Probability-driven simulated annealing for optimizing digital FIR filters

E. Boutillon, C. Roland, and M. Sevaux

Université Européenne de Bretagne
UBS - LESTER - Centre de Recherche
F-56321 Lorient – France
marc.sevaux@univ-ubs.fr

Summary. In this paper, we propose to mimic some well-known methods of control theory to automatically fix the parameters of a multi-objective Simulated Annealing (SA) method. Our objective is to allow a decision maker to efficiently use advanced operation research techniques without a deep knowledge of this domain. Classical SA controls the probability of acceptance using an a priori temperature scheduling (Temperature Driven SA, or TD-SA). In this paper, we simply propose to control the temperature using an a priori probability of acceptance scheduling (Probability Driven SA, or PD-SA). As an example, we present an application of signal processing and particularly the design of digital Finite Impulse Response (FIR) filters for very high speed applications. The optimization process of a FIR filter generally trades-off two metrics. The first metric is the quality of its spectral response (measured as a distance between the ideal filter and the real one). The second metric is the hardware cost of the filter. Thus, a Pareto-based approach obtained by a multi-objective simulated annealing is well suited for the decision maker. In this context, TD-SA and PD-SA method are compared. They show no significant differences in terms of performance. But, while TD-SA requires numerous attempts to set an efficient temperature scheduling, PD-SA leads directly to a good solution.

Keywords: *filter design, FIR, simulated annealing, multiobjective optimization, temperature regulation, feedback loop.*

1 Introduction

Implementing a Simulated Annealing (SA) algorithm is quite an easy task and should be done in a few hours. But tuning the parameters for having good and interesting results is much more difficult. Most of the time, based on a set of instances (sometimes with known results), the parameters, one by one, are changed and set to their best values. Of course, interaction between the different parameters complicates the task.

What motivates this work is to let a decision maker (who often is not a specialist in optimization, and even less in tuning SA parameters) use the solver with a minimum number of comprehensive parameters. To achieve this goal, we try to translate the classical SA parameters to what could be easily understood by the decision maker: a probability function and a number of iterations (a total running time).

In this paper, we consider that the temperature is controlled by a feedback loop. The feedback is given by the difference between the estimated probability of acceptance at a given iteration number and the desire probability of acceptance at this moment. This technique is applied on a signal processing problem: the joint optimization (i.e. multiobjective function) of the performance of a numerical Finite Impulse Response (FIR) filter and its related hardware complexity. Related work is presented in [4]. Note that the FIR filter is one of the key tools of the signal processing domain. The domain of application of FIR filter is thus very large (radar, sonar, communication, sysmography, ...).

The rest of the paper is divided in five sections. Section 2 describes the problem of FIR filter design and the relative metrics associated to the FIR filter performances and its hardware cost. Section 3 proposes a literature review of known works in the same area, followed by the proposed approach in section 4. Numerical experiments are conducted in section 5 before a conclusion in the last section.

2 The digital FIR filter problem design

This section presents the problem of digital FIR filter design for a high speed dedicated architecture. After recalling the definition of a FIR filter, the classical design flow is given. Then, an alternative method is proposed and the cost function of performances and complexity are presented. General information can be found in [8, 12, 13].

2.1 Definition of a FIR filter

A FIR filter is a common tool in signal processing. The input signal of a FIR filter is a numerical series (typically, the samples of a captor) $x(n)$ indexed by an integer n. Generally, the signal of interest is corrupted by noise or other non significant signals. The FIR filter processes the input signal $x(n)$ and generates a filtered output signal $y(n)$ that rejects part of the jamming signal and noise. A FIR filter of order N is characterized by its finite impulse response (FIR) of length N given by $H = (h(0), h(1), ..., h(N-1)\ 1$. The output $y(n)$ at time n of the filter H is given by the equation:

$$y(n) = \sum_{k=0}^{N-1} h(k)x(n-k) \tag{1}$$

This operation is noted $y(n) = h(n) * x(n)$, where $*$ stands for convolution.

The coefficients H of the filter are invariant over time and identical to the impulse response of the filter (see Figure 1-a).

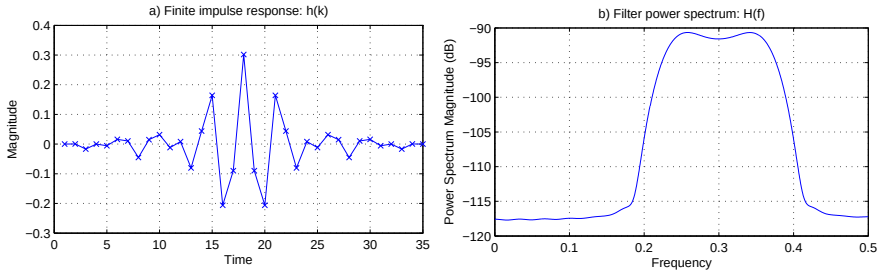

Fig. 1. Representation of a FIR filter

In signal processing theory, filters are characterized by their frequency response. The frequency response $H(f)$ is obtained with the Fourier Transform (FT) of the finite impulse response (see Figure 1-b). In the sequels, the phase response will not be considered and we only get focused on the amplitude response (i.e. $|H(f)|$) of the FIR filter (the majority of FIR application in signal processing).

2.2 The problem of FIR filter synthesis

The classical process of FIR synthesis is divided in 3 steps: first, the ideal filter is defined according to the spectral characteristic of the signal and the target of the application. In general, this filter has cliff transition and this results in an infinite impulse response. In order to obtain an implementable filter, the filter constraints are relaxed and the template of an acceptable filter is defined. For example, a template of band-pass filter is defined by several parameters: the bounds of the passband frequencies (fi_1, fi_2), the absolute value of the maximum gain in the passband frequencies, the size of the transition bands $(f_1, fi_1$ and $f_2, fi_2)$ and the rejection factor in the rejection band. Va_2 (resp. Va_3) is the maximum (resp. minimum) level for the passband. Va_1 is the maximum level for the rejection band (see Figure 2).

Given a template, the generation of H can be obtained by several methods: the Hamming method, the Hanning method, the Remez method, the Kaiser method and the window method to cite some of the most popular[1] [8].

All those methods provide real values of H. The next step is then to represent the real value in a fix precision format for the implementation. This task can be tricky because quantization impacts on both performance and

[1] These methods are all available in Matlab.

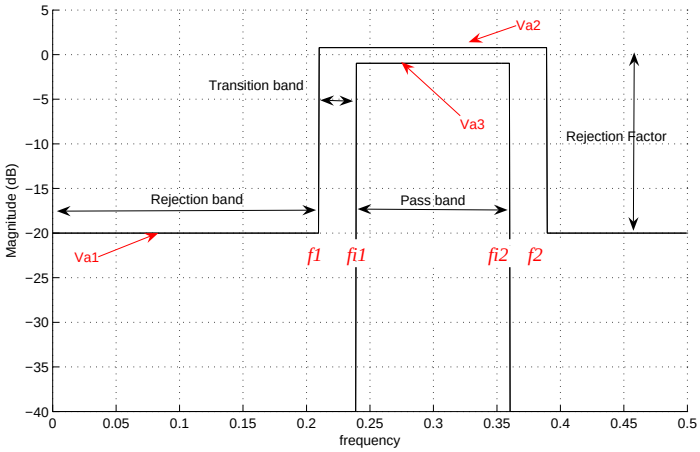

Fig. 2. Template of a FIR filter

hardware complexity. Some papers work research an optimal design of FIR with only one constraint. The effort is concentrated on the quantization of the filter coefficients to obtain H_q, the quantized impulse response (the value of H_q are then integer). In the general case, the authors have one objective: how to limit the degradation of the frequency performance between the real and the quantized FIR filter.

2.3 Proposed FIR designs

In this paper, we propose to proceed in a different way than the straightforward method defined above. The idea is to start from a classical method to generate H, then quantize H to obtain H_q^{init} and then, directly optimize H_q^{init} by considering jointly the performance and the complexity of the filter. This approach is quite new and few recent papers proposed to perform FIR design in a similar way [2,7,10,11,14]. Note that except for [14], all those papers are single objective optimization methods, and the bi-objective proposal concerns different criteria using genetic algorithms.

Those techniques require two types of metrics: a measure of the performance of the filter and a measure of the hardware complexity of the filter. Since the details of FIR design problem is out of the scope of the paper, the next two sections describe briefly the algorithm that will be used to compute those two metrics.

2.4 Performance measures

The cost function C between the template $G(f)$ and the actual filter $H_q(f)$ defines the "quality" of the design. Ideally, when $C(H_q, G) = 0$, then $H_q(f)$

is inside the template for all frequencies of the application. In the following, we will consider that, for a given frequency f, if $H(f)$ is inside the template, then $C(H_q(f), G(f)) = 0$, otherwise, $C(H_q(f), G(f))$ is equal to a weighted distance of the actual response $H_q(f)$ and the closest limit of the template for the frequency f.

In practice, the cost function of the template is given by the summation of $C(H_q(f_k), G(f_k))$, $f_k = k/N_{fft}$ for $f = 0 \cdots N_{fft} - 1$. The N_{fft} value of $H_q(f_k)$ are obtained by a Fast Fourier Transform of H_q on N_{fft} points. The details on the computation of the distance is provided in Algorithm 1.

Algorithm 1: Template cost function

initialization step:
  fix parameters cost function (N, N_{fft}, Va_1, Va_2 and Va_3)
  fix band-limited and stop-band frequencies (f_1, f_2, fi_1 and fi_2) `// (see Figure 2 for some of these parameters)`
find a new solution h_i with SA algorithm
calculate: $H = log|fft(h_i)|$
$d_1 = d_2 = d_3 = d_4 = 0$
i = 1
repeat
  if $H(i) > Va_1$ **then**
    $d_1 = d_1 + (H(i) - Va_1)^\alpha$
  endif
  i = i+1
until $i < f_1$
repeat
  if $H(i) > Va_2$ **then**
    $d_2 = d_2 + (H(i) - Va_2)^\alpha$
  endif
  if $i > fi_1$ *and* $i < fi_2$ *and* $H(i) < Va_3$ **then**
    $d_3 = d_3 + (Va_3 - H(i))^\alpha$
  endif
  i = i+1
until $i < f_2$
repeat
  if $H(i) > Va_1$ **then**
    $d_4 = d_4 + (H(i) - Va_1)^\alpha$
  endif
  i = i+1
until $i < fft\_size/2$
$d = d_1 + d_2 + d_3 + d_4$

2.5 Hardware complexity of a filter $H_q(z)$

In this paper, we assume that the hardware is dedicated to the filter and that the input rate of the filter is equal to the clock frequency of the hardware. In other words, an output sample is computed every clock cycle. The architecture requires N hardware multipliers, each one dedicated to a specific multiplication with a fix constant (i.e. $h_q(k)$ for the k^{th} multiplier). The architecture of the FIR filter is presented Figure 3.

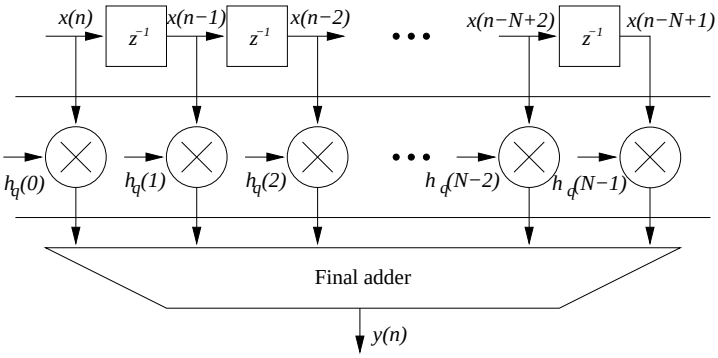

Fig. 3. Architecture of a FIR filter

Roughly speaking, the complexity of the architecture is composed of three terms, the first cost is related to the register required to store the N previous received input samples $(x(n-k))_{k=0,\ldots,N-1}$, the second term is the total cost of the N multipliers and the last term is the cost of the final adder to sum the N partial results $(h_q(k)x_{n-k})_{k=0,\ldots,N-1}$. The first and third terms can be assumed independent of the implementation (it is a constant cost). The second term depends on the implementation choice and on the value of the coefficient $h_q(k)_{k=0,\ldots,N-1}$ (variable term). This third term alone will be used as a metric for the hardware complexity of the design. The hardware cost depends on the binary representation of the coefficients. The exact formulation of the hardware cost is out of the scope of then paper but is non-linear. To give a brief idea, a multiplication by 31, 32, 33 has a complexity of 5, 1, and 2 respectively, *i.e.* the number of non-zero elements in the binary representation. Using the Canonical Signed Digit representation (a more efficient representation of number, where, for example, 31 is coded as $32-1$ [7]), the hardware cost function can be computed by Algorithm 2.

In the following sections, we consider the cost function C^{csd}, defined as $C^{csd} = \sum_{k=0}^{N-1} C^{csd}_{h_q(k)}$, where $C^{csd}_{h_q(k)}$ is the number of non zero value of the coefficient $h_q(k)$ in the CSD representation. This number is given by Algorithm 2 where C is the CSD decomposition of x and C^{csd}_x the number of non-zero values of C.

Algorithm 2: Complexity (CSD decomposition of h_q)

```
initialization step:
    C^csd = 0
    Nb = 12 // Number of bits for quantization
for i = 1 to length(h_q) do
    c = 0
    x = |h_q(i)|
    for k = Nb-1 to 1 (step -1) do
        M = 2^(k-1)
        if x > M/2 then
            x = M - x
            c = c + 1
        endif
    endfor
    C^csd = C^csd + c
endfor
return C^csd
```

3 Description of the proposed approach

The method itself is rather simple. Everything is based on the multiobjective simulated annealing scheme where the temperature is controlled by a feedback loop.

3.1 Multiobjective optimization by Temperature Driven Simulated Annealing (TD-SA)

We briefly recall in this section the multiobjective optimization framework using a simulated annealing algorithm.

When m objectives $f_i,\ i \in [1, m]$ are simultaneously considered for minimization, we need to define the concept of Pareto dominance. Instead of giving an absolute value for a solution, a partial order is defined based on dominance. A solution is said to dominate another solution when it is better on one objective, and not worse on the other objectives. Thus a solution x dominates a solution y if and only if $\exists i \in [1, m] : f_i(x) < f_i(y)$ and $\forall j \in [1, m], j \neq i : f_j(x) \leq f_j(y)$. A solution is said to be non-dominated if no solution can be found that dominates it.

The definition of the dominance relation gives rise to the definition of the Pareto optimal set, also called the set of non-dominated solutions. This set contains all solutions that balance the objectives in a unique and optimal way. The aim of multi-objective optimization is to induce this entire set. Picking a single solution from this set is then an a posteriori judgement, which can be done in terms of concrete solutions with concrete trade-offs, rather than

in terms of possible weightings of objectives. The question for multiobjective optimization is now how to find this Pareto optimal set.

We recall here that our objective is not to design the best possible multiobjective algorithm for solving the filter design problem but to propose an easy implementable solution.

Based on previous asumptions, we use the work of Nam and Park [9] and the multiobjective algorithm will be based on simulated annealing. The literature on this topic is important. A good introduction on evolutionary algorithms for multiobjective optimization can be found in [3, 5].

Algorithm 3: Basic multiobjective simulated annealing

```
1  initialization step:
2    find an initial solution x
3    fix an annealing schedule 𝒯
4    set initial temperature T = T_0
5  repeat
6  |   neighborhood search:
7  |     find a solution x' ∈ 𝒩(x)
8  |   if x does not dominates x' then
9  |   |   x' ← x
10 |   else
11 |   |   determine ΔC (the variation of the cost function)
12 |   |   draw p ~ 𝒰(0,1)
13 |   |   if e^{-ΔC/T} > p then
14 |   |   |   x' ← x
15 |   |   endif
16 |   endif
17 |   update temperature T according to 𝒯
18 until stopping criterion satisfied
```

Algorithm 3 presents a basic multiobjective simulated annealing framework. In the *initialization step*, as in all neighborhood search metaheuristics, an initial solution should be provided. Fixing the annealing schedule (Alg 3, line 3) and setting the initial temperature (Alg 3, line 4) are two empiric tasks that partially motivate the work presented in this paper. For general explanations on standard settings, we refer the reader to the original paper [9]

The general loop (Alg 3, lines 5-18), as in all metaheuristic algorithms, does not differ much from a single objective simulated annealing. First a neighbor solution x' of current solution x is randomly generated. In single objective optimization, if the new solution x' is better than x, it is accepted as the new current solution. In the multiobjective case, it is accepted if x' is not dominated by x (Alg 3, line 8), which means if it is not worse than the current solution.

Now, when x' is dominated by x, as in the classical simulated annealing algorithm, x' can become the new current solution (in order to escape local optima) under a probability condition. In [9], the authors call it *probability transition* and expose six different criteria. We use the random cost criterion. ΔC (Alg 3, line 11) is computed by Equation 2.

$$\Delta C = \sum_{j=1}^{m} \beta_j (f_j(x') - f_j(x)) \tag{2}$$

where β_j is a random variable with uniform probability distribution. Acceptance probability is given by the so-called Boltzmann's equation (see [1] for more information).

At the end of the loop iteration, the temperature is updated according to the cooling schedule. Classical cooling schedules refer to geometric evolution of the temperature $T_k = \alpha^k T_0$, where T_k is the temperature at iteration k, α is the cooling rate ($0 < \alpha < 1$) and T_0 the initial temperature.

3.2 Parameter reduction in SA – from TD-SA to PD-SA

In the description of the TD-SA (Algorithm 3), the following parameters have to be set by the decision maker or by the end-user of the solver :

- the initial solution x
- the annealing schedule $\mathcal{T}$
- the initial temperature T_0
- the stopping conditions

The initial solution x can be generated randomly or provided by the decision maker (from previous runs of the solver, or from experience). If the annealing schedule follows the description of the geometric evolution of the previous section, parameter α should be provided, as well as the initial temperature T_0 and the stopping conditions of the algorithm.

The stopping conditions are easy to set. Usually, a running time is wished by the decision maker and is converted into a maximum number of iterations. But the initial temperature T_0 and the cooling rate α requires several attempts for a good setting. Moreover, this setting should be done again when the set of instances changes.

To avoid this, we propose to replace the annealing schedule by a function representing at each iteration the probability of accepting a non-improving move. This new method will be called Probability-Driven Simulated Annealing (PD-SA).

Figure 4 draws such a simple potential function. In a solver with a graphical user interface, the function could be chosen from a library. In our case, we use $p(x) = P_0 \times (1 - 5/\texttt{ItMax})^x$ as a sample function. P_0 is the probability of accepting a non-improving solution at the beginning of the search. P_0 and `ItMax` are incorporated into the library (in Figure 4, $P_0 = 0.3$, and `ItMax` =

100). Note that the value of $P_0 = 0.3$ can be fixed for once, thus, the only parameter of the system simply becomes `ItMax`.

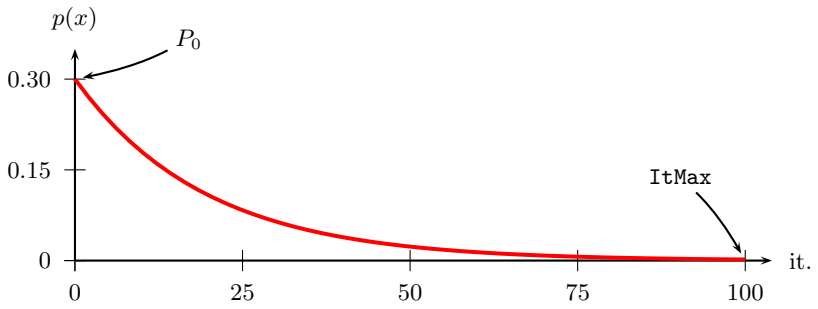

Fig. 4. A potential probability function

A decision maker is able to fully understand the purpose of this function and its two parameters. In order not to change the simulated annealing algorithm, we will transform the current probability (the one at the current iteration) into a temperature for the SA algorithm (see next section). For the classical SA algorithm, T_0 can be reversely computed from P_0 with the first non-improving move of the algorithm ($P_0 = e^{-\Delta C/T_0} \Leftrightarrow T_0 = -\Delta C/ln(P_0)$).

3.3 Controlling the temperature parameter in PD-SA

With the probability scheme proposed in the previous section, the major question coming is "why keeping a temperature-based system if we know in advance the probability for accepting non-improving moves?". First, removing the temperature will remove a degree of freedom in the SA approach, second, removing the temperature will change the SA scheme and will need partial re-writing of the program. Instead, we will use a feedback loop as in automatic control to keep a temperature as close as possible of the desired temperature (*i.e.*, of the probability value provided by the probability function).

In a classical TD-SA scheme, each time a new generated solution has an objective function value worse than the current solution (i.e. a cost function $C > 0$), this degrading solution is accepted if the value of $e^{-\Delta C/T}$ is greater than an uniform random value between 0 and 1. The event of accepting a degrading solution as exactly a probability of $\tilde{p} = e^{-\Delta C/T}$ to occur.

In the new PD-SA approach, we keep the same process to accept or not a degrading solution. The main difference is that, instead of cooling blindly the temperature in a deterministic way ($T(i+1) = \alpha \cdot T(i)$), we try to control the temperature T_c so that $\tilde{p}(i) = e^{-\Delta C/T_c}$ equals exactly $p(i)$ the desire probability of acceptance at iteration number i. Thus, each time a non-improving solution is generated, 3 cases can occur:

1. $\tilde{p}(i) > p(i)$, than the probability of acceptance is too high and T_c should be decreased.
2. $\tilde{p}(i) = p(i)$, than the probability of acceptance is good and T_c is correct.
3. $\tilde{p}(i) < p(i)$, than the probability of acceptance is too low and T_c should be increased.

To perform such an update, the temperature is adjusted using a feedback loop as:

$$T_c = T_c + T_c\epsilon(p(i) - \tilde{p}(i)) \tag{3}$$

where ϵ is a parameter of the feedback loop that weights the correction factor (ϵ is set to 1 in our simulation).

One can note that this feedback loop is similar to the Proportional Integral (PI) corrector [6] if we consider T_c in the log domain. In fact:

$$log(T_c) = log(T_c) + log(1 + \epsilon(p(i) - \tilde{p}(i)) \tag{4}$$

and thus:

$$log(T_c) \approx log(T_c) + \epsilon(p(i) - \tilde{p}(i)) \tag{5}$$

The PI corrector is well known in automatic control system to be a very robust corrector, with no bias and generally stable for a large range of values of ϵ. Performing this kind of feedback loop in unusual in metaheuristics but its efficiency has been proved in automatic control systems since a very long time.

One can note an advantage of this type of feedback control compared to the classical cooling temperature scheme. In fact, if the local solution is a local minima and if the current temperature is too low, then the SA algorithm is trapped in this local minima until the end of the SA process. On the contrary, with the feedback loop, in such a situation, the temperature will increase in a geometrical way so that, at one moment, it will escape this local minima and restart a worthwhile research process.

Numerical experiments will show the difference between PD-SA and TD-SA.

4 Numerical experiments

This section presents the comparison between TD-SA and PD-SA. The parameters of the SA algorithms are first presented (coding, neighborhood and initial solution). Then computational results are presented and discussed.

4.1 Coding, neighborhood and initial solution

A solution is represented by 33 integer values (stored in an array, $s[i], i = 1, \ldots, 33$). Each value belongs to the range $[-1023, +1023]$, corresponding

to the physical size of the component (a power of 2). These values are the coefficients of the filter *i.e.*, the impulse response of the filter. To obtain the frequency response of the filter, it is necessary to use a FFT. The impulse response is symmetric so the values in the array are also symmetric ($s[i] = s[33 - i + 1]$) and the sum of coefficients should be constant (the square root of the energy remains the same along the search).

The neighborhood is defined so that we respect the constraints on the encoding. To move from one solution to a neighbor, one of the coefficients is either increased by one or two or decreased. The symmetric value is changed accordingly. To keep the constant sum property, another symmetric pair of coefficients is inversely modified.

An initial solution for the proposed method is obtained from classical design filters, *e.g.* Hanning window multiply iFFT of ideal filter. This solution is usually blindly used by practitioners for designing filters. By doing so, we ensure the designer to have at least a feasible and "classical" solution.

4.2 Computational results

Numerical experiments have been conducted in order to assert that the proposed method is at least as good as a classical simulated annealing algorithm. To do so, several figures are presented and commented below.

Even if the general purpose of the algorithm is to reduce the number of parameters, some of them are necessary. First, the number of coefficients of the filter –sometimes considered as a meta-parameter– is left to the decision maker. For the experiments, we set it to the most commonly used value, *i.e.* 33 coefficients (*e.g.* like for a standard FIR1 function). Second, the size of the FFT is usually set to 2048 values. Then, several parameters presented in the cost function (see parameters Va_1, Va_2, Va_3 in Algorithm 1) and in the design of the template (see parameters f_1, f_2, fi_1, fi_2 in Figure 2) have to be chosen by the decision maker. In figure 2, some of them depend on the final application. We propose in the future to develop a graphical user interface that will help the designer to set these parameters. When it is not explicitly mentioned, the maximum number of iterations is 50 000 corresponding to a reasonable amount of time (5 minutes).

For all the experiments, we try to compare the execution of the standard simulated annealing algorithm and the version with automatic temperature setting. On one hand, simulation operates with an initial temperature T_0, and on the other hand, the parameter used at the beginning of the search is P_0, the initial probability of accepting non-improving moves. As an example, we auto-magically set $T_0 = 30$ and arbitrarily set $P_0 = 0.3$.

First we compare the influence of the maximum number of iterations. Figure 5 compares the Pareto solutions of the two approaches obtained after different maximum number of iterations. These Pareto solutions are the best solutions over 250 different runs for each approach.

In these figures, Pareto solutions are presented for the two solution techniques, TD-SA and PD-SA when $T_0 = 30$. The initial solution blindly used by decision makers is also noted as "Initial".

Figure 5 shows just 3 particular cases of the Pareto sets obtained with the TD-SA and the proposed PD-SA. It did not give any insight of which technique is more efficient. In order to obtain an objective comparison between the two methods, we perform a series of "match racing" between TD-SA and PD-SA. Each match racing starts from the same initial condition and process the same number of iteration. Once the two Pareto function are obtained, they are merge to create a new Pareto function. Let N_{PD-SA} (respectively N_{TD-SA}) be the number of point of the PD-SA Pareto curve (respectively TD-SA Pareto curve) that are not dominated by a point of the TD-SA Pareto curve (respectively TD-SA Pareto curve). Then the result of the match racing is given by:

$$Q = \frac{N_{PD-SA}}{N_{PD-SA} + N_{TD-SA}} \tag{6}$$

Thus Q is a number between 0 and 1. $Q = 0$ means that TD-SA dominates PD-SA, $Q = 1$ means that PD-SA dominates TD-SA. Note also that is the Pareto curve of PD-SA and TD-SA are identical, then $N_{PD-SA} = N_{TD-SA}$ and thus $Q = 0.5$. We perform each time 250 match racing to obtain an estimation of the expectation of $E[Q]$ and the standard deviation σ_Q of Q. The values of $E[Q]$ are obtain with an marginal error of $\pm\ \sigma_E[Q] = \frac{\sigma_Q}{\sqrt{250}} = 0.025$ for $T_0 = 30$ and 0.01 for $T_0 = 1$. Table 1 gives the results of the match racing for 5000, 10000 and 25000 iterations. Tests have been conducted for two different values of the initial temperature T_0 (30 or 1) in TD-SA. It appears clearly that tunning correctly the temperature can lead to better results in TD-SA than in PD-SA. As already mentioned, an end-user has to manually tune the temperature and this might be long and difficult.

Iterations	5 000	10 000	25 000
$\boldsymbol{T_0 = 30}$			
Average	0.60	0.65	0.63
St.Dev.	0.40	0.37	0.36
$\boldsymbol{T_0 = 1}$			
Average	0.33	0.34	0.33
St.Dev.	0.13	0.13	0.12

Table 1. Dominance ratio (Q) of PD-SA over TD-SA

To be sure that the produced Pareto solutions are of good quality in terms of distance to template, we plot the initial solution and some Pareto solutions represented as power spectrum. Hence it is possible to draw the template on the same figure. In Figure 6 dashed lines represent these solutions whereas

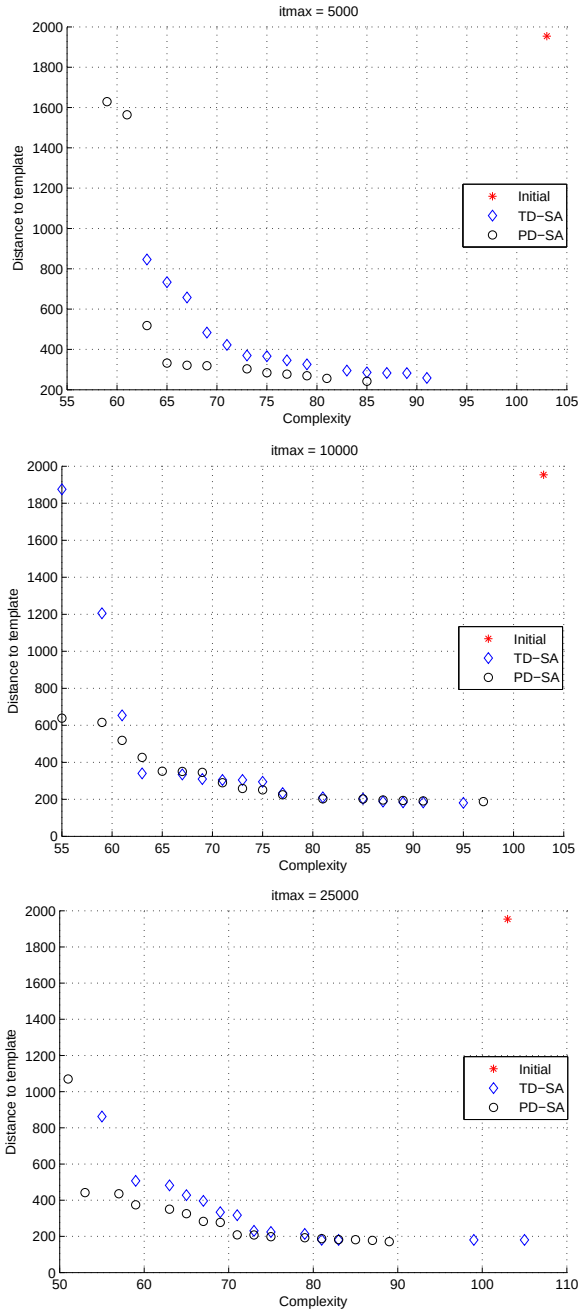

Fig. 5. Solutions after 5 000 it. (up), 10 000 it. (middle) and 25 000 it. (bottom)

the solid line corresponds to the initial solution. It is clear in the figure that the Pareto solutions are largely better than the initial solution, especially the part of the response of the filter outside the transition band is much smaller for all Pareto solutions represented here.

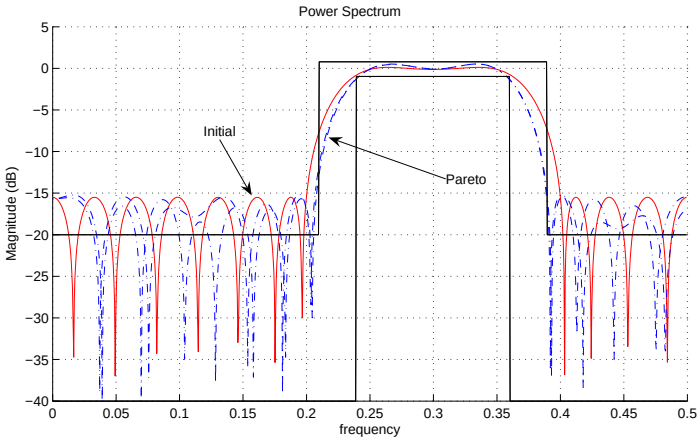

Fig. 6. Power spectrum

We now compare the resolution of the TD-SA approach and the new PD-SA method (see Figure 7) after 50 000 iterations. Note that the constraints for the template have been tightened. For this figure, it is clear that none of the approaches is better than the other. But from the decision maker point of view, the PD-SA gives the *same* results without the parameter setting phase (to find the best values of the parameters needed in the TD-SA, more than 10 different attempts were necessary).

To show how the feedback loop influences the temperature, we draw in Figure 8 the theoretical temperature and the temperature computed afterwards in PD-SA. Since the practical temperature is computed not at every iteration, it results a stepwise curve that oscillate around the theoretical temperature.

5 Conclusion

In this paper, we have presented an alternative approach for the design of a digital FIR filter minimizing two objectives, namely the distance to a template and the complexity of the filter. We have to mentioned here that, to our best knowledge, it is the first time that these two objectives are dealt simultaneously in a approximate Pareto-based approach using simulated annealing.

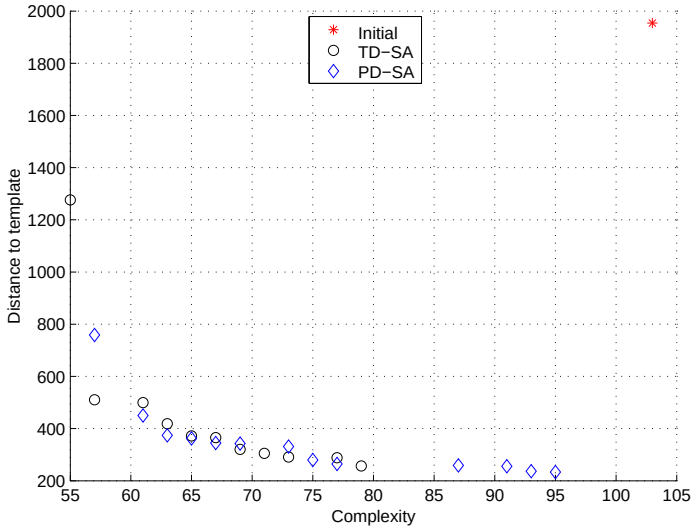

Fig. 7. Comparison of solutions

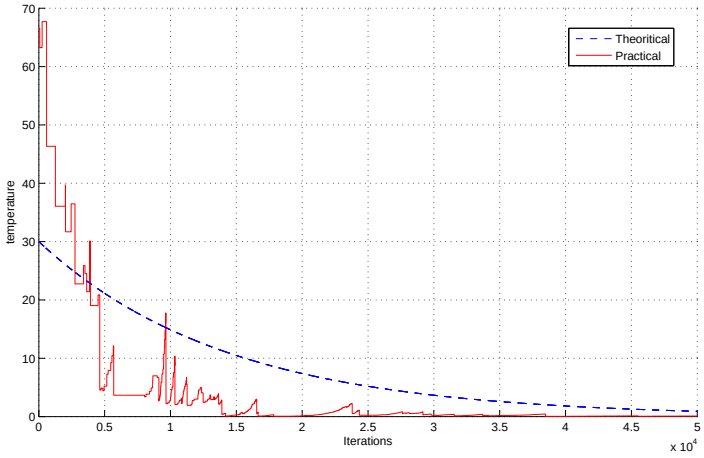

Fig. 8. Comparison of the temperature evolution in PD-SA

In the field of the filter design, it is of course much better to present several solutions to the decision maker which finally can choose the most appropriate alternative for his application.

Numerical experiments do not show any advantage in terms of performances to the probability-driven simulated annealing method but no clear drawbacks either. Of course, the proposed approach contains less parameters that have to be set and represents a progress for non-specialist people in the field of optimization and for end-users.

Through this paper, the reader can notice that several other parameters need to be set, even in the new method, and are not always explicitly mentioned here. The reason is that a graphical user interface might help the decision maker to set these parameters by choosing general templates from libraries with best known values and/or experimented designer knowledge. Several templates will be also included in the library. For example, the number of coefficients of the filter (here set to 33) can become a "meta" parameter. In that case, a dedicated neighborhood procedure will be designed for finding the most adapted number of coefficients.

We believe that the proposed probability-driven simulated annealing approach can be extended with success in many other application domains and that this approach will help the spreading of advanced SA techniques in the engineering community.

References

1. E.H.L. Aarts and J. Korst. *Simulated Annealing and Boltzmann Machines.* John Wiley, Chichester, 1989.
2. L. Cen and Y. Lian. Complexity reduction of high-speed fir filters using micro-genetic algorithm. In *First International Symposium on Control, Communications and Signal Processing/*, pages 419–422, 2004.
3. C. Coello. EMOO web pages. `http://www.lania.mx/~ccoello/EMOO/`.
4. N. Damera-Venkata and B.L. Evans. An automated framework for multicriteria optimization of analog filter designs. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(8):981–990, 1999.
5. K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & sons, New York, 2001.
6. M.A. Johnson and M.H. Moradi, editors. *PID Control.* Springer, London, UK, 2005.
7. S.M. Kilambi and B. Nowrouzian. A genetic algorithm employing correlative roulette selection for optimization of FRM digital filters over CSD multiplier coefficient space. In *IEEE Asia Pacific Conference on Circuits and Systems, 2006. APCCAS 2006.*, pages 732–735, Dec. 2006.
8. Z. Mou and P. Duhamel. Fast FIR filtering : Algorithms and implementations. *Signal Processing*, 13(4):377–384, 1987.
9. D. Nam and C.H. Park. Multiobjective simulated annealing: A comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems*, 2(2):87–97, 2000.

10. M. Oner. A genetic algorithm for optimisation of linear phase fir filter coefficients. In *Conference Record of the Thirty-Second Asilomar Conference on Signals, Systems & Computers, 1998.*, volume 2, pages 1397–1400, Nov. 1998.
11. J. Qiao, P. Fu, and S. Meng. A combined optimization method of finite wordlength fir filters. In *First International Conference on Innovative Computing, Information and Control, 2006. ICICIC '06.*, volume 3, pages 103–106, Aug. 2006.
12. P. Siohan and A. Benslimane. Synthèse des filtres numériques non récursifs à phase linéaire et coefficients de longueur finie. *Annales des Télécommunications*, 39(7-8):307–322, 1984.
13. P. Siohan and A. Benslimane. Finite precision design of optimal linear phase 2-D FIR digital filters. *IEEE Transactions on Circuits and Systems*, 36(1):11–22, 1989.
14. R. Thomson and T. Arslan. An evolutionary algorithm for the multi-objective optimisation of VLSI primitive operator filters. In *Proceedings of the 2002 Congress on Evolutionary Computation, 2002. CEC '02*, volume 1, pages 37–42, 2002.