Discrete Lagrangian Methods for Optimizing the Design of Multiplierless QMF Banks*

Benjamin W. Wah, Yi Shang, and Zhe Wu

Abstract

In this paper, we present a new discrete Lagrangian method for designing multiplierless QMF (quadrature mirror filter) banks. The filter coefficients in these filter banks are in powers-of-two (PO2), where numbers are represented as sums or differences of powers of two (also called Canonical Signed Digit–CSD–representation), and multiplications are carried out as additions, subtractions and shifts. We formulate the design problem as a nonlinear discrete constrained optimization problem, using reconstruction error as the objective, and stopband and passband energies, stopband and passband ripples and transition bandwidth as constraints. Using the performance of the best existing designs as constraints, we search for designs that improve over the best existing designs with respect to all the performance metrics. We propose a new discrete Lagrangian method for finding good designs, and study methods to improve the convergence speed of Lagrangian methods without affecting their solution quality. This is done by adjusting dynamically the relative weights between the objective and the Lagrangian part. We show that our method can find designs that improve over Johnston's benchmark designs using a maximum of three to six ONE bits in each filter coefficient instead of using floating-point representations. Our approach is general and is applicable to the design of other types of multiplierless filter banks.

Keywords and Phrases. Adaptive weighted Lagrangian search, canonical signed digit representation, discrete Lagrangian formulation, global search, multiplierless filter banks, quadrature mirror filtering, simulated annealing.

*Research supported by National Science Foundation Grant MIP 96-32316 and by a gift from Rockwell International.

B. W. Wah is with the Department of Electrical and Computer Engineering and the Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA, e-mail: b-wah@uiuc.edu, URL: http://manip.crhc.uiuc.edu. Between 1998-1999, he is on leave at the Department of Computer Science and Engineering, Chinese University of Hong Kong, Shatin, Hong Kong. Y. Shang is with Department of Computer Engineering and Science, University of Missouri, Columbia, MI 65211, USA, e-mail: yshang@cecs.missouri.edu. Z. Wu is with the Department of Computer Science and the Coordinated Science Laboratory, University of Illinois, Urbana, IL 61801, USA, e-mail: zhewu@manip.crhc.uiuc.edu;

A preliminary version of this paper was presented in the IEEE International Conference on Application-Specific Array Processors, 1997 [28].

Accepted to appear in *IEEE Trans. on Circuits and Systems, Part II, 1999.*

1 Introduction

Digital filter banks have been applied in many engineering fields. Figure 1 summarizes the various design objectives for measuring quality. In general, filter-bank design problems are multi-objective, continuous, nonlinear 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 [24] 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 [10, 8]. On the other hand, filter bank-design problems have been solved by non-optimization-based algorithms, which include spectral factorization [12, 25] and heuristic methods (as in IIR-filter design). These methods generally do not continue to find better designs once a suboptimal design has been found [25].

In this paper, we study discrete Lagrangian and global-search methods for designing multiplierless QMF banks. These filter banks are an important class of filter banks that have been studied extensively. In a two-band QMF bank, the reconstructed signal is:

$$\hat{X}(z) \;=\; \frac{1}{2}\left[H_0(z)F_0(z) \;+\; H_1(z)F_1(z)\right]\; X(z) + \frac{1}{2}\left[H_0(-z)F_0(z) \;+\; H_1(-z)F_1(z)\right]\; X(-z) \qquad (1)$$

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. To perfectly reconstruct the original signal based on $\hat{X}$, we have to eliminate aliasing, amplitude, and phase distortions. QMF banks with FIR filters implement perfect reconstruction by setting $F_0(z) = H_1(-z)$, $F_1(z) = -H_0(-z)$ and $H_1(z) = H_0(-z)$, leading to a filter bank with one prototype filter $H_0(z)$, linear phase, and no aliasing distortions.

Filter	Minimization Objectives
Overall Filter Bank	Amplitude distortion
	Aliasing distortion
	Phase distortion
Single Filter	Stopband ripple (δ_s)
	Passpand ripple (δ_p)
	Stopband energy (E_s)
	Passband energy (E_p)
	Transition bandwidth (T_t)

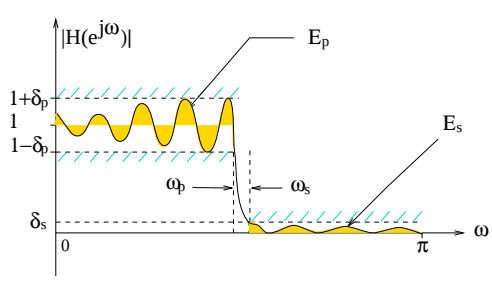


Figure 1: Possible design objectives of filter banks and an illustration of the design objectives of a single low-pass filter. ($[0, \omega_p]$ is the pass band; $[\omega_s, \pi]$, the stop band; $[\omega_p, \omega_s]$, the transition band.)

Traditional FIR filters in QMF banks use real numbers or fixed-point numbers as filter coefficients. Multiplications of such long floating point numbers generally limit the speed of FIR filtering. To overcome this limitation, *multiplierless* (*powers-of-two* or *PO2*) filters have been proposed. These filters use filter coefficients that have only a few bits that are ones. When multiplying a filter input (multiplicand) with one such coefficient (multiplier), the product can be found by adding and shifting the multiplicand a number of times corresponding to the number of ONE bits in the multiplier. For example, the multiplication of y by 0100001001 can be written as the sum of three terms, $y \cdot 2^8 + y \cdot 2^3 + y \cdot 2^0$, each of which can be obtained by shifting y. A limited sequence of shifts and adds are usually much faster than full multiplications. Without using full multiplications, each filter tap takes less area to implement in VLSI, and more filter taps can be accommodated in a given area to implement filter banks of higher performance.

The frequency response of a PO2 filter, $H(z)$, is

$$H(z) = \sum_{i=0}^{\gamma-1} x_i z^{-i} = \sum_{i=0}^{\gamma-1} \left(\sum_{j=0}^{d-1} e_{i,j} 2^j \right) z^{-i} \quad \text{where} \sum_{j=0}^{d-1} |e_{i,j}| \leq l \text{ for all } i, \quad e_{i,j} = -1, 0, 1. \tag{2}$$

Here, γ is the length of the PO2 filter, l is the maximum number of ONE bits used in each coefficient, and d is the number of bits in each coefficient.

The design of multiplierless filters has been solved by integer programming that optimizes filter coefficients with restricted values of powers-of-two. Other techniques used include combinatorial search [17], simulated annealing [2], genetic algorithms [18], linear programming [11], and continuous Lagrange-multiplier methods in combination with a tree search [20].

In this paper, we present a discrete Lagrange-multiplier search for designing multiplierless QMF banks. We formulate in Section 2 the design problem as a single-objective constrained optimization problem. Section 3 summarizes the principles behind discrete Lagrangian methods. In Section 4, we present our DLM-98 (Discrete Lagrangian Method, 1998 version) that finds saddle points in discrete space and examines the issues related to the implementation of DLM-98 to design multiplierless filter banks. Finally, Section 5 presents experimental results, and conclusions are drawn in Section 6.

2 Problem Formulation

The design of QMF banks can be formulated as a multi-objective unconstrained optimization problem or as a single-objective constrained optimization problem.

2.1 Multi-Objective Unconstrained Formulation

In a multi-objective formulation, the goals can be to:

- Minimize the amplitude distortion (reconstruction error) of the overall filter bank, or
- Optimize the individual performance measures of the prototype filter $H_0(z)$.

One possible formulation using a subset of the measures in Figure 1 is as follows:[1]

$$\text{Minimize} \quad E_r \quad \text{and} \quad E_s \tag{3}$$

$$\text{where} \quad E_r = \int_{\omega=0}^{\frac{\pi}{2}} (|H_0(e^{j\omega})|^2 + |H_0(e^{j(\omega-\pi)})|^2 - 1)^2 \, d\omega \quad \text{and} \quad E_s = \int_{\omega=\omega_s}^{\pi} |H_0(e^{j\omega})|^2 d\omega$$

Unfortunately, optimal solutions to (3) are not necessarily optimal solutions to the original problem that considers all the performance measures. Oftentimes, performance measures not included in the formulation are compromised.

In general, optimal solutions of a multi-objective problem form a *Pareto optimal frontier* such that one solution on this frontier is not dominated by another. One approach to find a point on the frontier is to optimize a weighted sum of all the objectives [10, 6, 24, 3, 16]. This approach has difficulty when frontier points of certain characteristics are desired, such as those with certain transition bandwidth. Different combinations of weights must be tested by trial and error until a desired solution is found. When the desired characteristics are difficult to satisfy, trial and error is not effective in finding feasible designs. Instead, constrained formulations should be used.

2.2 Single-Objective Constrained Formulation

Another approach to solve 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 measures constrained may be application- and filter-dependent [24].

Constraint-based methods have been applied to design QMF banks in both the frequency [10, 3, 5, 12, 21, 23] and time domains [15, 22]. In the frequency domain, the most often considered objectives are E_r (reconstruction error) and δ_s (stopband ripple). As stopband ripples cannot be formulated in closed form, stopband attenuation is used instead (represented as E_s in Figure 1). In the time domain, Nayebi [15] gave a time-domain formulation with constraints in the frequency domain and designed filter banks using an iterative time-domain design algorithm.

[1] Note that in QMF banks, E_r is non-zero. A multi-rate filter bank that enforces perfect reconstruction ($E_r = 0$) can be formulated as a constrained optimization problem with a goal of minimizing E_s [9, 8].

In this paper, we formulate the design of QMF banks in the most general form as a nonlinear constrained optimization problem using the reconstruction error as the objective and other measures (stopband ripple, stopband energy, passband ripple, passband energy and transition bandwidth) as constraints:

$$\begin{aligned} &\text{Minimize} \quad E_r \\ &\text{subject to} \quad E_p \leq \theta_{E_p}, \quad E_s \leq \theta_{E_s}, \quad T_t \leq \theta_{T_t}, \quad \delta_p \leq \theta_{\delta_p}, \quad \delta_s \leq \theta_{\delta_s} \end{aligned} \tag{4}$$

where θ_{E_p}, θ_{E_s}, θ_{δ_p}, θ_{δ_s} and θ_{T_t} are constraint bounds found in the best-known design (with possibly some bounds relaxed or tightened in order to obtain designs of different trade-offs). The goal here is to find filter banks of a finite word length whose performance measures are better than or equal to those of the reference design. Since the objective and the constraints are nonlinear, the problem is multi-modal with many local minima.

The original optimization problem with inequality constraints (4) can be transformed into an optimization problem with equality constraints as follows:

$$\text{Minimize} \quad f(x) = V_{E_r} = \frac{E_r - \theta_{E_r}}{\theta_{E_r}} \tag{5}$$

$$\begin{aligned} \text{subject to} \quad V_{E_p} &= \max\left(\frac{E_p - \theta_{E_p}}{\theta_{E_p}}, 0\right) = 0 \qquad V_{E_s} = \max\left(\frac{E_s - \theta_{E_s}}{\theta_{E_s}}, 0\right) = 0 \\ V_{\delta_p} &= \max\left(\frac{\delta_p - \theta_{\delta_p}}{\theta_{\delta_p}}, 0\right) = 0 \qquad V_{\delta_s} = \max\left(\frac{\delta_s - \theta_{\delta_s}}{\theta_{\delta_s}}, 0\right) = 0 \\ V_{T_t} &= \max\left(\frac{T_t - \theta_{T_t}}{\theta_{T_t}}, 0\right) = 0 \end{aligned} \tag{6}$$

where x is a vector of discrete coefficients, θ_{E_r} is the reconstruction error of the best-known design, and all functions have been normalized with respect to the values of the best-known design.

3 Lagrangian Formulations and Methods

In this section we first summarize past work on Lagrangian formulations and methods for solving continuous constrained optimization problems. We then extend them to discrete constrained optimization problems [19, 29, 26].

3.1 Continuous Lagrangian Formulations and Methods

Lagrangian methods are classical methods for solving continuous constrained optimization problems [14]. We first review briefly the theory of Lagrange multipliers.

Define a continuous constrained optimization problem as follows:

$$\begin{aligned} \min_{x \in E^m} \quad & f(x) \\ \text{subject to} \quad & g(x) \leq 0 \qquad x = (x_1, x_2, \ldots, x_n) \\ & h(x) = 0 \end{aligned} \tag{7}$$

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.

Since Lagrangian methods cannot deal directly with inequality constraints, we transform inequality constraint $g_i(x) \leq 0$ into an equality constraint by adding a slack variable $z_i(x)$, transforming it into $p_i(x) = g_i(x) + z_i^2(x) = 0$. The transformation is done in such a way that guarantees the existence of gradients in the x space when $g_i(x) = 0$. The corresponding *Lagrangian function* is defined as follows:

$$L_c(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T p(x) \tag{8}$$

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

To eliminate z_i from (8), we differentiate L_c with respect to z_i for a given x and set the result to zero. After solving the equality and substituting the result into (8) [14], we have:

$$L_c(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \frac{1}{2} \sum_{i=1}^{k} \left[\max^2(0, \mu_i + g_i(x)) - \mu_i^2 \right] \tag{9}$$

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.

According to classical optimization theory [14], all the extrema of (9) that satisfy the constraints and that are regular points are roots of the following set of first-order necessary conditions:

$$\nabla_x L_c(x, \lambda, \mu) = 0, \qquad \nabla_\lambda L_c(x, \lambda, \mu) = 0, \qquad \nabla_\mu L_c(x, \lambda, \mu) = 0. \tag{10}$$

These conditions are necessary to guarantee the (local) optimality of the solution to (7).[2]

There are many ways to find solutions that satisfy (10), including sequential quadratic programming and first-order search methods. The *first-order search method* expresses the search in a

[2] There are second-order conditions to guarantee that the extremum found is a local minimum [14].

dynamic system of ordinary differential equations:

$$\frac{d}{dt}x(t) = -\bigtriangledown_x L_c(x(t), \lambda(t), \mu(t)), \qquad \frac{d}{dt}\lambda(t) = \bigtriangledown_\lambda L_c(x(t), \lambda(t), \mu(t)), \tag{11}$$

$$\frac{d}{dt}\mu(t) = \bigtriangledown_\mu L_c(x(t), \lambda(t), \mu(t)).$$

They perform *local search* involving simultaneous descents in the original-variable space of x and ascents in the Lagrange-multiplier space of λ and μ. They evolve over time t, and reach a feasible local extremum when they stop at an *equilibrium point* where all gradients are zeroes.

3.2 Discrete Lagrangian Formulations and Methods

For discrete optimization problems, all the variables x_i ($i = 1, 2, \cdots, n$) take discrete values (e.g., integers). Little work has been done in applying Lagrangian methods to solve discrete constrained combinatorial optimization problems [7]. The difficulty in traditional Lagrangian methods lies in the lack of a differentiable continuous space to find an equilibrium point. In this subsection, we describe the theory of Lagrange-multiplier methods in discrete space [29, 19].

For nonlinear discrete problems with inequality constraints (similar to that in (7) where x is now a vector of discrete variables), we first transform inequality constraint $g_i(x) \leq 0$ into an equality constraint $\max(g_i(x), 0) = 0$. This transformation does not use a slack variable as in the continuous case because searches in discrete Lagrangian space do not require the existence of gradients in the x space when $g_i(x) = 0$.

After transforming inequality constraints into equality constraints, the resulting discrete Lagrangian function is written as follows:

$$L_d(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \sum_{i=1}^{k} \mu_i \max(0, g_i(x)), \tag{12}$$

where x is a discrete variable and λ and μ can be continuous.

The discrete Lagrangian function defined in (12) cannot be used to derive similar conditions in (10) 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 decreases. To this end, we define in discrete space a *direction of maximum potential drop* for Lagrangian function $L_d(x, \lambda, \mu)$ at point x for fixed λ and μ as a

vector[3] that points from x to a neighborhood point of $x \in \mathcal{N}(x)$ with the minimum L_d value:

$$\Delta_x L_d(x, \lambda, \mu) = \vec{\nu}_x = y \ominus x = (y_1 - x_1, y_2 - x_2, \ldots, y_n - x_n) \tag{13}$$
$$\text{where } y \in \mathcal{N}(x) \cup \{x\} \text{ and } L_d(y, \lambda, \mu) = \min_{\substack{x' \in \mathcal{N}(x) \\ \cup \{x\}}} L_d(x', \lambda, \mu).$$

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, $\vec{\nu}_x$ 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, when x itself has the minimum L_d, then $\vec{\nu}_x = \vec{0}$.

Having defined $\Delta_x L_d(x, \lambda, \mu)$, the direction of maximum potential drop in the x space, we define the concept of *saddle points* in discrete space similar to those in continuous space [14]. A point (x^*, λ^*, μ^*) is a saddle point when:

$$L(x^*, \lambda, \mu) \leq L(x^*, \lambda^*, \mu^*) \leq L(x, \lambda^*, \mu^*), \tag{14}$$

for all (x^*, λ, μ) and all (x, λ^*, μ^*) sufficiently close to (x^*, λ^*, μ^*). Starting from (14), we can prove similar first-order necessary conditions in discrete space that are satisfied by all saddle points [29].

$$\Delta_x L_d(x, \lambda, \mu) = 0, \qquad \nabla_\lambda L_d(x, \lambda, \mu) = 0, \qquad \nabla_\mu L_d(x, \lambda, \mu) = 0. \tag{15}$$

Note that the notation in the first condition defines the direction of maximum potential drop 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. For brevity, the proofs showing the correctness of these conditions are omitted here [29].

The first-order necessary conditions in (15) lead to the following first-order search method in discrete space. Here, we seek discrete equilibrium points similar to those of continuous problems. The following equations are discrete approximations to implement the first-order conditions in (15).

General Discrete First-Order Search Method

$$x(k+1) = x(k) \oplus \Delta_x L_d(x(k), \lambda(k)) \tag{16}$$
$$\lambda(k+1) = \lambda(k) + c_1 h(x(k)) \tag{17}$$
$$\mu(k+1) = \mu(k) + c_2 \max(0, g(x(k))) \tag{18}$$

[3] We assume that points in the x space are represented as vectors without explicitly denoting them using the vector notation, whereas λ and μ are scalars.

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** search has not converged **and** number of iterations $< i_{max}$ **do** {
7. update x to x' only if this will result in $L_d(x', \mu) < L_d(x, \mu)$;
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 2: 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.)

where $\oplus$ is the vector-addition operator ($x \oplus y = (x_1 + y_1, x_2 + y_2, \ldots x_n + y_n)$), 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 $h(x) = 0$, 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 does 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

Based on (5) and (6), the discrete Lagrangian function for optimizing PO2 filter banks is:

$$L_d(x, \lambda, \mu) \;=\; f(x) + \sum_{\substack{i \in \{E_p, E_s, \\ \delta_p, \delta_s, T_t\}}} \mu_i \cdot V_i \tag{19}$$

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 2 shows an implementation of the discrete first-order method (16) and (18) 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.

- *Starting points* (Line 3). We choose a starting point based on a discrete approximation of an existing QMF bank with real coefficients (Section 4.1).

- *Initial Lagrange-multiplier values* (Line 4). We initialize all Lagrange multipliers to zero in order to allow our results to be reproduced easily. An optimal initial setting is difficult because it depends on the amount of constraint violation.
- *Time constraint* (Lines 2 and 6). This limits the number of iterations through the loop.
- *Updating x* (Line 7). Here, we evaluate all possible neighboring points of x in order to find improvements in its Lagrangian value (Section 4.2).
- *Updating μ* (Lines 1 and 8). The Lagrange multipliers are updated when the search reaches a local minimum in the objective space. We do not update the multipliers more frequently due to instability of the trajectory. The amount of update is controlled by an application-dependent constant c and other filter-related parameters (Section 4.3).
- *Dynamic weight adaptation* (Lines 5 and 9). Weight adaptation adjusts the weight between the objective and the constraints in order to adjust their relative importance and to improve convergence (Section 4.4).

4.1 Generating a Starting Point

There are two alternatives to select a starting point (Line 3 in Figure 2): using the parameters of an existing PO2 QMF bank, or using a discrete approximation of an existing QMF bank with real coefficients. The first alternative is not always possible because not many such filter banks are available in the literature. In this section, we discuss the second alternative.

In the second approach, we first transform the real coefficients of the best-known design to PO2 forms using a CSD representation. Given a real coefficient and b, the maximum number of ONE bits to represent the coefficient, we apply Booth's algorithm [1] to represent consecutive 1's using two ONE bits and then truncate the least significant bits of the coefficients. This approach generally allows a number to be represented in a few ONE bits. 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 ONE bits:

$$0.10011101100 \xRightarrow[\text{Booth's Algorithm}]{} 0.101000\bar{1}0\bar{1}00 \xRightarrow[\text{Truncation}]{} 2^{-1} + 2^{-3}.$$

Previous work [13, 17, 4] shows that scaling has a significant impact on the optimization of coefficients in PO2 filters. That is, if each coefficient is scaled properly before the search starts (based on a heuristic objective), the quality of the final design can be improved significantly. In

Table 1: Comparison of a PO2 filter bank obtained by truncating the real coefficients of Johnston's 32e QMF bank [10] to 3 bits and a similar PO2 filter bank whose coefficients were scaled by 0.5565 before truncation. (Performance has been normalized with respect to the performance of the original filter bank.)

Performance Metrics	E_r	E_p	E_s	δ_p	δ_s	T_t
Filter bank A with Truncated Coefficients	6.93	9.61	1.09	1.89	1.05	1.00
Filter bank B with Scaling and Truncation	0.99	1.08	0.96	1.20	0.98	0.99

procedure *find_scaling_factor*
1. *LeastSum* $= +\infty$;
2. **for** *ScaleFactor* := 0.5000 **to** 1.0 **step** 0.0001 **do** {
3. Multiply each filter coefficient by *ScaleFactor*;
4. Get the PO2 form of the scaled coefficients;
5. Compute the weighted sum of constraint violation: $sum := \sum_{i=1}^{5} w_i \cdot g_i$;
6. **if** ($sum < LeastSum$) **then** { $LeastSum := sum$; $BestScale := ScaleFactor$ }}
7. **return** *BestScale*

Figure 3: Algorithm for finding the best scaling factor, where w_i is the weight of constraint i.

our case, the performance of a PO2 filter obtained by truncating its real coefficients to a fixed maximum number of ONE bits is not as good as one whose real coefficients were first multiplied by a scaling factor. We illustrate this observation in the following example.

Consider Johnston's 32e filter bank [10] as a starting point. Table 1 shows the metrics of two PO2 filters: Filter Bank A was obtained by truncating each of the original coefficients to a maximum of 3 ONE bits, whereas Filter Bank B was obtained by multiplying each of the coefficients by 0.5565 before truncation. Filter Bank B performs better and is almost as good as the original design with real coefficients. In fact, a design that is better than Johnston's 32e design can be obtained by using Filter B as a starting point, but no better designs were found using Filter A. This example illustrates that multiplying the filter coefficients by a scaling factor changes the bit patterns of the coefficients, which can improve the quality of the starting point when the coefficients are truncated.

Experiments also show that it is possible to find good designs without requiring the PO2 coefficients to have the same degree of precision as that of continuous coefficients. For instance, in our experiments, we restrict the minimum exponent of the ONE bits in each coefficient (in the range $[-1, 1]$) to be -22, even though the real coefficients have a minimum exponent of -31.

To find the best scaling factor, we enumerate over different scaling constants and scale all the coefficients by a common constant before the search begins. Figure 3 shows a simple but effective algorithm to find the proper scaling factor to be multiplied before the coefficients are truncated. We evaluate the quality of the resulting starting point by a weighted sum of its performance metrics. Since under most circumstances, the constraint on transition bandwidth is more difficult to satisfy,

we give it a weight of 100 and a weight of 1 for the other four metrics. Note that our objective in finding a good scaling factor is different from that in the previous work [13, 17, 4]. Further, note that the filter output in the final design will need to be divided by the same scaling factor.

Experimental results show that the algorithm in Figure 3 works fast and can complete in a few minutes, and that the scaling factors chosen are reasonable and suitable. It is important to point out that scaling does not help when the number of ONE bits allowed to represent each coefficient is large. For instance, when the maximum number of ONE bits allowed is larger than 6, the performance of all the filters is nearly the same for all scaling factors.

As an illustration, consider the design of a PO2 QMF bank [28] based on Johnston's 32d design [10] as our constraints. Assuming a minimum exponent of -22 in each ONE bit, we enumerate and find the best scaling factor for all the coefficients to be 0.9474.

4.2 Updating x

The value of x is updated in Line 7 in Figure 2. There are two ways in which x can be updated: greedy update and hill climbing. In greedy updates, the update of x leading to the maximum improvement of $L_d(x, \mu)$ 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.

We process all the bits of all the coefficients in a round-robin manner. Suppose γ is the filter length, l is maximum number of ONE bits that can be used for each coefficient, and the ith coefficient is composed of l elements $b_{i,1}, b_{i,2}, \ldots, b_{i,l}$. We process the elements in the following order repetitively:

$$b_{1,1}, b_{1,2}, \ldots, b_{1,l}, b_{2,1}, \ldots, b_{\gamma,1}, \ldots, b_{\gamma,l}.$$

For each element $b_{i,j}$, we perturb it to be $b'_{i,j}$ that differs from $b_{i,j}$ by either the sign or the exponent or both, while maintaining $b'_{i,j}$ to be not the same in exponent as another element of the ith coefficient. Using $b_{i,1}, \cdots, b_{i,j-1}, b'_{i,j}, \cdots, b_{i,l}$ while keeping other coefficients the same, we compute the new value $L_d(x', \mu)$ and accept the change if $L_d(x', \mu) < L_d(x, \mu)$.

4.3 Updating μ

Lines 1 and 8 in Figure 2 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 7 of DLM-98 for all the bits in all the coefficients. This is the approach we have taken in solving satisfiability problems [19, 29, 26]. However, we have found that more frequent updates of μ may lead to better PO2 filters. In our implementation, we update μ every time three coefficients have been processed. Since μ is updated before all the filter coefficients have been perturbed, the guidance provided by μ may not be exact.

When updating μ before the search reaches a local minimum of $L_d(x, \mu)$, we set c in Line 8 of Figure 2 to be a normalized value as follows:

$$c = \frac{\theta_{speed}}{\max_{i=1}^{n} g_i} \tag{20}$$

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

When the search reaches a local minimum of $L_d(x, \mu)$, perturbing any single bit in any coefficient will result in no improvement of $L_d(x, \mu)$. At this point, we need to update μ differently in order to bring the search out of the local minimum. This is done by choosing a proper value of c in Line 8 of DLM-98. If μ is increased too fast, then the search will restart from a random starting point. On the other hand, if μ is increased too slowly, then the trajectory will remain in the current local minimum, and updates of x in the next iteration of DLM-98 will bring the search to the same local minimum! Hence, we like to set c so that it will bring the search out of the current local minimum in one step, and local descents in the next iteration will head to an adjacent local minimum. This means that, after μ has been changed to μ', there exists x' in $\mathcal{N}(x)$ such that

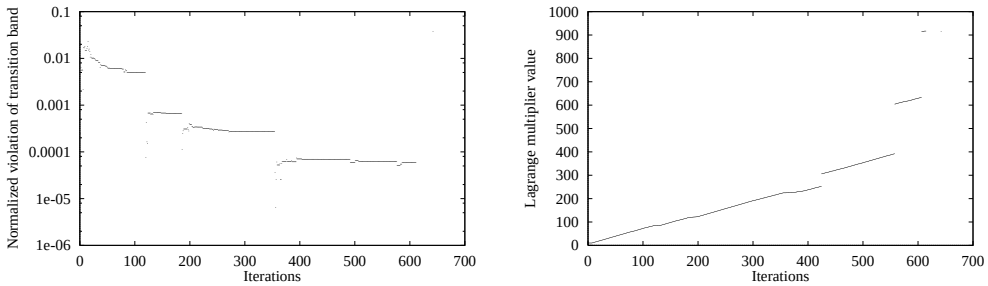
$$L_d(x, \mu) \leq L_d(x', \mu) \quad \text{and} \quad L_d(x', \mu') < L_d(x, \mu') \tag{21}$$

Replacing $L_d(x, \mu)$ by $f(x) + \sum_{i=1}^{n} \mu \cdot max(0, g_i(x))$ in (21), we get the condition before μ changes:

$$f(x) + \sum_{i=1}^{n} \mu \cdot max(0, g_i(x)) \leq f(x') + \sum_{i=1}^{n} \mu \cdot max(0, g_i(x')) \tag{22}$$

and that after μ is updated to $\mu'_i = \mu_i + c \cdot max(0, g_i(x))$:

$$f(x) + \sum_{i=1}^{n} \mu'_i \cdot max(0, g_i(x)) > f(x') + \sum_{i=1}^{n} \mu'_i \cdot max(0, g_i(x')) \tag{23}$$



(a) Violation values of T_t

(b) Corresponding μ_{T_t}

Figure 4: Performance progress measured during the search of Problem 32e.

where $max(0, g_i(x'))$ is the new violation of the ith constraint at x'. After transformations, we get

$$c > \frac{L_d(x', \mu) - L_d(x, \mu)}{\sum_{i=1}^{n} max(0, g_i(x)) \cdot (max(0, g_i(x)) - max(0, g_i(x')))} \tag{24}$$

When c is large enough to satisfy (24) for all x', and μ is increased according to Line 8 of DLM-98, we are assured that there is new x' that will cause L_d to decrease in the next iteration.

As an example, consider in Figure 4a the violation of transition bandwidth T_t in a typical search based on the constraints derived from Johnston's 32e filter bank [10]. Figure 4a shows that the value of the violation on T_t can be extremely small, on the order of 10^{-5} in the later part of the search. For such small violation values, the update of μ_{T_t} using c defined in (20) will result in a large number of iterations before the violation can be overcome. Using c defined in (24) to increase μ_{T_t}, we see in Figure 4b that μ_{T_t} jumps three times when the condition for updating μ was satisfied. These saved at least half of the total search time in order to find the solution.

4.4 Weighted Discrete First-Order Methods

As discussed in Section 3.1, 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, depends 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.

$$L_d^w(x, \mu) = w\ f(x) + \sum_{i=1}^{k} [\mu_i\ \max(0, g_i(x))] \tag{25}$$

Table 2: Multiplierless 32d QMF banks found by DLM-98 with static and adaptive weights. (The objective is the reconstruction error E_r.)

Weight	Static Weight		Adaptive Weight	
w	Objective	Time (minutes)	Objective	Time (minutes)
10000.0	-	-	0.998	195.9
1000.0	-	-	0.998	168.3
100.0	-	-	0.885	277.2
10.00	-	-	0.883	119.3
1.00	-	-	0.836	190.8
0.1	0.87	197.1	0.837	161.4
0.01	0.87	115.2	0.87	124.3
0.001	0.87	4.8	0.87	34.5
0.0001	0.88	12.0	0.878	10.8
0.00001	0.9	23.7	0.924	12.0

where $w > 0$ is a user-controlled weight on the objective. By applying DLM-98 in Figure 2 on (25) 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.

Obviously, the first two cases are desirable, and 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 of DLM-98 with static weights. DLM-98 does not converge when the static weight w is large. 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 [19] 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 (18), and a Lagrangian space with large Lagrange multipliers is more rugged and more difficult to search. In our previous approach [19], the period between

```
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 ← 0 (number of iterations since last divergence)

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

Figure 5: Procedures for weight initialization and adaptation in Figure 2. (The initial values of parameters are indicated here unless specified otherwise in the text.)

scaling and the scaling factor are application dependent and 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 all the m constraints:

$$v_{max}(i) = \max\left[\max_{1\le j\le m}\{|max(0, g_j(x(t)))|\}, \max_{1\le j\le k}\{g_j(x(t), 0\}\right] \tag{26}$$

To monitor the progress of the search, we divide time into non-overlapping major windows of size N_u iterations (Line 2), each of which is then divided into minor windows of δ_t iterations (Line 3). We further record some statistics, such as $v_{max}(i)$ 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 ($u = 1, 2, \cdots$) as follows:

$$\bar{v}_u = \frac{1}{N_u} \sum_{i=(u-1)N_u+1}^{uN_u} v_{max}(i) \quad \text{or} \quad \bar{v}_u = \underset{\substack{(u-1)N_u+1 \\ \leq i \leq uN_u}}{\text{median}} \{v_{max}(i)\} \tag{27}$$

$$\bar{f}_u = \frac{1}{N_u} \sum_{i=(u-1)N_u+1}^{uN_u} f_i(x) \quad \text{or} \quad \bar{f}_u = \underset{\substack{(u-1)N_u+1 \\ \leq i \leq uN_u}}{\text{median}} \{f_i(x)\} \tag{28}$$

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 is not changed.

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, 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

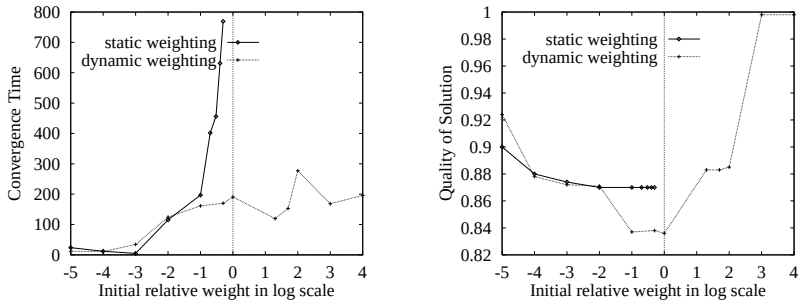
We have applied DLM-98 to solve the QMF-bank design problems formulated by Johnston [10]. In this section, we compare the performance of designs found by DLM-98 and those by Johnston [10], Chen *et al.* [4], *Novel* [27], simulated annealing (SA), and genetic algorithms (GA). All the experiments were run on Pentium Pro 200 computers with Linux unless specified otherwise.[4]

Our goal is to find designs that are better than the baseline results across all six performance measures. Hence, we use (5) with the constraint bounds defined by those of the baseline designs.

5.1 Performance of Lagrangian Methods with Dynamic Weights

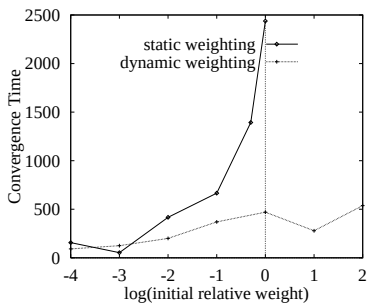
To design multiplierless QMF banks, we allow the maximum number of ONE bits to be 6 and the minimum exponent to be -22 for each filter coefficient. The Lagrangian method uses both static weights and dynamic weights to solve 32d and 48e problems. We compare both the convergence time and the quality of solution in terms of reconstruction error. The starting points were obtained from Johnston's design, and the control parameters were the same as those used in the previous subsection except that the window size N_u is 10.

[4] All the filter-bank coefficients are placed at *ftp://manip.crhc.uiuc.edu/pub/papers/PostScript/J66/J66.coefficients.*

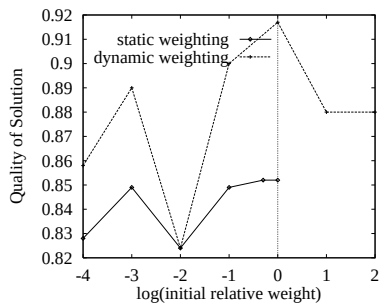


(a) Problem 32d with static weights

(b) Problem 32d with dynamic weights



(c) Problem 48e with static weights



(d) Problem 48e with dynamic weights

Figure 6: Comparison of convergence time and quality of solution between static weighting and dynamic weighting for multiplierless QMF-bank design problems 32d and 48e.

A comparison of DLM-98 with static weights and that with dynamic weights is shown in Figure 6. Even though the initial weights have very large ranges, $[10^{-5}, 10^{4}]$ for 32d and $[10^{-4}, 10^{2}]$ for 48e, the dynamic weight-adaptation algorithm converges in less than 300 minutes for the 32d problem and 510 minutes for 48e. However, using DLM-98 with static weights, when the initial w is larger than 1.0, the search cannot converge within 15 hours for 32d and 32 hours for 48e.

Note that, for 48e, the solution quality of DLM-98 with static weights is slightly better than our dynamic weight-adaptation algorithm for some initial weights. This happens because the latter may change the terrain during the search and find different solutions.

Finally, Table 3 shows the results of solving all the Johnston's benchmarks using filter coefficients with a maximum of six ONE bits. Our results show that we were able to find designs that have better reconstruction errors, while the other performance metrics are either the same or better.

5.2 Comparison of DLM-98 with Johnston's Designs

In this section, we compare the performance of designs found by DLM-98 and those by Johnston [10].

There are two parameters in a PO2 filter bank design: the maximum number of ONE bits in each filter coefficient and the number of filter taps. In our experiments, we have varied one while

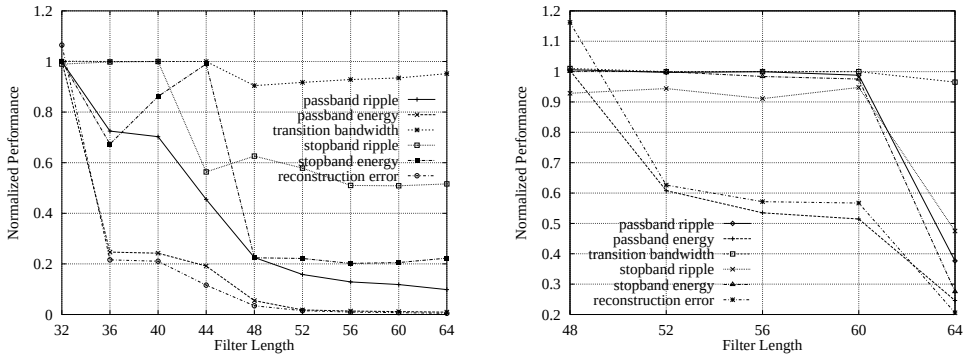
Table 3: Experimental results of DLM-98 in solving multiplierless QMF-bank design problems. The initial points of the run were from six ONE-BIT expressions of scaled Johnston's solutions.

Filter	E_r	δ_p	E_p	δ_s	E_s	T_r	Scaling Factor	Time (hrs)
16a	0.99	0.99	0.94	0.99	0.95	0.99	0.9747	1.6
16b	0.99	0.99	0.90	0.99	0.98	0.99	0.8524	2.1
16c	0.96	0.99	0.98	0.99	0.99	0.99	0.5967	3.0
24b	0.97	0.99	0.87	0.96	0.99	0.99	0.9661	5.6
24c	0.89	0.99	0.58	0.99	0.99	0.99	0.6413	12.0
24d	0.81	0.99	0.83	0.99	0.99	0.99	0.5342	13.1
32c	0.96	0.99	0.75	0.99	0.99	0.99	0.5706	12.0
32d	0.83	0.95	0.61	0.99	0.99	0.99	0.6971	3.1
32e	0.72	0.99	0.90	0.99	0.99	0.99	0.5019	7.2
48c	0.88	0.95	0.85	0.99	0.99	0.99	0.7914	18.0
48d	0.95	0.99	0.75	0.99	0.99	0.99	0.7138	23.0
48e	0.91	0.99	0.80	0.99	0.99	0.99	0.5793	8.0
64d	0.87	0.99	0.83	0.76	0.99	0.99	0.9955	9.5
64e	0.85	0.99	0.73	0.99	0.99	0.99	0.8026	24.1

keeping the other fixed when evaluating a PO2 design with respect to a benchmark design.

We have used closed-form integration to compute the performance values. In contrast, Johnston [10] used sampling to compute energies. Hence, designs found by Johnston are not necessarily at the local minima in a continuous sense. To demonstrate this, we applied local search in a continuous formulation of the 24D design, starting from Johnston's design. We found a design with a reconstruction error of 3.83E-05, which is better than Johnston's result of 4.86E-05. By applying global search, we can further improve the design to have a reconstruction error of 3.66E-05.

We have evaluated PO2 designs obtained by DLM-98 with respect to Johnston's designs whose coefficients are 32-bit real numbers. Using the performance of Johnston's 32e design as constraints [10], we ran DLM-98 from 10 different starting points obtained by randomly perturbing 1% of all the coefficients of Johnston's design [10]. Each run was limited so that each ONE bit of the coefficient was processed in a round robin fashion 400 times. We then picked the best solution of the 10 runs and plotted the result in Figure 7, which shows the normalized performance of PO2 designs with increasing number of filter taps, while each filter coefficient has a maximum of 3 ONE bits. (The best design is one with the minimum reconstruction error if all the constraints are satisfied; otherwise, the one with the minimum violation is picked.) Our results show a design with 32 taps that is nearly as good as Johnston 32e's design. For filters with 32, 36, 40 and 44 taps, we used a starting point derived from Johnston's 32e design with filter coefficients first scaled by 0.5565 and truncated to a maximum of 3 ONE bits, and the filter coefficients of the remaining taps



(a) Problem 32e (b) Problem 48e

Figure 7: Normalized performance for PO2 filter banks with a maximum of 3 ONE bits per coefficient and different number of filter taps.

set to zeroes initially. Starting points for filters with longer than 44 taps were generated similarly, except that a scaling factor of 0.5584 was used instead. Our results show that, as the filter length is increased, all the performance metrics improve, except the transition bandwidth, which remains close to that of the benchmark design.

With respect to Johnston's 48e design [10], we set a limit so that each ONE bit of the coefficient was processed in a round-robin fashion 800 times, and ran DLM-98 once from the truncated Johnston's 48e design. (The scaling factor was 0.5584 for filters with 48, 52, 56, and 60 taps. The scaling factor was 0.6486 for filters with 64 taps.) Our results show that our 48-tap PO2 design is slightly worse than that of Johnston's, while PO2 designs with 52 taps or longer have performance that are either the same or better than those of Johnston's 48e design. In particular, the reconstruction error of our 52-tap PO2 design is 62% of Johnston's 48e design, while that of our 64-tap PO2 design is only 21% of Johnston's 48e design.

In the next set of experiments, we kept the same number of taps as Johnston's 48e design and increased the maximum number of ONE bits in each coefficient from 3 to 6. We set a limit so that each ONE bit of the coefficient was processed in a round-robin fashion 800 times, and ran DLM-98 once from the truncated Johnston's 48e design. Figure 8 shows a design that is better than Johnston's 48e design when the maximum number of ONE bits per coefficient is 6. In this case, the reconstruction error is 91% of Johnston's 48e design. (The scaling factors used are 0.5584 for 3 bits, 0.8092 for 4 bits, 0.7409 for 5 bits, and 1.0 for 6 bits.)

With respect to Johnston's 64d and 64e designs, Table 4 shows improved PO2 designs obtained by DLM-98 using a maximum of 6 ONE bits per coefficient and 64 taps. No improvements were found when the maximum number of ONE bits is less than 6.

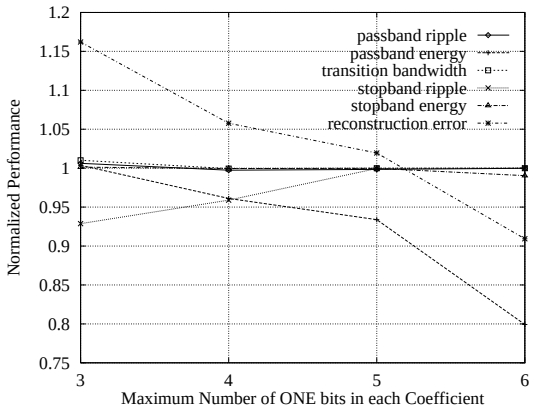


Figure 8: Normalized performance with respect to Johnston's 48e QMF bank [10] for PO2 filters with 48 taps and different maximum number of ONE bits per coefficient.

Table 4: Comparison of normalized performance of filter banks with discrete coefficients designed by DLM-98 with respect to those with continuous coefficients designed by Johnston, Chen, *Novel*, simulated annealing (SIMANN), and genetic algorithms (EA-Ct and EA-Wt). Columns 2-4 show the performance of DLM-98 using 3 ONE bits for 32-tap filters and 6 ONE bits for 64-tap filters normalized with respect to that of Johnston's 32e, 64d, and 64e filter banks [10]. Columns 5-6 show the performance of DLM-98 using 3 ONE bits normalized with respect to that of Chen *et al.*'s 64-tap and 80-tap filter banks [4]. Columns 7-10 show the performance of 32-tap filter banks designed using *Novel* [27], SA, and EA, normalized with respect to that of Johnston's 32e filter bank and using Johnston's design as constraints.

Type	Discrete Coefficients					Continuous Coefficients			
Method	DLM-98					*Novel*	SA	EA-Ct	EA-Wt
Problem	J-32e	J-64d	J-64e	C-64	C-80	J-32e	J-32e	J-32e	J-32e
E_r	0.83	0.90	0.89	0.91	0.95	0.712	0.500	0.724	0.507
E_p	1.00	0.82	0.83	0.80	0.96	0.896	0.582	0.905	0.590
E_s	1.00	1.00	1.00	1.00	0.86	1.000	1.000	1.000	0.999
δ_p	1.00	0.97	1.00	1.00	1.00	1.000	1.000	1.000	0.997
δ_s	0.99	0.75	1.00	1.00	1.00	1.000	1.000	1.000	0.999
T_t	1.00	1.00	1.00	1.00	1.00	1.000	1.013	1.000	1.013

5.3 Comparison of DLM-98 with Other Optimization Methods

In this section, we compare the performance of designs found by DLM-98 and those by Chen *et al.* [4], *Novel* [27], SA, and GA. Table 4 shows improved designs found by DLM-98 with respect to Chen *et al.*'s designs with, respectively, 64 and 80 taps, all using a maximum of 3 ONE bits per coefficient. In these designs, we used Chen *et al.*'s designs as starting points and ran DLM-98 once with a limit so that each ONE bit was processed in a round-robin fashion 1,000 times.

We also compare in Table 4 the performance of 32e PO2 filter banks obtained by DLM-98 with a maximum of 3 ONE bits per coefficient, and those obtained by *Novel*, simulated annealing (SA),

and evolutionary algorithms (EAs). *Novel* uses a continuous trace function to bring a search out of local minima rather than restarting the search from a new starting point when the search finds a feasible design. The SA we have used is SIMANN from netlib that works on a weighted-sum formulation. The EA is Sprave's Lice (Linear Cellular Evolution) that can be applied to both constrained and weighted-sum formulations. SIMANN and EA-Wt use weighted-sum formulations with weight 1 for the reconstruction error and weight 10 for the remaining metrics. EA-Ct works on the same constrained formulation defined in (4). All methods were run significantly long with over 10 hours on a SUN SS20 workstation in each run.

We have tried various parameter settings and report the best solutions in Table 4. *Novel* improves Johnston's designs consistently. SIMANN and EA-Wt have difficulty in improving over Johnston's design across all measures and have found designs with larger transition bandwidth. EA-Ct found a design that improves Johnston's across all measures, although it is not as good as the one found by *Novel*. Note that all these designs have continuous coefficients that will need either a complex carry-save adder or a 32-bit multiplier in each tap in their hardware implementations. In contrast, DLM-98 obtained a design that improves E_r, while the other metrics are either exactly the same or slightly better than those of Johnston's. Moreover, the design uses a maximum of five additions in each tap, leading to very cost-effective implementations.

Since existing optimization packages like SIMANN and EA works in continuous space, we have also constructed our own simulated annealing package call *discrete simulated annealing* (DSA) that works directly in discrete space. As SA cannot handle constraints directly, we create a single objective based on a weighted sum of the objective and the constraints using static weights:

$$F = w_0 V_{E_r} + w_1 V_{\delta_p} + w_2 V_{E_p} + w_3 V_{T_t} + w_4 V_{\delta_s} + w_5 V_{E_s} \tag{29}$$

DSA first defines an initial temperature T_0, and selects a starting point and scaling factor in the same way as that in Section 4.1. It then generates a new x' in discrete space and accepts the new point at the current temperature T according to the following probability:

$$\text{probability of accepting } x' = e^{-\frac{((L(x')-L(x)))^+}{T}} \quad \text{where } a^+ = \begin{cases} a & \text{if } a > 0 \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

Periodically, T is scaled down by $scale_T$ when the maximum violation does not decrease over a period of time (set to 10 round robins in our experiments). Finally, DSA reports the best solution when the search converges.

In our experiments using DSA, we found it very difficult to set T_0, $scale_T$, and the static weights in (29) that lead to better feasible PO2 designs. A set of improperly chosen parameters will lead

Table 5: Experimental results of DSA in designing multiplierless QMF-bank problem 24c, starting from a six ONE-BIT expression of scaled Johnston's solutions.

T_0	E_r	δ_p	E_p	δ_s	E_s	T_r	Search Time (Hours)
5.0	-	-	-	-	-	-	1.5
1.0	0.81	0.81	0.65	1.04	0.83	0.94	0.9
0.5	0.99	1.008	0.99	0.99	0.99	0.99	1.0
0.1	0.91	0.76	0.72	1.01	0.94	1.01	1.5
0.05	0.96	0.97	0.88	1.009	0.90	1.01	1.6
0.01	0.88	0.91	0.75	1.006	0.97	0.99	1.2
0.005	0.98	0.96	0.93	0.99	0.99	0.99	2.4
0.001	0.87	0.94	0.70	1.003	0.96	0.99	1.7

to violations of certain constraints. This phenomenon is obvious because the weights define the relative importance of the constraints.

Our experience on DSA is illustrated in the search of a better design of Johnston's 24c filter bank. After extensive experimentation, we initialized the weights to be $w_0 = 1.0$, $w_1 = w_2 = w_4 = 5.0$, $w3 = 15.0$, $w_5 = 25.0$, and $scale_T = 0.95$. We further set the scaling factor to be 0.6413, the same as that in DLM-98 for 24c. Table 5 lists the eight designs found by DSA. When the initial temperature was too high (≥ 5.0), DSA did not find any meaningful design, but found near feasible designs when the initial temperature is lower. When the initial temperature is 0.005, DSA found a feasible PO2 design with six ONE bits that is slightly better than Johnston's 24c. Note that we did not find any feasible design after trying many other combinations of parameters.

In short, we found it difficult to use global search strategies, like SA and GA, to design PO2 filter banks formulated as weighted sum of the objective and the constraints. Without dynamically changing the weights as in DLM-98, it is hard to choose a proper set of weights (except by trial and error) that will allow SA or GA to converge to feasible designs. The best that SA and GA can find are designs with trade-offs on different metrics. For this reason, the method studied in this paper represents a significant advance in solving discrete constrained optimization problems.

6 Conclusions

We have presented a new discrete Lagrangian method (DLM-98) for designing multiplierless powers-of-two (PO2) QMF banks. Our results show that DLM-98 can find better PO2 filter banks with very few ONE bits in each filter coefficient than other discrete and continuous optimization methods. Our design method is unique because it starts from a constrained formulation, with the objective of finding a design that improves over a benchmark design. In contrast, existing methods for designing

PO2 filter banks can only obtain designs with different trade-offs among the performance metrics and cannot guarantee that the final design is always better than the benchmark design with respect to all the performance metrics.

References

[1] A. D. Booth. A signed binary multiplication technique. *Quart. J. Mech. Appl. Math.*, 4:236–240, 1951.

[2] R. A. Caruana and B. J. Coffey. Searching for optimal FIR multiplierless digital filters with simulated annealing. *Technical report, Philips Laboratories*, 1988.

[3] 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*, 39(9):593–605, September 1992.

[4] C.-K. Chen and J.-H. Lee. Design of linear-phase quadrature mirror filters with powers-of-two coefficients. *IEEE. Trans. Circuits Syst.*, 41(7):445–456, 7 1994.

[5] C. D. Creusere and S. K. Mitra. A simple method for designing high-quality prototype filters for m-band pseudo QMF banks. *IEEE Trans. on Signal Processing*, 43(4):1005–1007, April 1995.

[6] M. H. Er and C. K. Siew. Design of FIR filters using quadrature programming approach. *IEEE Trans. on Circuits and Systems - II*, 42(3):217–220, March 1995.

[7] A. M. Geoffrion. Lagrangian relaxation and its uses in integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

[8] B. R. Horng and A. N. Willon, Jr. Lagrange multiplier approaches to the design of two-channel perfect-reconstruction linear-phase FIR filter banks. *IEEE Trans. on Signal Processing*, 40(2):364–374, February 1992.

[9] V. K. Jain and R. E. Crochiere. Quadrature mirror filter design in the time domain. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 32(2):353–361, April 1984.

[10] J. D. Johnston. A filter family designed for use in quadrature mirror filter banks. *IEEE Proc. of Int'l Conf. on ASSP*, pages 291–294, 1980.

[11] D. Kodek and K. Steiglitz. Comparison of optimal and local search methods for designing finite wordlength FIR digital filters. *IEEE Trans. Circuits Syst.*, 28:28–32, 1 1981.

[12] R. D. Koilpillai and P. P. Vaidyanathan. A spectral factorization aparoach to pesudo-QMF design. *IEEE Trans. on Signal Processing*, 41(1):82–92, January 1993.

[13] Y. C. Lim and S. R. Parker. FIR filter design over a discrete power-of-two coefficient space. *IEEE Trans. Acoust. Speech, Signal Processing*, ASSP-31:583–519, June 1983.

[14] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, 1984.

[15] K. Nayebi, T. P. Barnwell III, and M. J. T. Smith. Time-domain filter bank analysis: A new design theory. *IEEE Transactions on Signal Processing*, 40(6):1412–1429, June 1992.

[16] T. Q. Nguyen. Digital filter bank design quadratic-constrained formulation. *IEEE Trans. on Signal Processing*, 43(9):2103–2108, September 1995.

[17] H. Samueli. An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients. *IEEE Transactions on Circuits and Systems*, 36(7):1044–1047, 1989.

[18] J. D. Schaffer and L. J. Eshelman. Designing multiplierless digital filters using genetic algorithms. In *Proc. Int'l Conf. on Genetic Algorithms*, pages 439–444, San Mateo, CA, 1993. Morgan Kaufmann.

[19] Y. Shang and B. W. Wah. A discrete Lagrangian-based global-search method for solving satisfiability problems. *J. of Global Optimization*, 12(1):61–99, January 1998.

[20] J.-J. Shyu and Y.-C. Lin. A new approach to the design of discrete coefficient FIR digital filters. *IEEE Transactions on Signal Processing*, 41(1), 1 1995.

[21] M. J. T. Smith and T. P. Barnwell III. Exact reconstruction techniques for tree-structured subband coders. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 34(3):434–441, June 1986.

[22] Iraj Sodagar, Kambiz Naybei, and Thomas P. Barnwell III. Time-varying filter banks and wavelets. *IEEE Trans. on Signal Processing*, 42(11):2983–2996, November 1994.

[23] A. K. Soman, P. P. Vaidyanathan, and T. Q. Nguyen. Linear phase paraunitary filter banks: Theory, factorizations and designs. *IEEE Trans. on Signal Processing*, 41(12):3480–3496, December 1993.

[24] T. E. Tuncer and T. Q. Nguyen. General analysis of two-band QMF banks. *IEEE Trans. on Signal Processing*, 43(2):544–548, February 1995.

[25] P. P. Vaidyanathan. *Multirate Systems and Filter Banks.* Printice-Hall Inc., 1993.

[26] B. W. Wah and Y. Shang. A discrete Lagrangian-based global-search method for solving satisfiability problems. In Ding-Zhu Du, Jun Gu, and Panos Pardalos, editors, *Satisfiability Problem: Theory and Applications*, pages 365–392. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1997.

[27] B. W. Wah, Y. Shang, T. Wang, and T. Yu. Global optimization of QMF filter-bank design using NOVEL. In *Proc. Int'l Conf. on Acoustics, Speech and Signal Processing*, volume 3, pages 2081–2084. IEEE, April 1997.

[28] B. W. Wah, Y. Shang, and Z. Wu. Discrete Lagrangian method for optimizing the design of multiplierless QMF filter banks. In *Proc. Int'l Conf. on Application Specific Array Processors*, pages 529–538. IEEE, July 1997.

[29] Zhe Wu. *Discrete Lagrangian Methods for Solving Nonlinear Distrete Constrained Optimization Problems.* M.Sc. Thesis, Dept. of Computer Science, Univ. of Illinois, Urbana, IL, May 1998.