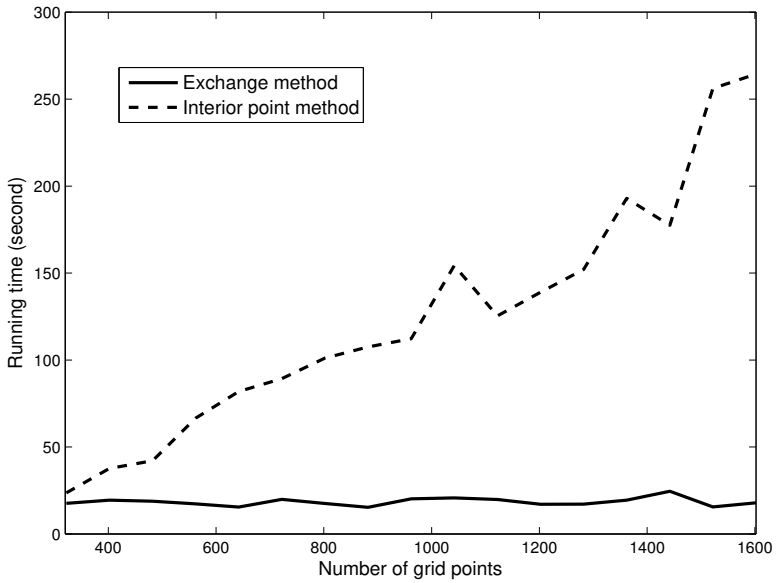

Fig. 1 Running times of the exchange method and interior point method.

Next, we fix the interval as 0.001π and set m from 5 to 50. We execute the program 100 times by starting from a different randomly selected $\boldsymbol{\sigma}$ as above. The comparison of the total running times for the exchange method and the interior point method is depicted in Figure 2.

It can be seen from Figure 2 that the complexities of the exchange method and the interior point method increases if m increases, and the exchange method is more efficient than the interior point method.

6.2 Example of discrete search method

Then, we implement the discrete search method to solve Problem (3.2). For the comparison, we first consider the examples in [1], where a design of uniformly linear beamformer is considered. The desired beam pattern is chosen with mainlobe region given by $[0, 0.0436\pi]$ and the sidelobe region given by $[0.0872\pi, \pi]$. The mainlobe magnitude is within ± 0.5dB of unity and the sidelobe magnitude is below -20, -30, and -40 dB, respectively. The comparison results are summarized in Table 1, 2 and 3. We can see that the proposed method is better than the methods in [1].

To show the filter response, we plot its magnitude in Figure 3 in the case of -20dB sidelobe level, where the filter length is chosen as 45. The nonzeros terms are obtained as $\{1 - 12, 20, 22, 23\}$.

Next, we consider the example in [13], where the mainlobe region is given by $[0, 0.55\pi]$ and the sidelobe region is given by $[0.6\pi, \pi]$. The maximum value is chosen as -33.42dB. The filter length is chosen as $L = 65$. We apply the

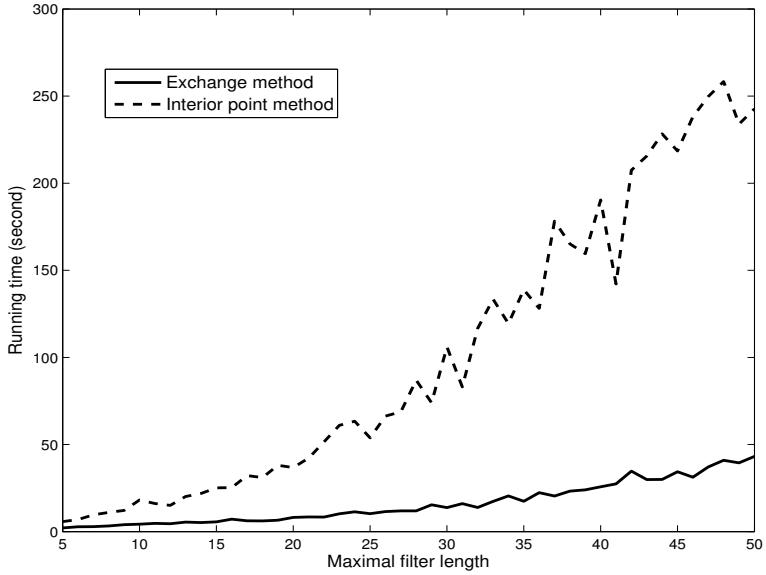

Fig. 2 Running times of the exchange method and interior point method.

Method	Nonzero terms	Filter length (L)
Parks-McClellan	43	43
Minimum-increase in [1]	29	49
Smallest-coefficient in [1]	31	47
Minimum 1-norm in [1]	29	51
Proposed method	31	43
	29	45
	28	48

Table 1 Numbers of nonzero terms and the filter lengths, where the sidelobe level is -20dB.

Method	Nonzero terms	Filter length (L)
Parks-McClellan	55	55
Minimum-increase in [1]	47	55
Smallest-coefficient in [1]	47	55
Minimum 1-norm in [1]	47	55
Proposed method	47	55
	46	56

Table 2 Numbers of nonzero terms and the filter lengths, where the sidelobe level is -30dB.

Method	Nonzero terms	Filter length (L)
Parks-McClellan	79	79
Minimum-increase in [1]	65	83
Smallest-coefficient in [1]	69	83
Minimum 1-norm in [1]	73	83
Proposed method	66	78
	67	79
	65	81

Table 3 Numbers of nonzero terms and the filter lengths, where the sidelobe level is -40dB.

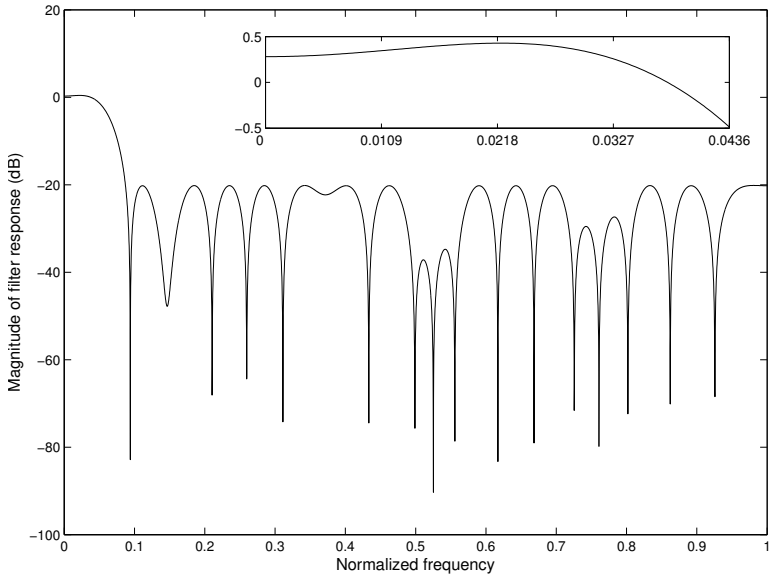

Fig. 3 Magnitude of the filter response, where the filter length is 45 and the sidelobe level is -20dB.

proposed method to solve this problem and the total number of nonzero filter coefficients can achieve 51, which is better than the method in [13]. The nonzeros terms are obtained as $\{1-14, 16-19, 21, 23-26, 28, 30, 33\}$ and the corresponding magnitude of filter response is depicted in Figure 4.

7 Conclusion

In this paper, we have studied the sparse filter design problem and proposed a new algorithm. First, we have formulated a linear semi-infinite programming problem to find the optimal filter coefficients for the subproblems where the set of nonzero terms is given. Then, we developed a dedicated exchange algorithm for solving this problem and showed that it's more efficient than the interior point algorithm. Second, by introducing the neighborhood concept for the discrete space, we have proposed a discrete search method based on the discrete filled function to find the minimal number of nonzero terms. Numerical results indicates that it's better than existed methods.

Acknowledgements This work is supported by RGC Grant PolyU. (152200/14E) and PolyU Grant 4-ZZGS. The first author is also supported by the National Natural Science Foundation of China (No. 61673078), the grant of Chongqing Science and Technology Commission (No. cstc2017jcyjAX0161) and the grant of Chongqing Normal University (No. 17XLB010).

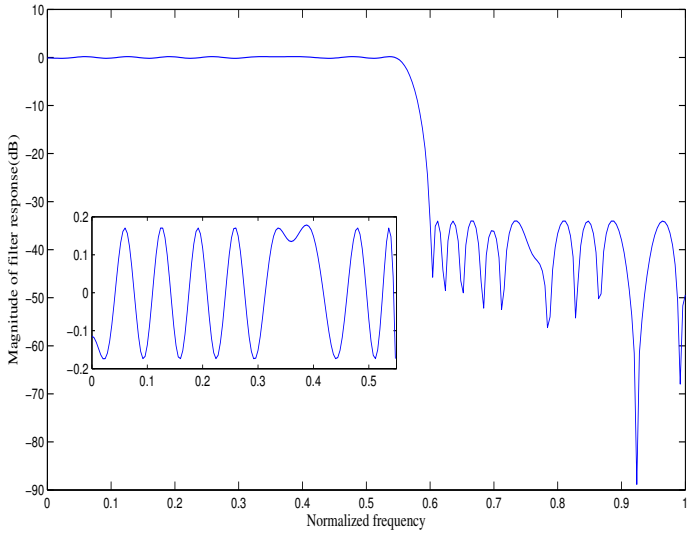

Fig. 4 Magnitude of the filter response, where the filter length is 65 and the sidelobe level is -33.42dB.

References

1. Baran, T., Wei, D., Oppenheim, A.V.: Linear programming algorithms for sparse filter design. IEEE Transactions on Signal Processing **58**(3) (2010)
2. Chen, C.L., Willson, A.N.: A trellis search algorithm for the design of FIR filters with signed-powers-of-two coefficients. IEEE Trans. Circuits Syst. II **46**, 29–39 (1999)
3. Feng, Z.G., Teo, K.L.: A discrete filled function method for the design of FIR filters with signed-powers-of-two coefficients. IEEE Trans. Signal Processing **56**(1), 134–139 (2008)
4. Feng, Z.G., Yiu, K.F.C., Nordholm, S.: A two-stage method for the design of near-field multi-dimensional broadband beamformer. IEEE Trans. Signal Processing **59**(8), 3647–3656 (2011)
5. Feng, Z.G., Yiu, K.F.C., Nordholm, S.: Performance limit of broadband beamformer designs in space and frequency. Journal of Optimization Theory and Application **164**, 316–341 (2015)
6. Ge, R.P.: A filled function method for finding a global minimizer of a function of several variables. Mathematical Programming **46**, 191–204 (1990)
7. Kim, J.T., Oh, W.J., Lee, Y.H.: Design of nonuniformly spaced linear-phase FIR filters using mixed integer linear programming. IEEE Transactions on Signal Processing **44**(1) (1996)
8. Leahy, R.M., Jeffs, B.D.: On the design of maximally sparse beamforming arrays. IEEE Transactions on Antennas Propag. **39**(8), 1178–1187 (1991)
9. Liang, J.K., Figueiredo, R.D., Lu, F.: Design of optimal Nyquist, partial response, Nth band, and nonuniform tap spacing FIR digital filters using linear programming techniques. IEEE Transactions on Circuits Syst. **CAS-32**(4), 386–392 (1985)
10. Lim, Y.C.: Frequency-response masking approach for the synthesis of sharp linear phase digital filters. IEEE Trans. Circuits Syst. **ASSP-33**, 357–364 (1986)
11. Lim, Y.C., Lian, Y.: Frequency-response masking approach for digital filter design: Complexity reduction via masking filter factorization. IEEE Trans. Circuits Syst. **41**, 518–525 (1994)

12. Lim, Y.C., Parker, S.R.: FIR filter design over a discrete powers-of-two coefficient space. IEEE Trans. Acoust. Speech Signal Process **31**, 583–591 (1983)
13. Lu, W.S., Hinamoto, T.: Digital filters with sparse coefficients. In: Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 169–172. Paris, France (2010)
14. Ng, C.K., Zhang, L.S., Li, D., Tian, W.W.: Discrete filled function method for discrete global optimization. Computational Optimization and Applications **31**, 87–115 (2005)
15. Song, Y.S., Lee, Y.H.: Design of sparse FIR filters based on branch and bound algorithm. Proc. 40th Midwest Symp. Circuits and Systems **2** (1997)
16. Webb, J.L.H., Jr., D.C.M.: Chebyshev optimization of sparse FIR filters using linear programming with an application to beamforming. IEEE Transactions on Signal Processing **44**(8), 1912–1922 (1991)
17. Yiu, K.F.C., Gao, M.J., Shiu, T.J., Tran, T., Wu, S.Y., Claesson, I.: A fast algorithm for the optimal design of high accuracy windows in signal processing. Optimization Methods and Software **28**(4), 900–916 (2013)
18. Yu, Y.J., Lim, Y.C.: Design of linear phase FIR filters in subexpression space using mixed integer linear programming. IEEE Transactions on circuits and systems I **54**(10), 2330–2338 (2007)