Applications of Simulated Annealing for the Design of Special Digital Filters

Nevio Benvenuto, *Senior Member, IEEE,* Michele Marchesi, and Aurelio Uncini, *Member, IEEE*

Abstract—**This paper describes the salient features of using a simulated annealing (SA) algorithm in the context of designing digital filters with coefficient values expressed as the sum of power of two. A procedure for linear phase digital filter design, using this algorithm, is first presented and tested, yielding results as good as known optimal methods. The algorithm is then applied to the design of Nyquist filters, optimizing at the same time both frequency response and intersymbol interference, and to the design of cascade form FIR filters.**

Although SA is not a solution to all design problems, and is computationally very expensive, it may be an important method for designing special digital filters where numerous or conflicting constraints are present.

I. Introduction

The realization of low-cost and high-speed special purpose DSP hardware requires fast fixed-point arithmetic, and consequently filters with very coarse coefficients' values are highly valuable. Efficient design procedures for these kinds of filters are therefore very important.

A variety of algorithms exist for the design of digital filters. Unfortunately, each of them is suitable for a particular application. For example, integer programming methods are used for the design of FIR digital filters with discrete valued coefficients when the performance criterion is the classical minmax [1]–[3]; i.e., the maximum weighted difference between the desired frequency response and the actual filter frequency response. These methods require a high computation time, and a large amount of memory, so that suboptimal techniques have also been extensively used. Among other techniques we recall the use of integer programming methods to a mean-square-error criterion [4].

In this paper we are interested mainly in FIR filters whose coefficients can be written as a power of two, or as a sum (or difference) of two power of two. It has been shown that while both of the above mentioned integer programming techniques can still be applied [3]–[5], some of the approximation techniques, which determine the discrete solution by rounding of the infinite precision solution, can no longer be used because they yield very poor performance [5]. For higher order filters, we should mention the suboptimal design method of Zhao and Tadokoro [6] which first determines the filter gain, and then performs a local optimization of the filter coefficients in order to reduce the maximum weighted frequency error.

The basic limitation of all the above methods is that they can mainly be used to design FIR digital filters with specifications in the frequency domain, or whenever the constraints on the coefficients are of linear type. On the other hand, to design Nyquist filters [7], [8] or cascade form FIR filters [9], we may have a mixture of requirements and some of those may not be linear in the coefficients.

This paper describes another method based upon the global optimization algorithm, known as simulated annealing (SA) [10]. The algorithm has been directly derived from the SA algorithm for functions of continuous variables presented in [11]. This could be accomplished because the range of variation of the coefficients, although discrete, is sufficiently broad. The original algorithm has been modified in such a way that the independent variables are forced to be discrete (sum of two power-of-two). Other approaches of using SA to digital filter design could also be applied [12]–[15]. However, the proposed method is more general because it can be used to simultaneously optimize functions of both continuous and discrete variables. Furthermore, new features have been added with respect to traditional SA algorithms with the goal of reducing the computational complexity.

A detailed description of the algorithm is reported in Section II. To test the optimization procedure, the proposed algorithm was implemented in C language on a SUN 3/60 workstation for the same filter designs presented in [5, fig. 4], and a comparable performance, up to 0.5 dB, was obtained (see Section III). The drawback of using SA is that the computation time is quite long; in the order of 1–2 hours for each filter design, on the Sun 3/60. However, this was more than compensated by the extreme versatility of the new algorithm, which can be used to design filters with multiple constraints. Two examples of applications will be reported: 1) design of

Manuscript received December 20, 1989; revised December 13, 1990. This work was supported in part by the Ministero dell'Università e della Ricerca Scientifica and in part by the Consiglio Nazionale delle Ricerche of Italy.

N. Benvenuto was with the Dipartimento di Elettronica e Informatica, Università di Ancona, Ancona, Italy. He is now with the Dipartimento di Elettronica e Informatica, Università di Padova, Padova, Italy.

M. Marchesi was with the Dipartimento di Elettronica e Automatica, Università di Ancona, Ancona, Italy. He is now with the Dipartimento di Ingegneria Biofisica e Elettronica, Università di Genova, Genova, Italy.

A. Uncini is with the Dipartimento di Elettronica e Automatica, Università di Ancona, 60131 Ancona, Italy.

IEEE Log Number 9104883.

1053-587X/92$03.00 © 1992 IEEE

Nyquist filters, and 2) cascade form FIR filters (two stages). In this second case, the algorithm is able to perform an optimization of the two stages at the same time. Indeed, it is seen that this approach yields a better performance than a procedure which designs one stage at a time iteratively [9].

II. An Algorithm for Designing Digital Filters

In this section we present a general algorithm which uses the SA method for the design of FIR digital filters.

A. Variable Space

In general terms, the problem of designing a digital filter consists of finding the minimum of a real function $f(\boldsymbol{x})$ whose variables $\{x_i\}$ may assume only discrete values [5]. For a minmax criterion, the function represents the maximum weighted difference between the desired and the actual frequency response. Variables are the coefficients $\{h(n)\}$ of the filter, which in our case can be written as a power of two, or sum of two power of two terms. In the latter case, the domain of x_i is given by

$$D = \left\{ \alpha\colon \alpha = \sum_{k=1}^{2} c_k 2^{-g_k}, \quad c_k \in \{-1, 0, 1\} \text{ and } g_k \in \{1, 2, \cdots, B\} \right\} \tag{1}$$

which is ordered as follows:

$$D = \{\alpha_1, \alpha_2, \cdots, \alpha_L\}, \text{ with } \alpha_1 < \alpha_2 < \cdots < \alpha_L. \tag{2}$$

In (1), B is an integer representing the maximum number of shifts that can be performed on the filter input signal. It is seen that values of x_i are within the interval $[-1, 1]$ and they are not distributed uniformly. However, if B is sufficiently large, the values of x_i are numerous enough that we may consider the domain of x_i "almost continuous." For this reason, a modified version of the SA algorithm for continuous variables [11] has been used. As well, a similar approach has been used by the authors to design FIR filters with finite word length coefficients [14]. The difference is that now the filter coefficient values are very coarse and, as a consequence, the filter gain G assumes a particular relevance. Indeed, performance may deteriorate up to 6 dB if the choice of G is not accurate [6]. The filter design then becomes a mixed optimization problem, namely, continuous in the filter gain, and discrete in the coefficient values. The SA algorithm has been adapted to solve this type of problem, thus avoiding the use of an exhaustive search over a fine grid of G's.

To summarize, on designing linear phase FIR filters of length N, the variables are the gain G, with domain I_G, and $(N + 1)/2$ coefficients, each with domain D as defined by (1). This holds true if N is odd. For other cases we refer to [16]. If $\boldsymbol{x}$ denotes the vector of independent variables with M components, its domain C is simply the Cartesian product of I_G with $D^{(N+1)/2}$, i.e.,

$$C = I_G \times D^{(N+1)/2} \tag{3}$$

and in this case $M = (N + 3)/2$. In general, I_G is an interval belonging to R_+.

B. SA Algorithm

SA is a well-known powerful global optimization algorithm, introduced in combinatorial optimizations [10]. It is based on random moves, and has the ability to overcome local minima, found on the way toward a better minimum, with uphill moves.

The SA algorithm actually used is derived from the SA algorithm for functions of continuous variables proposed in [11]; a proper discretization of the variables' values and various controls of the computational effort, during the search, have been added.

1) Generalities: Let $\boldsymbol{x}$ be a vector of M components in C and $f(\boldsymbol{x})$ the function to minimize:

$$f(\boldsymbol{x})\colon C \rightarrow R. \tag{4}$$

The algorithm proceeds iteratively: from the starting point $\boldsymbol{x}_0$ it generates a sequence of points $\{\boldsymbol{x}_k\}$ that tend towards the minimum. The latest point added to the sequence is called the current point $\boldsymbol{x}_c$. Some symbols are now introduced:

r a random number uniformly distributed in the range $[-1, 1]$,

$\boldsymbol{s}$ the step vector. Its components $\{s_p\}$, $p = 1, 2, \cdots, M$, control the trial point generation (see Section II-B3),

$x_{c,p}$ the pth component of the vector $\boldsymbol{x}_c$.

Fig. 1 shows a Pascal-like diagram of the main steps of the minimization algorithm. It is composed of an initialization phase, a main loop (from which it is possible to escape if a termination test has been satisfied), and four nested loops. In the following, we shall describe the algorithm from the inner to the outer loop.

2) Algorithm Parameters: The algorithm has many parameters controling its behavior and their proper setting is crucial to obtain good performance. A partial list is now given:

T_0 the starting temperature,

$T_{\min}$ the minimum temperature,

r_T the temperature reduction factor,

ϵ the terminating criterion,

N_ϵ the number of temperature reductions to test for termination,

N_s the number of cycles between step adjustments,

$N_{\min}$ the minimum number of step adjustments between temperature reductions,

$N_{\max}$ the maximum number of step adjustments between temperature reductions,

N_r the number of temperature reductions between restarts from the optimum value reached so far.

```
begin
  INITIALIZE PARAMETERS;
  while true do
  begin
    j:=1;
    while j<=$N_r$ do
    begin
      k:=1;
      while k<=$N_k$ do
      begin
        m:=1;
        while m<=$N_s$ do
        begin
          p:=1;
          while p<=M do
          begin
            SEARCH MOVE;
            if METROPOLIS TEST PASSED then
            begin
              ACCEPT NEW POINT;
              COMPARE WITH $x_{opt}$;
            end;
            p:=p+1;
          end;
          m:=m+1;
        end;
        STEP ADJUSTMENT;
        k:=k+1;
      end;
      COMPUTE <f>, <$f^2$>, A, $N_k$;
      if TERMINATION TEST PASSED then goto 1;
      T:=$r_T$*T;
      if T<=$T_{min}$ then goto 1;
      j:=j+1;
    end;
    RESTART FROM $x_{opt}$;
  end;
  1: return $x_{opt}$;
end.
```

Fig. 1. The proposed simulated annealing optimization algorithm.

The meanings and suitable values of these parameters will be given after a general description of the algorithm.

3) Search Cycle: In the inner loop, the algorithm performs the basic search of the function to minimize. New candidate points x are generated at random along each coordinate direction p ($p = 1, \cdots, M$, in sequence), according to the following formulas:

A move for a continuous variable: ($p = 1$ for C given by (3))

$$x_p := x_{c,p} + rs_p. \tag{5}$$

If x_p lies outside I_G, then (5) is applied again until $x_p \in I_G$.

A move for a discrete variable: ($p > 1$ for C given by (3)).

Let $x_{c,p} = \alpha_{k_p}$. Selection of index value k_p' associated to each element of D in (2), $\alpha_{k'_p}$, as follows:

$$k_p' := k_p + \lfloor rs_p + 0.5 \rfloor . \tag{6}$$

If $k_p' < 1$ or $k_p' > L$ or $k_p' = k_p$, then (6) is applied again until $1 \leq k_p' \leq L$ and $k_p' \neq k_p$. In this case:

$$x_p := \alpha_{k_p'}. \tag{7}$$

With this method, trial coordinate values are distributed in intervals of size proportional to s_p and centred around $x_{c,p}$. The generation of an admissible trial point x is called a *move*. A series of moves along all M coordinate directions is called a *cycle*.

A trial point x becomes the new current point if the metropolis test is satisfied [10], [11]. Namely, if

$$f(x) \leq f(x_c) \tag{8}$$

x is always accepted. However, if

$$f(x) > f(x_c) \tag{9}$$

x is accepted with probability p where

$$p := \exp\left(\frac{f(x_c) - f(x)}{T_k}\right). \tag{10}$$

T_k is a control parameter called temperature. If $T_k > 0$, it is possible that a trial point x is accepted even if it has a higher value of f than the current point.

The method starts with a "high" temperature T_0, for which most trial points are accepted, and then gradually reduces T_k in order to focus on the minimum. If the values of T_k are very high, most trial points are accepted, and the algorithm performs a random walk of the domain C. At lower (but still high enough) values of T_k only the gross behavior of the cost function is relevant to the search. Eventually, as T_k is further decreased, finer details can be developed to obtain a good final point.

The best point reached by the search (i.e., the point with the lowest function value so far found) is recorded as x_{opt}.

4) Initialization: The initialization phase consists of choosing the following parameters: x_0: the starting point; s_0: the starting step vector.

Selection of the starting point depends on the problem. In general, for problems with a low value of M, the SA algorithm is robust enough to yield good results even with random starting points [14], [15]. However, when M is high the choice of a "good" starting point is very important for good performance of the algorithm. As a starting point for filter design, we adopted the quantized version (see (1)) of a good infinite precision solution.

Based upon (5) and (6) we have the correspondence: if x_p is a continuous variable, s_p should also be continuous; while if x_p is discrete, the corresponding s_p should be a positive integer number. In all presented tests the starting values are respectively equal to one quarter of the length of I_G in the continuous case, and to 2 in the discrete case. However, this choice is not very important, since s is quickly adapted to the problem by the step adjustment algorithm, illustrated in the next paragraph.

5) Step Adjustments: Step vector s is periodically adjusted every N_s cycles to follow the function behavior without wasting function evaluations with search steps too big (in such case most of the trial points are rejected) or too small (the sequence tends too slowly toward the minimum). The criterion used to adjust the steps is to maintain a 1 : 1 ratio between accepted and rejected candidate points along each coordinate direction.

For continuous variables, the formula used for step adjustments is the same as reported in [11], with the further constraint being that component s_p should not be greater than one quarter of the length of interval I_G. The same formula is used in the discrete case, with its value rounded to the nearest integer. Now s_p must lie between 2 and $L/4$. Limit values are assumed, both in the continuous and discrete case, when the formula yields a value outside bounds. These constraints guarantee that a new trial component x_p both differs from the current point component $x_{c,p}$ and probably lies inside its domain.

6) Termination Test: The proposed algorithm has two termination criteria to decide when the global minimum has been reached. The first one depends on parameters ϵ and N_ϵ [11]. Specifically, the function value f_j is recorded in the current point immediately before each temperature reduction from T_j to T_{j+1}. When all search cycles at temperature T_k are made, the search is stopped if

$$|f_k - f_{k-u}| \le \epsilon, \qquad u = 1, \cdots, N_\epsilon$$

and

$$f_k - f(\boldsymbol{x}_{\text{opt}}) \le \epsilon. \tag{11}$$

The other termination criterion stops the search when the temperature falls below a given minimum value $T_{\min}$.

7) Temperature Reductions: In the proposed algorithm, the number of cycles between step adjustments N_s is constant, while the number of step adjustments N_k at a fixed temperature T_k, is dynamically varied according to a criterion similar to White's [17]. During the search at temperature T_{k-1}, the average value of $f(\boldsymbol{x})$ and $f^2(\boldsymbol{x})$, for the accepted points only, are evaluated. Let A be defined as

$$A := \frac{\sqrt{\langle f^2 \rangle - \langle f \rangle^2}}{T_{k-1}}. \tag{12}$$

N_k is computed as follows:

$$N_k := \begin{cases} N_{\min}, & \text{if } A \le A_1 \\ N_{\min} + \dfrac{(A - A_1)(N_{\max} - N_{\min})}{A_2 - A_1}, & \\ & \text{if } A_1 < A < A_2 \\ N_{\max}, & \text{if } A \ge A_2. \end{cases} \tag{13}$$

The behavior of $N_k(A)$ is the simplest possible: N_k increases linearly with A if $A_1 \le A \le A_2$, otherwise a minimum or maximum value is assumed. Incidentally, if the distribution of $f(\cdot)$ is Gaussian with standard deviation σ then A is an estimate of the ratio σ/T_{k-1} [17]. The initial value of N_k, N_0, is set equal to $N_{\min}$.

Fig. 2 shows a typical annealing curve of $\langle f \rangle$ (solid line) and the corresponding behavior of A (dashed line) versus the temperature T, on designing a FIR filter with a minmax criterion. For higher values of T, the annealing curve exhibits the characteristic plateau that denotes the random sampling of $f(\cdot)$. Values of A are lower than 1, and from (13) the number of function evaluations is the lowest possible. When the annealing takes place, the $\langle f \rangle$ curve decreases while A rises, allowing many more function evaluations to be performed. Eventually, the annealing curve reaches its final plateau closer to the value of the global minimum found. Correspondingly, A decreases, and this avoids wasting function evaluations as the minimum has been found.

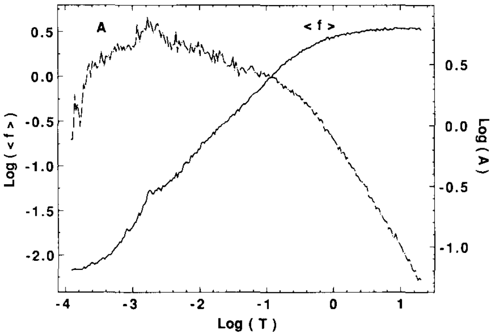


Fig. 2. Average of cost function $\langle f \rangle$ and parameter A versus temperature.

Parameter N_k is crucial for good sampling of the function $f(\cdot)$, without wasting computation effort. By keeping N_k a variable, it allows us to save more than 50% of function evaluations. Furthermore, this improves reliability with respect to the algorithm [11].

8) Restart From the Optimum: In the algorithm [11], after each temperature reduction, the current point $\boldsymbol{x}_c$ was set equal to $\boldsymbol{x}_{\text{opt}}$. This choice was justified by the fact that the functions considered were continuous, with many local minima, and defined over possible unbounded domains. This approach led to better results with respect to the original SA algorithm [10], where the sequence $\{\boldsymbol{x}_k\}$ was never alterated. The algorithm presented here generalizes the former choice, and restarts from the optimal point after consecutive N_r temperature reductions.

9) Parameters Values: In this section, we will report the typical values assigned to the various parameters, and we will briefly justify the choices.

T_0, the starting temperature. From Section II-B7 T_0 must be "high enough" to allow a sampling of the search space before the annealing takes place. In practice, a good value of T_0 is one which yields a behavior of $\langle f \rangle$ and A as shown in Fig. 2.

In [17], it has been proposed to choose T_0 proportional to the standard deviation of the cost function. From our experience, we believe that a better choice of T_0 is based on the direct observation of the behavior of $\langle f \rangle$ and A. In the tests performed, T_0 was chosen between 2 and 0.1.

$T_{\min}$, the minimum temperature. When a new minimization problem is tackled, it is better to set $T_{\min}$ to zero. As experience is gained on the problem, we should find at which temperature the optimum is "frozen," then this is the appropriate value of $T_{\min}$. This choice avoids wasting computation time at lower temperatures. In our cases, $T_{\min}$ was chosen between 2×10^{-4} and 10^{-5}, depending on the filter design.

r_T, the temperature reduction factor. This parameter, together with N_s, $N_{\min}$, and $N_{\max}$, determines the number N_{eval} of function evaluations performed in a single optimization run, in the following way:

$$N_{\text{eval}} \propto MN_s\langle N_k\rangle \log^{-1}\left(\frac{1}{r_T}\right). \tag{14}$$

In turn, $\langle N_k\rangle$ depends on $N_{\min}$ and $N_{\max}$. We can say that N_{eval} approximately determines the computation time of the design procedure. In the SA literature, r_T has been given values between 0.5 and 0.99, sometimes with an adaptive schedule. In our algorithm, r_T is constant, with a value between 0.85 and 0.95, while N_k varies adaptively. For values of r_T closer to 1, we must scale down N_s, $N_{\min}$, and $N_{\max}$ to avoid N_{eval} from becoming too large.

N_s, the number of cycles between two step adjustments. This parameter controls how often there is a step adjustment. Therefore, if N_s is very low, step adjustments are performed very often and create convergence problems. On the other hand, if N_s is very high, there is a poor adaptation to the function behavior. From extensive tests on continuous functions, values between 10 and 20 were found to be good choices [11], and these were also the values used in the presented algorithm.

ϵ, N_ϵ, the parameters that control the termination test. N_ϵ has been kept equal to 4, a value found in exhaustive tests in continuous optimization [11]. ϵ is a classical "termination test" of optimization algorithms, and its value depends on the particular problem. In our opinion, a good practice is to set ϵ to a very low value (say: 10^{-6}–10^{-9}), and observe at which temperature there is no further improvement: this will be the appropriate value of $T_{\min}$.

$N_{\min}$, $N_{\max}$, the minimum and maximum number of step adjustments between two temperature reductions. We can say that $N_{\max}$ ($N_{\min}$) controls the number of function evaluations at temperatures where annealing (no annealing) takes place. In order to reduce the computation complexity both $N_{\min}$ and $N_{\max}$ should be as low as possible. However, at the same time we should avoid minimization getting trapped into a local minimum. After many tests, we set their values to $N_{\min}$ = 10–20, and $N_{\max}$ = 150–300, depending on the design problem. However, the value of these parameters is very problem dependent, and its final setting should be left to the user's experience in the particular problem.

A_1, A_2, the limit values for the computation of N_k. In our algorithm, $A_1 = 1$ and $A_2 = 6$. These values are the results of many observations of diagrams like that reported in Fig. 2, where A is seldom higher than 6 and the annealing starts taking place when A is approximately equal to 1.

N_r, the number of temperature reductions between two restarts from x_{opt}. This parameter is very critical. If N_r is set to 1, as in the continuous optimization [11], we observe that the search too often restarts from the same point, not allowing a proper exploration of the function domain. In this case, it is likely to be trapped in a local minimum. On the other hand, if N_r is too high, the information on the best point reached may be far away from the current point, and again, the probability to end the search in a local minimum is very high. We tried many values of N_r and found that in our case a good choice was 5. However, selection of this parameter is problem dependent.

III. Design Algorithm

To show the effectiveness and versatility of the proposed algorithm, three types of filter design are presented. All filters have power-of-two or sum of power-of-two coefficients. First, classical linear phase FIR filters, with specifications given by a mask in the frequency domain, will be designed. This case shows that the SA algorithm's performance compares favorably with other optimal methods [5], [9]. Second, in Section IV, we consider the design of FIR Nyquist filters, whose requirements are to have maximum stopband attenuation and minimum intersymbol interference [7], [8]. In the same section, we also consider the design of cascade form FIR filters composed of two stages [9].

A. Design of FIR Digital Filters

As mentioned in the introduction, when filter coefficient space is very coarse, determination of the optimum value of the filter gain G is very important [6].

If

$$\left\{h\left(-\frac{N-1}{2}\right), \cdots, h(0), \cdots h\left(\frac{N-1}{2}\right)\right\}$$

and G are, respectively, the impulse response and gain of a linear phase FIR filter of length N (odd), the corresponding vector $\boldsymbol{x}$ of independent variables is defined as follows:

$$x_1 = G$$

$$x_p = h(p-2), \qquad p = 2, \cdots, \frac{N+3}{2}. \tag{15}$$

As indicated by (3), the domain of $\boldsymbol{x}$ is continuous with respect to x_1 and discrete with respect to x_p, $p \geq 2$.

To appreciate the similarities and differences with respect to our design strategy, Zhao and Tadokoro's method [6] is briefly reported. First, they determine the value of gain G by minimization of

$$E(G) = \sum_{n=0}^{(N-1)/2} \{h_0(n) - \frac{1}{G} Q[Gh_0(n)]\}^2, \qquad G \in [0.5, 1] \tag{16}$$

where $\{h_0(n)\}$ are the filter coefficients calculated by the Remez exchange algorithm [16], and $Q[a]$ denotes the α_i in (2) closest to a. Once G has been determined by (16), a simple procedure is used to determine the filter coefficients (with a sum of power-of-two constraint) which minimize the maximum weighted frequency ripple. Since G is strictly related to the filter coefficients values, it is seen that this two step procedure yields a suboptimal value of G.

In our experience we obtained better results by replacing $E(\text{G})$ with

$$M(G) = \max_n \left| h_0(n) - \frac{1}{G} Q[Gh_0(n)] \right|, \qquad G \in [0.5, 1]. \tag{17}$$

We then determine values of G corresponding to some minimum values of $M(G)$. Small intervals centered around these values are used as domain I_G in the global optimization algorithm of Section II. Although the SA algorithm could theoretically solve the general problem with I_G equal to the entire interval [0.5, 1], in practice, in this case only a local minimum was usually found.

This fact deserves a comment. G is not a variable like the filter coefficients $\{h(n)\}$. Varying G means to change drastically $f(\cdot)$ and this affects all coefficients $\{h(n)\}$. In particular, the global minimum of $f(\cdot)$ varies, both as a point (the values of $\{h(n)\}$) and in the value. Therefore, letting G vary in the whole interval of [0.5, 1] yields very high perturbations of $f(\cdot)$ and our SA algorithm, which is necessarily limited in the total number of function evaluations, tends to get trapped in some local minimum. This problem could be solved only with an enormous increase in the computation cost.

Fig. 3 illustrates a typical behavior of $E(G)$ and $M(G)$ for a particular filter design. Note that although the two curves exhibit the same behavior, their local minima do not coincide. In the figure this is especially true for higher values of G.

The step-by-step design procedure is the following:

1) For a given order N, determine the infinite precision coefficients $\{h_0(n)\}$, $n = 0, \cdots, (N-1)/2$, by the Remez exchange algorithm [16].

2) By varying the filter gain G from 0.5 to 1 with steps of 0.001, determine corresponding $M(G)$. Store the value of G_1 (for which $M(G)$ is minimum) and the Z other best local minima G_m which have $M(G_m) < 5\,M(G_1)$, $(1 < m \le Z + 1)$.

3) For each G_m determined in step 2, run the SA algorithm by letting G vary in the interval $[G_m - 0.15, G_m + 0.15]$. Starting point for G is G_m and for the discrete variables is $\{Q[G_m h_0(n)]\}$, $n = 0, \cdots, (N-1)/2$. At each run, store the optimum value of $f(x)$, and the corresponding values of x (which includes G and the discrete variables $\{h(n)\}$). The global optimum is determined in correspondence with the minimum value of $f(x)$ over the various runs.

4) If the best filter does not satisfy the specifications, the whole procedure must be repeated with a higher value of N.

The cost function $f(\cdot)$ used in the minimization is the maximum weighted ripple in a finite number N_f of normalized frequencies (sampling frequency equal to 1) at the limits, and inside the stopband and the passband regions. More formally, let $\{\phi_k\}$, $k = 1, \cdots, N_f$, be the sequence of such frequencies, $T(\phi)$ the desired value of the frequency response, and $W(\phi)$ a weighting function, the formula yielding $f(x)$ is

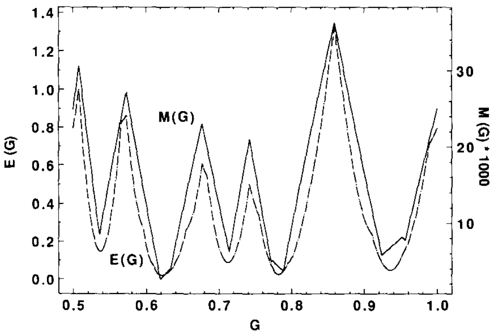


Fig. 3. Quantization error: mean square $E(G)$ in dashed line and maximum value $M(G)$ in solid line versus gain G.

$$f(x) = \max_{k=1,\cdots,N_f} \{ |W(\phi_k)[G^{-1}H(\phi_k) - T(\phi_k)]| \} \tag{18}$$

where $H(\phi)$ is the frequency response of the FIR filter and $G = x_1$. For a linear phase filter with order N (odd)

$$\begin{aligned} H(\phi) &= h(0) + \sum_{n=1}^{(N-1)/2} h(n) \cos(2\pi\phi n) \\ &= x_2 + \sum_{n=1}^{(N-1)/2} x_{n+2} \cos(2\pi\phi n) \end{aligned} \tag{19}$$

by using (15).

The value of N_f must be kept to a minimum, since it linearly influences the total computation cost of the algorithm, but it must be high enough to yield significant results. In our computations, generally $N_f = 3M$, however, the final reported results of $f(\cdot)$ are for a grid of 512 points.

It is worth noting that at each move of the proposed SA algorithm, only one filter coefficient $h(n_i)$ is changed. Consequently, the computation of the cost function in candidate points is made incrementally by changing in (19) only the term with $h(n_i)$ and thus saying a lot of computations.

B. Examples

For comparison purposes the above algorithm was implemented in C language on a SUN 3/60 workstation with the same filter designs of [5] and [9]. In all the following examples, the value of B in (1) is equal to 9, while the number of starting values of G in step 2 of global procedure is equal to 4.

The filter specifications are a low-pass filter with normalized passband and stopband edges of 0.15 and 0.25, respectively, and the weighting function equal to one. Let δ be the peak weighted ripple as determined by (18) and b the mean value of the passband gain [5], we report as performance criterion the normalized peak weighted ripple δ/b.

TABLE I
RESULTS OF 5 FILTER DESIGNS. NORMALIZED CUTOFF FREQUENCIES ARE 0.15 AND 0.25

(Starting Point)			(Optimum Point)		
Filter Length N	G_0	δ/b [dB] (in x_0)	G_{opt}	δ/b [dB] (in x_{opt})	Nr. Function Eval. $\times 10^{-3}$
27	0.925	−33.5	0.928	−41.3	849
29	0.924	−33.5	0.926	−43.1	939
31	0.924	−33.5	0.926	−43.1	1060
33	0.924	−35.7	0.927	−44.7	1026
35	0.925	−33.5	0.927	−44.7	1105

In Table I results of five filter designs are reported. For a given order N, among the various initial values of G, we only report that value (G_0) which yielded the final best performance. To show the improvement of using the optimization procedure with respect to rounding, we also report the initial normalized ripple as determined in step 3 of the general procedure. G_{opt} is the value of G in correspondence with the best δ/b found by using the SA algorithm. The computational cost, reported in the last column, is expressed by the total number of function evaluations.

Let us note that these results are almost identical (up to 0.5 dB) to those presented in [5] and [6], where an optimal integer programming algorithm has been used. In any case, they are better than the results obtained by using the Zhao and Tadokoro's suboptimal procedure [6].

The drawback of using SA is the computation time: it took one hour to compute a single run (see step 3) with $N = 27$, and the computation time increases as longer filters are considered. Furthermore, since SA is an optimization method that does not guarantee finding the global optimum, one cannot rely on a single run. The test cases reported in Table I were each run four times, starting with different seeds of the random number generator. However, only in the filter design with $N = 35$, was the global optimum not found in the first run. Again, more confidence in the results could be gained by increasing the values of the parameters introduced in Section II, with a consequent higher computation cost.

To emphasize the importance of the choice of G in the final performance, let us consider the filter design with $N = 31$ more thoroughly. Plots of $E(G)$ and $M(G)$, as defined by (16) and (17), respectively, are shown in Fig. 3. The global minimum of $M(G)$ in the range [0.5, 1] occurs at the values $G_1 = 0.620$, while the other three deepest local minima are located at $G_2 = 0.788$, $G_3 = 0.924$, and $G_4 = 0.714$. The results of running the SA algorithm, initiating from the four starting values of G, are reported in Table II. It is evident that the global minimum of $f(\boldsymbol{x})$ was obtained by starting neither from the global minimum of $M(G)$ nor from the second best minimum.

TABLE II
FILTER DESIGN WITH $N = 31$ STARTING FROM 4 DIFFERENT GAIN VALUES

G_0	$M(G_0)$ $\times 10^3$	δ/b [dB] (in x_0)	G_{opt}	δ/b [dB] (in x_{opt})
0.620	2.65	−36.3	0.624	−42.6
0.788	3.75	−34.7	0.794	−40.9
0.924	5.83	−33.5	0.926	−43.1
0.714	6.28	−31.1	0.699	−38.0

TABLE III
RESULTS OF 6 FILTER DESIGNS. NORMALIZED CUTOFF FREQUENCIES ARE 0.15 AND 0.22

(Starting Point)			(Optimum Point)		
Filter Length N	G_0	δ/b [dB] (in x_0)	G_{opt}	δ/b [dB] (in x_{opt})	Nr. Function Eval. $\times 10^{-3}$
31	0.844	−34.0	0.844	−39.7	795
33	0.844	−35.5	0.844	−39.7	868
35	0.844	−35.8	0.835	−41.0	946
37	0.845	−37.4	0.835	−41.5	995
39	0.844	−36.5	0.841	−41.6	1043
41	0.844	−38.9	0.840	−42.7	1168

In Table III we report additional examples of low-pass filter designs, varying N from 31 to 41, and with the band edge frequencies equal to 0.15 and 0.22, respectively. All other parameters are the same as for previous filters. Again, it is seen that the SA algorithm can achieve performance equal to other optimal methods [5], [6].

At this point one may wonder why we should use another design method, which requires quite a long computation time, when other well-established algorithms already exist? The answer may be that the SA algorithm is a general-purpose method, and finds its main application especially in the design of filters with special requirements; cases where other methods cannot be used. Two examples will be presented in the next section.

IV. DESIGN OF SPECIAL FILTERS

A. Linear Phase Nyquist Filters

These filters can be used in modem design [7] because they minimize intersymbol interference (ISI). If $\{h(n)\}$ is the impulse response (assume it is even with respect to the central coefficient $h(0)$), a requirement of such filters is to have minimum peak ISI, defined as [8]

$$\text{ISI} = \frac{2 \sum_{i=1}^{(N-1)/2K} |h(Ki)|}{|h(0)|} \tag{20}$$

where K is the data symbol duration in number of samples. Moreover, in order to minimize interchannel interference, the frequency response of these filters must have maximum attenuation in the stopband. We considered here only linear phase FIR filters.

The cost function which will be used in the minimization is the following:

$$F(\boldsymbol{h}) = \frac{\delta}{|h(0)|} + w\,\text{ISI} \tag{21}$$

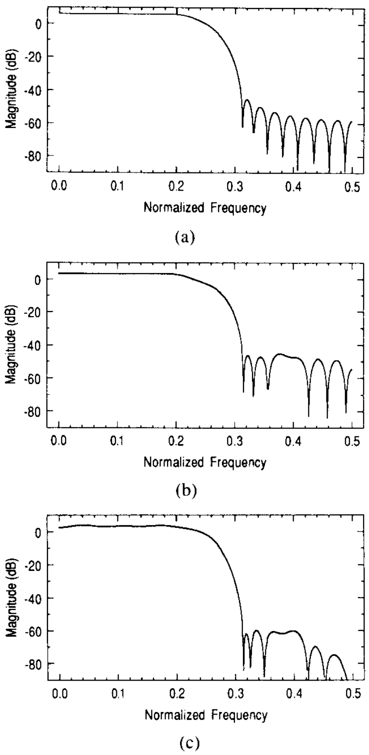
TABLE IV
STOPBAND RIPPLE $\delta/h(0)$ (IN DECIBELS) AND ISI (WITHIN PARENTHESIS) FOR NYQUIST FILTERS WITH SUM OF TWO POWER-OF-TWO COEFFICIENTS

	Filter Length		
Filter Type	27	31	35
Truncated raised-cosine filter	−38.5 (0)	−40.7 (0)	−45.3 (0)
Rounded to power-of-two coeff.	−36.0 (0)	−39.5 (0)	−36.0 (0)
Optimized with ISI = 0	−41.9 (0)	−42.3 (0)	−42.8 (0)
Optimized with $w = 0.05$	−50.4 (0.027)	−53.2 (0.038)	−57.3 (0.057)

where δ is the stopband ripple and ISI is weighted by parameter w. With respect to the algorithm of Section II, this optimization involves only the filter coefficients $\{h(n)\}$, whose values must be discrete (sum of two power-of-two), discarding the optimization with respect to the continuous gain G. To summarize, the filter requirements are linear phase, filter coefficients of the form sum of power of two, and tradeoff between stopband ripple and peak ISI. The starting point of the minimization was the rounded truncated infinite precision impulse response given by a raised-cosine filter [9].

In Table IV we report results concerning three cases of filter designs, each of different length, and the following parameters: oversampling factor K equal to 2, and a fractional excess bandwidth (i.e., rolloff factor) of 0.25. As a consequence, the normalized stopband edge is 0.3125. The choice of K was dictated by requiring a minimum filter length N. For each case we consider 1) the truncated infinite precision raised-cosine filter, 2) the same filter with coefficients rounded to a difference or sum of power of two (the starting point of the optimization), 3) the optimized sum of two power-of-two coefficients FIR filter with ISI = 0, and 4) the optimized sum of power-of-two coefficients FIR filter with $w = 0.05$ (i.e., ISI $\neq$ 0). The latter two filters are designed with the proposed SA algorithm. For filters of length 35, Fig. 4 shows the corresponding frequency responses, while Table V reports the coefficients' values of the optimum filters with $N = 35$.

It may be interesting to consider the tradeoff between the weighted stopband ripple $\delta/|h(0)|$ and ISI, which is reported in Fig. 5 for a given filter length $N = 31$. These points have been obtained by optimizing functional (21) for seven values of w and performing five runs for each w. For a given ISI, only the lowest value of $\delta/|h(0)|$ has been reported. It is seen that by allowing ISI to increase up to 5% , the stopband ripple can be lowered to more than 12 dB.



(a)

(b)

(c)

Fig. 4. Frequency responses of Nyquist filters of length 35 and power-of-two coefficients obtained by (a) rounding a truncated raised-cosine function, (b) using the SA method with ISI = 0, and (c) using the SA method with $w = 0.05$.

TABLE V
IMPULSE RESPONSE, $[h(n)]$, $n = 0, 1, \cdots, (N-1)/2$, OF OPTIMIZED NYQUIST FILTERS ($N = 35$) WHOSE FREQUENCY RESPONSES ARE REPORTED IN FIG. 4

ISI = 0:	ISI = 0.057:
7.5000000e − 01	7.50000000e − 01
4.6875000e − 01	4.68750000e − 01
0.0000000e + 00	−3.90625000e − 03
1.3281250e − 01	−1.40625000e − 01
0.0000000e + 00	1.95312500e − 03
5.4687500e − 02	6.44531250e − 02
0.0000000e + 00	−3.90625000e − 03
−1.9531250e − 02	−3.22265625e − 02
0.0000000e + 00	−9.76562500e − 04
2.9296875e − 03	2.92968750e − 03
0.0000000e + 00	−1.46484375e − 02
5.8593750e − 03	−8.78906250e − 03
0.0000000e + 00	9.76562500e − 04
−7.8125000e − 03	−8.78906250e − 03
0.0000000e + 00	−1.66015625e − 02
4.8828125e − 03	−8.78906250e − 03
0.0000000e + 00	0.00000000e + 00
−2.9296875e − 03	9.76562500e − 04

B. Cascade Form FIR Filters

We conclude this section with the design of cascade form FIR filters whose performance is far superior to direct form filters [5,], [9]. The previous design strategy for these structures was to design one stage at a time iteratively. In our case, by using the SA algorithm, we are able to design all stages at the same time. The result is improved filter performance. In particular we compared results of our global filter design method with [9, fig. 6]. We should remark that in this example each coefficient is simply a power of two.

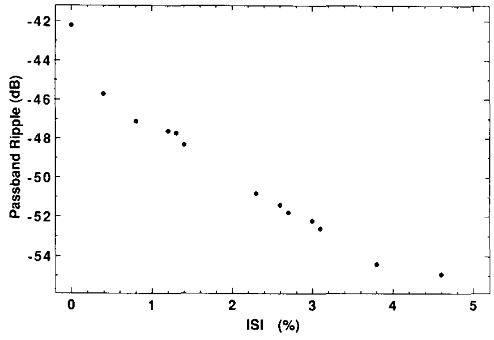


Fig. 5. The tradeoff between $\delta/|h(0)|$ and ISI for a Nyquist filter of length $N = 31$.

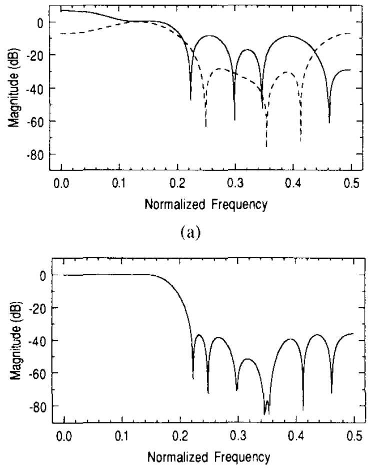
TABLE VI
PERFORMANCE COMPARISON BETWEEN TWO METHODS ON DESIGNING A DIRECT FORM FILTER WITH POWER-OF-TWO COEFFICIENTS

Filter Length N	δ/b (dB)	G_{opt}	(From [9, fig. 6]) δ/b (dB)
21	−17.6	0.844	−17.4
23	−19.0	0.841	−18.5
25	−19.0	0.841	−18.7
27	−19.8	0.850	−19.4
29	−20.7	0.845	−20.5
31	−20.9	0.854	−20.5

TABLE VII
PERFORMANCE COMPARISON BETWEEN TWO METHODS ON DESIGNING A CASCADE FILTER WITH POWER-OF-TWO COEFFICIENTS

Global Filter Length N	Stage 1-2 Filter Length N_1, N_2	δ/b (dB)	(From [9, fig. 6]) δ/b (dB)
21	11, 11	−24.8	−23.3
23	13, 11	−28.3	−24.7
25	13, 13	−29.1	−27.2
27	15, 13	−29.2	−27.7
29	15, 15	−30.4	−29.5
31	17, 15	−32.2	−32.8
33	17, 17	−35.9	—

On designing a low-pass filter with passband and stopband edges of 0.15 and 0.22, the results of the direct form configuration are presented in Table VI where, for various filter lengths N, the optimum value of δ/b and corresponding G_{opt} are reported. Data from [9, fig. 6] are reported in the last column. We can say that for a direct form the two methods yield similar results. This was also Section III's conclusion. However, when we consider a cascade form filter configuration with two stages, the iterative procedure of [9] provides inferior results, as shown in Table VII. Indeed, up to 3.5 dB improvement (the case for $N = 23$) was obtained by using the global optimization SA algorithm. Incidentally, these results were obtained by setting the filter gain G to one, because we did not notice any improvement by keeping it variable. Fig. 6 shows the frequency responses of the case for $N = 33$.



(a)

(b)

Fig. 6. Frequency responses of a cascade form configuration with $N = 33$: (a) first and second stage, and (b) overall filter.

V. CONCLUSIONS

A simulated annealing algorithm for the design of digital filters with multiple constraints has been presented. This algorithm yields good results in applications where previous approaches use heuristics to find a solution. Three examples of filter designs have been given to illustrate possible applications. The drawback of the proposed algorithm is its high computation time, especially for higher order filters. Furthermore, many runs of the same filter design have to be made in order to "trust" the solution found.

In fact, this method does not guarantee finding the global optimum, as do all general purpose nonlinear global optimization algorithms. In our experience, the SA algorithm yielded the same solution (optimal?) in most runs of the same filter design. However, if the number of filter coefficients N is higher than 39, it started providing many local-optimal solutions. Therefore, many more runs were needed to find a good solution.

REFERENCES

[1] D. M. Kodek, "Design of optimal finite wordlength FIR digital filters using integer programming techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 304-308, June 1980.

[2] V. B. Lawrence and A. C. Salazar, "Finite precision design of linear-phase FIR filters," *Bell Syst. Tech. J.*, vol. 59, pp. 1575-1598, Nov. 1980.

[3] Y. C. Lim, S. R. Parker, and A. G. Costantinides, "Finite word length FIR filter design using integer programming over a discrete coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 661-664, Aug. 1982.

[4] Y. C. Lim and S. R. Parker, "Discrete coefficients FIR digital filter design upon a LMS criteria," *IEEE Trans. Circuits Syst.*, vol. CAS-30, pp. 723-739, Oct. 1983.

[5] Y. C. Lim and S. R. Parker, "FIR filter design over a discrete powers-of-two coefficient space," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 583-591, June 1983.

[6] Q. F. Zhao and Y. Tadokoro, "A simple design of FIR filters with powers-of-two coefficients," *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 566-570, May 1988.

[7] H. Baher and J. T. Coffey, "On the design of digital Nyquist channel filters," *IEEE Trans. Circuits Syst.*, vol. CAS-33, pp. 460–462, Apr. 1986.

[8] L. Lo Presti, "FIR design of raised-cosine filters," in *Proc. IEEE Int. Symp. Circuits Syst.* (Helsinki, Finland), 1988, pp. 146–149.

[9] Y. C. Lim and B. Liu, "Design of cascade form FIR filters with discrete valued coefficients," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 36, pp. 1735–1739, Nov. 1988.

[10] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, May 1983.

[11] A. Corana, M. Marchesi, C. Martini, and S. Ridella, "Minimizing multimodal functions of continuous variables with the simulated annealing algorithm," *ACM Trans. Math. Software*, vol. 13, pp. 262–280, Sept. 1987.

[12] E. J. Diethorn and D. C. Munson, Jr., "Finite word length FIR digital filter design using simulated annealing," in *Proc. IEEE Int. Symp. Circuits Syst.* (San Jose, CA), May 1986.

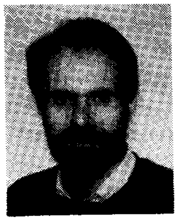
[13] F. Catthoor, H. De Man, and J. Vandewalle, "Simulated-annealing-based optimization of coefficient and data word-lengths in digital filters," *Int. J. Circuit Theory Appl.*, vol. 16, pp. 371–390, 1988.

[14] N. Benvenuto and M. Marchesi, "Digital filters design by simulated annealing," *IEEE Trans. Circuits Syst.*, vol. 36, pp. 459–460, Mar. 1989.

[15] R. V. Kacelenga, P. J. Graumann, and L. E. Turner, "Design of digital filters using simulated annealing," in *Proc. IEEE Int. Symp. Circuits Syst.* (New Orleans, LA), May 1990, pp. 642–645.

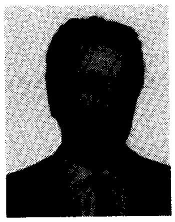
[16] J. H. McClellan, T. W. Parks, and L. R. Rabiner, "A computer program for designing optimum FIR linear-phase digital filter," *IEEE Trans. Audio Electroacoust.*, vol. AU-21, pp. 506–526, Dec. 1973.

[17] S. R. White, "Concepts of scale in simulated annealing," in *Proc. IEEE ICCD'84* (New York, NY), Oct. 1984, pp. 646–651.



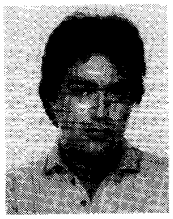
Nevio Benvenuto (S'81–M'82–SM'88) received the Laurea degree in electrical engineering from the University of Padova, Italy, in 1976, and the Ph.D. degree in electrical engineering from the University of Massachusetts, Amherst, in 1983.

From 1983 to 1985 he was with AT&T Bell Laboratories, Holmdel, NJ, working on signal analysis problems. He spent the next three years alternating between the University of Padova, where he worked on communication systems research, and Bell Laboratories as a Visiting Professor. Since 1990 he has been an Associate Professor of electrical engineering at the University of Padova, having been on the faculty of the University of Ancona, Italy, from 1987 to 1990. His research interests are in the areas of voice and data communications, filter design, and signal processing.



Michele Marchesi received the Laurea degree from the University of Genova, Italy, in 1975 in electronic engineering, and in 1980 in applied mathematics.

From 1976 to 1987 he was a Researcher at the Institute for Electronic Circuits, C.N.R., Genova. From 1987 to 1990 he was Associate Professor at the University of Ancona, Italy. Since November 1990, he has been Associate Professor of network theory at the University of Genova, in its Department of Biophysical and Electronic Engineering. His main fields of interest are optimization techniques applied to circuit design and neural networks.



Aurelio Uncini (M'88) was born in Cupra Montana, Italy, in 1958. He received the Laurea degree in electrical engineering from the University of Ancona, Italy, in 1983.

From 1984 to 1986 he was with Fondazione Ugo Bordoni, Rome, Italy, engaged in research on digital speech processing. Since 1987 he has been a Researcher associated with the Department of Electronics and Automatics, University of Ancona. His research interests include digital signal processing, digital circuit theory, and neural networks.