A New Global-Search Method for Designing Filter Banks [*]

Yi Shang[a] and Benjamin W. Wah[b]

[a]Dept. of Computer Engineering/Science, University of Missouri, Columbia, MO 65211, USA

[b] Center for Reliable and High Performance Computing, UIUC, Urbana, IL 61801, USA

ABSTRACT

In this paper, we present a new global-search method for designing QMF (quadrature-mirror-filter) filter banks. We formulate the design problem as a nonlinear constrained optimization problem, using the reconstruction error as the objective, and the other performance metrics as constraints. This formulation allows us to search for designs that improve over the best existing designs. Due to the nonlinear nature of the performance metrics, the design problem is a nonlinear constrained optimization problem with many local minima. We propose to solve this design problem use global-search methods based on Lagrangian formulations. After transforming the original constrained optimization problem into an unconstrained form using Lagrange multipliers, we apply a new global-search method to find good solutions. The method consists of a coarse-level global-search phase, a fine-level global-search phase, and a local search phase, and is suitable for parallel computation due to the minimal dependency between various key components. In our experiments, we show that our method finds better designs than existing global-search methods, including simulated annealing and genetic algorithms.

Keywords: Filter bank, global search, nonlinear constrained optimization, quadrature-mirror-filtering (QMF), parallel processing, signal processing.

1. INTRODUCTION

The design of digital filter banks is important because filter banks have been used in many applications, including modems, data transmission, digital audio broadcasting, speech and audio coding, and image and video coding.[26,22,5] In this paper, we study the design of quadrature-mirror-filtering (QMF) filter banks because they are amongst the simplest filter banks with defined benchmarks. Our design method is general enough to be applicable to the design of other types of filter banks, such as multirate filter banks.

Algorithms for designing filter banks can be classified into non-optimization-based and optimization-based. Non-optimization-based algorithms generally do not continue to find better designs once a suboptimal design has been found.[23] In contrast, in an optimization-based approach, a design problem is formulated as a multi-objective nonlinear optimization problem[21] whose form may be application- and filter-dependent. This multi-objective problem can then be solved by various existing methods, such as penalty methods[9] and genetic algorithms. An alternative way is to reformulate the multi-objective problem into a single-objective optimization problem, using one of the original objectives as the new objective, and the remaining objectives as constraints with respect to an existing design. The advantage of this single-objective formulation is that its solution allows designs that are better than an existing design to be found with respect to all the design objectives. For this reason, we have chosen to formulate the design problem as a single-objective constrained optimization problem.

Mostly, the single-objective formulation in filter bank design cannot be solved in closed form with nonlinear objective and constraints, but can be solved numerically by many existing methods, such as gradient-descent, Lagrange-multiplier, quasi-Newton, simulated-annealing, and genetics-based methods.[8,7] Among them, only the Lagrangian method can guarantee constraint satisfaction.

In this paper, we focus on two issues in finding *constrained local minima*. First, we study the convergence of Lagrangian methods in designing filter banks and develop methods to improve the convergence speed. Second, we

[*] Research supported in part by National Science Foundation Grant MIP 96-32316 and Research Board Grant of University of Missouri System.

Other author information:
Y. Shang: E-mail: yshang@cecs.missouri.edu; URL: http://www.cecs.missouri.edu/~yshang
B. B. Wah: E-mail: wah@manip.crhc.uiuc.edu; URL: http://www.manip.crhc.uiuc.edu

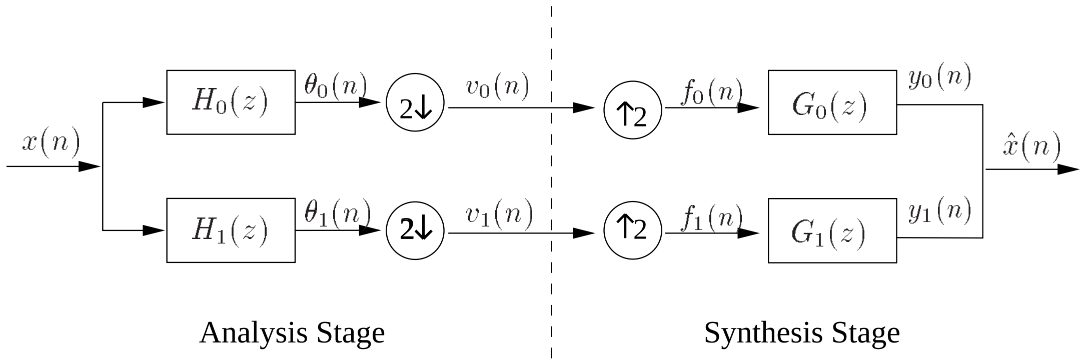

Figure 1. A two-channel filter bank.

apply global-search methods to find high-quality solutions. We propose a new global-search method that uses a predefined traveling trace to pull the search trajectory from one constrained local minimum to another, without having to restart the search. This avoids missing good local minima if the search were already in their vicinity. Furthermore, the global-search method can be implemented on parallel computers efficiently because of the minimal dependency between its various components and the parallel execution of the most time-consuming parts.

This paper is organized as follows. In Section 2, we introduce QMF filter banks, identify their performance metrics, and present our constraint-based formulation. Section 3 introduces Lagrangian methods, studies the issue on convergence speed, and presents our dynamic weight-adaption method. In Section 4, we present our new trace-based global-search method, and a prototype (*Novel*) that implements it. In Section 5, we apply *Novel* to design some QMF filter banks, and show improved designs over existing ones.

2. QMF FILTER BANKS AND DESIGN FORMULATION

Figure 1 shows a two-band filter bank that first decomposes an input signal $x(n)$ in time domain into two frequency subbands, and then combines them into a single output signal $\hat{x}(n)$. The filter bank consists of an analysis stage and a synthesis stage. In the analysis stage, the input signal is divided into two equal subbands using the analysis filters $H_0(z)$ and $H_1(z)$. According to Nyquist Theorem, each subband signal is then down-sampled by 2 to form the outputs of the analysis stage. These signals can then be analyzed or processed in various ways depending on the application. In the synthesis stage, each subband signal is up-sampled by 2 to form $f_0(n)$ and $f_1(n)$, processed by the synthesis filters $G_0(z)$ and $G_1(z)$, and summed at the output to form the reconstructed signal $\hat{x}(n)$.

The Z-transform of the reconstructed signal, which is a function of $x(n)$ and the filters, is[14,1]

$$\begin{aligned}\hat{X}(z) &= \frac{[G_0(z)H_0(z)+G_1(z)H_1(z)]X(z)+[G_0(z)H_0(-z)+G_1(z)H_1(-z)]X(-z)}{2} \\ &= T(z)X(z)+S(z)X(-z) \qquad (1)\end{aligned}$$

where $T(z)=T(e^{jw})=|T(e^{jw})|e^{j\phi(w)}$ and $S(z)=S(e^{jw})=|S(e^{jw})|e^{j\psi(w)}$.

There are three types of undesirable distortions of $x(n)$ in a filter bank. *Aliasing distortions* include aliasing caused by sub-sampling and images caused by up-sampling. The aliasing term $S(z)X(-z)$ in (1) represents the aliasing distortion. *Amplitude distortions* represent deviations of $|T(z)|$ in (1) from unity. *Phase distortions* represent deviations of $\phi(w)$ from the desired phase property, such as linear phase.

Extensive research has been conducted to remove undesirable distortions of filter banks. Aliasing distortions can be removed by selecting synthesis filters based on analysis filters. Infinite-impulse-response (IIR) all-pass filters can be used to eliminate magnitude distortions, whereas a linear-phase finite-impulse-response (FIR) filter removes phase distortions. Perfect reconstruction of the original signal by a filter bank requires $S(z)=0$ for all z, and $T(z)=cz^{-n_0}$, where c and n_0 are constants. Therefore, to perfectly reconstruct a signal, the transfer function is a pure delay with no aliasing, no amplitude change, and linear phase.

Table 1. Possible design objectives of filter banks. Refer to (1) for explanation.

	Design Objectives	Metrics
Overall Filter Bank	Minimize amplitude distortion E_r	$E_r = \int_0^\pi (\lvert T(\omega)\rvert^2 - 1)^2 d\omega$
	Minimize aliasing distortion ρ_a	$\rho_a = \int_0^\pi \lvert S(\omega)\rvert^2 d\omega$
	Minimize phase distortion ρ_p	$\rho_p = \int_0^\pi \lvert \phi(\omega) - \phi_0(\omega)\rvert d\omega$
Single Low-pass Filter	Minimize stopband ripple (δ_s)	$\delta_s = \max(\lvert H(\omega)\rvert,\ \omega \in [\omega_s, \pi])$
	Minimize passband ripple (δ_p)	$\delta_p = \max(\lvert H(\omega) - 1\rvert,\ \omega \in [0, \omega_p])$
	Minimize transition bandwidth (T_t)	$T_t = \omega_s - \omega_p$
	Minimize stopband energy (E_s)	$E_s = \int_{\omega_s}^\pi \lvert H(\omega)\rvert^2 d\omega$
	Maximize passband flatness (E_p)	$E_p = \int_0^{\omega_p} (\lvert H(\omega)\rvert - 1)^2 d\omega$
$[0, \omega_p]$ — pass band; $[\omega_s, \pi]$ — stop band; $[\omega_p, \omega_s]$ — transition band; $\phi_0(\omega)$ — desired linear phase.		

In QMF filter banks, the filters are chosen in the following manner.

$$G_0(w) = H_0(-w), \qquad G_1(w) = H_1(-w), \qquad H_1(w) = e^{jw} H_0(\pi - w). \tag{2}$$

Hence, $\hat{X}(w)$ in (1) becomes

$$\hat{X}(w) \quad = \quad \frac{[H_0(-w)H_0(w) + H_0(\pi - w)H_0(\pi + w)]X(w)}{2} = T(w)X(w)$$

Note that the aliasing term $S(w)$ is zero, independent of $H_0(w)$. Filter $H_0(w) = H(w)$ is called the *prototype filter* of the QMF filter bank. The design problem is now reduced to finding a filter with Fourier transform $H(w)$ such that $T(w)$ is a pure delay.

$H(w)$ has linear phase (so does $T(w)$) when the filter is a symmetric low-pass FIR filter. In order to have perfect reconstruction, the amplitude response needs to be a constant, such as 1.

$$|T(w)| = |H(w)|^2 + |H(w + \pi)|^2 = 1 \tag{3}$$

It can be shown that once the filters are chosen as in (2), it is not possible to obtain perfect reconstruction of the signal (*i.e.*, $|T(w)| \neq 1$) except for trivial, two-tap FIR filters. However, by numerically minimizing the reconstruction error, filter banks with longer FIR filters can be designed to achieve extremely high quality.[9]

Due to the particular way of selecting filter pairs, the design of a two-band QMF filter bank becomes the design of one symmetric low-pass prototype filter. This design problem is a multi-objective nonlinear optimization problem, whose objectives consist of performance metrics of both the overall filter bank and the single prototype low-pass filter. Table 1 summarizes the various design objectives.

The design objectives of filter banks have the following features:

- They are not unique and may be conflicting, leading to designs with different tradeoffs.
- Some objectives and their derivatives are not in closed forms and need to be evaluated numerically.
- The design objectives are nonlinear.

In general, the 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 Pareto frontier is to optimize a weighted sum of all the objectives.[9,4,21,2,13] This approach has difficulty when Pareto 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 filter bank is found. When the desired characteristics are difficult to satisfy, trial and error is not effective in finding feasible designs.

Another approach to solve a multi-objective problem is to turn all except one objectives into constraints and solve the problem as a constrained optimization problem. In this formulation, the constraints are defined with respect to a reference design. The specific measures constrained may be application- and filter-dependent.[21] Constraint-based methods have been applied to design QMF filter banks in both the frequency[9,2,3,10,17,19] and the time domains.[12,18] In the frequency domain, the most often considered objectives are the reconstruction error, E_r, and the stopband attenuation.

In this paper, we formulate the design of a QMF filter bank in the most general form as a nonlinear constrained optimization problem as follows:

$$\begin{array}{llll} \textit{minimize} & E_r(X)/\theta_{E_r} & & \\ \textit{subject to} & E_p(X)/\theta_{E_p} \leq 1 & E_s(X)/\theta_{E_s} \leq 1 & \delta_p(X)/\theta_{\delta_p} \leq 1 \\ & \delta_s(X)/\theta_{\delta_s} \leq 1 & T_t(X)/\theta_{T_t} \leq 1 & \end{array} \tag{4}$$

where X is a vector of variables (filter coefficients), and θ_{E_r}, θ_{E_p}, θ_{E_s}, θ_{δ_p}, θ_{δ_s}, and θ_{T_t} are performance values of the baseline design. Reconstruction error $E_r(X)$ is the objective to be minimized, and all other metrics of the prototype filter are used as constraints. This formulation allows us to improve over the best existing design (such as those reported by Johnston[9]) with respect to all performance metrics. Our formulation is more general than existing formulations because we include metrics that are not in closed forms (stopband and passband ripples and transition bandwidth).

3. LAGRANGIAN METHOD WITH ADAPTIVE CONTROL

A general *nonlinear constrained optimization problem* has the following form.

$$\begin{array}{lll} \textit{minimize} & f(X) & \\ \textit{subject to} & g(X) \leq 0 & X = (x_1, x_2, \ldots, x_n) \\ & h(X) = 0 & \end{array} \tag{5}$$

where X is a vector of real numbers, $f(X)$ is the objective function, $g(X) = [g_1(X), \cdots, g_k(X)]^T$ is a set of k inequality constraints, and $h(X) = [h_1(X), \cdots, h_m(X)]^T$ is a set of m equality constraints.

There are two distinct strategies to handle constraints. One way is to absorb all the constraints into the objective and weigh them by penalty terms. This is not effective because it is hard to choose appropriate penalty terms when constraints are violated. Another way is to absorb the constraints into a Lagrangian function, which is the sum of the objective and the constraints weighted by Lagrange multipliers. Lagrange-multiplier methods solve (5) by introducing Lagrange multipliers to gradually resolve constraints iteratively. It is an exact method that optimizes the objective $f(X)$ to meet the Kuhn-Tucker conditions.[11]

The augmented Lagrangian function corresponding to (5) is[11]

$$L_z(X, \lambda, \mu) = f(X) + \lambda^T h(X) + ||h(X)||_2^2 + \sum_{i=1}^{k} \left[max^2(0, \mu_i + g_i(X)) - \mu_i^2 \right] \tag{6}$$

where $\lambda = [\lambda_1, \cdots, \lambda_m]^T$ and $\mu = [\mu_1, \cdots, \mu_k]^T$ are two sets of Lagrange multipliers for the equality and inequality constraints respectively. We use the augmented Lagrangian function in this paper since it provides better numerical stability than the simple Lagrangian function.

According to classical optimization theory,[11] all the extrema of (6), whether local or global, are roots of the following set of *first-order necessary conditions* when these extrema are regular points.

$$\bigtriangledown_X L_z(X, \lambda, \mu) = 0 \qquad \bigtriangledown_\lambda L_z(X, \lambda, \mu) = 0 \qquad \bigtriangledown_\mu L_z(X, \lambda, \mu) = 0 \tag{7}$$

The gradients of the Lagrangian function $L_z(X, \lambda, \mu)$ become 0 at the extrema.

The set of points satisfying the necessary conditions can be found by a *first-order search method* that is expressed in a dynamic system of differential equations:

$$\begin{array}{ll} \frac{d}{dt}X(t) = -\bigtriangledown_X L_z(X(t), \lambda(t), \mu(t)) & \frac{d}{dt}\lambda(t) = \bigtriangledown_\lambda L_z(X(t), \lambda(t), \mu(t)) \\ \frac{d}{dt}\mu(t) = \bigtriangledown_\mu L_z(X(t), \lambda(t), \mu(t)) & \end{array} \tag{8}$$

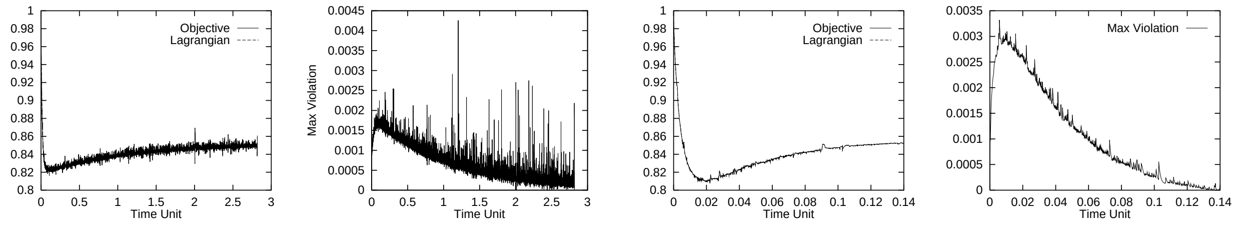

Figure 2. Comparison of search profiles with static weight $w = 10^{-4}$ (left two graphs) and dynamic w (right two graphs) in designing a 48e QMF filter bank. The convergence time to the equilibrium point with the static weight is much longer than that with the dynamic weight.

which perform gradient 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 minimum when it reaches an *equilibrium point* where all gradients become zero. In this sense, first-order methods can be considered as *local-search methods.*

Basically, Lagrangian methods rely on two counteracting forces to resolve constraints and find high-quality solutions. When constraints are satisfied, Lagrangian methods rely on gradient descents in the objective space to find high-quality solutions. On the other hand, when constraints are not satisfied, they rely on gradient ascents in the Lagrange-multiplier space to increase the penalties on unsatisfied constraints and to force the constraints into satisfaction. The balance between gradient descents and gradient ascents depends on the relative magnitudes of the Lagrange multipliers. At an equilibrium point, the forces due to descent and ascent reach a balance through appropriate Lagrange-multiplier values.

To solve the QMF filter bank design problem (4) using the Lagrangian method, we need to evaluate the performance metrics and their derivatives. We derive closed-form formulas for the reconstruction error, stopband energy, and passband energy. Metrics that do not have closed-form formulas, such as stopband ripple, passband ripple, and transition bandwidth, are evaluated using Newton's method to find their values and use finite-difference methods to approximate their derivatives. We solve the dynamic system (8) using either Euler's method or *LSODE*[†].[6]

Due to errors introduced by numerical methods in function and gradient evaluations, *LSODE* was forced to choose very small step sizes, such as 10^{-5}, in order to keep the results within certain error tolerance. This leads to larger variations in the convergence time of using a Lagrangian method to design QMF filter banks, which range from minutes to hours, even days.

The convergence time and/or solution quality further depends on the relative weights between the objective and the constraints. We show this phenomenon by adding a weight w in the objective part of the Lagrangian function as follows:

$$L_o(X, \lambda, \mu) = w\, f(X) + \lambda^T h(X) + ||h(X)||_2^2 + \sum_{i=1}^{k} \left[max^2(0, \mu_i + g_i(X)) - \mu_i^2 \right] \tag{9}$$

where $w > 0$ is a static weight on the objective. When $w = 1$, $L_o(X, \lambda, \mu) = L_z(X, \lambda, \mu)$, which is the original Lagrangian function in (6).

The convergence speed of Lagrangian methods using static weights is illustrated by the 48e QMF filter-bank design problem.[9] We use Johnston's solution as our starting point, which is a feasible solution whose performance measures are used as bounds in our formulation. However, it is not a local minimum of the objective function. Experimentally, using static w of $10^{-4}, 10^{-5}$, and 10^{-6} represents three convergence behaviors: over-weighted objective, balanced objective and constraints, and over-weighted constraints. Due to the space limit, we only show the search profile for $w = 10^{-4}$.

In Figure 2, the left two graphs show the dynamic changes of the objective, the Lagrangian-function value, and the maximum constraint violation as the search progresses for static weight $w = 10^{-4}$. Note that the trajectories of the objective and the Lagrangian-function values are overlapped because constraint violations are small. As the

[†]*LSODE* is a solver for first-order ordinary differential equations, a public-domain package available from http://www.netlib.org. It implements both Adams' method and the method based on backward differentiation formulas. It solves the system of ODEs to within a prescribed degree of accuracy using adaptive step-size adjustment.

```
1.  Select control parameters:
        time unit $\triangle t$
        initial weight $w(t = 0)$
        maximum number of iterations $i_{max}$
2.  Set window size $N_w$, e.g. 10
3.  $j := 1$          /* $j$ is the iteration number */
4.  while $j \leq i_{max}$ and stopping condition is not satisfied do
5.      advance search trajectory by $\triangle t$ time unit to get to $(X_j, \lambda_j, \mu_j)$
6.      if trajectory diverges then
            reduce $w$; restart the algorithm by going to Step 2
        end if
7.      if $(mod(j, N_w) == 0)$ then
8.          change $w$ according to search process behavior
        end if
11. end while
```

Figure 3. Framework of dynamic weight-adaptation algorithm

starting point is not a local minimum of the objective, the search descends in the original-variable X space as the objective value decreases. In the meantime, the constraints are getting violated. As constraint violations become large, the Lagrangian part slowly gains ground and pushes the search back towards the feasible region, leading to increases in the objective value and decreases in the constraint violation. Eventually, the constraint violation becomes 0 (within error tolerance), and the objective value stabilizes. The overall convergence speed to the equilibrium point is slow (949.0 CPU minutes on a Pentium Pro running Linux at $t = 2.819$). Note that fluctuations of the maximum constraint violation and the objective are due to inaccuracy in numerical estimation of the function values and gradients.

For static weight $w = 10^{-5}$, the objective and the constraints are more balanced and the convergence time to the equilibrium point is shorter (125.7 CPU minutes at $t = 1.577$ time units). For static weight $w = 10^{-6}$, the constraints are over-weighted, and constraint satisfaction dominates the search process. The trajectory is kept inside or very close to the feasible region. However, due to the small weight on the objective, improvements of the objective is slow, causing slow convergence to the equilibrium point (293.8 CPU minutes at $t = 7.608$).

To overcome the difficulty in selecting the appropriate static weights a priori for a given problem instance, we have developed a strategy to adapt weight w based on the search profile in order to obtain high-quality solutions with short convergence times.[25] Figure 3 outlines the algorithm. Its basic idea is to monitor the behavior of the search trajectory $(X(t), \lambda(t), \mu(t))$ periodically, keep track of the objective value and the constraint violation, and adapt the weight w accordingly to improve convergence time or solution quality. The total search time is divided into small units of $\triangle t$ and the search trajectory is divided into non-overlapping windows of size $N_w \times \triangle t$. In each window, we measure the progress of the search relative to that of previous windows. Generally speaking, the weight w is increased when the search is within a feasible region and the objective is not improving fast enough in successive windows. On the other hand, w is decreased when the trajectory moves slowly back to the feasible region. Divergence of the search trajectory happens when the constraint violation is extremely large. When this happens, we reduce w by a large amount.

In Figure 2, the right two graphs show the search profile of our Lagrangian method with adaptive weight control in solving the 48e QMF filter bank problem. We use time unit $\triangle t = 10^{-4}$ and window size $N_w = 10$. The weight-adaption method converges at $t = 0.142$ (35.1 CPU minutes), much faster than the static-weight method.

4. GLOBAL SEARCH FOR EQUILIBRIUM POINTS

The Lagrangian method presented in the last section only looks for a single equilibrium point, behaving like a local search algorithm. To find multiple equilibrium points, *global-search methods* are needed to bring the search out of local equilibrium points. In this section, we present our global-search strategy, followed by a description of our trace-based search method.

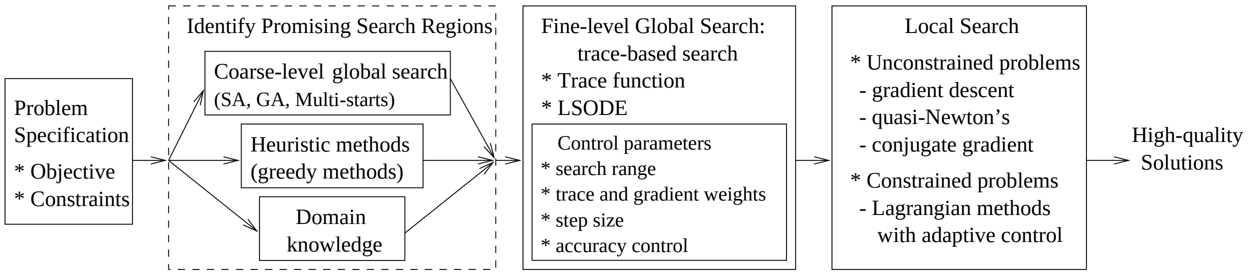

Figure 4. Proposed hybrid global-search strategy.

4.1. Global-Search Strategy

There are two classes of global-search methods: deterministic and probabilistic. Deterministic methods, such as covering methods and generalized descent methods, do not work well when the search space is large. Probabilistic methods, on the other hand, are weak in either their local or their global search. For instance, gradient information is not used well in simulated annealing and evolutionary algorithms. In contrast, gradient descent algorithms with multistarts and random probing are weak in their global-search strategy.

We present in this section a hybrid search strategy that combines coarse-level and fine-level global search, as well as local search.[15] Figure 4 shows the framework of the global-search strategy. Constrained problems are first transformed into unconstrained problems using a Lagrangian formulation. Global and local searches are then performed on the unconstrained functions to find high-quality solutions.

The search space of a continuous optimization problem is usually large. Identifying promising search regions and concentrating the search in smaller sub-spaces can effectively reduce the computation time in finding good solutions. Promising regions can be identified in three ways: domain knowledge, heuristic methods, and coarse-level global search, depending on the amount of problem-specific information available. In designing QMF filter banks, very little domain knowledge is available to define good starting points. A coarse-level global search is not very useful either because the solution is highly infeasible once the starting point is far from a feasible point. Existing coarse-level global-search methods, such as simulated annealing, genetic algorithms, and multi-start of local descents, are not able to generate good starting points. For this reason, we just use the best existing solution as our starting point for the fine-level global search.

Given a starting point, we apply a fine-level global search to look for equilibrium points in the surrounding search region. Existing global-search methods do not work well for this purpose. Algorithms based on random restarts, simulated annealing, genetic algorithms, and clustering do not work well when the problem size is large. Moreover, they may have difficulty in satisfying the constraints when the search stops. Trajectory-based methods that rely on internal forces to move the trajectory do not work well because they have difficulty to adapt to the rugged terrain in a Lagrangian search. To overcome this problem, we use a trace-based fine-level global search presented in the next subsection. The method relies on two counteracting forces: the local gradient force that drives the search to a local equilibrium point, and a deterministic problem-independent trace to leads the search out of local equilibrium points.

The search trajectories formed in the fine-level global search are used to provide good starting points in the local-search stage. We have used two heuristics to select the initial points: the best solutions in periodic time intervals, or the local minima along each trajectory.

4.2. Trace-Based Fine-Level Global Search

According to the dynamic system (8) for finding equilibrium points of (9), the Lagrangian method performs descents in the original-variable space to locate local minima of the objective function when the constraints are satisfied, and performs ascents in the Lagrange-multiplier space when the constraints are not satisfied. Eventually, the search converges to an equilibrium point. To find globally optimal solutions, we are interested to move the search trajectory from one equilibrium point to another, without having to restart the search. To do so, we add an *external* force to pull the search out of a local equilibrium point in the original-variable space continuously and escapes from it without restarts.[24,16]

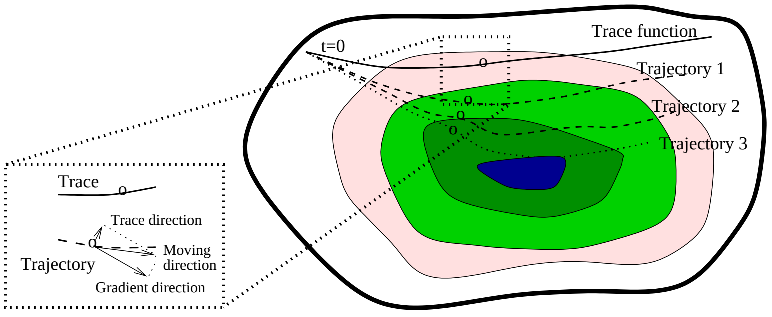

Figure 5. Illustration of our trace-based global search defined in the original-variable space. The search trajectory shows the combined effect of gradient descents and pulls exerted by the moving trace. Three cascaded global-search stages generate Trajectories 1, 2 and 3, respectively.

Figure 5 illustrates the process of the trace-based global search. The search process uncovers promising local optimal regions in the search space without going deep into each one. The global-search trajectory is formed by the combination of two vectors: the problem-independent force provided by the trace, and the problem-dependent force provided by the local gradient. These two counteracting forces form a composite vector that represents the route taken by the trajectory.

Generally, there can be a number of bootstrapping stages in the trace-based global search, coupling one stage to the next by feeding its output trajectory as the trace of the next stage. The problem-independent trace is fed into the first stage. By cascading the stages, one stage is coupled to the next by feeding its output trajectory as the trace of the next stage. Figure 5 shows three cascaded global-search stages in which the trace and local gradient generate Trajectory 1. Trajectory 1 and local gradient then generate Trajectory 2, and so on.

Based on the dynamic system (8) for the Lagrangian function L_o in (9), the system of ODEs that characterize each stage of the trace-based global search has the following form:

$$\frac{d}{dt}X(t) = -\mu_g \bigtriangledown_X L_o(X(t), \lambda(t), \mu(t)) - \mu_t \left(X(t) - T(t)\right) \tag{10}$$

$$\frac{d}{dt}\lambda(t) = \bigtriangledown_\lambda L_o(X(t), \lambda(t), \mu(t)), \qquad \frac{d}{dt}\mu(t) = \bigtriangledown_\mu L_o(X(t), \lambda(t), \mu(t)) \tag{11}$$

where μ_g and μ_t are coefficients specifying the weights for descents in local minimum regions and for exploration of broader space, and $T(t)$ is the trace function.

Trace functions traverse the search space in an aperiodic and uniform fashion. Empirically, we have designed an n-dimensional trace function as follows.

$$T_i(t) = \rho_i \sin\left[2\pi \left(\frac{t}{s}\right)^{1-a_0-\frac{(1-a_0-d_0)(i-1)}{k}}\right], \qquad i = 1, \cdots, n \tag{12}$$

where i represents the i'th dimension, a_0, d_0, s, ρ and k are parameters that control the speed and shape of the function. The parameter values we used in our experiments are $a_0 = 0.05$, $d_0 = 0.5$, $s = 2$, $\rho = 1$ and $k = n$.

The global-search strategy containing the trace-based global-search method described here and the weight-adaptation Lagrangian method described in the last section has been implemented in a prototype called *Novel* (*Nonlinear Optimization via External Lead*).[24,16]

4.3. Potential for Parallel Processing

Parallel processing is essential in solving large nonlinear optimization problems. These problems are computationally intensive because the number of local minima and the number of variables can be large; the objectives, constraints, and derivatives can be expensive to evaluate; and many evaluations of the objectives, constraints, and derivatives may be needed.

As shown in Figure 4, the proposed global-search strategy is suitable for parallel processing because many components can be executed in parallel. Generally speaking, the parallelism exists at three levels: (1) parallel global searches, (2) parallel local searches, and (3) parallel evaluation of objective function and constraints, and their corresponding derivatives.

First, the coarse-level global search provides initial points for the fine-level global search. Parallel search methods can be used in the coarse-level global search, such as parallel genetic algorithms, parallel simulated annealing, and parallel multi-starts of descents. Other heuristic methods for coarse-level global search can also be parallelized. For instance, in the trajectory-based method, sample points are selected along a 1-dimensional trajectory function over the search space. Then, descents from these sample points, which take much more time than selecting the sample points, can be done in parallel.

Second, starting from different initial points generated by the coarse-level search, multiple trace-based global searches can be executed fully in parallel. All of them are independent to each other and have minimal communication and synchronization overhead. Within each trace-based global search, multiple stages can also be executed concurrently in a pipelined fashion. However, since the number of stages is usually not large, the amount of parallelism is limited in this case.

Third, starting from initial points generated by fine-level global searches, the local searches are independent to each other and can be executed in parallel. For some problems such as the QMF filter bank design, the local descent is time-consuming and constitutes a large portion of the total execution time. Performing multiple local descents in parallel can speed up the search process significantly.

Finally, the search for optimal solutions is based on the values and derivations of objective function and constraints that can be expensive to evaluate. In the QMF filter bank design problem, for the three performance metrics that have closed-form formulas, their values and corresponding derivatives can be computed in very short time. However, for the other three performance metrics that do not have closed-forms, computing their values numerically is expensive. Furthermore, approximating their derivatives using finite-difference methods requires a number of function evaluations proportional to the number of variables. It makes the derivatives even more expensive to evaluate. Thus, computing function values and derivatives in parallel can also greatly reduce the execution time.

5. EXPERIMENTAL RESULTS

We have applied our global-search method, *Novel*, to solve the 14 QMF filter-bank design problems formulated by Johnston.[9] Our goal is to find designs that are better than Johnston's results across all the six performance measures. Hence, we use (4) with the constraint bounds defined by those of Johnston's designs.

Note that Johnston used sampling in computing energy values, whereas we use closed-form integration. As a result, Johnston's designs may not be locally optimal in a continuous formulation. 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 0.789 of Johnston's result. By applying global search, *Novel* can further improve the design to result in a reconstruction error of 0.753 of Johnston's result.

Figure 6 compares the performance of our adaptive Lagrangian method with that of the static-weight Lagrangian method. It shows the convergence times of the static method normalized with respect to that of the adaptive method in solving the 24d and 48e design problems. We always start the search from Johnston's solutions as starting points. For the static method, we have used, respectively, weight values of $10^{-4}, 5 \times 10^{-5}, 10^{-5}, 5 \times 10^{-6}$, and 10^{-6} in our experiments. In solving the 24d (resp. 48e) problem, our adaptive method takes 6.6 (resp. 35.1) minutes to converge to an equilibrium point with an objective value of 0.789 (resp. 0.852), whereas the static method converges to the same equilibrium point most of the time, but with vastly different convergence times.

Next, we compare the performance of *Novel*, simulated annealing (SA), and evolutionary algorithms (EA). In *Novel*, the trace-based global search consists of one global-search stage that starts from Johnston's solution and searches in the range ± 0.01 in each dimension around Johnston's solution. The global-search stage is run for one time unit, *i.e.*, from $t = 0$ to $t = 1$. In every 0.1 time units, a starting point for local search is selected, resulting in a total of at most 10 descents using the adaptive Lagrangian method. On a sequential computer, the global search and the local search are implemented as interleaving each other, i.e., the first 0.1 time unit of global search is followed by a complete local search, and then the second 0.1 time unit of global search, and so on. In case that the local search is too expensive and the execution time is over the limit, e.g., 30 hours on the 200-MHz Pentium Pro, the search is

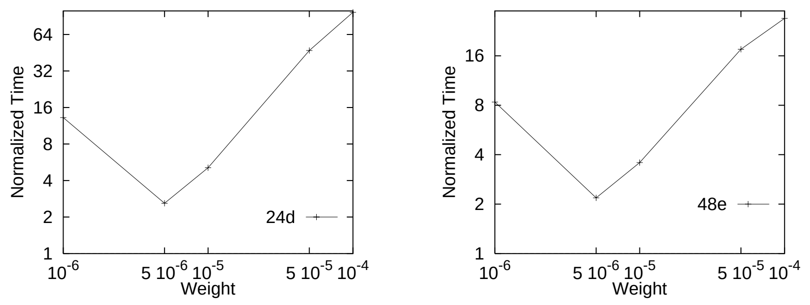

Figure 6. Comparison of the convergence speeds of Lagrangian methods with and without adaptive weight control for the 24d (left) and 48e (right) QMF filter-bank design problems. The convergence times of the static method are normalized with respect to the corresponding times of the adaptive method.

Table 2. Experimental results of *Novel* and simulated annealing (SA) in designing QMF filter banks. Novel uses trace-based global search and the adaptive Lagrangian method as local search. The cooling rates of SA for 16-, 24-, 32-, 48-, and 64-tap filter banks are 0.98, 0.98, 0.95, 0.85, and 0.80, respectively. The total number of evaluations for 16-, 24-, 32-, 48-, and 64-tap filter banks are 20, 10, 10, 3, and 2 millions, respectively. δ_p, δ_s, and E_s in all the cases are equal to or slightly less than 1. All experiments were run on a 200-MHz Pentium Pro with Linux.

Filter-type	*Novel*					*SA*			
	E_r	E_p	T_r	CPU time (hrs)	# Descents	E_r	E_p	T_r	CPU time (hrs)
16a	0.986	0.858	1.000	35.7	10	0.986	0.862	1.000	12.4
16b	0.985	0.893	1.000	6.9	10	0.985	0.895	1.000	12.8
16c	0.822	0.919	1.000	6.2	10	0.419	0.423	1.015	12.0
24b	0.964	0.778	1.000	33.1	3	0.964	0.801	1.000	12.9
24c	0.910	0.768	1.000	10.9	10	0.853	0.551	1.003	13.4
24d	0.753	0.770	1.000	7.6	10	0.399	0.404	1.018	13.1
32c	0.959	0.738	1.000	40.2	4	0.959	0.748	1.000	16.5
32d	0.870	0.800	1.000	15.2	10	0.617	0.570	1.015	17.0
32e	0.716	0.889	1.000	14.1	10	0.501	0.583	1.013	16.3
48c	0.793	0.810	1.000	48.32	1	0.753	0.802	1.000	13.1
48d	0.947	0.756	1.000	63.6	2	0.943	0.757	1.001	14.4
48e	0.852	0.838	1.000	43.4	10	0.618	0.585	1.015	14.4
64d	0.784	0.787	1.000	52.39	1	0.867	0.794	1.000	30.1
64e	0.842	0.737	1.000	36.4	2	0.541	0.514	1.032	31.2

terminated after the current local search finishes. We use *LSODE* to generate both the global- and the local-search trajectories.

The SA we have used is SIMANN from netlib that works on the following weighted-sum formulation:

$$\begin{aligned}\min_X f(X) &= w_1 \frac{E_r(X)}{\theta_{E_r}} + w_2 \max\left(\frac{E_p(X)}{\theta_{E_p}} - 1, 0\right) + w_3 \max\left(\frac{E_s(X)}{\theta_{E_s}} - 1, 0\right) \\ &\quad + w_4 \max\left(\frac{\delta_p(X)}{\theta_{\delta_p}} - 1, 0\right) + w_5 \max\left(\frac{\delta_s(X)}{\theta_{\delta_s}} - 1, 0\right) + w_6 \max\left(\frac{T_t(X)}{\theta_{T_t}} - 1, 0\right)\end{aligned} \tag{13}$$

In our experiments, we assign $w_1 = 1$ for the reconstruction error and $w_i = 10$, $i = 2, \cdots, 6$, for the other performance measures. We tried various parameter settings and report the best solutions in Table 2. Like *Novel*, SA started from Johnston's solutions as initial points and searched in the range ± 0.01 in each dimension around Johnston's solutions.

Table 2 compares the experimental results of *Novel* and SA in solving QMF filter-bank design problems. The values are normalized with respect to Johnston's design. A value less than 1 for a performance metric means that the design is better on this metric as compared to Johnston's design. The column of "# Descents" shows the number of applications of the adaptive Lagrangian method in *Novel* within the corresponding amount of CPU time. As

Table 3. Experimental results of the evolutionary algorithm in solving a constrained formulation (*EA-Constr*) and a weighted-sum formulation (*EA-Wt*) of the QMF filter-bank design problems. The population size of EA is $10n$, where n is the number of variables, and the number of generations for 16-, 24-, 32-, 48-, and 64-tap filter banks are 2000, 2000, 2000, 1000, and 1000, respectively. δ_p, δ_s, and E_s in all the cases are equal to or slightly less than 1. All experiments were run on a Sun SparcStation 10/51.

Filter-type	*EA-Constr*				*EA-Wt*			
	E_r	E_p	T_r	CPU time (hrs)	E_r	E_p	T_r	CPU time (hrs)
16a	937.0	0.479	1.604	2.7	1.520	0.622	1.322	4.1
16b	154.2	1.000	1.176	2.7	0.986	0.893	1.000	2.9
16c	1.056	0.854	1.000	2.7	0.418	0.422	1.015	2.8
24b	889.2	1.000	1.564	7.8	1.049	0.651	1.448	8.3
24c	421.9	1.000	1.287	7.5	0.825	0.511	1.011	8.8
24d	6.075	1.000	1.003	7.6	0.399	0.403	1.019	8.1
32c	2283.8	1.000	1.647	22.0	0.842	0.671	1.025	17.8
32d	4.914	1.000	1.011	17.9	0.560	0.504	1.026	20.9
32e	0.724	0.905	1.000	24.5	0.501	0.582	1.013	19.2
48c	1656.9	0.999	1.166	25.2	0.780	0.794	1.020	25.4
48d	1472.6	1.000	1.438	27.7	0.757	0.606	1.049	26.9
48e	2.350	1.000	1.001	29.2	0.566	0.530	1.023	28.1
64d	3862.7	0.987	0.974	62.9	0.803	0.816	1.000	58.7
64e	0.940	0.854	1.000	68.0	0.492	0.445	1.033	61.8

shown in Table 2, *Novel* always find designs that have better objective value, E_r, and are also better in one or more constraints and no worse in others. In contrast, SA improves Johnston's solutions on all six performance measures for some problems, but obtains solutions with worse transition bands for some other problems. These happen because the selected weights in the weighted-sum formulation cannot force all the objectives to be smaller than certain values.

The EA used in our experiments is Sprave's Lice (Linear Cellular Evolution)[20] that solves both the constrained formulation (*EA-Constr*) and the weighted-sum formulation (*EA-Wt*). In *EA-Constr*, the fitness value of each individual is based on its feasibility and its objective value. Feasible individuals have higher fitness value than infeasible ones. Among infeasible individuals, the one with a smaller constraint violation has a higher fitness value. In *EA-Wt*, as in the experiments with SA, we assigned a weight of 1 for the reconstruction error and a weight of 10 for the other performance measures. As before, we set the search range to be ± 0.01 in each dimension around Johnston's solutions, We have tried various population sizes and number of generations, and report the best solutions we have obtained in Table 3.

Both *EA-Constr* and *EA-Wt* do not perform very well. *EA-Constr* has difficulty in finding good feasible solutions. This happens because the constraints based on Johnston's solutions form a tiny feasible region in the search space, and randomly generated points have little chance of being feasible. *EA-Wt* is better than *EA-Constr*. However, except in the "16b" and "64d" design problems, where *EA-Wt* improves Johnston's solutions in all six performance measures, *EA-Wt* only finds solutions with trade-offs, often with larger transition bandwidths than those of Johnston's and better in the remaining performance measures.

To summarize, the performance improvement in *Novel* comes from three sources. First, the closed-form formulation used in *Novel* is more accurate than the sampling method used in Johnston's approach. Local optima found by *Novel* are true local optima, whereas Johnston's solution are local optima in a discrete approximation of the design problem. Second, *Novel* uses a constrained formulation which allows it to find designs that are guaranteed to be better than or equal to Johnston's design with respect to all performance measures. Third, *Novel* employs effective global-search strategies that allows it to explore a large part of the search space without first committing to many expensive local searches. In addition, the small amount of dependency between various components of *Novel* makes it suitable for parallel implementation.

REFERENCES

1. A. N. Akansu and R. A. Haddad. *Multiresolution Signal Decomposition.* Academic Press, Inc., San Diego, CA, 1992.
2. 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.
3. 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.
4. 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.
5. N. J. Fliege. *Multirate Digital Signal Processing.* John Wiley and Sons, 1994.
6. A. C. Hindmarsh. ODEPACK, a systematized collection of ODE solvers. In R. S. Stepleman, editor, *Scientific Computing*, pages 55–64. North Holland, Amsterdam, 1983.
7. 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.
8. 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.
9. J. D. Johnston. A filter family designed for use in quadrature mirror filter banks. In *Proc. of Int'l Conf. on ASSP*, pages 291–294, 1980.
10. 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.
11. D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley Publishing Company, 1984.
12. 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.
13. T. Q. Nguyen. Digital filter bank design quadratic-constrained formulation. *IEEE Trans. on Signal Processing*, 43(9):2103–2108, September 1995.
14. T. Q. Nguyen and P. P. Vaidyanathan. Two-channel perfect-reconstruction FIR QMF structures which yield linear-phase analysis and synthesis filters. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 37(5):676–690, May 1989.
15. Y. Shang. *Global Search Methods for Solving Nonlinear Optimization Problems.* Ph.D. Thesis, Dept. of Computer Science, Univ. of Illinois, Urbana, IL, August 1997.
16. Y. Shang and B. W. Wah. Global optimization for neural network training. *IEEE Computer*, 29:45–54, March 1996.
17. 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.
18. 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.
19. 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.
20. J. Sprave. Linear neighborhood evolution stategies. In *Proc. of the third Annual Conf. on Evolutionary Programming*, San Diego, CA, 1994. World Scientific.
21. 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.
22. P. P. Vaidyanathan. Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial. *Proc. of the IEEE*, 78(1):56–93, January 1990.
23. P. P. Vaidyanathan. *Multirate Systems and Filter Banks.* Printice-Hall Inc., 1993.
24. B. W. Wah and Y.-J. Chang. Trace-based methods for solving nonlinear global optimization problems. *J. of Global Optimization*, 10(2):107–141, March 1997.
25. B. W. Wah, T. Wang, Y. Shang, and Z. Wu. Improving the performance of weighted Lagrange-multiple methods for constrained nonlinear optimization. In *Proc. 9th Int'l Conf. on Tools for Artificial Intelligence*, pages 224–231. IEEE, November 1997.
26. J. W. Woods, editor. *Subband Image Coding.* Kluwer Academic Publishers, 1991.