Journal of VLSI Signal Processing 21, 131–149 (1999)

Discrete Lagrangian Methods for Designing Multiplierless Two-Channel PR-LP Filter Banks

BENJAMIN W. WAH
Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA; Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong

ZHE WU
Department of Computer Science and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

Received May 30, 1998; Revised January 20, 1999

Abstract. In this paper, we present a new search method based on the theory of discrete Lagrange multipliers for designing multiplierless PR (perfect reconstruction) LP (linear phase) filter banks. To satisfy the PR constraints, we choose a lattice structure that, under certain conditions, can guarantee the resulting two filters to be a PR pair. Unlike the design of multiplierless QMF filter banks that represents filter coefficients directly using PO2 (powers-of-two) form (also called Canonical Signed Digit or CSD representation), we use PO2 forms to represent the parameters associated with the lattice structure. By representing these parameters as sums or differences of powers of two, multiplications can be carried out as additions, subtractions, and shifts. Using the lattice representation, we decompose the design problem into a sequence of four subproblems. The first two subproblems find a good starting point with continuous parameters using a single-objective, multi-constraint formulation. The last two subproblems first transform the continuous solution found by the second subproblem into a PO2 form, then search for a design in a mixed-integer space. We propose a new search method based on the theory of discrete Lagrange multipliers for finding good designs, and study methods to improve its convergence speed by adjusting dynamically the relative weights between the objective and the Lagrangian part. We show that our method can find good designs using at most four terms in PO2 form in each lattice parameter. Our approach is unique because our results are the first successful designs of multiplierless PR-LP filter banks. It is general because it is applicable to the design of other types of multiplierless filter banks.

1. Introduction

Digital filter banks have been applied in many engineering fields. Their design objectives consist of their overall metrics and the metrics of each individual filter. Figure 1 summarizes the various objectives for measuring design quality. In general, filter bank-design problems are multi-objective, nonlinear, continuous optimization problems.

Algorithms for designing filter banks are either optimization-based or non-optimization based. In optimization-based methods, a design problem is formulated as a multi-objective nonlinear optimization problem [1] whose form may be application- and filter-dependent. The problem is then converted into a single-objective optimization problem and solved by existing optimization methods, such as gradient-descent, Lagrange-multiplier, quasi-Newton, simulated-annealing, and genetics-based methods [2, 3]. On the other hand, filter bank-design problems have been solved by non-optimization-based algorithms that include spectral factorization [4, 5] and

Filter	Design Objectives
Overall	Minimize amplitude distortion
Filter	Minimize aliasing distortion
Bank	Minimize phase distortion
	Minimize stopband ripple (δ_s)
Single	Minimize passband ripple (δ_p)
Filter	Minimize transition bandwidth (T_t)
	Minimize stopband energy (E_s)
	Maximize passband flatness (E_p)

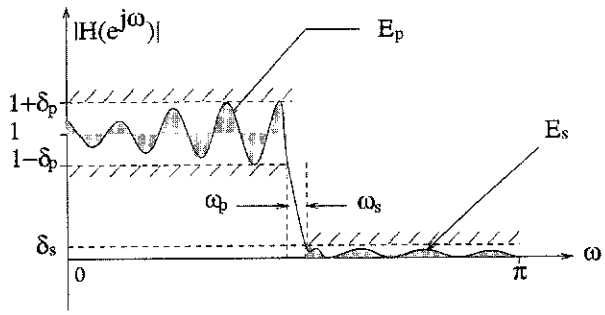


Figure 1. Possible design objectives of filter banks and an illustration of the design objectives of a single low-pass filter. ($[0, \omega_p]$ is the pass band; $[\omega_s, \pi]$, the stop band; $[\omega_p, \omega_s]$, the transition band.)

heuristic methods (as in IIR-filter design [6, 7]). These methods generally do not continue to find better designs once a suboptimal design has been found [5].

In this paper, we design new two-channel *multiplierless* PR (perfect-reconstruction) LP (linear-phase) filter banks. In multiplierless filters, each filter coefficient or parameter is represented by a limited number of terms in PO2 form rather than by a full floating point number. Such a representation is also called a Canonical Signed Digit or powers-of-two (PO2) representation. Multiplying a filter input (multiplicand) by a multiplierless coefficient (multiplier) is then carried out as a limited number of adds and shifts of the multiplicand, corresponding to the number of terms in the multiplier in PO2 form. For example, multiplying y by 0100001001 $(= 2^8 + 2^3 + 2^0)$ can be written as the sum of three terms, $y \times 2^8 + y \times 2^3 + y \times 2^0$, each of which can be obtained by shifting y. Besides faster, multiplierless operations take less area to implement in VLSI, while allowing more stages of the filter to be implemented in the same area. There is no previous study on such designs, and all existing PR-LP designs in signal processing are based on real coefficients [8–11].

In a general two-band filter bank, the reconstructed signal is:

$$\hat{X}(z) = \frac{X(z)}{2}[H_0(z)F_0(z) + H_1(z)F_1(z)] + \frac{X(-z)}{2} \times [H_0(-z)F_0(z) + H_1(-z)F_1(z)] \quad (1)$$

where $X(z)$ is the original signal, and $H_i(z)$ and $F_i(z)$ are, respectively, the response of the analysis and synthesis filters. Choosing $F_0(z) = 2H_1(-z)$ and $F_1(z) = -2H_0(-z)$ lead to a filter bank with only two prototype filters $H_0(z)$ and $H_1(z)$ and the following output:

$$\hat{X}(z) = [H_0(z)H_1(-z) - H_0(-z)H_1(z)]X(z) \quad (2)$$

Further, if we impose the pure-delay constraint as follows, where c is a non-zero real coefficient:

PR Constraints:

$$H_0(z)H_1(-z) - H_0(-z)H_1(z) = cz^{-2k+1}$$
$$\Longrightarrow \hat{X}(z) = cz^{-2k+1}X(z). \quad (3)$$

In this way, perfect reconstruction is achieved because the output is a delayed replica of the input. To reduce aliasing and phase distortions, we need to further restrict $H_0(z)$ and $H_1(z)$ to be of linear phase as follows:

LP Constraints:

$$H_0(n) = H_0(N - n + 1);$$
$$H_1(n) = -H_1(N - n + 1) \quad (4)$$

for all $n \in [1, N]$, where N is the filter length. These constraints can be enforced by choosing $H_0(z)$ to be symmetric and $H_1(z)$ to be antisymmetric.

There are two general approaches to the design of PR-LP filter banks. One approach [3] begins from a starting point (represented by $H_0(z)$ and $H_1(z)$) and tries to optimize some performance metrics, like passband energy, stopband energy, and transition bandwidth, while keeping constraints (3) and (4) satisfied. This is not practical when the coefficients are multiplierless because the equality constraints (3) imposed by the PR condition are very hard to satisfy when the search space is discrete and a feasible design occurs at only a few points in discrete space. Intuitively, this is true because values represented by a small number of terms in PO2 form are a proper subset of all floating point values, and equality constraints generally have less feasible solutions in the restricted space.

The second approach [12] formulates a PR-LP filter bank in a lattice structure in order to enforce some constraints in its design. Figure 2 shows an example lattice structure, where $k_1, \ldots, k_{N-1}$, B_1 and B_2 are unknown

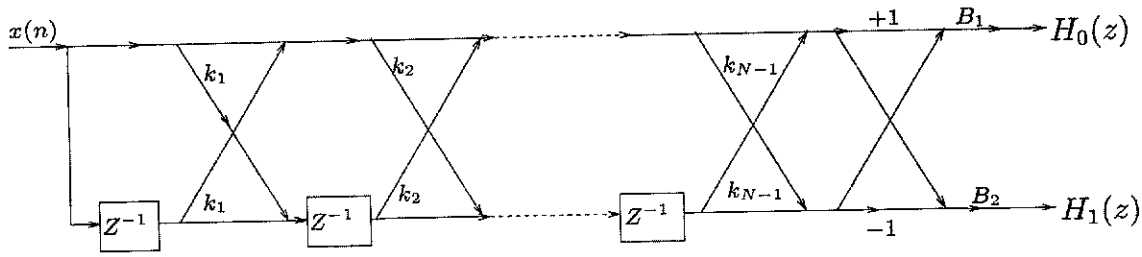


Figure 2. Two-channel lattice structure.

parameters. We state without proof [13] that when the filter length N is even and $k_i = 0$ for all even i, the resulting filters $H_0(z)$ and $H_1(z)$ in Fig. 2 are a PR pair that satisfy (3). Moreover, the LP constraints (4) are satisfied automatically. In short, the design a PR-LP filter bank based on a lattice structure only involves searching the space of variables $k_1, k_3, \ldots, k_{N-1}$, B_1, and B_2.

In matrix form, $H_0(z)$ and $H_1(z)$ can be rewritten as follows:

$$\begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix} = \begin{bmatrix} B_1 & 0 \\ 0 & B_2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & k_{N-1} \\ k_{N-1} & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & k_1 \\ k_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & z^{-1} \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{5}$$

The design of multiplierless PR-LP filter banks in this paper is based on the second approach that represents each $k_i, i = 1, \ldots, N-1$, using a small, finite number (say 3 or 4) of terms in PO2 form. This multiplierless representation allows the lattice structure to be implemented efficiently in hardware. We consider only the design of two-channel filter banks with the same even length N for both filters.

Existing designs of multiplierless filter banks have been restricted to those with FIR filters due to limitations in the available search methods. Optimization methods that have been applied include integer programming, combinatorial search [14], simulated annealing [15], genetic algorithms [16], linear programming [17], and continuous Lagrange-multiplier methods in conjunction with a tree search [18]. In our previous work, we have applied a search method based on discrete Lagrange multipliers to design multiplierless QMF filter banks [19, 20]. This design problem is easier than our current design problem as it involves only inequality constraints and discrete variables. In contrast, our current design problem involves mixed (integer as well as real) variables and equality constraints, and cannot be solved without the development of new search methods.

We extend our previous local-search method based on the theory of *discrete Lagrange multipliers* to design two-channel multiplierless PR-LP filter banks formulated as single-objective nonlinear mixed optimization problems with equality constraints (Section 2). Since the original formulation is too hard to be solved, it is decomposed into a sequence of four design subproblems. Section 3 summarizes the theory of discrete Lagrange multipliers. In contrast to pure descent methods, Lagrangian methods have additional strategies to escape from infeasible local minima once the search gets there. Based on the theory of discrete Lagrange multipliers, we then present a discrete local-search method that finds equilibrium points in discrete space and show that the equilibrium points found correspond to local minima in the feasible space. The methods studied are local search because they stop at feasible local minima and do not have strategies to escape from the feasible local minima once they are found. Section 4 examines issues related to the implementation of our discrete Lagrangian method (DLM-98) to design multiplierless PR-LP filter banks, discusses the convergence speed of discrete Lagrangian methods, and presents our discrete Lagrangian method with dynamic weight adaptation. Section 5 presents experimental results, and conclusions are drawn in Section 6.

2. Problem Formulation

The design of a two-channel PR-LP filter bank is a multi-objective constrained optimization problem with objectives measured by metrics in Fig. 1, the PR constraints defined in (3), and the LP constraints defined in (4). As mentioned in the last section, the LP constraints are satisfied automatically by using a lattice structure.

2.1. *General Formulation*

Since there is no general method to solve multi-objective constrained optimization problems, we first transform our design problem into a single-objective constrained optimization problem. There are two ways for this transformation.

First, the multiple performance objectives are combined into a single objective represented by a weighted sum [1, 2, 21–23]. For instance, to minimize the passband and stopband energies of the two filters in a two-channel filter bank, one possible formulation using a subset of the metrics is as follows:

$$\min \quad w_1 E_p^0 + w_2 E_p^1 + w_3 E_s^0 + w_4 E_s^1 \qquad (6)$$

where

$$E_p^i = \int_{\omega=0}^{\omega_p} (|H_i(e^{j\omega})| - 1)^2 \, d\omega$$

and

$$E_s^i = \int_{\omega=\omega_s}^{\pi} |H_i(e^{j\omega})|^2 \, d\omega, \quad i = 0, 1$$

subject to PR and LP constraints.

Unfortunately, solutions obtained by solving (6) depend heavily on the relative magnitudes of w_1, w_2, w_3, and w_4. To obtain designs with certain characteristics, different combinations of weights must be tested by trial and error. It is difficult to find the right combination of weights to use in the objective when the required characteristics of a feasible design are hard to satisfy. Further, unless the formulation considers all possible performance metrics, metrics not involved in the formulation are generally compromised.

Another approach in solving a multi-objective problem is to turn all but one objectives into constraints, and define the constraints with respect to a reference design. The specific metrics constrained may be application- and filter-dependent [3].

For instance, given a reference design with five performance metrics for each filter, we can formulate the design into a problem with ten constraints, in addition to the PR and LP constraints:

$$\begin{array}{lll} \min & f(x) = E_s^0 + E_s^1 + E_p^0 + E_P^1 & \\ \text{subject to} & E_p^i \le \theta_{E_p^i}, & E_s^i \le \theta_{E_s^i}, \\ & \delta_p^i \le \theta_{\delta_p^i}, & \delta_s^i \le \theta_{\delta_s^i}, \\ & T_t^i \le \theta_{T_t^i}, & i = 0, 1 \\ & \text{PR constraints,} & \text{LP constraints} \end{array} \qquad (7)$$

where $x = \{k_1, k_3, \ldots, k_{N-1}\}$ is a vector of $N/2$ discrete coefficients, B_1 and B_2 are two continuous parameters, and the θ's are performance metrics of the reference design.

2.2. *Four Design Subproblems*

The problem formulated in (7) is difficult to be solved directly because it is a nonlinear mixed integer optimization problem involving integer and real variables. The PR constraints in (3) are particularly hard to satisfy in multiplierless designs because they are nonlinear equality constraints involving mixed variables. In this paper, we propose an approach that solves (7) using a sequence of four subproblems specified in Table 1.

The goal of the first three subproblems is to find designs that have proper "shapes" in their frequency responses ($H_0(z)$ in the form of a low-pass filter and $H_1(z)$ in the form of a high-pass filter), while ignoring ripples and transition bandwidth. Here, we focus only on the energies because the original problem (7) is too difficult to be solved when additional metrics are included. The simplified problem involving only the passband and stopband energies is as follows:

$$\begin{array}{lll} \min & E_p^0 + E_p^1 + E_s^0 + E_s^1 & \\ \text{subject to} & E_p^0 \le \theta_{E_p}^0, & E_p^1 \le \theta_{E_p}^1 \\ & E_s^0 \le \theta_{E_s}^0, & E_s^1 \le \theta_{E_s}^1 \end{array} \qquad (8)$$

where $\theta_{E_p}^0$, $\theta_{E_s}^0$, $\theta_{E_p}^0$ and $\theta_{E_p}^1$ are performance bounds of the reference design.

The first subproblem involves a search space of $\frac{N}{2} + 2$ continuous variables ($k_1, k_3, \ldots, k_{N-1}$ and B_1 and B_2), and computes the energies from stopband and passband cutoff frequencies at $\frac{\pi}{2}$ (i.e., $\omega_p = \omega_s = \frac{\pi}{2}$). The reason for choosing such inaccurate cutoff frequencies is that the initial filters we start with may not

Table 1. Specifications of the four subproblems solved. (The objective in subproblem 4 only considers $E_s^0 + E_s^1$ because we found experimentally that the magnitudes of the passband energies are much smaller than those of the stopband energies and do not play any role in the final design.)

Subproblem	Formulation	$k_1, k_3, \ldots, k_{N-1}$	B_1 and B_2	ω_s and ω_p
1	Eq. (8)	Continuous	Continuous	$\frac{\pi}{2}$
2	Eq. (8)	Continuous	Continuous	From ref. design
3	Eq. (8)	PO2	Continuous	Comp. numerically
4	Eq. (7)	PO2	Continuous	Comp. numerically

be in proper shapes that allow the calculation of cutoff frequencies.

The second subproblem is solved in the same search space as the first, but have the stopband and passband cutoff frequencies computed based on the reference design. The exact cutoff frequencies are not needed because they are expensive to evaluate and are not crucial at this point.

The third subproblem has a mixed-integer space consisting of $\frac{N}{2}$ variables $(k_1, k_3, \ldots, k_{N-1})$ in PO2 form and B_1 and B_2 in continuous form. It uses the solution of the second subproblem as a starting point to find solutions with parameters having a constrained number of terms in PO2 form. The solution to this subproblem should be very close to the final desired solution.

The last subproblem uses the same variable space as the third subproblem but uses the exact formulation in (7) and computes all cutoff frequencies and energies by either numerical methods or closed-form formulae. These calculations are delayed until this point due to their high computational costs.

3. Lagrange-Multiplier Methods

In this section, we extend Lagrangian methods to solve the four subproblems described in the last section. We first summarize existing work on Lagrangian methods for solving continuous constrained optimization problems. We then present the theory of discrete Lagrange multipliers and its application to solve discrete and mixed optimization problems [24–26].

3.1. *Continuous Lagrangian Methods*

Lagrangian methods are classical methods for solving continuous constrained optimization problems [27]. We first review briefly the theory of Lagrange multipliers.

Define a constrained optimization problem as follows:

$$\begin{aligned} &\min_{x \in E^m} && f(x) \\ &\text{subject to} && g(x) \leq 0 \quad x = (x_1, x_2, \ldots, x_n) \\ &&& h(x) = 0 \end{aligned} \tag{9}$$

where x is a vector of real numbers, $f(x)$ is an objective function, $g(x) = [g_1(x), \ldots, g_k(x)]^T$ is a set of k inequality constraints, and $h(x) = [h_1(x), \ldots, h_m(x)]^T$ is a set of m equality constraints. Further, $f(x)$, $g(x)$ and $h(x)$, as well as their derivatives, are continuous functions.

Lagrange-multiplier methods introduce Lagrange multipliers to resolve constraints iteratively. It is an exact method that optimizes the objective $f(x)$ to meet the Kuhn-Tucker conditions [27].

Since Lagrangian methods cannot directly deal with inequality constraints $g_i(x) \leq 0$ in general, we transform inequality constraints into equality constraints by adding slack variables $z_i(x)$, which results in $p_i(x) = g_i(x) + z_i^2(x) = 0$. The corresponding *Lagrangian function* is defined as follows:

$$L_c(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T p(x) \tag{10}$$

where $\lambda = [\lambda_1, \ldots, \lambda_m]^T$ and $\mu = [\mu_1, \ldots, \mu_k]^T$ are two sets of Lagrange multipliers, and $p(x) = [p_1(x), \ldots, p_k(x)]^T$.

To eliminate z_i from (10), we differentiate L_c with respect to z_i for given x and set the result to zero. After solving the equality and substituting the result into (10) [27], the Lagrangian function becomes:

$$\begin{aligned} L_c(x, \lambda, \mu) = f(x) + \lambda^T h(x) \\ + \frac{1}{2} \sum_{i=1}^{k} \left[\max^2(0, \mu_i + g_i(x)) - \mu_i^2 \right] \end{aligned} \tag{11}$$

Note that the derivation applies to both the continuous and the discrete cases because the differentiation of L_c with respect to z_i is for a fixed x, and z_i is assumed continuous.

First-Order Necessary Conditions. According to classical optimization theory [27], all the (local and global) extrema of (11) that satisfy the constraints and that are regular points are roots of the following set of first-order necessary conditions:

$$\begin{aligned} \nabla_x L_c(x, \lambda, \mu) = 0, \\ \nabla_\lambda L_c(x, \lambda, \mu) = 0, \\ \nabla_\mu L_c(x, \lambda, \mu) = 0. \end{aligned} \tag{12}$$

These conditions are necessary to guarantee the (local) optimality of the solution to (9).[1]

First-Order Methods. The set of points satisfying the necessary conditions can be found by a first-order

method that can be expressed as a system of ordinary differential equations:

$$\begin{aligned}\frac{d}{dt}x(t) &= -\nabla_x L_c(x(t), \lambda(t), \mu(t)),\\ \frac{d}{dt}\lambda(t) &= \nabla_\lambda L_c(x(t), \lambda(t), \mu(t)),\\ \frac{d}{dt}\mu(t) &= \nabla_\mu L_c(x(t), \lambda(t), \mu(t)).\end{aligned} \tag{13}$$

These perform simultaneous descents in the original variable space of x and ascents in the Lagrange-multiplier space of λ and μ. The dynamic system evolves over time t, and reaches a feasible local extremum when it stops at an *equilibrium point* where all gradients become zero. Since there is no strategies to help escape from a feasible local extremum, first-order methods are considered *local-search methods*. Moreover, they require a continuous differentiable space in order to find equilibrium points satisfying the first-order necessary conditions.

3.2. *Discrete Lagrangian Methods*

In this section, we summarize the theory of discrete Lagrange multipliers we have developed [24, 26] for solving discrete optimization problems.

Since Lagrangian methods in discrete space only handle problems with equality constraints, an inequality constraint must first be transformed into an equality constraint. We transform $g_i(x) \leq 0$ into $\max(g_i(x), 0) = 0$ without using a slack variable as in the continuous case because searches in discrete space do not require the existence of gradients in the x space.

The resulting discrete Lagrangian function is formulated as follows [24]:

$$\begin{aligned}L_d(x, \lambda, \mu) &= f(x) + \lambda^T h(x)\\ &\quad + \sum_{i=1}^{k} \mu_i \max(0, g_i(x)),\end{aligned} \tag{14}$$

where x is the space of discrete variables x_i, $i = 1, \ldots, n$, and λ and μ are Lagrange multipliers that can be continuous.

The discrete Lagrangian function defined in (14) cannot be used to derive first-order necessary conditions similar to those in continuous space [27] because there are no gradients and differentiation in discrete space. Without these concepts, none of the mechanisms of calculus in continuous space is applicable in discrete space.

An understanding of gradients in continuous space shows that they define directions in a small neighborhood in which function values decrease. To this end, [24] defines in discrete space a *direction of maximum potential drop* (*DMPD*) for $L_d(x, \lambda, \mu)$ at point x for fixed λ and μ as a vector[2] that points from x to a neighbor of $x \in \mathcal{N}(x)$ with the minimum L_d:

$$\begin{aligned}\Delta_x L_d(x, \lambda, \mu) &= \vec{v}_x = y \ominus x\\ &= (y_1 - x_1, \ldots, y_n - x_n)\end{aligned} \tag{15}$$

where

$$y \in \mathcal{N}(x) \cup \{x\}$$

and

$$L_d(y, \lambda, \mu) = \min_{\substack{x' \in \mathcal{N}(x)\\ \cup\{x\}}} L_d(x', \lambda, \mu).$$

Here, $\ominus$ is the vector-subtraction operator for changing x in discrete space to one of its "user-defined" neighborhood points $\mathcal{N}(x)$. Intuitively, $\vec{v}_x$ is a vector pointing from x to y, the point with the minimum L_d value among all neighboring points of x, including x itself. That is, if x itself has the minimum L_d, then $\vec{v}_x = \vec{0}$.

Having defined DMPD $\Delta_x L_d(x, \lambda, \mu)$ in the x space, we define the concept of *saddle points* in discrete space [24, 26] similar to those in continuous space [27]. A point (x^*, λ^*, μ^*) is a saddle point when:

$$L(x^*, \lambda, \mu) \leq L(x^*, \lambda^*, \mu^*) \leq L(x, \lambda^*, \mu^*), \tag{16}$$

for all (x^*, λ, μ) and all (x, λ^*, μ^*) sufficiently close to (x^*, λ^*, μ^*). Starting from (16), we prove stronger necessary and sufficient conditions that are satisfied by all saddle points in discrete space [26].

Necessary and Sufficient Conditions for Saddle Points in Discrete Space.

$$\begin{aligned}&\Delta_x L_d(x, \lambda, \mu) = 0,\\ &\nabla_\lambda L_d(x, \lambda, \mu) = 0 \quad \text{or } \max(0, g(x)) = 0,\\ &\nabla_\mu L_d(x, \lambda, \mu) = 0 \quad (\text{or } h(x) = 0).\end{aligned} \tag{17}$$

Note that the first condition defines the DMPD of L_d in discrete space of x for fixed λ and μ, whereas the differentiations in the last two conditions are in continuous space of λ and μ for fixed x. Readers should refer

to the proofs in [24, 26] for the correctness of these conditions.

When all the constraints are non-negative, we have proved the following results.

Necessary and Sufficient Conditions for Feasible Local Minima in Discrete Space [26]. When all constraint functions are non-negative, the set of saddle points is the same as the set of feasible local minima. Hence the conditions defined in (17) are necessary and sufficient.

The first-order necessary and sufficient conditions in (17) lead to the following first-order method in discrete space that seeks discrete saddle points. The following equations are discrete approximations to implement (17).

General Discrete First-Order Method.

$$x(k+1) = x(k) \oplus \Delta_x L_d(x(k), \lambda(k)) \tag{18}$$

$$\lambda(k+1) = \lambda(k) + c_1 h(x(k)) \tag{19}$$

$$\mu(k+1) = \mu(k) + c_2 \max(0, g(x(k))) \tag{20}$$

where $\oplus$ is the vector-addition operator ($x \oplus y = (x_1 + y_1, \ldots, x_n + y_n)$), and c_1 and c_2 are positive real numbers controlling how fast the Lagrange multipliers change.

It is easy to see that the necessary condition for the discrete first-order method to converge is when $h(x) = 0$ and $g(x) \leq 0$, implying that x is a feasible solution to the original problem. If any of the constraints is not satisfied, then λ and μ on the unsatisfied constraints will continue to evolve. Note that, as in continuous Lagrangian methods, the first-order conditions are only satisfied at saddle points, but do not imply that the time to find a saddle point is finite, even if one exists.

4. DLM-98: An Implementation of Discrete First-Order Method

By transforming the inequality constraints in (7) into equality constraints, the design of multiplierless PR-LP filter banks is formulated as follows:

$$\begin{aligned} \min \quad & f(x) = E_s^0 + E_s^1 + E_p^0 + E_p^1 \\ \text{subject to} \quad & V_{E_p^i} = \max\left(E_p^i - \theta_{E_p^i}, 0\right) = 0 \\ & V_{E_s^i} = \max\left(E_s^i - \theta_{E_s^i}, 0\right) = 0 \\ & V_{\delta_p^i} = \max\left(\delta_p^i - \theta_{\delta_p^i}, 0\right) = 0 \\ & V_{\delta_s^i} = \max\left(\delta_s^i - \theta_{\delta_s^i}, 0\right) = 0 \\ & V_{T_t^i} = \max\left(T_t^i - \theta_{T_t^i}, 0\right) = 0 \\ & \text{PR constraints,} \\ & \text{LP constraints,} \quad i = 0, 1 \end{aligned} \tag{21}$$

In a similar way, the inequality constraints in (8) can be converted into equality constraints.

The discrete Lagrangian function corresponding to (21) is as follows:

$$L_d(x, \mu) = f(x) + \sum_{i \in \{E_p, E_s, \delta_p, \delta_s, T_t\}} \mu \times V_i \tag{22}$$

where x is a vector of coefficients, each of which is in CSD form of the sum of several signed binary bits, such as $2^{-1} + 2^{-3} - 2^{-6}$. Since we have only equality constraints transformed from inequality constraints, we use μ as our Lagrange multipliers in the following discussion.

Figure 3 shows an implementation of the discrete first-order method (18) and (20) for designing PO2 filter banks formulated as nonlinear discrete constrained minimization problems. The procedure shows several aspects that can be tuned in order to improve its performance.

4.1. *Generating Starting Points and Initial Constraints*

As discussed in Section 2, the design problem is broken into a sequence of four subproblems. We only need to

procedure *DLM-98*
1. set c (positive real constant for controlling the speed of change of Lagrange multipliers);
2. set i_{max} (maximum number of iterations);
3. set starting point x;
4. set initial value of μ (set to 0 in the experiments);
5. **if** using dynamic weight adaptation **then** *weight_initialization*;
6. **while** not converged **and** no. iterations $< i_{max}$ **do** {
7. update x to x' iff $L_d(x', \mu) < L_d(x, \mu)$;
8. **if** condition for updating μ is satisfied **then** $\mu_i \leftarrow \mu_i + c \cdot \max(0, g_i)$;
9. **if** using dynamic weight adaptation **then** *dynamic_weight_adaptation* }

Figure 3. DLM-98: An implementation of discrete first-order method for designing PO2 filter banks. (The initial values of parameters are indicated here unless specified otherwise in the text.)

generate starting points for the first subproblem, as the remaining subproblems take their starting points from the solutions of the previous subproblem.

To find a starting point for the first subproblem, the easiest way is to start from an existing lattice structured PR-LP filter bank. This approach is not practical because not many such filter banks exist in the literature. Hence, we need to start from scratch without assuming an initial feasible design.

Our approach to generate an initial PR-LP filter bank consisting of a pair of low-pass and high-pass filters has the following steps:

(a) Generate an initial filter bank by using a symmetric low-pass filter $G_0(z)$ in one QMF filter bank and an antisymmetric high-pass filter $G_1(z)$ in another QMF filter bank. These are called the "base" pair of filters because they are the very initial point of our design procedure. For instance, $G_0(z)$ can be the symmetric low-pass filter in 32C [2], and $G_1(z)$ be the antisymmetric high-pass filter in 32D (after changing 32D to antisymmetric using $h'(n) = (-1)^n h(n)$) (Fig. 4(a)). Note that we cannot use the low-pass and high-pass filters in the same QMF filter bank as our starting point because they are mirrors of each other at $\frac{\pi}{2}$, resulting in $k_1 = k_3 = \cdots = k_{N-1} = 0$ and the failure of the next two steps of the procedure.

(b) Compute parameters $k_0, k_1, \ldots, k_{N-1}$ of the filter bank using the iterative procedure in [3]. Given an initial pair of LP filters $G_0(z)$ and $G_1(z)$, where $G_0(z)$ is symmetric and $G_1(z)$ is antisymmetric, we set initially $B_1 = B_2 = 1.0$ and compute a pair of z transformations $T_{N-1}(z)$, $U_{N-1}(z)$ as follows:

$$\begin{bmatrix} T_{N-1}(z) \\ U_{N-1}(z) \end{bmatrix} = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} \tag{23}$$

For $j = 2, \ldots, N-1$, let $T_{N-j+1}(z)$ be represented by $\sum_{i=0}^{N-j+1} t_i z^{-i}$. Compute $k_{N-j+1} = \frac{t_{N-j+1}}{t_0}$, and $T_{N-j}(z)$ and $U_{N-j}(z)$ based on $T_{N-j+1}(z)$ and $U_{N-j+1}(z)$ as follows:

$$\begin{bmatrix} T_{N-j}(z) \\ z^{-1}U_{N-j}(z) \end{bmatrix} = \frac{1}{(1-k_{N-j+1}^2)}\begin{bmatrix} 1 & -k_{N-j+1} \\ -k_{N-j+1} & 1 \end{bmatrix} \times \begin{bmatrix} T_{N-j+1}(z) \\ U_{N-j+1}(z) \end{bmatrix} \tag{24}$$

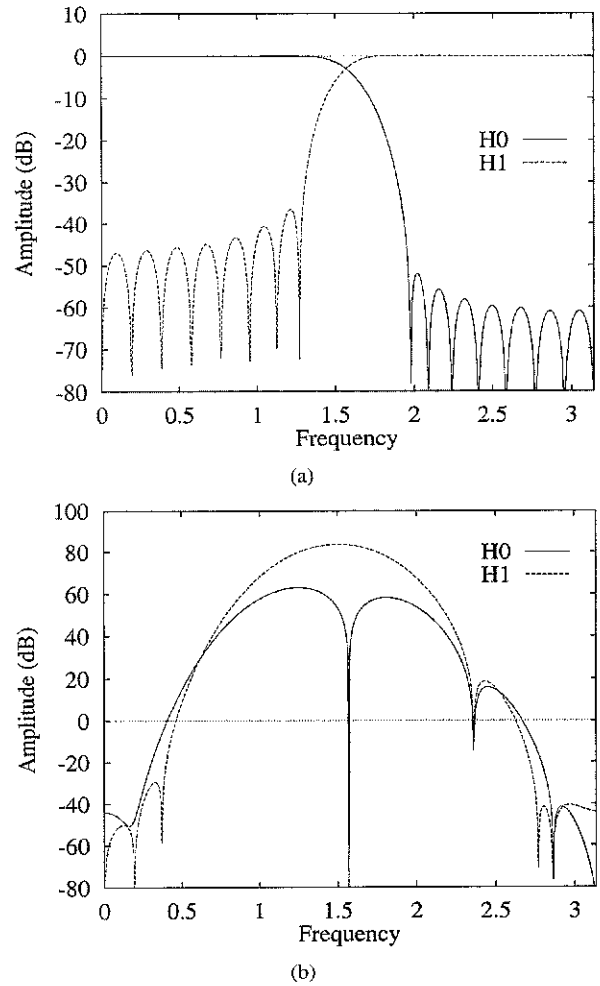


Figure 4. Generating a starting point in subproblem 1: (a) frequency responses $G_0(z)$ and $G_1(z)$ of the initial non-PR filter bank and (b) $H_0(z)$ and $H_1(z)$ of the PR filter bank supplied to subproblem 1 as its starting point.

Finally, let $T_1(z)$ be of the form $\sum_{i=0}^{1} w_i z^{-i}$, and compute $k_1 = \frac{w_1}{w_0}$. It can be proved [3] easily that if $G_0(z)$ is symmetric and $G_1(z)$ is antisymmetric, then T_j and U_j are in the jth order. Further, to ensure that the denominator is nonzero, we can apply a strategy in [3] to ensure that k_i is not 1.0.

(c) Force $k_2, k_4, \ldots, k_{N-2}$ to be zero in order to guarantee the PR property in the lattice structure. The resulting filters $H_0(z)$ and $H_1(z)$ are a PR-LP pair satisfying (3) and (4), although their behaviors can be far from desirable. Figure 4(b) plots $H_0(z)$ and $H_1(z)$ after applying steps 2 and 3. Obviously, it is difficult to identify the passband cutoff frequency in $H_0(z)$ and the

stopband cutoff frequency in $H_1(z)$ based on the design in Fig. (b).

In addition to generating starting points, we also need to define initial constraints in all the subproblems. To this end, we use a reference (possibly non-PR) filter bank $R(z)$ from an existing design. This reference filter bank serves as a guide and provides some performance metrics like transition bandwidth, energies, and ripples. Since the PR-LP filter banks designed by our procedure may not have the same quality as $R(z)$, our procedure should only aim to minimize the maximum violation of the constraints defined. In our experiments, we used one of Johnston's QMF filter bank as our reference filter bank.

In generating a starting point for the third subproblem, we need to transform the real coefficients $k_1, k_3, \ldots, k_{N-1}$ found in the second subproblem to PO2 forms. Given a real parameter and b, the maximum number of terms in PO2 form to represent the parameter, we apply Booth's algorithm [28] to represent a consecutive sequence of 1's using two terms in PO2 form and truncate the least significant bits of the parameter so that the number of terms in PO2 form is within a fixed limit.

As an example, consider a binary fixed-point number 0.10011101100. After applying Booth's algorithm and truncation, we can represent the number in 2 terms in PO2 form:

$$0.10011101100 \xrightarrow{\text{Booth's Algorithm}} 0.1010001\bar{0}1\bar{0}0 \xrightarrow{\textit{Truncation}} 2^{-1} + 2^{-3}.$$

4.2. *Updating x*

The value of x is updated in Line 7 of DLM-98 in Fig. 3. There are two ways to update x: greedy update and hill climbing. In greedy updates, the update of x leading to the maximum improvement of $L_d(x, \mu)$ in (26) is found before an update is made. This approach is very time consuming and may not lead to the best filter bank when DLM-98 stops. On the other hand, in hill climbing, x is updated as soon as an improvement in $L_d(x, \mu)$ is found. This approach is efficient and generally leads to good designs. For this reason, we use hill climbing as our update strategy.

Our update strategy examines all the coefficients ($k_1, k_3, k_5, \ldots, k_{N-1}, B_1$ and B_2) in a predefined round-robin order, first evaluating changes to k_1 and last evaluating changes to B_2, before returning to k_1 again.

For a given variable x at $\hat{x}$, we search its neighborhood for possible improvements in the Lagrangian value. Each of the four subproblems to be solved has a different definition of neighborhood, depending on whether its variable are continuous or discrete. In the first two subproblems with continuous variables, we define the neighborhood of a continuous variable x at $\hat{x}$ to be a finite number of discretized points around $\hat{x}$, with more sample points closer to $\hat{x}$ than further away while keeping all the other variables fixed. For instance, we define the neighborhood of $\hat{x} \neq 0$ to be the set $\{+\hat{x}p, -\hat{x}p\}$, where $p \in \{8, 7.5, 7, 6.5, 6\} \cup \{5.75, 5.5, 5.25, \ldots, 2.0\} \cup \{1.5, 1.498, 1.496, \ldots, 0.666\}$. ($\hat{x}$ is set heuristically to 10^{-7} if $\hat{x}$ were zero initially.) The reason for using a variable density is to limit the search complexity, while providing finer coverage of the space closer to $\hat{x}$.

It is important to point out that we are not discretizing the variables of a subproblem but rather the neighborhood points of a point. After discretizing the neighborhood points, we can apply DLM-98 to solve the subproblems. Experimentally, we found little difference in execution time and solution quality between using DLM-98 and algorithms that search in continuous space.

In the last two subproblems, we have $\frac{N}{2}$ discrete variables in PO2 form and two continuous variables. The neighborhood points of a specific point in continuous space are similar to those in the first two subproblems. For discrete variable k_i, let l be the maximum number of terms in PO2 form, and k_i be composed of l elements $b_{i,1}, b_{i,2}, \ldots, b_{i,l}$. We define the neighborhood points of k_i as a set $\{b'_{i,j}\}$, where $b'_{i,j}$ differs from $b_{i,j}$ by either the sign or the exponent or both, while maintaining $b'_{i,j}$ to be different in sign and exponent to another elements of k_i.

4.3. *Initializing and Updating μ*

The value of μ is initialized in Line 4 of DLM-98 in Fig. 3. To allow our experiments to be repeated and our results reproducible, we always initialize μ to be zero.

Line 8 of DLM-98 in Fig. 3 is related to the condition when μ should be updated. In traditional Lagrangian methods on continuous variables, μ is updated in every iteration. This approach does not work in DLM-98 because if μ were updated after each update of x, then the search behaves like random probing and restarts from a new starting point even before a local minimum is reached. For this reason, μ for violated constraints

should be updated less frequently, only when no further improvement in $L_d(x, \mu)$ can be made in Line 8 of DLM-98 for all its neighborhood points. This is the approach we have taken in solving satisfiability problems [24, 25]. However, we have found that more frequent updates of μ may lead to better designs here. In our implementation, we update μ after every six round robins. Since μ is updated before all the lattice parameters have been perturbed, the guidance provided by μ may not be accurate.

When updating μ before the search reaches a local minimum of $L_d(x, \mu)$, we set c in Line 8 of DLM-98 to be a normalized value as follows:

$$c = \frac{\theta_{\text{speed}}}{\max_{i=1}^{n} g_i}, \tag{25}$$

where θ_{speed} is a real constant used to control the speed of increasing μ. Experimentally, we have determined θ_{speed} to be 0.6818.

4.4. *Weighted Discrete First-Order Methods*

Lagrangian methods rely on ascents in the Lagrange-multiplier space and descents in the objective space in order to reach equilibrium. The convergence speed and solution quality, however, depend on the balance between objective $f(x)$ and constraints $h(x)$ and $g(x)$. Although changes in μ lead to different balance between ascents and descents, convergence can be improved by introducing a weight on the objective function. These considerations lead to a new Lagrangian function as follows.

$$L_d^w(x, \mu) = wf(x) + \sum_{i=1}^{k} [\mu_i \max(0, g_i(x))] \tag{26}$$

where $w > 0$ is a user-controlled weight on the objective. By applying DLM-98 on (26) using different w, we observe four possible behaviors of the search trajectory:

- The trajectory converges without oscillations.
- The trajectory gradually reduces in oscillations and eventually converges.
- The trajectory oscillates within some range but never converges.
- The magnitude of oscillations increases, and the trajectory eventually diverges.

Table 2. The design of multiplierless PR-LP filter banks with 24 taps by DLM-98 using static and adaptive weights. (Time is measured by the number of times each variable is examined in the search. The base filters are from Johnston's 24C and 24D QMF filter banks [2], and the reference filter bank is Johnston's 24E QMF filter bank. Each lattice parameter has a maximum of 4 terms in PO2 form.)

	Static weight		Adaptive weight	
Weight, w	Objective	Time	Objective	Time
0.01	0.836	44	0.836	44
0.1	0.836	44	0.836	44
1.0	0.833	44	0.838	64
10.0	0.835	932	0.844	654
100.0	0.839	586	0.837	249
1000.0	0.814	2692	0.819	175
2000.0	—	—	0.828	386
4000.0	—	—	0.846	398

Obviously, the first two cases are desirable, and the latter two are not. Moreover, we would like to reduce the amount of oscillations and improve convergence time.

The second and third columns of Table 2 show the objective-function values of the designs found and the corresponding convergence times with static weights. DLM-98 does not converge when the static weight w is large. Note that time is measured by number of round robins (the number of times a variable is examined in the search), an architecture-independent metric. On a Pentium PRO 200 MHz computer, it takes one minute of CPU time to perform 10 round robins.

These results demonstrate that the choice of w is critical in controlling both the convergence time and solution quality. There is, however, no effective method for choosing a fixed w except by trial and error.

In the rest of this subsection, we present a strategy to adapt w based on run-time search progress in order to obtain high-quality solutions and short convergence time. This approach is more general than our previous approach [24] that scales the Lagrange multipliers periodically in order to prevent them from growing to be very large when all constraint functions are positive. The Lagrange multiplier of a non-negative constraint may grow without limit because its value is always non-decreasing according to (20), and a Lagrangian space with large Lagrange multipliers is more rugged and more difficult to search. In our previous approach [24], the period between scaling and the scaling factor are application dependent and are

chosen in an ad hoc fashion. Our current approach adjusts the weight between the objective and the constraints, which is equivalent to scaling the Lagrange multipliers. It is more general because it adjusts the weight according to the convergence behavior of the search.

In general, changing w may speed up or delay convergence before a trajectory reaches an equilibrium point, and may bring the trajectory out of equilibrium after it reaches there. In this section, we design weight-adaptation algorithms to speed up convergence. Strategies to bring a trajectory out of equilibrium by modifying w will be studied in the future.

Figure 5 outlines the procedures for weight initialization and adaptation. Its basic idea is to first estimate the initial weight $w(0)$ (Line 1), measure the performance of the search trajectory $(x(t), \mu(t))$ periodically, and adapt $w(t)$ to improve convergence time or solution quality.

Let (x_i, μ_i) be the point of the ith iteration, and $v_{\max}(i)$ be its maximum violation over the m constraints:

$$v_{\max}(i) = \max\Big[\max_{1\le j\le m}\{|\max(0, g_j(x(t)))|\}, \max_{1\le j\le k}\{g_j(x(t), 0\}\Big] \tag{27}$$

To monitor the search progress, we divide time into non-overlapping major windows of size N_u iterations (Line 2), each of which is divided into minor windows of δ_t iterations (Line 3). We further record statistics like $v_{\max}(i)$ and $f_i(x)$ that will be used to calculate the performance in each minor/major window (Line 5).

At the beginning of a minor window (Line 7), we test whether the trajectory diverges or not (Line 8). Divergence happens when $v_{\max}(i)$ is larger than an extremely large value (say 10^{20}). If it happens, we reduce w, say $w \leftarrow \frac{w}{10}$, and restart the window markers by resetting j to 0.

At the beginning of a major window (Line 9), we compute some metrics to measure the progress of the search relative to that of previous major windows (Line 10). In general, application-specific metrics, such as the number of oscillations of the trajectory, can be used. In our current implementation, we compute the averages (or medians) of $v_{\max}(i)$ and objective $f_i(x)$ in the uth major window ($u = 1, 2, \ldots$) as follows:

$$\bar{v}_u = \frac{1}{N_u}\sum_{i=(u-1)N_u+1}^{uN_u} v_{\max}(i) \quad \text{or}$$
$$\bar{v}_u = \underset{\substack{(u-1)N_u+1\\ \le i\le uN_u}}{\text{median}}\{v_{\max}(i)\} \tag{28}$$

$$\bar{f}_u = \frac{1}{N_u}\sum_{i=(u-1)N_u+1}^{uN_u} f_i(x) \quad \text{or}$$
$$\bar{f}_u = \underset{\substack{(u-1)N_u+1\\ \le i\le uN_u}}{\text{median}}\{f_i(x)\} \tag{29}$$

Based on these measurements, we adjust w accordingly (Line 11). Note that when comparing values between two successive major windows $u-1$ and u, both must use the same w; otherwise, the comparison is not meaningful because the terrain may be totally different. Hence, after adapting w, we should wait at least two major windows before changing it again.

To understand how weights should be updated in step 10, we examine all the possible behaviors of the search trajectory in successive major windows. We have identified four possible cases.

First, the trajectory does not stay within a feasible region, but goes from one feasible region to another through an infeasible region. During this time, $v_{\max}(i)$ is zero when the trajectory is in the first feasible region, increased when it travels from the first feasible region to an infeasible region, and decreased when going from the infeasible region to the second feasible region. No oscillations will be observed because oscillations normally occur around an equilibrium point in one feasible region. In this case, w should not be changed.

procedure *weight_initialization*
1. set $w(0)$ (initial weight, set to 0.00001 in the experiments);
2. set N_u (major window for changing w, set to 30 in the experiments);
3. set δ_t (minor window for changing w, set to 5 in the experiments);
4. $j \leftarrow 0$ (number of iterations since last divergence)

procedure *dynamic_weight_adaptation*
5. record useful information for calculating performance;
6. $j \leftarrow j + 1$;
7. **if** ($j \mod \delta_t = 0$) **then**
8. if trajectory diverges **then** { reduce w; $j \leftarrow 0$ }
9. **if** ($j \mod N_u = 0$) **then** {
10. compute performance based on data collected;
11. change w when certain conditions are satisfied (see text) }

Figure 5. Procedures for weight initialization and adaptation in Fig. 3. (The initial values of parameters are indicated here unless specified otherwise in the text.)

Second, the trajectory oscillates around an equilibrium point of a feasible region. This can be detected when the number of oscillations in each major window is larger than a certain threshold, and the trajectory is not always in a feasible region, and the trend of the maximum violation does not decrease. To determine whether the oscillations will subside eventually, we compute $\bar{v}_u - \bar{v}_{u+1}$, the difference of the average values of maximum violation $v_{\max}(i)$ for two successive major windows u and $u+1$. If the difference is not reduced reasonably, then we assume that the trajectory has not converged and decrease w accordingly.

Third, the search trajectory moves very slowly within a feasible region. This happens when w is very small, and the constraints dominate the search process. As a result, the objective value is improving very slowly and may eventually converge to a poor value. This situation can be identified when the trajectory remains within a feasible region in two successive major windows and is improving in successive major windows, but the improvement of the objective is not fast enough and is below an upper bound. Obviously, we need to increase w in order to speed up the improvement of the objective. If the objective remains unchanged, then the trajectory has converged, and no further modification of w is necessary.

Finally, the trajectory does not oscillate when it starts within a feasible region, goes outside the region, and converges to a point on the boundary. Here, a large w makes it more difficult to satisfy the constraints, causing the trajectory to move slowly to the feasible region. In this case, an appropriate decrease of w will greatly shorten the convergence time.

Table 2 illustrates the improvements in convergence times using adaptive weights. For all the initial weights considered, the adaptive algorithm is able to find converged designs in a reasonable amount of time, although the solution quality is not always consistent.

5. Experimental Results

In this section, we present the designs of four PR-LP filter banks. To help readers understand the design process, we present more details in the first design and summarize the results of the others.

Design Case 1. The base pair of filters is Johnston's low-pass filter in 32C and high-pass filter in 32D [2] (after changing it to antisymmetric). The reference filter bank $R(z)$ is Johnston's 32E.

In the first subproblem, we generate its starting point based on the procedure in Section 4.1. We then compute the passband and stopband energies of $R(z)$ and use the values as constraints. We define the objective as the sum of the passband and stopband energies of $H_0(z)$ and $H_1(z)$, using $\pi/2$ as the passband and stopband cutoff frequencies.

Since there is no guarantee that we can find a design of $H_0(z)$ and $H_1(z)$ that have the same passband and stopband energies as $R(z)$, we pick the solution with the minimum of the maximum violation after a fixed number of round robins. This subproblem takes less than 2 h (3000 round robins) to solve on a Pentium Pro 200 MHz computer. The frequency responses of the PR pair found (Fig. 6(a)) are much better than those in Fig. 4(b).

The second subproblem used the result of the first subproblem as its starting point and took less than 5 h (6500 round robins) to complete. It uses fixed passband and stopband cutoff frequencies defined by the reference filter bank $R(z)$. As before, we pick the solution with the minimum of the maximum violation. Figure 6(b) shows that the ripples and energies are much smaller than those in Fig. 6(a).

In the third subproblem, we first change k_1, $k_3, \ldots, k_{N-1}$ into PO2 forms with a fixed maximum number (3 or 4) of terms in each k_i. The mixed-integer problem took less than 2 h of CPU time (3000 round robins) to complete. Figure 6(c) (resp. 6(e)) plots the frequency responses of the PR-LP design with only three (resp. four) terms in PO2 form in any k_i. The ripples are slightly better when more terms in PO2 form are allowed in each parameter.

The last subproblem solved uses the complete formulation defined in (7) in order to get a high-quality design in all respects. The progress in this search is much slower than the other three subproblems because the ripples and transition bandwidth are expensive to be evaluated numerically. However, the solution of this subproblem took a relatively short amount of time (less than 2 h for 1500 round robins) because its starting point supplied by solving the third subproblem is already very good. Figure 6(d) (resp. 6(f)) shows the frequency responses of the multiplierless PR-LP designs with three (resp. four) terms in PO2 form in each k_i. Table 3 lists all the lattice parameters k_i, B_1, and B_2, both in floating point as well as in PO2 form. Figure 7 compares the frequency responses of the designs found and those of the reference design.

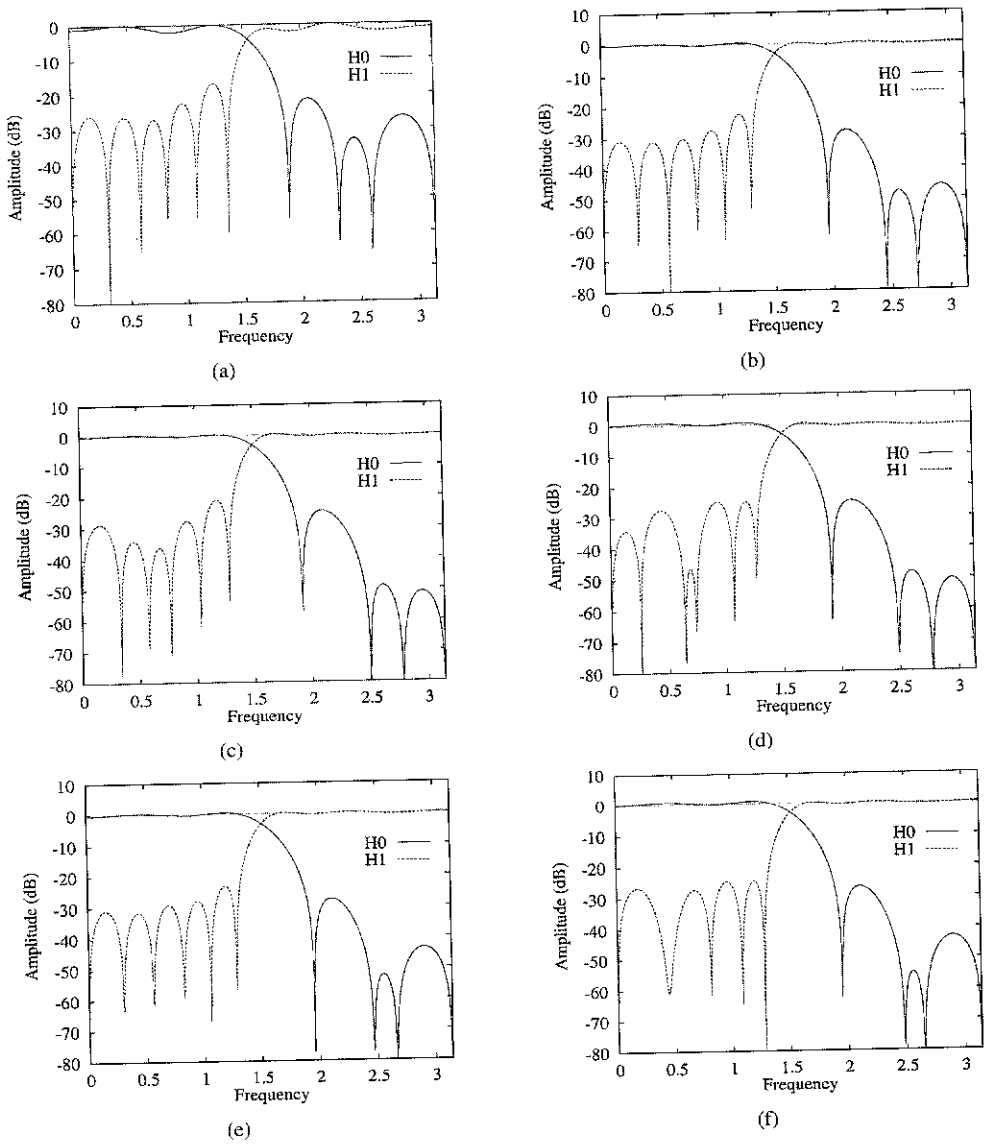


Figure 6. Design case 1: Frequency responses of the designs found by solving the four subproblems, using Johnston's 32C and 32D as the base pair and Johnston's 32E as the reference: (a) after solving subproblem 1; (b) after solving subproblem 2; (c) after solving subproblem 3 (3 terms in PO2); (d) after solving subproblem 4 (3 terms in PO2); (e) after solving subproblem 3 (4 terms in PO2) and (f) after solving subproblem 4 (4 terms in PO2).

Design Case 2. Using Johnston's 32C and 32D as our base pair and 32C as the reference, the filters all have proper shapes after solving the first subproblem (Fig. 6(a)). Figure 8(a) and (b) plot the frequency responses after solving the second and the third subproblems, respectively, using a maximum of 4 terms in PO2 form in each k_i. The performance of the final design is shown in Fig. 8(c) and compared with respect to the reference 32C in Fig. 8(d). Our design has similar transition bandwidth as the reference but has worse energies and ripples.

If we further limit the number of terms in PO2 form to be 3 in each k_i, the resulting filter bank performs slightly worse. Figure 8(e) and (f) show the frequency responses after solving subproblems 3 and 4, respectively.

Table 3. Lattice parameters (both in real and PO2 forms) and the 16 lattice parameters (B_1 = 0.0001951010603248444, B_2 = 0.0001061909416765278 9).

m	Lattice k_{2m+1}	PO2			$h_0(m)$	$h_1(m)$
0	−1.375	-2^1	2^{-1}	2^{-3}	0.000015242270281	0.000204085715984
1	42.0	2^5	2^3	2^1	−0.000020958121637	0.000280617859478
2	1.140625	2^0	2^{-3}	2^{-6}	−0.000216071379982	−0.003065275681529
3	0.09765625	2^{-3}	-2^{-5}	2^{-8}	−0.000273058025229	0.003419327251431
4	−1.375	-2^1	2^{-1}	2^{-3}	0.000441643704953	0.011091595601248
5	7.5625	2^3	-2^{-1}	2^{-4}	0.001847200986083	−0.024054212199478
6	0.78125	2^0	-2^{-2}	2^{-5}	0.003383148582433	−0.007544528934661
7	2.625	2^1	2^{-1}	2^{-3}	0.001610769396968	0.035890465697903
8	0.091796875	2^{-3}	-2^{-5}	-2^{-9}	−0.014350771362619	0.012097807744450
9	0.982421875	2^0	-2^{-6}	-2^{-9}	−0.028832464597731	−0.041720969306460
10	−127.25	-2^7	2^0	-2^{-2}	0.019550976540926	0.007821887696226
11	−0.935546875	-2^0	2^{-4}	2^{-9}	0.050236848223581	0.065778721213968
12	−0.75390625	-2^0	2^{-2}	-2^{-8}	−0.052502845341496	−0.032907418059305
13	−7.94140625	-2^3	2^{-4}	-2^{-8}	−0.095110664539566	−0.108146999185062
14	0.421875	2^{-1}	-2^{-4}	-2^{-6}	0.152566364065811	0.110672911111743
15	−0.921875	-2^0	2^{-4}	2^{-6}	0.473859657541803	0.472154047535480

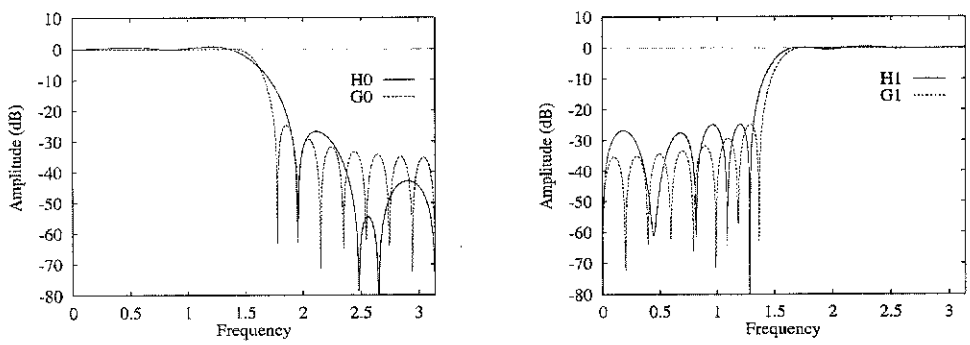


Figure 7. Comparison of the frequency responses of the multiplierless PR-LP filter bank designed with 32E, the reference LP filter bank [2]: (a) comparing $H_0(z)$ and $G_0(z)$ and (b) comparing $H_1(z)$ and $G_1(z)$. (Each lattice parameter is represented using 4 terms in PO2 form).

Design Case 3. Using Johnston's 32C and 32D as our base pair of filters and 32D as the reference, the frequency response after solving the first subproblem is the same as that of Design Case 1, since the two base filters are the same. Figure 9(a) and (b) plot the frequency responses after solving the second and third subproblems, respectively. Note that the number of terms in PO2 form in each each k_i is 4. Figure 9(c) shows the performance of the final design, and Fig. 9(d) compares the performance with that of the reference.

After limiting the number of terms in PO2 form in each lattice parameter to be 3, Fig. 9(e) and (f) plot the frequency responses of the filter banks designed. As before, the ripples and energies are worse than those of the reference.

Design Case 4. Using Johnston's 24C and 24D as the base pair of filters and 24C as the reference, Fig. 10(a) shows that reasonable performance is obtained after solving the first subproblem. Figure 10(b–d) plot the frequency responses after solving, respectively, the second, third, and fourth subproblems. The experiments are then repeated by using 3 terms in PO2 form in each k_i. Figure 10(e) and (f) plot the corresponding frequency responses.

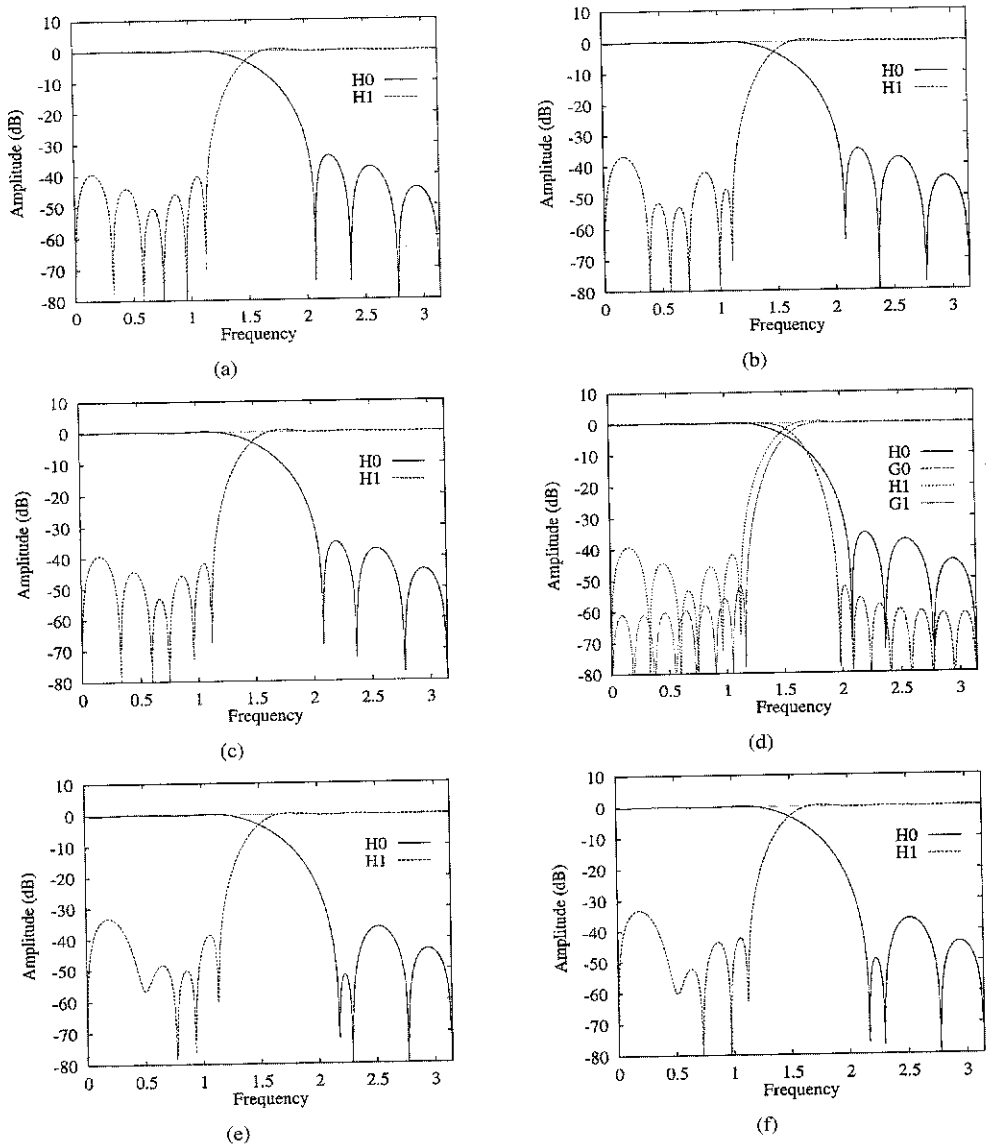


Figure 8. Design case 2: Frequency responses of the two-channel PR-LP filter banks after solving (a) subproblem 2, (b) subproblem 3, (c) subproblem 4, using 32C and 32D as the base pair, 32C as the reference, and 4 terms in PO2 form in each lattice parameter. (d) Comparison of the filter bank designed ($H_0(z)$ and $H_1(z)$) with respect to the reference filter bank ($G_0(z)$ and $G_1(z)$). Experiments are repeated using 3 terms in PO2 form in each lattice parameter and the results after solving subproblems 3 and 4 are shown in (e) and (f), respectively.

6. Conclusions

In this paper, we have presented a new discrete Lagrangian method (DLM-98) for designing multiplierless perfect-reconstruction (PR) linear-phase (LP) filter banks. Such designs have not been attempted before because the design problem is a highly nonlinear discrete optimization problem with many equality constraints imposed by the PR conditions. Such constraints are especially hard to satisfy when the variable space is discrete. Moreover, some of the performance metrics used in the objective and the constraints are not in closed forms and require expensive numerical methods to evaluate.

We have chosen a lattice structure in our PR-LP filter bank because the PR and LP conditions are automatically satisfied in the structure by suitable choices of the lattice parameters. This allows us to eliminate the

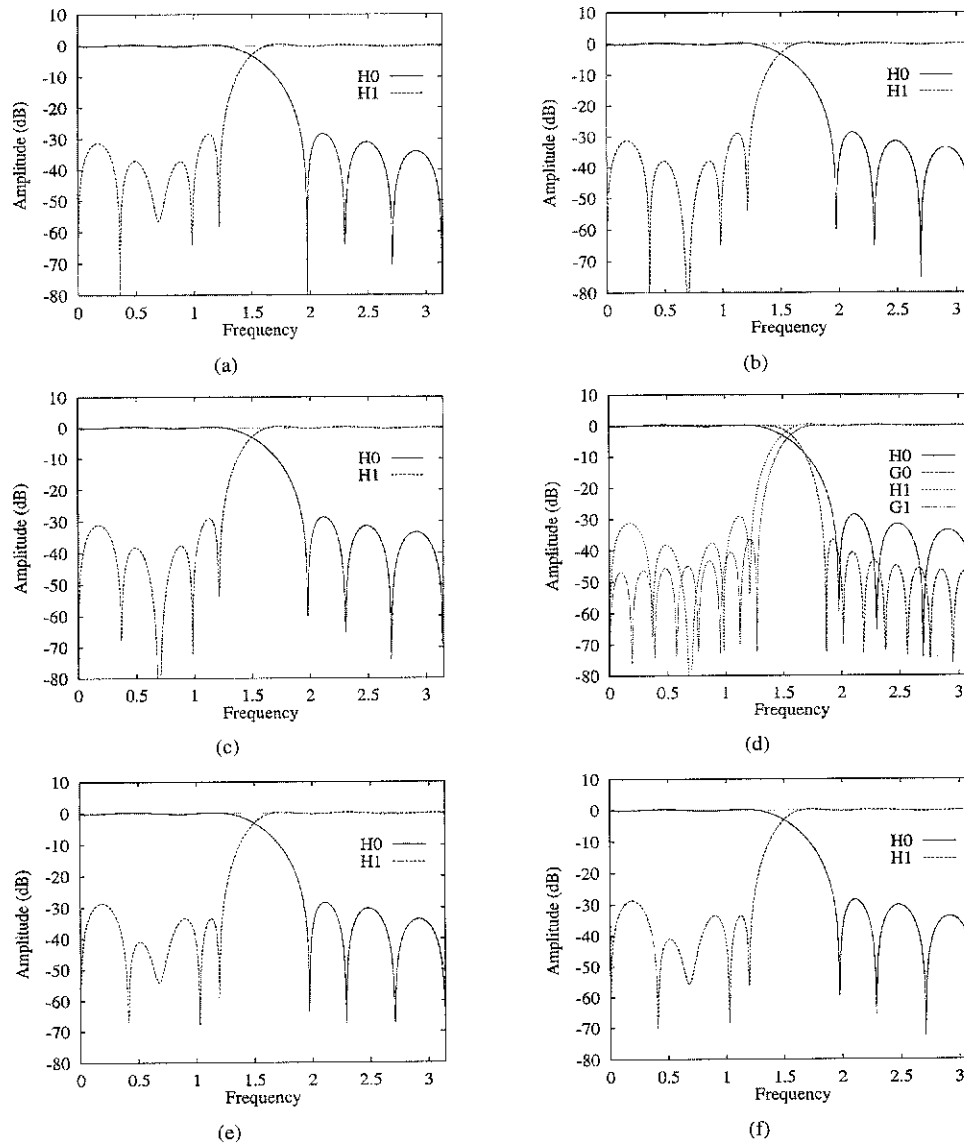


Figure 9. Design case 3: Frequency responses of designs under similar conditions as in Design case 2 (Fig. 8) except a different reference filter bank (32D): (a) after solving subproblem 2; (b) after solving subproblem 3 (4 terms in PO2); (c) after solving subproblem 4 (4 terms in PO2); (d) comparison with reference filter bank 32D; (e) after solving subproblem 3 (3 terms in PO2) and (f) after solving subproblem 4 (3 terms in PO2).

equality constraints imposed by the PR conditions in our formulation.

We have described a discrete Lagrangian method and the first-order necessary and sufficient conditions for convergence. The derivation of these conditions requires a new definition of gradients in discrete space and the observation that traditional calculus in continuous space does not work in discrete space. These conditions lead to the first-order discrete Lagrangian method (DLM-98) that we use in this paper.

We have studied and evaluated efficient weight-adaptation algorithms in DLM-98 and have illustrated through examples that such a balance is very sensitive to the relative weights between the objective and the

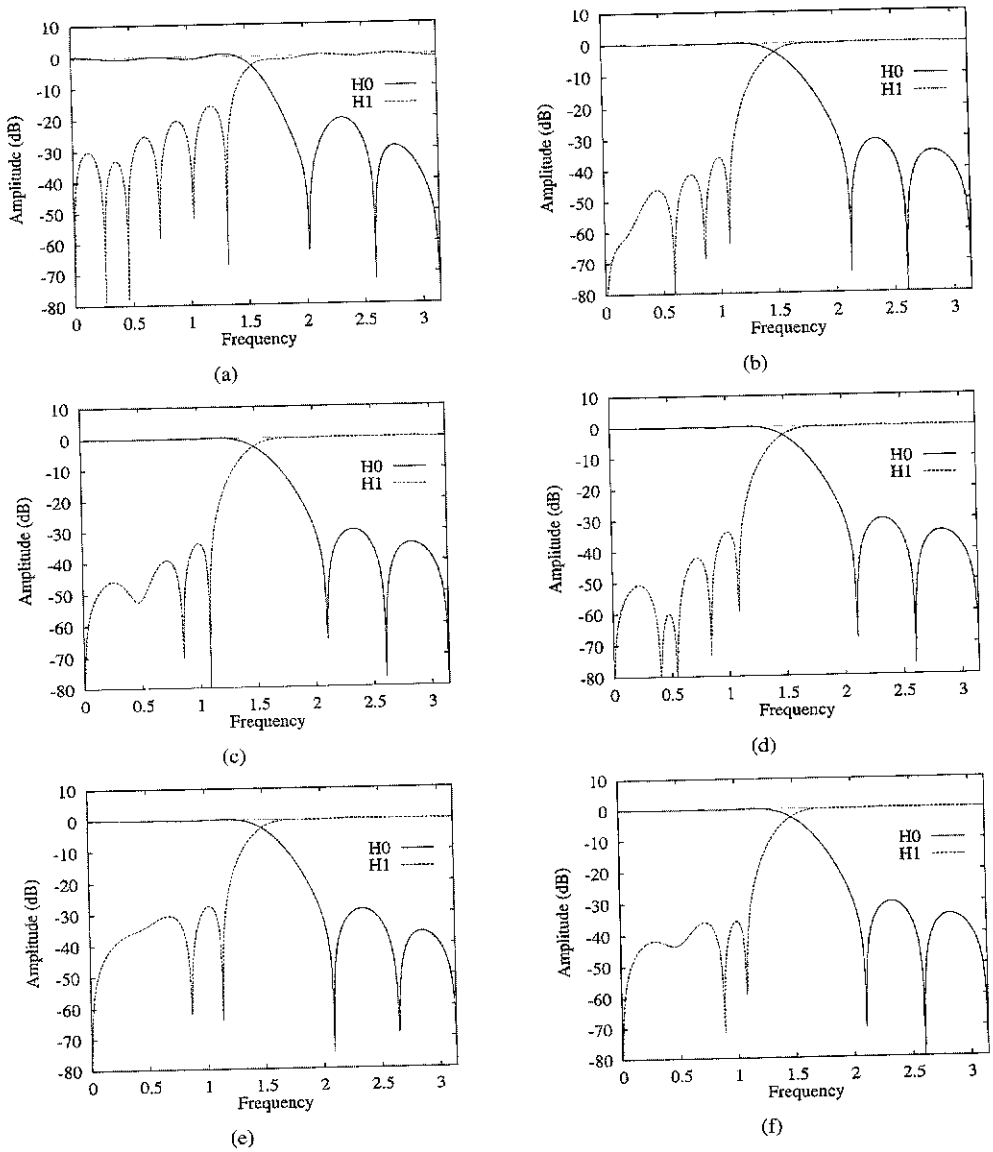


Figure 10. Design case 4: Frequency responses of the designs found by solving the four subproblems, using Johnston's 24C and 24D as the base pair and Johnston's 24D as the reference. (a) after solving subproblem 1; (b) after solving subproblem 2; (c) after solving subproblem 3 (4 terms in PO2); (d) after solving subproblem 4 (4 terms in PO2); (e) after solving subproblem 3 (3 terms in PO2) and (f) after solving subproblem 4 (3 terms in PO2).

constraint part in the Lagrangian function. Without a good balance, the search trajectory may not converge. To cope with this problem, we have proposed a dynamic weight-adaptation algorithm that adjusts the weight of the objective relative to the constraints based on statistics collected during the search. Our experience in designing multiplierless PR-LP filter banks shows that our adaptive method leads to fast convergence with similar solution quality.

We have applied DLM-98 to design four PR-LP filter banks, starting from an initial non-PR filter bank. Our designs were based on a non-PR filter bank with continuous coefficients as a reference. In each case, we have obtained feasible multiplierless PR-LP

lattice-structured designs that perform slightly worse than the non-PR reference. Such degradations are expected because we have imposed the PR and the multiplierless conditions on the reference design. Overall, our design method is effective because it finds designs with very few terms in PO2 form in each filter coefficient, while allowing a cost-effective design to be implemented.

Achnowledgments

Research supported by National Science Foundation Grant MIP 96-32316.

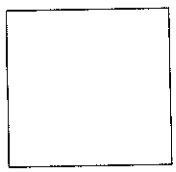
Notes

1. There are second-order conditions to guarantee that the extremum found is a local minimum [27].
2. To simplify our symbols, we represent points in the x space without the explicit vector notation.

References

1. T.E. Tuncer and T.Q. Nguyen, "General analysis of two-band QMF banks," *IEEE Trans. on Signal Processing*, Vol. 43, No. 2, pp. 544–548, Feb. 1995.
2. J.D. Johnston, "A filter family designed for use in quadrature mirror filter banks," *Proc. of Int'l Conf. on ASSP*, pp. 291–294, 1980.
3. B.R. Horng and A.N. Willson, Jr., "Lagrange multiplier approaches to the design of two-channel perfect-reconstruction linear-phase FIR filter banks," *IEEE Trans. on Signal Processing*, Vol. 40, No. 2, pp. 364–374, Feb. 1992.
4. R.D. Koilpillai and P.P. Vaidyanathan, "A spectral factorization approach to pesudo-QMF design," *IEEE Trans. on Signal Processing*, Vol. 41, No. 1, pp. 82–92, Jan. 1993.
5. P.P. Vaidyanathan, *Multirate Systems and Filter Banks*, Printice-Hall, 1993.
6. A.V. Oppenheim and R.W. Schafer, *Discrete-Time Signal Processing*, Prentice Hall, 1989.
7. J.G. Proakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, Maxwell Macmillan, 1992.
8. O. Alkin and H. Caglar, "Design of efficient m-band coders with linear-phase and perfect-reconstruction properties," *IEEE Trans. on Signal Processing*, Vol. 43, No. 7, pp. 1579–1590, July 1995.
9. Y.P. Lin and P.P. Vaidyanathan, "Linear phase cosine modulated maximally decimated filter banks with perfect reconstruction," *IEEE Trans. on Signal Processing*, Vol. 42, No. 11, pp. 2525–2539, Nov. 1995.
10. M. Vetterli and D. Le Gall, "Perfect reconstruction FIR filter banks: Some properties and factorizations," *IEEE Trans. on Acoutics, Speech, and Signal Processing*, Vol. 37, No. 7, pp. 1057–1071, July 1989.
11. K. Nayebi, T.P. Barnwell III, and M.J.T. Smith, "Nonuniform filter banks: A reconstruction and design theory," *IEEE Trans. on Signal Processing*, Vol. 41, No. 3, pp. 1114–1127, March 1993.
12. T.Q. Nguyen and P.P. Vaidyanathan, "Structures for m-channel perfect-reconstruction FIR QMF banks which yield linear-phase analysis filters," *IEEE Trans. on Acoutics, Speech, and Signal Processing*, Vol. 38, No. 3, pp. 433–446, March 1990.
13. T.Q. Nguyen and P.P. Vaidyanathan, "Two-channel perfect reconstruction FIR QMF structure which yield linear-phase analysis and synthesis filters," *IEEE Trans. on Acoutics, Speech, and Signal Processing*, Vol. 37, No. 5, pp. 676–690, May 1989.
14. H. Samueli, "An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients," *IEEE Transactions on Circuits and Systems*, Vol. 36, No. 7, pp. 1044–1047, 1989.
15. R.A. Caruana and B.J. Coffey, "Searching for optimal FIR multiplierless digital filters with simulated annealing," Technical report, Philips Laboratories, 1988.
16. J.D. Schaffer and L.J. Eshelman, "Designing multiplierless digital filters using genetic algorithms," *Proc. Int'l Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 439–444, 1993.
17. D. Kodek and K. Steiglitz, "Comparison of optimal and local search methods for designing finite wordlength FIR digital filters," *IEEE Trans. Circuits Syst.*, Vol. 28, No. 1, pp. 28–32, 1981.
18. J.-J. Shyu and Y.-C. Lin, "A new approach to the design of discrete coefficient FIR digital filters," *IEEE Transactions on Signal Processing*, Vol. 41, No. 1, p. 1, 1995.
19. B.W. Wah, Y. Shang, and Z. Wu, "Discrete Lagrangian method for optimizing the design of multiplierless QMF filter banks," *Proc. Int'l Conf. on Application Specific Array Processors*, IEEE, pp. 529–538, July 1997.
20. B.W. Wah, Y. Shang, and Z. Wu, "Discrete Lagrangian method for optimizing the design of multiplierless QMF filter banks," *IEEE Transactions on Circuits and Systems, Part II*, (accepted to appear) 1999.
21. M.H. Er and C.K. Siew, "Design of FIR filters using quadrature programming approach," *IEEE Trans. on Circuits and Systems-II*, Vol. 42, No. 3, pp. 217–220, March 1995.
22. C.-K. Chen and J.-H. Lee, "Design of quadrature mirror filters with linear phase in the frequency domain," *IEEE Trans. on Circuits and Systems-II*, Vol. 39, No. 9, pp. 593–605, Sept. 1992.
23. T.Q. Nguyen, "Digital filter bank design quadratic-constrained formulation," *IEEE Trans. on Signal Processing*, Vol. 43, No. 9, pp. 2103–2108, Sept. 1995.
24. Y. Shang and B.W. Wah, "A discrete Lagrangian-based global-search method for solving satisfiability problems," *J. of Global Optimization*, Vol. 12, No. 1, pp. 61–99, Jan. 1998.
25. B.W. Wah and Y. Shang, "A discrete Lagrangian-based global-search method for solving satisfiability problems," in *Satisfiability Problem: Theory and Applications*, D.-Z. Du, J. Gu, and P. Pardalos (Eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, pp. 365–392, 1997.
26. Zhe Wu, *Discrete Lagrangian Methods for Solving Nonlinear Discrete Constrained Optimization Problems*, M.Sc. thesis, Dept. of Computer Science, Univ. of Illinois, Urbana, IL, May 1998.

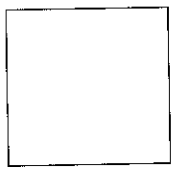
27. D.G. Luenberger. *Linear and Nonlinear Programming*, Addison-Wesley Publishing Company, 1984.
28. A.D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, Vol. 4, pp. 236–240, 1951.



Benjamin W. Wah received his Ph.D. degree in computer science from the University of California, Berkeley, CA, in 1979. He is currently the Robert T. Chien Professor of Engineering and a Professor in the Department of Electrical and Computer Engineering, the Coordinated Science Laboratory, and the Beckman Institute of the University of Illinois at Urbana-Champaign, Urbana, IL. During 1998–1999, he serves as a Professor of Computer Science and Engineering, Chinese University of Hong Kong. Previously, he had served on the faculty of Purdue University (1979–1985), as a Program Director at the National Science Foundation (1988–1989), as Fujitsu Visiting Chair Professor of Intelligence Engineering, University of Tokyo (1992), and McKay Visiting Professor of Electrical Engineering and Computer Science, University of California, Berkeley (1994). In 1989, he was awarded a University Scholar of the University of Illinois, and in 1998, he received the IEEE Computer Society Technical Achievement Award.

Wah's current research interests are in the areas of nonlinear search and optimization, knowledge engineering, multimedia signal processing, and parallel and distributed processing.

Wah was the Editor-in-Chief of the *IEEE Transactions on Knowledge and Data Engineering* between 1993-1996. He currently serves as the Honorary Editor-in-Chief of *Knowledge and Information Systems*, and on the editorial boards of *Information Sciences*, *International Journal on Artificial Intelligence Tools*, and *Journal of VLSI Signal Processing*. He had chaired a number of international conferences and is currently serving as the International Program Committee Chair of the IFIP World Congress in 2000. He had served in the IEEE Computer Society in various capacities and is currently the elected First Vice President for Publications. He is a Fellow of the IEEE and the Society for Design and Process Science.
b-wah@uiuc.edu, URL: http://www.manip.crhc.uiuc.edu



Zhe Wu was born in Anhui, China, on January 4, 1974. He received B.Sc. degree (June 1996) in Department of Special Class for Gifted Young from the University of Science and Technology of China and M.S. degree (May 1998) in Department of Computer Science from the University of Illinois at Urbana-Champaign. He is currently pursuing his Ph.D. degree in the University of Illinois at Urbana-Champaign.

His main research interests are in optimization, signal processing, software engineering and computer networks.
zhewu@www.manip.crhc.uiuc.edu