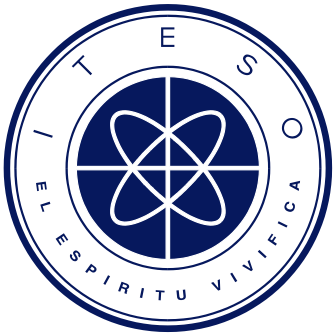
INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación el 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

MAESTRÍA EN DISEÑO ELECTRÓNICO



HIGH-FREQUENCY ELECTRONIC DESIGN OPTIMIZATION USING SIMULATED ANNEALING

Tesis que para obtener el grado de

MAESTRO EN DISEÑO ELECTRÓNICO

Presenta: Jesús Ricardo Alejos Jiménez

Director de tesis: Dr. José Ernesto Rayas Sánchez

Tlaquepaque, Jalisco. Noviembre de 2017.

MAESTRÍA EN INGENIERÍA (2017) ITESO

Maestría en Diseño Electrónico Tlaquepaque, Jal., México

TÍTULO: **High-Frequency Electronic Design Optimization using Simulated Annealing**

AUTOR: Jesús Ricardo Alejos-Jiménez

Ingeniero en Electrónica (ITESO, México)

DIRECTOR DE TESIS: José Ernesto Rayas-Sánchez

Departamento de Electrónica, Sistemas e Informática, ITESO

Ingeniero en Electrónica (ITESO, México)

Maestro en Sistemas Electrónicos (ITESM Campus Monterrey, México)

Doctor en Ingeniería Eléctrica (Universidad McMaster, Canadá)

Senior, IEEE

NÚMERO DE PÁGINAS: viii, 53

Dedication

To my family, who has supported and guided me through my life and career.

Summary

This thesis presents a methodology for improving the performance of the Simulated Annealing (SA) algorithm to optimize high-frequency electronic circuits. It starts by introducing the algorithm together with the fundamental concepts that support its functionality. Then, a set of new features are added to the SA algorithm to control its behavior, which include: knobs for controlling the step-size, search-space limits, and the functions that govern the evolution of the algorithm. The introduction of such features is accompanied by a set of experiments to demonstrate the functionality of the modified SA, and compare its performance against Nelder-Mead and Conjugated Gradients Fletcher-Reeves methods. This is followed by the definition and application of a methodology to configure the algorithm and improve its consistency and efficiency, accompanied by a test (optimization of a simple high-frequency filter) to verify its effectivity. Next, SA is configured to optimize a more complex circuit, consisting of a microstrip low-pass filter implemented in the full-wave electromagnetic simulator Sonnet. Finally, such optimization problem is solved by using other optimization algorithms (Nelder-Mead, Sequential Quadratic Programming, and Genetic Algorithm) to make an overall assessment of the proposed SA algorithm, identifying what kind of problems may take advantage of the features and improvements added to SA, or may expose its caveats.

Contents

Introduction

The design optimization during the development of electronic devices has become a bottleneck due to the increasing complexity of such devices, and the limited availability of computing resources. This leads to a trade-off between cost, time, and design-quality. There is a need for more efficient optimization methods to deal with such trade-off.

Global optimization algorithms (such as Genetic Algorithms [Whitley-94], Particle Swarm Optimization [Poli-07], and Ant Colony Optimization [Dorigo-06], among others) are used for working around poor solutions that might be given by other algorithms when they get stuck in local optima (which lead to poor designs). However, the employment of global optimization algorithms usually requires either more time, or more expensive computer hardware. This happens because such algorithms involve many objective function evaluations before getting to a result candidate.

In this thesis, we focus on improving the performance of Simulated Annealing (SA) for electronic circuits design optimization. This algorithm, in contrast to other global optimization algorithms, yields solution candidates with fewer objective function evaluations. This is achieved at the cost of sacrificing the accuracy and repeatability of its results. Across this document, we describe a methodology that involves adding some extra features to the SA algorithm, and a process to adjust them to reduce its unwanted behaviors. Moreover, we show how this algorithm can optimize functions with numerical noise, which appears, for instance, due to quantization errors in EM simulations with coarse discretization).

Chapter 1 introduces the reader to the foundations of SA: the ideas that inspired it, its components, and its interactions. The SA algorithm is implemented with additional features: custom temperature profiles, variable step-size, normalized objective-function-input generation, and search-region restriction. Then, we proceed to test this SA implementation with mathematical functions and a simple microstrip filter design optimization problem. Such test is complemented by a comparison against the results given by the Conjugated Gradients Fletcher-Reeves (CGFR) and Nelder-Mead (NM) optimization algorithms.

Chapter 2 describes a methodology for improving the accuracy of the optimization results given by SA. Such methodology consists of changing the temperature function, step-size function,

and acceptance function in such a way that allows choosing the combination of them that best fits the optimization problem being solved. The proposed methodology is shown in action by solving a simple high-frequency circuit design optimization problem.

Chapter 3 provides an example case consisting of a more complex high-frequency circuit design optimization problem, which is solved using the proposed SA implementation, together with the tuning methodology proposed in Chapter 2. Three versions of the objective function are used, which differ in the electromagnetic simulation resolution used to get its response. This exposes the capability of SA of dealing with discretized search spaces.

In Chapter 4, the same optimization problem presented in Chapter 3 is solved but now using Genetic Algorithms, Sequential Quadratic Programming, and the Nelder-Mead algorithm. This is realized with the purpose of comparing their results and performance with those delivered by our SA implementation. The strengths and weaknesses of enhanced SA algorithm are identified, suggesting which scenarios are good candidates for its effective usage.

The author expresses his sincere appreciation to Dr. José Ernesto Rayas-Sánchez, Numerary Professor in the Department of Electronics, Systems, and Informatics, and director of the Research Group on Computer-Aided Engineering of Circuits and Systems (CAECAS) at ITESO, who kindly guided the efforts that resulted in this document. The author also thanks to the people who encouraged and supported this effort through the journey in the Master's program, mainly to Ricardo Alejos Vizcarra, and Rosa de Lourdes Jiménez, my parents.

1. Basic Concepts on Simulated Annealing

This chapter introduces the Simulated Annealing optimization method. We review its most basic theory. Then, the algorithm is implemented and applied for solving some mathematical and electronic design problems. With this data, an assessment is made to identify in what scenarios would be better to use it instead of using other classical optimization methods, such as Conjugate Gradients or Nelder-Mead.

1.1. Introduction

Electronics is becoming part of our everyday life through its inclusion in different items making them "smart". Mobile technology, wearables, smart-cars, domotics and other product lines are proof of such trend. The effort to achieve such integration involves different trade-offs that have to be overcome by product developers in order to achieve competitive products and prices.

One of the key challenges during the product development is their design process. As the market demands the usage of low-cost materials to achieve competitive prices, it becomes more challenging. Moreover, as the usage of such product becomes more versatile, it is subjected to more kinds of physical stress (for example: electromagnetic, thermal and mechanical).

In order to design products with these factors taken into account, the development engineers can make use of multi-physical modeling, which happens to be computationally intensive. Given that such process gets more expensive as more details are needed, it becomes a key area for potential improvement.

One way of improvement is by decreasing the computational cost of the design process by using computationally inexpensive models. However, the behavior of such models often does not correlate with sufficient accuracy with the behavior of the manufactured product, and they may introduce numerical noise. In order to overcome such difficulties, optimization algorithms can be employed. However, it may not be trivial to decide which would be the best algorithm to use because some of them have greater computational cost and some others are not effective for noisy objective functions.

Finding the global optimum in a numerical optimization problem can be a difficult problem

since the employed algorithm can get stuck in local minima. The algorithm for solving such problems usually samples the objective function across its domain, achieving a high probability of finding a near-global-optimal solution, and lend itself to efficient implementation. Such criteria are met by Simulated Annealing, which was introduced in the early 1980s by Kirkpatrick et al., and independently by Cerny.

Simulated Annealing (SA) is a heuristic and stochastic algorithm derived from statistical mechanics for finding near globally-minimum-cost solutions to large optimization problems [Gall-14]. It models problems as a system of interacting components where the magnitude of such interactions vary proportionally to the energy state of the system: when the system is in steady state, the probability to change from one energy state to another is given by Boltzmann distribution, which is dependent on the system temperature and the magnitude of the desired energy state change. This is done in such a way that higher temperatures allow more random changes, and as it cools down, it settles down to a final state.

In contrast to many classical optimization processes, SA is not based on gradients and it does not have a deterministic convergence: the same seed and parametric configuration may make the algorithm converge to a different solution from one run to another. This is because the decision for taking the next step towards the final solution has a random component.

With such behavior, SA may not converge to the global optimum. However, it has other exploitable advantages, for instance: it can escape from local optimum points, it can deal with noisy objective functions, and it can be used for discrete optimization. All these benefits can be available with few iterations and function evaluations, in comparison to other optimization methods. When applicable, SA can be combined with other processes to increase the accuracy of the final solution.

In this chapter, the basic theory of this algorithm is explained and some of its benefits are verified with practical examples.

1.2. Basic Theory of Simulated Annealing

The name of this algorithm is inspired from metallurgy, where the word “annealing” refers to a tempering technique that involves heating a metal and cooling it down so that the shape

imperfections are eliminated and the material becomes harder. The principle behind this process is that the atoms that compose the material gain mobility when temperature increases, and they lose it as it cools down, forming well-defined structures. This behavior of controlled cooling is emulated by SA to regulate the acceptance probability of candidate solutions as the algorithm evolves.

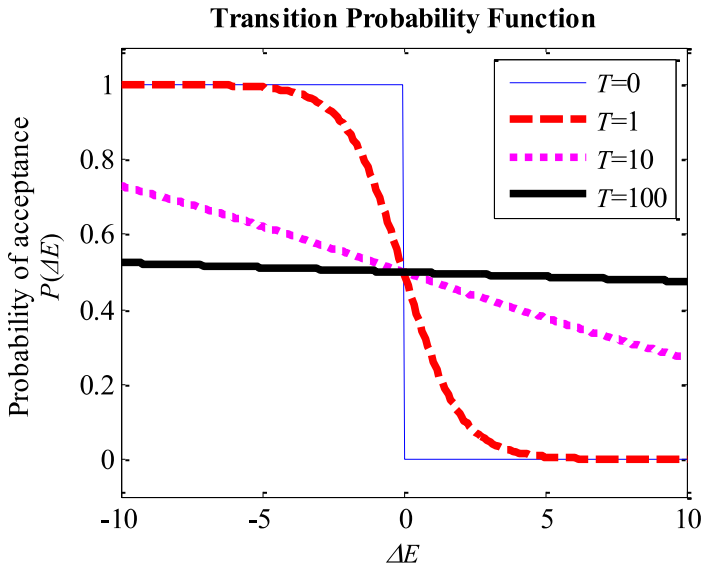


Fig. 1-1. Plotted transition probability function. It describes how probable is to accept or reject a step given the energy difference between the current point and the proposed one ($\Delta E = u(\boldsymbol{x}_i) - u(\boldsymbol{x}_j)$), where T represents the temperature value for each case.

The objective function u emulates the energy states of the annealed material by being modeled as a node network, where each node (represented by a vector $\boldsymbol{x}$) has an assigned energy state with value $E = u(\boldsymbol{x})$. Starting from $\boldsymbol{x}_0$, SA "travels" through the node network one node at a time. In each iteration, a transition from the current-node $\boldsymbol{x}_i$, to another node $\boldsymbol{x}_j$ is proposed within a limited range (given by the current step size), and the algorithm decides whether it is accepted or not.

Such decision is made in a way that the nodes that represent better candidates to be closer to the optimum point $\boldsymbol{x}_{\text{opt}}$ have greater probability of being accepted than those that worsen the value of the objective function. This behavior changes as the temperature parameter value decreases: the lower the temperature, the stricter the acceptance criteria. Such trend is continued towards only accepting the nodes that show an improvement with respect the current objective function value.

For a multivariable-scalar objective function $u(\boldsymbol{x})$, the probability P that the transition from the current-node $\boldsymbol{x}_i$ to a next-node $\boldsymbol{x}_j$ takes place is given by

$$P(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{1}{1+e^{\Delta E/T}} \tag{1-1}$$

where $\Delta E = u(\boldsymbol{x}_j) - u(\boldsymbol{x}_i)$ for minimization problems, or $\Delta E = u(\boldsymbol{x}_i) - u(\boldsymbol{x}_j)$ for maximization problems, and T is the temperature parameter. Notice that $P \to 0.5$ when $T \gg \Delta E$. This shows how the random behavior of SA becomes more dominant as temperature increases.

As $T \to 1$ the behavior of P becomes very similar to a horizontally-inverted Heaviside function (or sigmoidal function) in terms of ΔE: the probability of accepting that transition goes close to zero when $\Delta E >> 0$, and to 1 when $\Delta E << 0$. Such behavior is illustrated in Fig. 1-1 for different values of T.

In our implementation, the temperature parameter T is updated in every iteration by indexing the value of a temperature profile (which is modeled as a vector $\boldsymbol{T}$) so that $T = T_i$ and $T_i \in \boldsymbol{T}$. Changing the temperature profile can lead to different SA behaviors. Fig. 1-2 shows different temperature profiles. Some of the observed behaviors are that periodic profiles allow the algorithm to escape from local optimum regions, while exponential decay profiles allow the algorithm to have a larger coverage of the objective function domain.

Also, for our implementation, we consider a variant of SA that varies the step size in proportion to the temperature parameter. This allows the algorithm not just to cover wider areas when the temperature has high values, but also become stricter and finer as the temperature goes low.

The SA implementation used for this study is based on the pseudo-code shown in Fig. 1-3 [Rossmanith-11]. Additionally, some extra features were added, which include: custom temperature profiles, temperature-dependent step-sizes, returning the best solution found instead of the last one (allowing precision improvements when the algorithm is ran periodically), domain restrictions, custom exit criteria (the original algorithm exits until the entire temperature profile has been swept), among others. Most of these features have been added to the algorithm version used in the corresponding internal research report. The MATLAB code used for this work can be found in Fig. 1-4. This implementation has the characteristic that its number of iterations I is always equal to the number of elements of the temperature profile $\boldsymbol{T}$. This leads to a tradeoff between the average precision and the computational cost.

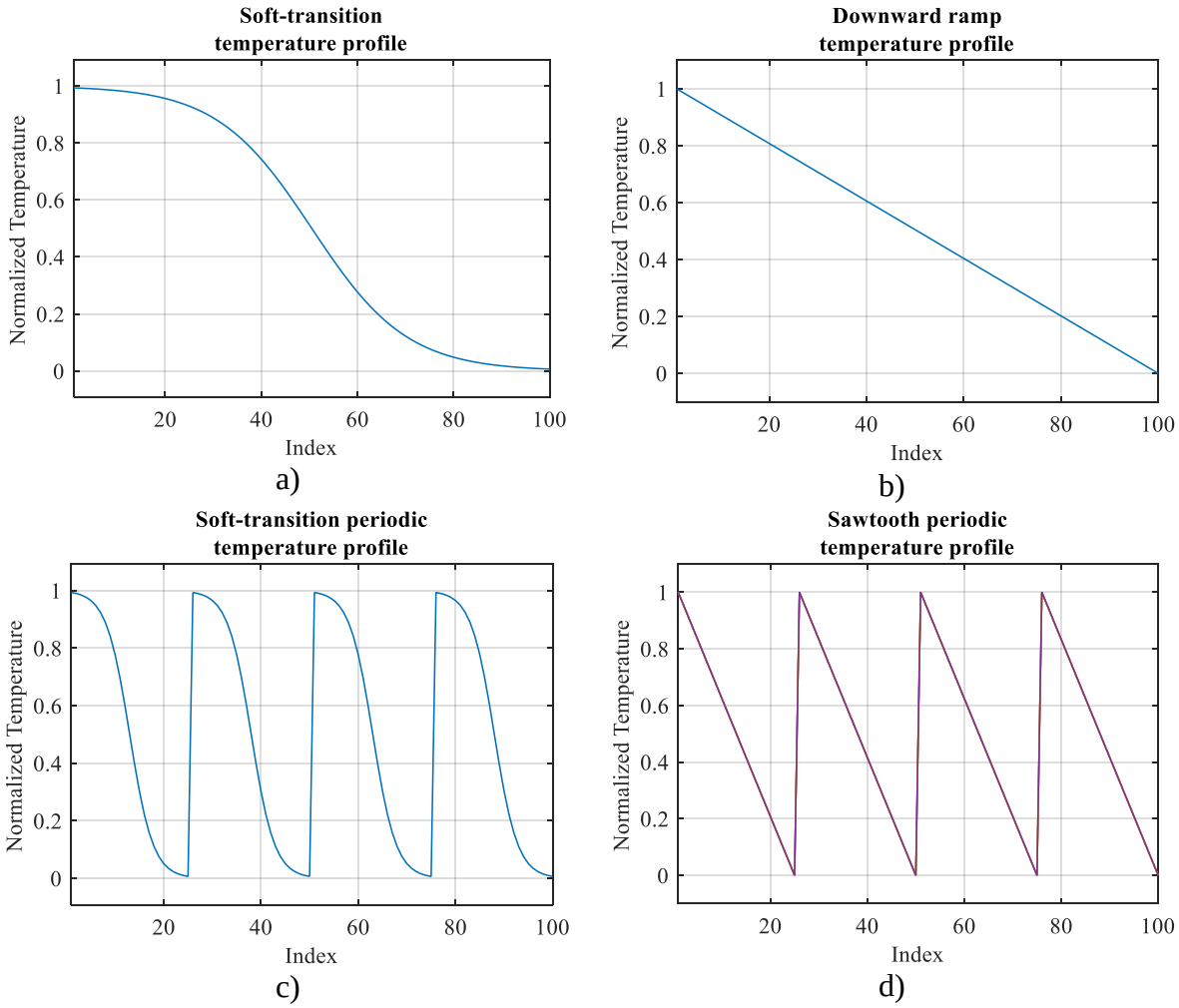


Fig. 1-2. Different temperature profiles used for SA: a) soft-transition; b) downwards ramp; c) periodic soft-transition; d) periodic downwards ramp (saw tooth).

1.3. Examples using Mathematical Functions

In this section, SA is compared against Conjugate Gradients Fletcher-Reeves (CGFR) and Nelder-Mead (NM) optimization algorithms in terms of precision and computational cost. Such comparison is done by minimizing a collection of multivariable-scalar functions.

To measure precision, we use the Euclidean norm of the difference between the exact solution and that one found by each algorithm. The Euclidean norm gives the notion of distance between two points when the problem has one or two independent variables. This happens to be helpful because it makes the algorithm behavior become graphically intuitive in such cases.

The computational cost has two main components: memory usage and time complexity. For now, we focus on the time complexity, which can be measured both in terms of the number of iterations, and the number of times that the objective function is evaluated. In some cases, the number of function evaluations is taken into account because it has more impact on the overall running-time than the number of iterations. The number of iterations become data of interest when its pre-configured maximum is exceeded, which usually means that the algorithm has diverged.

```
begin
        i = 0
        x_i = x_0
        for each element in T:
                x_j = generate a node one step away with random direction
                if P(x_i, x_j) > random(0, 1)
                        x_i = x_j
end
```

Fig. 1-3. Pseudo-code for Simulated Annealing. The acceptance of a new point happens with a probability P given by (1-1).

The testing mathematical function are next described.

1.3.1 Bowl Function

The bowl function is a quadratic multi-variable scalar function with only one minimum. It can be written mathematically as

$$u_{\text{bowl}}(\boldsymbol{x}) = (x_1 - 6)^2 + \frac{1}{25}(x_2 - 4.5)^4 \tag{1-2}$$

and its minimum value is zero. This value is reached when $[x_1 \;\; x_2]^{\text{T}} = [6 \;\; 4.5]^{\text{T}}$.

The optimization results for the bowl function (starting the algorithms at the seed value $\boldsymbol{x}_0 = [1 \;\; 1]^{\text{T}}$) are shown in Table I. Notice that SA has a worse precision as compared to the other two algorithms. However it computes its final result with less function evaluations. Fig. 1-5 offers a visual aid on the path followed by SA through the bowl function. Notice how the density of accepted points increases as it approaches the solution. This happens because the step-size decreases when the temperature decreases, as mentioned in the previous section and illustrated in Fig. 1-6.

```
function [x_opt, u_val, XN, FN] = SimulatedAnnealing(u, x0, t, s, l)
  %{
    Simulated Annealing - Optimization Algorithm
    Inputs
      u  <function> - Function to be optimized
      x0 <n-element row vector> - Seed value of independent variable of "u"
      T  <m-element row vector> - Temperature parameter
      s  <2-element row vector> - Step size [Maximum Minimum]
      l  <2*n-element matrix>   - x limits [min(x); max(x)]
    Ouputs
      x_opt <n - lement vector> - optimal solution found.
      u_val <scalar> - value of "u" at x_opt
      XN <m*n-element matrix>   - x value history during the algorithm run
      UN <m-element row vector> - u value history during the algorithm run
    The more the cost of f is, the shorter the t vector should be.
  %}

  N = 1:1:length(t); % iterator
  P = @(DE,T) 1/(1+exp(DE/T)); % bigger when DE is more negative
  S = @(T) (s(2)-s(1))/(max(t)-min(t))*(T-min(t))+s(2); % linear step-size calculation in
function of temperature
  xn = x0;
  XN = zeros(length(t), length(x0));
  UN = zeros(length(t),1);
  c=0;
  u0 = feval(u,xn);
  un = u0;
  for n = N
    t = T(n);          % update current temperature
    while (1)
      xt = rand(size(xn))-0.5;  % generate random direction
      xt = xt/norm(xt,2);     % make direction vector unitary
      xt = xt*S(t);         % scale step size
      xt = xn + xt;         % advance that step
      if (sum(xt>l(1,:))==length(xt) && sum(xt<l(2,:))==length(xt))
        break
      end
    end
    ut = feval(u,xt);       % evaluate function at test point
    DE = ut-un;          % delta between current and test function values
    p = P(DE,t);         % probability for xt of being accepted
    r = rand();          % random decider
    if (r<p)
      xn=xt;
      un=ut;
      c=c+1;
    end
    XN(n,:)=xn;
    UN(n)=un;
  end
  XN = [x0;XN];
  UN = [u0;UN];
  u_val = min(UN);
  u_val = u_val(1);
  x_opt = XN(UN==u_val,:);
  x_opt = x_opt(1,:);
end
```

Fig. 1-4. MATLAB implementation of Simulated Annealing.

With the purpose of exploring the behavior of the algorithms when they begin their search far away from the solution, the algorithm is triggered using $[-20 \;\; 60]^T$ as the seed value. The results of such experiment are in Table II. Notice how CGFR requires more function evaluations than in the last case, while NM and SA keep the number of functions evaluations around the same values.

The SA evolution can be observed in Fig. 1-6.

TABLE I
OPTIMIZATION RESULTS FOR THE BOWL FUNCTION WITH $\boldsymbol{x}_0 = [1\ \ 1]^T$.

Algorithm	Solution found	Function evaluations	Euclidean norm error
CGFR	$[6.0000\ \ 4.4300]^T$	31	0.07
NM	$[6.0000\ \ 4.5000]^T$	174	3.6161e-06
SA (70 elements in downward ramp profile)	$[6.3730\ \ 5.7405]^T$	70	0.0855
SA (30 elements in downward ramp profile)	$[5.9731\ \ 3.5287]^T$	30	0.9717
SA (10 elements downward ramp profile)	$[5.8626\ \ 5.1233]^T$	10	0.6382

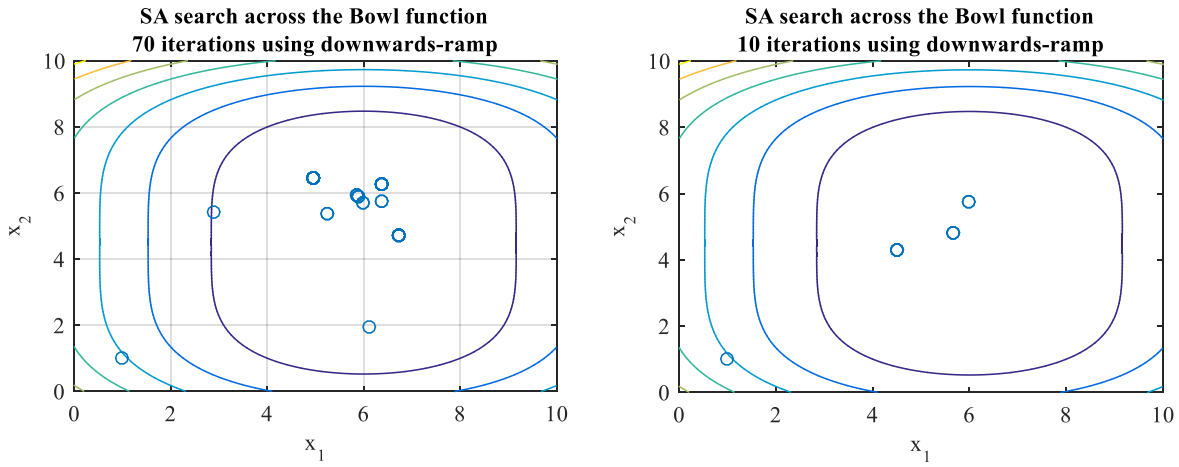
TABLE II
OPTIMIZATION RESULTS FOR THE BOWL FUNCTION WITH $\boldsymbol{x}_0 = [-20\ \ 60]^T$.

Algorithm	Solution found	Function evaluations	Euclidean norm error
CGFR	$[6.0000\ \ 4.4226]^T$	9688	0.0774
NM	$[6.0000\ \ 4.5000]^T$	180	1.3098e-06
SA (70 elements in downward ramp profile)	$[5.8363\ \ 4.8158]^T$	70	0.3557
SA (30 elements in downward ramp profile)	$[6.0663\ \ 4.0117]^T$	30	0.4928

1.3.2 Bowl Function with Gaussian Noise

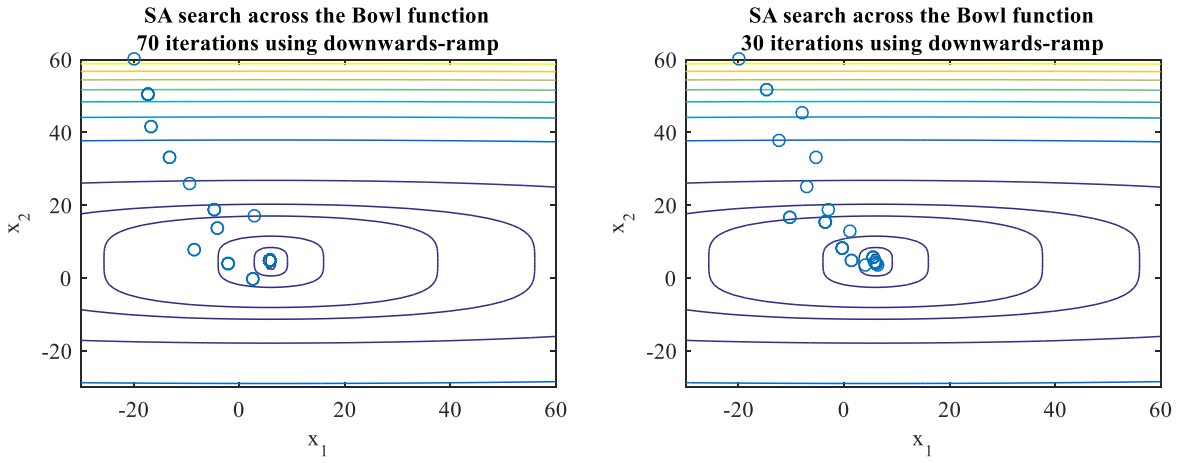
This test case basically consists of adding Gaussian noise to the previous objective function so that the average value of the function on each point is kept the same, but with a standard deviation of 0.5. The optimization results are shown in Table III. Now it is clear that SA got a more precise and faster result than the other algorithms, which diverged from the exact solution.

The algorithm evolution is shown in Fig. 1-7.



a) b)

Fig. 1-5. Evolution of SA for the bowl function starting from the point $[1 \;\; 1]^T$: a) evaluating 70 points in the objective function; b) evaluating 10 points.



a) b)

Fig. 1-6. Evolution of SA for the bowl function starting from $\boldsymbol{x}_0 = [-20 \;\; 60]^T$: a) using 70 temperature profile points; b) using 30 points.

1.3.3 Periodic and Exponential Function

For this test let us consider a periodic and exponential function as our objective function,

whose behavior is mathematically given by

$$u_{\text{perexp}}(\boldsymbol{x}) = -\sin^2\left(\frac{4\pi x_1}{13}\right)\cos^2\left(\frac{4\pi x_2}{13}\right)e^{\left(\frac{x_1+x_2}{10}\right)} \tag{1-3}$$

The solution space is limited to the range 0 to 12 for both x_1 and x_2. With this scenario the exact solution is at $[11.4285 \;\; 9.8035]^T$. Such limits are incorporated to the problem with penalty functions, which worsens the value of the function as it goes away from the ranges of interest.

The optimization results are shown in Table IV and Fig. 1-8. Notice that SA is the only algorithm that is capable of getting into the global minimum region. However, it also needed more runs in order to get such solutions (remember that each run gives a different result because of its stochastic nature).

TABLE III
OPTIMIZATION RESULTS FOR NOISY BOWL FUNCTION USING $\boldsymbol{x}_0 = [1 \;\; 1]^T$.

Algorithm	Solution found	Function evaluations	Euclidean norm error
CGFR	$[5.8477 \;\; -1.0434]^T$	37317 (diverges)	5.5455
NM	$[1.0590 \;\; 0.9977]^T$	401 (diverges)	6.0563
SA (10 elements downward ramp profile)	$[6.0581 \;\; 4.3855]^T$	10	0.1284

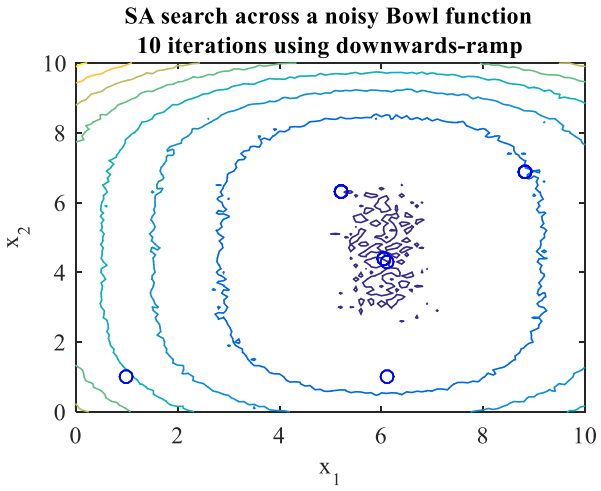


Fig. 1-7. Evolution of the Simulated Annealing algorithm within the noisy bowl function surface starting at $\boldsymbol{x}_0 = [1 \;\; 1]^T$.

TABLE IV
OPTIMIZATION RESULTS FOR PERIODIC AND EXPONENTIAL OBJECTIVE FUNCTION USING $\boldsymbol{x}_0 = [1 \;\; 1]^T$.

Algorithm	Solution found	Function evaluations	Euclidean norm error
CGFR	$[4.9285 \;\; 3.3035]^T$	54	9.1924
NM	$[4.9285 \;\; 3.3035]^T$	80	9.1924
SA (10 elements downward ramp profile)	$[5.2441 \;\; 7.0207]^T$	10	6.7817
SA (30 elements downward ramp profile) – 5 tries.	$[11.084 \;\; 9.8603]^T$	(30)(5) = 150	0.3490
SA (30 elements 3 cycle sawtooth) – 2 tries.	$[11.335 \;\; 10.085]^T$	(30)(2) = 60	0.2963
SA (90 elements 3 cycle sawtooth)	$[11.225 \;\; 9.8939]^T$	90	0.2229
SA (90 elements 3 cycle sawtooth)	$[11.623 \;\; 9.8049]^T$	90	0.1945

1.4. Examples using a Microstrip Filter

1.4.1 Microstrip Low-pass Filter

This optimization problem consists of finding the geometrical parameter values that allow the low-pass filter shown in Fig. 1-9 to match its design specification requirements (which are known to be strict). Being f the frequency value, the specifications are given by:

$$|S_{11}| > 0.9 \;\text{ for }\; 0.2\text{GHz} \le f \le 6\text{GHz}\,, \text{ and} \tag{1-4}$$

$$|S_{21}| < 0.1 \;\text{ for }\; 8\text{GHz} \le f \le 10\text{GHz}\,. \tag{1-5}$$

Let us consider $\boldsymbol{x} = [W_1 \;\; L_1 \;\; S_1]^T$ as our design variables, while preserving $\boldsymbol{z} = [H \;\; \varepsilon_r \;\; W_p \;\; L_p \;\; \tan(\delta) \;\; \sigma \;\; t]^T = [0.794\text{mm} \;\; 2.2 \;\; 2.45\text{mm} \;\; 12.25\text{mm} \;\; 0.01 \;\; 5.8\times10^7\text{S/m} \;\; 15.24\mu\text{m}]^T$ as fixed parameters.

This problem is solved by using a mini-max formulation: where the objective function is

the maximum error with respect to the specification requirements. In this kind of formulations, the maximum error has to become negative to meet all the specs.

Such objective function has many local minima. For this problem, SA is fed with a periodic temperature profile which helps it to escape from local minima by changing the step-size and the randomness of the energy state transitions.

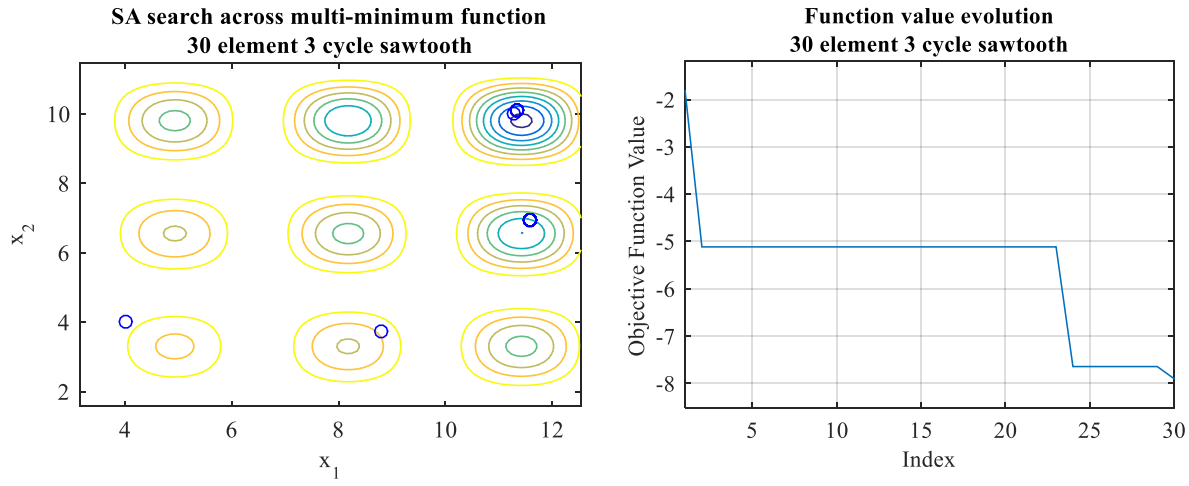
Table V shows the optimization results. Even though that NM was able to achieve a negative maximum error, it doubled the number of function evaluations that it took to SA to find its solution (which is not as precise as the one from NM). All this while CGFR went over its iterations limit before converging to a solution, which is not as precise as the ones found by SA and NM. Fig. 1-10 shows the evolution of the objective function for SA, which finds the best value at 26-th iteration.

1.4.2 Noisy Filter Optimization

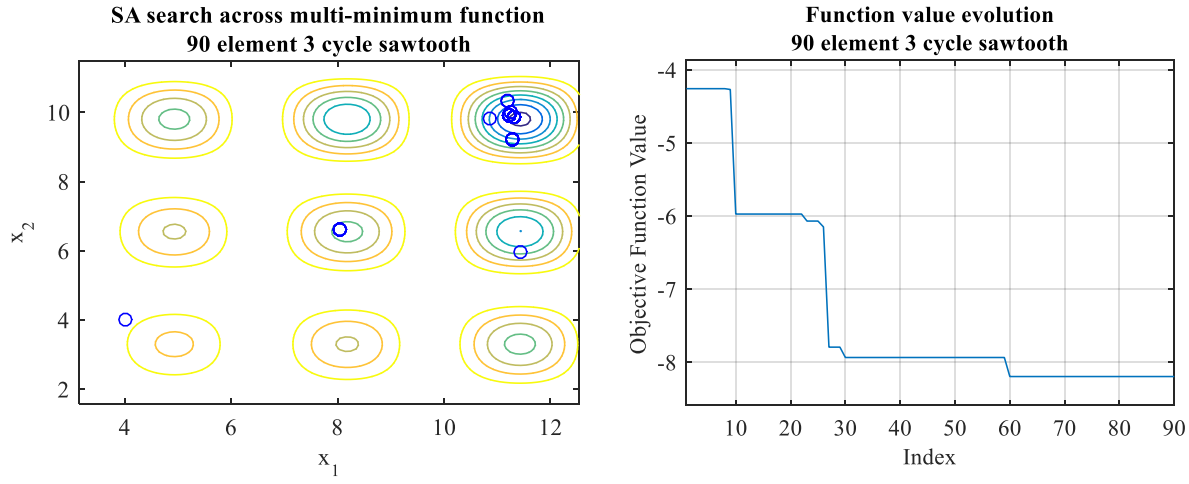
Similarly to what was done with the bowl function, now we add Gaussian noise to the filter response. This time, such noise has a standard deviation of 0.1 while the objective function keeps its average within its original range.

Refer to the experiment results in Table VI. Notice that NM is still reporting a close-to-zero final objective function value, however, it requires a much larger number of function evaluations with respect to SA. On the other hand, SA does not converge to the best solution, but it gets close enough.

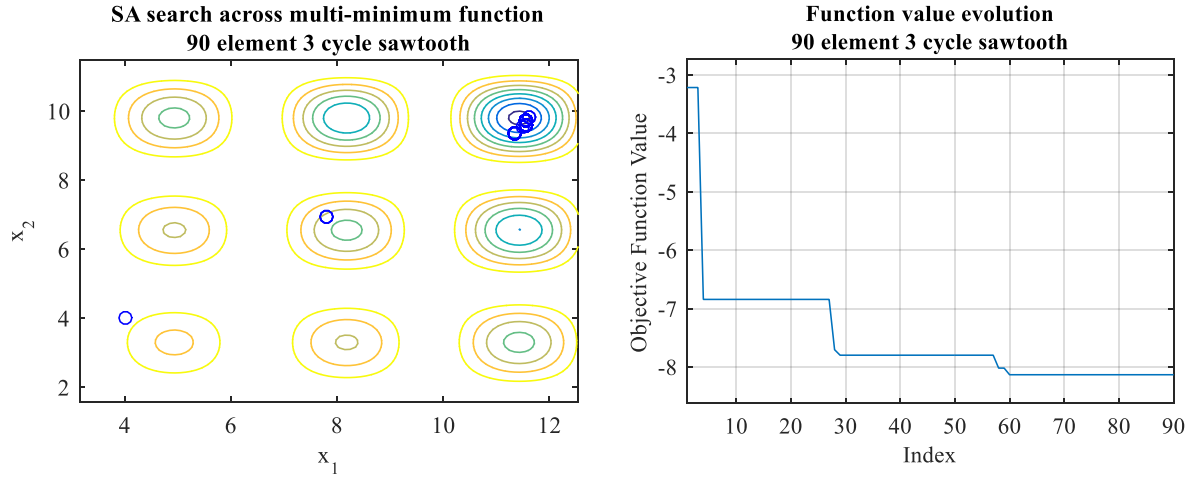
These results indicate that SA is not just good for solving complex problems, but also for finding seed values to feed finer optimization algorithms.



a)

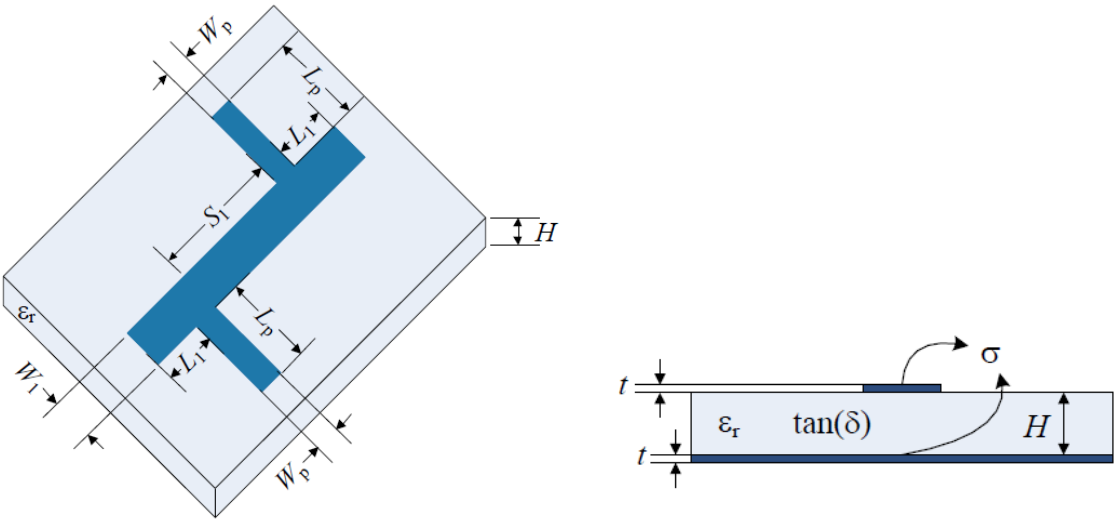


b)



c)

Fig 1-8. Evolution of SA applied to the periodic and exponential function (1-3) using the saw tooth temperature profile: a) evaluating the function 30 times; b) and c) evaluate the function 90 times.



a) b)

Fig. 1-9. Dimensional description of the low-pass filter on microstrip technology: a) top view, (b) transversal section.

TABLE V
OPTIMIZATION RESULTS FOR THE MICROSTRIP FILTER
USING $\boldsymbol{x}_0 = [3.5 \;\; 5.6 \;\; 4.2]^T$.

Algorithm	Solution found (mm)	Maximum error value at $\boldsymbol{x}^*$	Function evaluations
CGFR	$[4.9007 \;\; 34.515 \;\; 20.884]^T$	0.1753	37317
NM	$[0.9555 \;\; 6.7088 \;\; 3.8446]^T$	-0.0077	163
SA	$[1.9033 \;\; 5.4549 \;\; 6.9883]^T$	0.0837	63

1.5. Conclusions

As CGFR, NM and other optimization algorithms, SA has a trade-off between its precision and computational cost. However, the key difference is that the algorithm can vary both its precision and cost as the algorithm evolves. By taking control over both parameters, SA can make wide-area searches much faster and become more precise as it approaches its last iteration. This while the other two algorithms keep their cost constant for both wide and short searches and having the risk of converging to the next local minima they find (and therefore they may not explore other

regions with better optimum values).

When the search regions are wide enough to embrace more than one local minimum, it is convenient to use SA to find where the global minimum is located. This is achieved by keeping the temperature *T* parameter in high values, which implies large step sizes and higher randomness for node transition acceptance. Also, since the precision is still not important at this point, it can be done with fewer algorithm iterations. This is done by lowering the number of elements in ***T***.

For narrow search regions where there is only one minimum, SA should be configured with low-temperature as the step-size becomes shorter and the transition acceptance stricter.

Wide area searches using SA can be used as seed value generators for other algorithms that have better behavior in narrower regions. This avoids that other algorithms get stuck in local minimum regions.

Periodic temperature profiles can be used in order to increase the capability of SA to escape from local minimums, so the algorithm recovers its mobility towards the solution. It can also be programmed in such a way that it always returns the best node found during the search instead of the converged solution (which are not necessarily the same).

Having a variable step size allows SA to adapt its search characteristics. The step configuration is also helpful when the objective function is noisy, since such functions tend to make gradient based algorithms diverge. This happens because such algorithms take the neighbor objective function values to determine the search direction and step size.

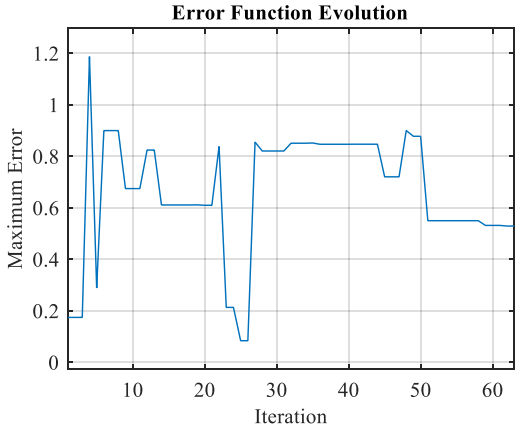


Fig. 1-10. Maximum error values as SA runs for the microstrip low-pass filter.

Noisy objective functions are frequently observed in real-world applications. Many of these are related to measurements because they have a range of uncertainty, which can be modeled as

noise. When it comes to noisy functions, one could think of averaging the values on each point to clear the noise. However, that implies taking various samples per point, and therefore, such solution increases the computational cost in terms of function evaluations.

TABLE VI
OPTIMIZATION RESULTS FOR THE MICROSTRIP FILTER WITH NOISE USING TWO DIFFERENT SEED VALUES

Algorithm	$\boldsymbol{x}_0$ (mm)	$\boldsymbol{x}^*$ (mm)	Max error at $\boldsymbol{x}^*$	Function evaluations
NM		$[0.8288 \quad 6.2988 \quad 3.6405]^T$	0.0115	601
SA (1st try)	$[3.5 \quad 5.6 \quad 4.2]^T$	$[1.1903 \quad 4.9173 \quad 8.5986]^T$	0.1593	100
SA (2nd try)		$[1.8494 \quad 5.8486 \quad 5.4776]^T$	0.1326	100
NM		$[14.095 \quad 22.336 \quad 16.739]^T$	0.8610	601
SA (1st try)	$[7.0 \quad 11.2 \quad 8.4]^T$	$[10.188 \quad 5.1777 \quad 20.235]^T$	0.3735	100
SA (2nd try)		$[6.4806 \quad 9.5782 \quad 18.175]^T$	0.2072	100

2. Configuring Simulated Annealing for High-Frequency Design Optimization

This chapter explains a tuning methodology to improve the mean error of the results given by the Simulated Annealing (SA) algorithm. The tuning methodology contemplates varying the mathematical functions that control the behavior of the algorithm: the temperature function, the step-size function, and the acceptance function. Since the behavior of the algorithm varies for each objective function, this methodology is illustrated by applying it to a high-frequency circuit design optimization problem.

2.1. Introduction

Simulated Annealing (SA) behaves differently for each objective function. Such variation can be measured with the mean error of the converged solutions from one objective function to another [Dymond-15] [Vanderbilt-84]. Such variations are smaller when the objective functions are similar, so it is possible to adapt the algorithm to a family of objective functions by optimizing its behavior for a base objective function representative of such family. In this case, the goal is to adapt SA to solve a simple high-frequency circuit design optimization problem.

To achieve such adaptation, a methodology that consists of manipulating the internal control functions that determine the behavior of SA is proposed: the temperature function (t), the step-size function (s), and the acceptance function (a).

We have defined three versions of each control function: typical, slow, and fast. To facilitate the notation, each version is identified with a letter “T”, “S”, and “F”, respectively. In general terms, slow functions have a smaller rate of change as compared to the typical version, while fast versions have a higher rate of change as compared to the typical version.

An “SA configuration” is defined as a collection of the three control functions. A notation that consists of three letters is used to represent an SA configuration: the version of the temperature function, the version of the step-size function, and the version of the acceptance function. For example, the configuration SFS refers to a configuration that involves the slow temperature function (S), the fast step-size function (F), and the slow acceptance function (S). The

configuration TTT is referred as the "default configuration" (typical in all aspects). The "order" of a configuration is defined as the number of control functions used that are different to their typical version (for example, the configuration TFT is a first order configuration, while STF refers to a second order configuration).

The adaptation methodology consists of applying the default configuration to an objective function of interest. Then all possible first-order configurations are applied and the results are compared against the default configuration. If only one of the first-order configuration results to be better than the default configuration, it is selected as the optimal configuration. If two/three first-order configurations get better results than the default configuration, such configurations are merged into a second/third-order configuration which is also tested and compared. Else, if there is no configuration better than the default configuration, this one is selected as the optimal configuration.

2.2. The Objective Function

The selected base problem consists of finding the width W (in millimeters) that minimizes the reflections of a microstrip transmission line in a 50 Ω system across the frequency range that goes from 0.1 GHz to 20 GHz. This considering that the transmission line has a metal thickness t = 0.66 mm, and a dielectric substrate height H = 0.66 mm with a relative permittivity $\varepsilon_r = 9$.

The objective function program calls a script which invokes Sonnet (a full-wave electromagnetic simulator) that runs a circuit template. Such template contains all the given specifications of the microstrip line. The objective function u is the average value of the reflection coefficient magnitude, $|\Gamma|$, evaluated at 50 frequency points linearly distributed across the frequency range of interest. Since $0 \leq |\Gamma| \leq 1$, then the objective function value is within a normalized range between 0 and 1.

Let $\boldsymbol{x}_j$ be the j-th iterate predicted by our SA algorithm. In this particular case, $\boldsymbol{x}_j$ corresponds to the predicted width W at each iteration. Since our implementation of SA bounds its iterates to the range $-1 < \boldsymbol{x}_j < 1$, the input is transformed so that $W = (10 \text{ mm}) \times (\boldsymbol{x}_j + 1)$. At the end, the search region embraces values from 0 mm to 10 mm.

2.3. The Control Functions

2.3.1 Temperature Functions

Temperature functions determine the temperature value for each iteration of SA. It is a decreasing function, which makes the search resolution increase (because the step-size gets smaller with temperature), and turns the search more deterministic (by interacting with the acceptance function).

Three temperature functions whose main difference is the number of iterations they spend in high or low temperature values (which are normalized to the range $0 < t \leq 1$) have been designed. As mentioned before, we have defined a typical version

$$t_{\mathrm{T}}(j,N) = (1+0.00001^{0.5-j/N})^{-1}, \tag{2-1}$$

a slow version,

$$t_{\mathrm{S}}(j,N) = 2 - 2^{(j/N)^3}, \tag{2-2}$$

and a fast version,

$$t_{\mathrm{F}}(j,N) = 0.01^{j/N}. \tag{2-3}$$

Their values depend on the current iteration index j and the maximum number of iterations N. The above three functions are illustrated in Figs. 2-1 to 2-3.

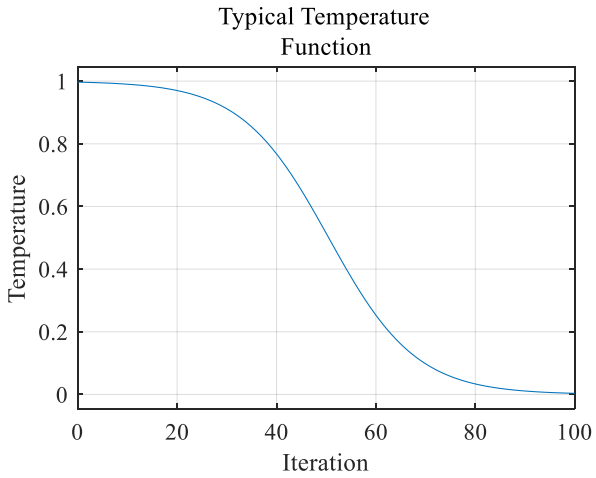


Fig. 2-1. Typical temperature function (2-1) for N = 101.

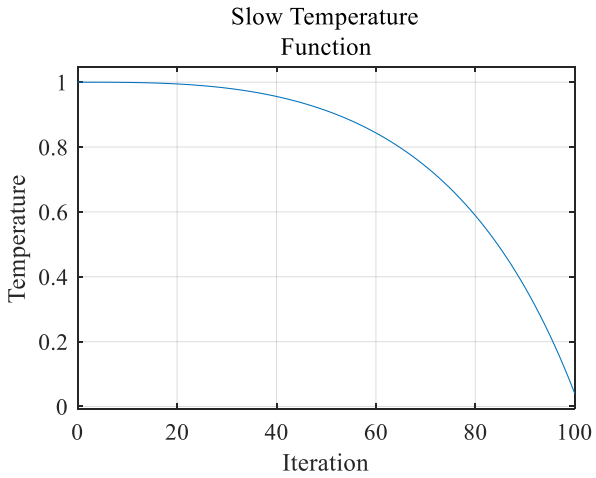


Fig. 2-2. Slow temperature function (2-2) for N = 101.

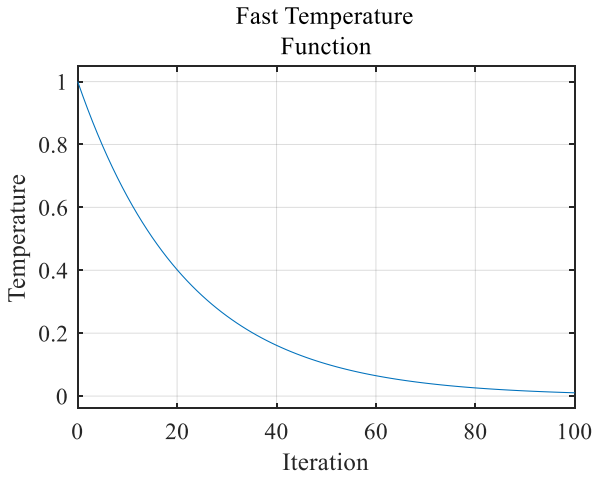


Fig. 2-3. Fast temperature function (2-3) for N = 101.

2.3.2 Step-size Functions

Step-size functions determine the distance between the last accepted $\boldsymbol{x}$ value ($\boldsymbol{x}_i$), and that one generated in the current iteration ($\boldsymbol{x}_j$). The values returned by this function increase in proportion to temperature. Following these guidelines, a typical step-size function is defined by

$$\boldsymbol{s}_{\mathrm{T}}(\boldsymbol{t}) = \boldsymbol{t}\,, \tag{2-4}$$

a slow step-size function by

$$s_{\mathrm{S}}(t) = \sqrt{t}\,, \tag{2-5}$$

and a fast step-size function by

$$s_F(t) = t^2. \tag{2-6}$$

Notice that the output range for all these functions is limited to $0 < s \leq 1$. These three functions are illustrated in Figs. 2-4 to 2-6.

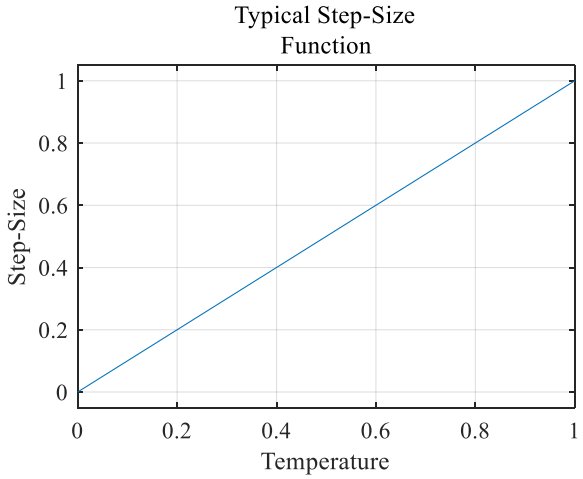


Fig. 2-4. Typical step-size function (2-4).

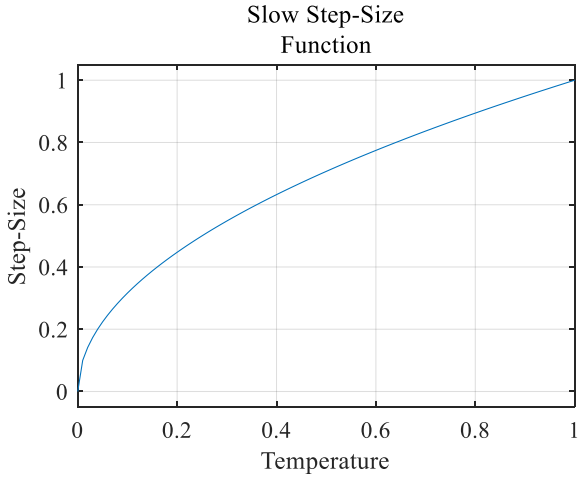


Fig. 2-5. Slow step-size function (2-5).

2.3.3 Acceptance Functions

Acceptance functions define how the difference Δu between the objective function value u_i, and that one from the current iteration u_j, interacts with the temperature value t to determine the

probability for $\boldsymbol{x}_j$ to be accepted as the new $\boldsymbol{x}_i$ for the next iteration. These functions must behave in such a way that the probability P decreases when the temperature decreases and increases when $\boldsymbol{x}_j$ is better than $\boldsymbol{x}_i$. In addition, P must be bounded within the range $0 < P < 1$ (because it is a probability function) [Frausto-Solis-14] [Henderson-03].

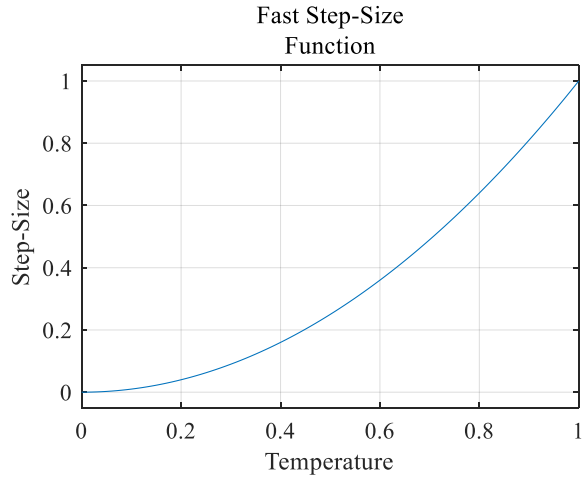


Fig. 2-6. Fast step-size function (2-6).

With these guidelines, a set of three functions which are based on the Boltzmann probability function has been designed: the typical acceptance function

$$a_{\mathrm{T}}(t,\Delta u) = (1+e^{0.7\Delta u/(t^2+6t^{10})})^{-1}; \tag{2-7}$$

the slow acceptance function,

$$a_{\mathrm{S}}(t,\Delta u) = (1+e^{0.7\Delta u/(4t^2+3t^{10})})^{-1}, \tag{2-8}$$

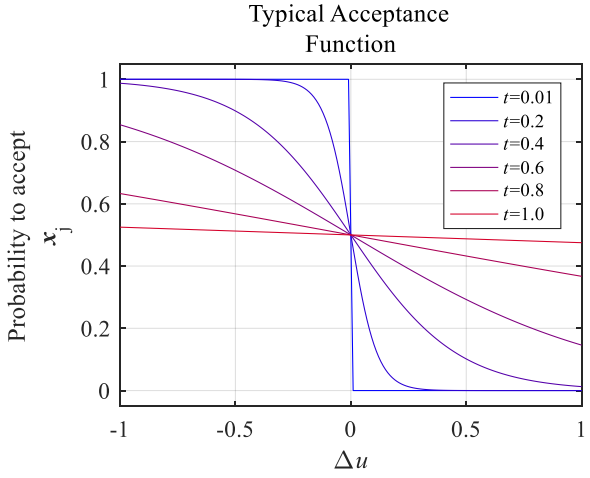


Fig. 2-7. Typical acceptance function (2-7).

which keeps the algorithm behavior stochastic for more iterations; and the fast acceptance function,

$$a_F(t,\Delta u) = (1+e^{0.7\Delta u/(0.4t^2+6.6t^{12})})^{-1}, \tag{2-9}$$

that makes the algorithm reach its deterministic behavior sooner than the typical acceptance function.

Functions (2-7) to (2-9) are illustrated in Figs. 2-7 to 2-9.

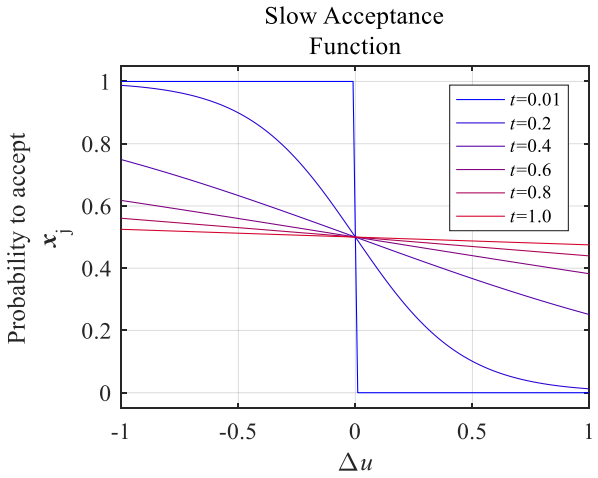


Fig. 2-8. Slow acceptance function (2-8).

2.4. Experiment Execution and Results

The experiment consisted of running SA over the described objective function, using the default configuration and all the first-order configurations. The measured mean-error for each configuration is shown in Table VII and Fig. 2-10. Notice that the smallest mean error is given by the run made with the default configuration, so there is no need to try the second-order configurations. In Fig. 2-10, the samples for each SA configuration are separated across the horizontal axis: the outliers of each sample are marked with dots, each quartile is represented with a horizontal line (together with the sample median, which is represented with a horizontal line within the box), and the sample mean is pointed by a cross. The sample means have been connected in order to facilitate their comparison.

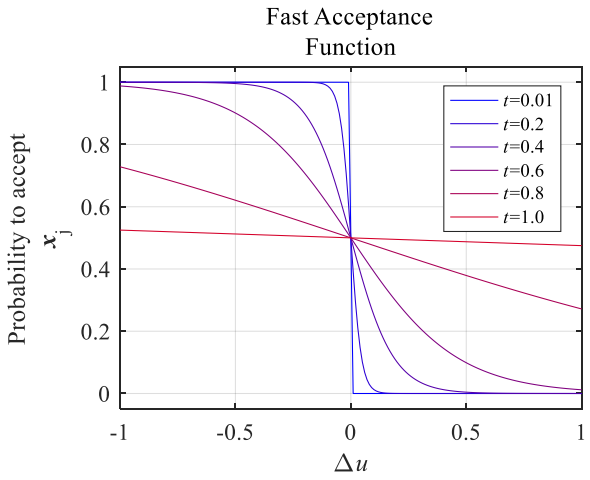


Fig. 2-9. Fast acceptance function (2-9).

TABLE VII
RESULTS OF THE SA ADAPTATION EXPERIMENT FOR THE HIGH-FREQUENCY CIRCUIT DESIGN OPTIMIZATION PROBLEM

SA Configuration	Mean Error
TTT	0.1129
STT	0.0823
FTT	0.0181
TST	0.0585
TFT	0.1083
TTS	0.0888
TTF	0.0171
FSF	0.0968

An F-test [McClave-12] was performed to verify that the results between tested configurations differ because the control functions were changed, and not because the intrinsic variability of SA. For this, the null hypothesis was defined to state that the different test-cases are not relevant for the observed result variation (which means that the variance between test-cases would be significantly smaller than the variance within each test-case). A significance level of α = 0.05 was used, which is mapped to a critical F-value of F_c = 2.1421 given that there are seven

test cases, each one was sampled thirty times. This test resulted in an $F = 2.3564$. Therefore, as $F > F_c$, the null hypothesis is rejected and the conclusion is that changing the control functions caused the differences between the results.

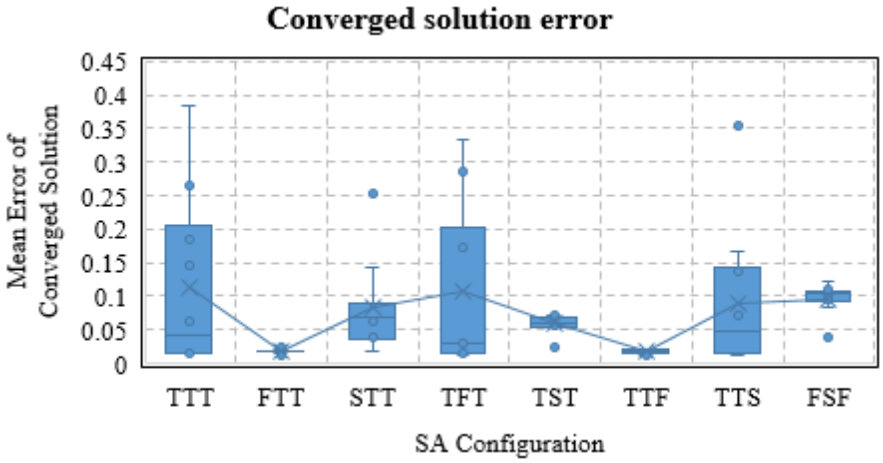


Fig. 2-10. Box chart of the converged solutions for the high-frequency circuit design optimization problem.

The best improvement in the results are observed when using the TTF configuration. Such improvement represents an improvement of ~84.8539% compared with the default configuration TTT.

When this methodology is applied to another family of objective functions, the conclusions in terms of the best configuration settings for SA are not necessarily the same. For instance, the same test-cases were reproduced over the periodic and exponential objective function

$$u_{\text{perexp}}(\boldsymbol{x}) = -\sin^2\left(\frac{4\pi x_1}{13}\right)\cos^2\left(\frac{4\pi x_2}{13}\right)e^{\left(\frac{x_1+x_2}{10}\right)}, \tag{2-10}$$

and the results are shown in Table VIII and Fig 2-11. In this case, there are three first-order configurations (STT, TST, and TTF) that have a smaller mean error than the typical configuration. Once they were combined into the SSF third-order configuration, an even smaller mean error value was gotten as compared to the rest of the configurations.

2.5. Conclusions

A methodology that improves the accuracy of SA by modifying its temperature, step-size,

and acceptance functions was shown. Additionally, by observing the interaction of such functions, it is possible to identify better configurations that involve changing more than one function simultaneously (second or third order configurations).

TABLE VIII
RESULTS OF THE SA ADAPTATION EXPERIMENT FOR THE PERIODIC AND EXPONENTIAL FUNCTION

SA Configuration	Mean Error
TTT	0.1705
STT	0.1665
FTT	0.2323
TST	0.1371
TFT	0.2806
TTS	0.1932
TTF	0.1640
SSF	0.0679

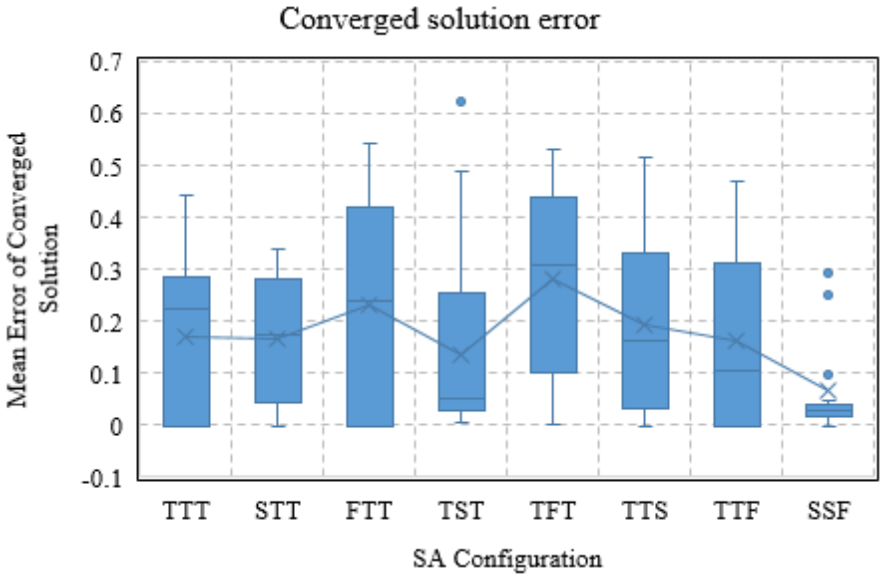


Fig. 2-11. Box chart of the converged solutions for the periodic and exponential objective function (10).

The SA configuration that is chosen as the best for the objective function used to execute

the methodology may not be the best one for other types of objective functions. This means that, in order to get the best SA accuracy, it is better to execute this methodology once per each objective function type or family (unless the differences between the objective functions are negligible).

This methodology can be extended by designing new control functions for SA, identifying their corner cases, and constructing configurations with them. For example, there is literature in which the authors decide to experiment with different acceptance functions than the Boltzmann-based functions [McClave-12].

3. Validation of the Simulated Annealing Configuration Methodology by Optimizing a Microstrip Filter

In this chapter, the Simulated Annealing (SA) configuration methodology presented in Chapter 2 is validated. It is used to find the best SA configuration for minimizing the response error of a microstrip filter with respect to its design specifications. To verify its portability, such configuration is used for optimizing the same microstrip filter but using different discretization resolutions. This is particularly relevant for direct optimization of coarsely discretized full-wave EM models.

3.1. Introduction

Our objective in this work is to validate that the Simulated-Annealing (SA) configuration found with the methodology defined in Chapter 2 improves the optimization results when compared to other SA configurations. Additionally, the portability of the found configuration is validated by using it to optimize the same microstrip filter but with different accuracy in the discretization resolution. This is especially important for optimizing coarsely discretized full-wave EM models, where the usage of direct optimization can be still computationally affordable.

First, an objective function that quantifies the response error of a microstrip filter with respect to its design specifications is defined. Then, three versions of such objective function are implemented by varying the resolution of the simulation model used to generate the filter response data. The models with lower resolution lead to low-accuracy response data with a low-computational cost. In contrast, a higher-resolution model takes longer to be simulated, but generates more accurate response data.

Then the methodology for finding the best SA configuration is used for minimizing the value of the objective function that corresponds to the lowest-resolution simulation model. Once such configuration is found, it is used for optimizing the other two versions of the objective function that consider higher-resolution simulation models. Per our hypothesis, the same SA configuration must lead to a successful filter design optimization for the three implementations of

the objective function.

3.2. The Microstrip Filter To Be Optimized

3.2.1 Microstrip Filter Description

The considered low-pass filter (taken from [Sheen-90] and [D'Inzeo-79]), shown in Fig. 3-1. It consists of two 50-Ω feeding microstrip lines that are symmetrically separated by a microstrip segment of length S_1 and width W_1. Such segment is terminated at both ends with an open stub of length L_1.

The filter feeding lines have a width W_p = 2.45 mm and a length $L_p = 2W_1$. The substrate has a thickness H = 0.794 mm, a relative dielectric permittivity of ε_r = 2.2, and a loss tangent $\tan(\delta)$ = 0.01. The microstrip traces are made with a metal with conductivity σ = 58 MS/m and thickness t = 15.24 μm.

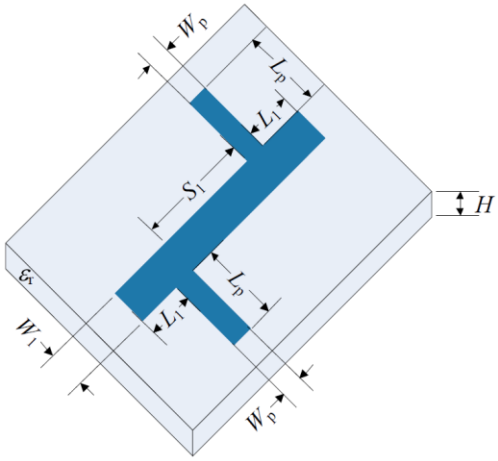


Fig. 3-1. The microstrip low-pass filter to be optimized.

3.2.2 Design Specifications

The magnitude of the transmission coefficient of the filter, $|S_{21}|$, must be greater than 0.9 for frequencies lower than 6 GHz. Additionally, it must be equal or less than 0.1 for frequencies within the range from 8 GHz to 10 GHz. This must be achieved by varying W_1, S_1, and L_1.

3.2.3 Sonnet Implementation

The simulation model implementation is based on that one used in [Rayas-Sánchez-12]. Both feeding lines are in contact with the Sonnet simulation box bounds, while the stubs are separated from the box walls by a distance $x_{gap} = 12H$. Additionally, all the box walls are implemented with lossless metals. And the top wall is separated from the filter structure by a layer filled with air, whose height is $H_{air} = 18H$.

The discretization resolution in the y-direction is $y_{res} = W_1 / 2$, while the resolution in the x-direction depends on the value of the parameter k as

$$x_{res} = W_p / k \,. \tag{3-1}$$

In each Sonnet simulation, the value of $|S_{21}|$ is captured across a set of frequency points that are linearly distributed in the range between 0.2 GHz and 10 GHz, using steps of 0.2 GHz.

3.3. Design of the Objective Function

Let vector $\boldsymbol{r}$ refer to the simulated filter response data. Its elements, denoted by r_i, contain the value of $|S_{21}|$ at each frequency point f_i, for the frequency vector $\boldsymbol{f} = [0.2 \quad 0.4 \quad \ldots \quad 9.8 \quad 10.0]^T$ (GHz).

The error vector $\boldsymbol{e}$ is defined to quantify the filter response error for each element in $\boldsymbol{r}$. Its elements, denoted by e_i, contain the normalized error value between the specification requirements for $|S_{21}|$ and $\boldsymbol{r}$. Such value is computed using

$$e_i = 1 - \frac{r_i}{0.9} \quad \text{for } i : \{ f_i \le 6\,\text{GHz} \}\,, \tag{3-2}$$

and

$$e_i = \frac{r_i}{0.1} - 1 \quad \text{for } i : \{ 8\,\text{GHz} \le f_i \le 10\,\text{GHz} \}\,. \tag{3-3}$$

This means that when all the elements within $\boldsymbol{e}$ are zero or negative, the specifications are fully satisfied.

The objective function value, u, is defined to be equal to the maximum-value element within $\boldsymbol{e}$ for a given value of the optimization variables in vector $\boldsymbol{x} = [W_1 \quad S_1 \quad L_1]^T$ (mm). Notice

that the optimization of such function consists of minimizing u by varying such variables. Finally, three versions of this objective function are defined. Such versions differ only in the value of the parameter k, denoted by $u_k(\boldsymbol{x})$ for $k \in \{2, 4, 8\}$, which correspond to the selected discretization resolutions.

For all our experiments, the initial value of $\boldsymbol{x}$ is considered to be $\boldsymbol{x}_0 = [1.2220 \;\; 5.4050 \;\; 2.5944]^T$ (mm). The elements of $\boldsymbol{x}$ can vary within the restricted range of ±30% around $\boldsymbol{x}_f^* = [1.5469 \;\; 4.7000 \;\; 2.5188]^T$ (mm). This $\boldsymbol{x}_f^*$, taken from [Rayas-Sánchez-12], satisfies the filter design specifications using a filter model for the COMSOL multi-physics simulator, as described in [Rayas-Sánchez-12]. Table IX shows the maximum and minimum values of each design variable after considering such restriction.

TABLE IX
DESIGN VARIABLES VALUE RANGES

Variable	Minimum (mm)	Maximum (mm)
W_1	1.0822	2.0110
S_1	3.2900	6.1100
L_1	1.7632	3.2744

The objective function only accepts input values in which the design variables lead to a structure that fits exactly in the simulation grid in Sonnet. This means that S_1 and L_1 must be an exact multiple of x_{res}, while W_1 can vary freely given that it is used to calculate y_{res}.

Our SA implementation generates continuous, and normalized inputs ($\boldsymbol{x}_{SA}$) whose elements vary between −1 and 1. Therefore, it is necessary to use an interface function g to discretize and de-normalize the values of the design variables (mathematically, $\boldsymbol{x} = g(\boldsymbol{x}_{SA})$).

3.4. Experiments and Results

First, the values of $u_k(\boldsymbol{x}_0)$ and the simulation times ($t_{sim,k}$) for $k = 2$, $k = 4$, and $k = 8$ are registered. Then, the SA configuration methodology is executed using $u_2(\boldsymbol{x})$ to find both $\boldsymbol{x}_2^*$, and the SA configuration that leads to the lowest u_2 value. Finally, the resulting SA configuration is

re-used to optimize $u_4(\boldsymbol{x})$ and $u_8(\boldsymbol{x})$ to verify if they can satisfy the design specifications with such configuration.

The resulting values of u_k and $t_{\text{sim},k}$ are shown in Table X, while the evolution of the u_k value during the SA optimization is shown in Fig. 3-2 for each $k = 2$, $k = 4$, and $k = 8$. It was found that the simulation time increases with a trend given by $t_{\text{sim}} = 1.25k + 5$ for this particular setup.

After executing the SA configuration methodology over the $u_2(\boldsymbol{x})$ objective function (using 10 optimizations per configuration, and 50 SA iterations per optimization) it was found that the SSS configuration, as defined in Chapter 2, produced the best optimization results (see Table XI). The filter response for the ten optimizations that were run with such configuration are shown in Fig. 3-3. The average value of the optimized objective function was $u_2{}^*_{\text{avg}} = -0.0072$. And the difference between $u_2(\boldsymbol{x}_0)$ and the best $u_2{}^*$ value was 0.0437.

TABLE X
VALUES OF $u_k(\boldsymbol{x}_0)$ AND SIMULATION TIMES FOR DIFFERENT DISCRETIZATION RESOLUTIONS (VALUES OF k)

k	$u_k(\boldsymbol{x}_0)$	t_{sim1} (s)	t_{sim2} (s)	t_{sim3} (s)
2	0.0365	7.2470	7.5900	7.3530
4	0.1289	9.6780	11.242	10.883
8	0.1333	15.199	15.400	14.940

Finally, the SSS configuration was used to optimize $u_4(\boldsymbol{x})$ and $u_8(\boldsymbol{x})$. The optimization results are summarized in Table XII, while the filter responses before and after optimization for each objective function are shown in Fig 3-4. Notice that all the optimizations resulted in filter designs that satisfy the design specifications.

3.5. Conclusions

Given that the three versions of the objective function were successfully optimized using the configuration found with the methodology described in Chapter 2, we have demonstrated in practice that this methodology improves the SA optimization results, and that configuration found is reusable for optimizing the same problem using models with different accuracies.

In this case of study, using our methodology has an extra benefit that consists on saving time by running the methodology over an objective function calculated from a low computational cost EM model, and reusing the found configuration for optimizing objective functions from higher computational cost EM models, without executing the methodology again (which would be slower).

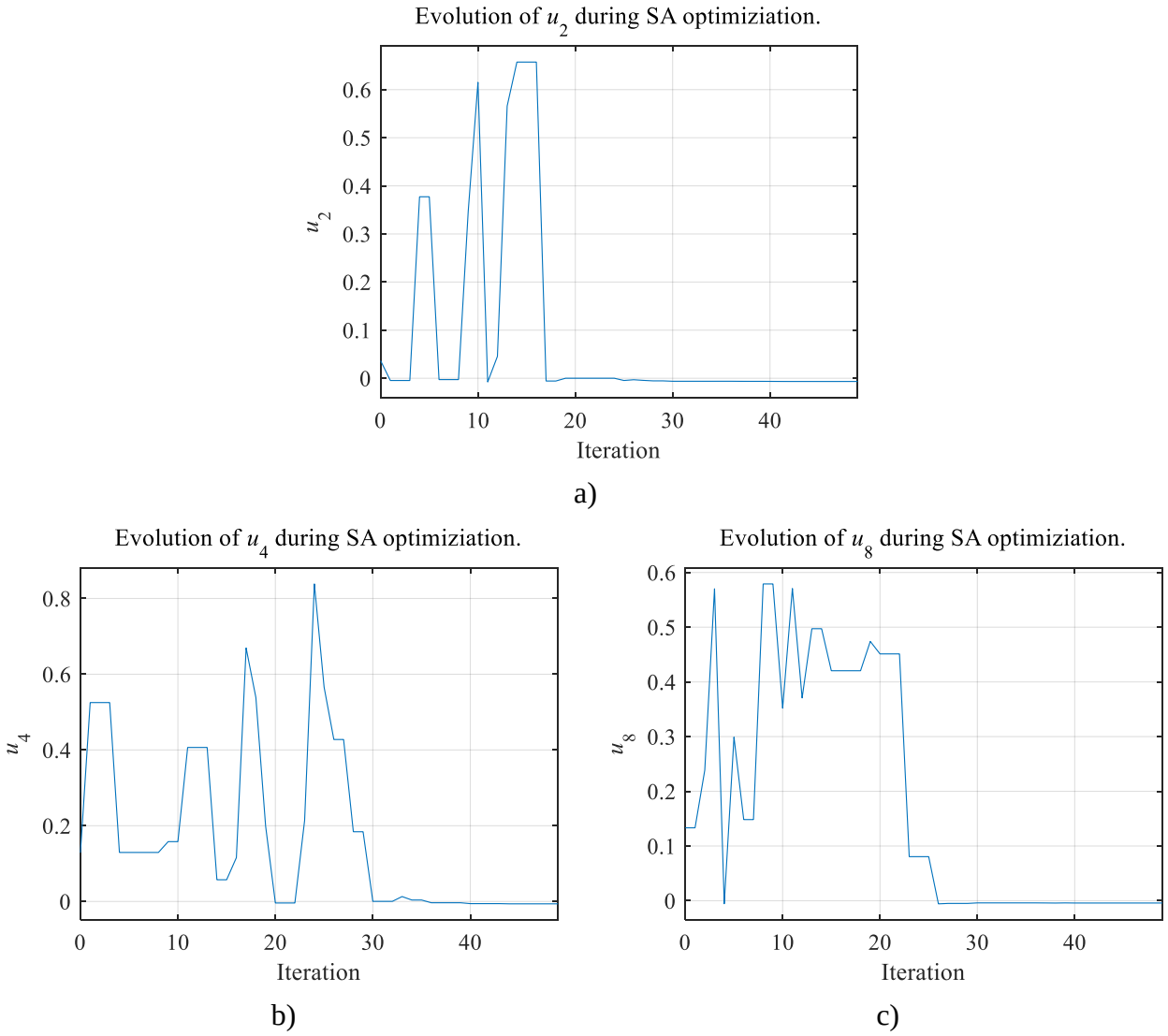


Fig. 3-2. Evolution of the value of u_k during the SA optimization for different discretization resolutions: a) k = 2, b) k = 4, c) k = 8.

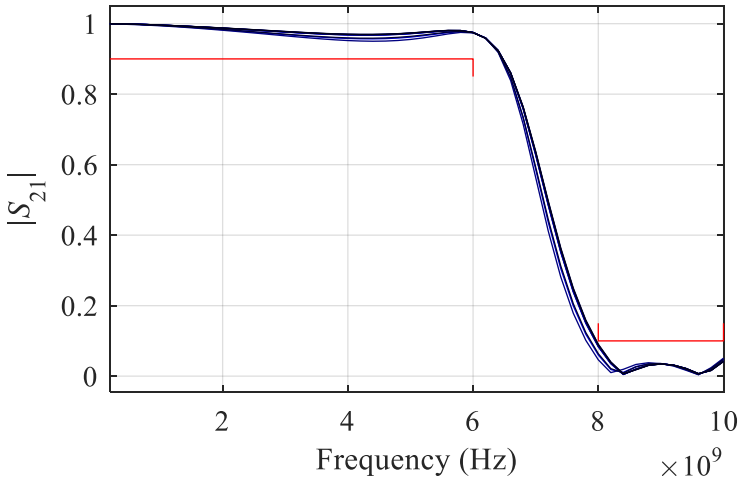


Fig. 3-3. Filter responses at the ten optimal solutions for u_2 found after running SA with the SSS configuration (blue), and the filter design specifications (red).

TABLE XI
AVERAGE OBJECTIVE FUNCTION VALUES FOR EACH SA CONFIGURATION

SA Configuration	$u_2^*{}_{avg}$
TTT	− 0.0066
STT	− 0.0061
FTT	− 0.0059
TST	− 0.0071
TFT	− 0.0023
TTS	− 0.0027
TTF	− 0.0023
SSS	− 0.0072

And last, but not least, it was shown that this methodology works for high-frequency design optimization problems which consider multiple design variables, and coarsely discretized electromagnetic simulation models.

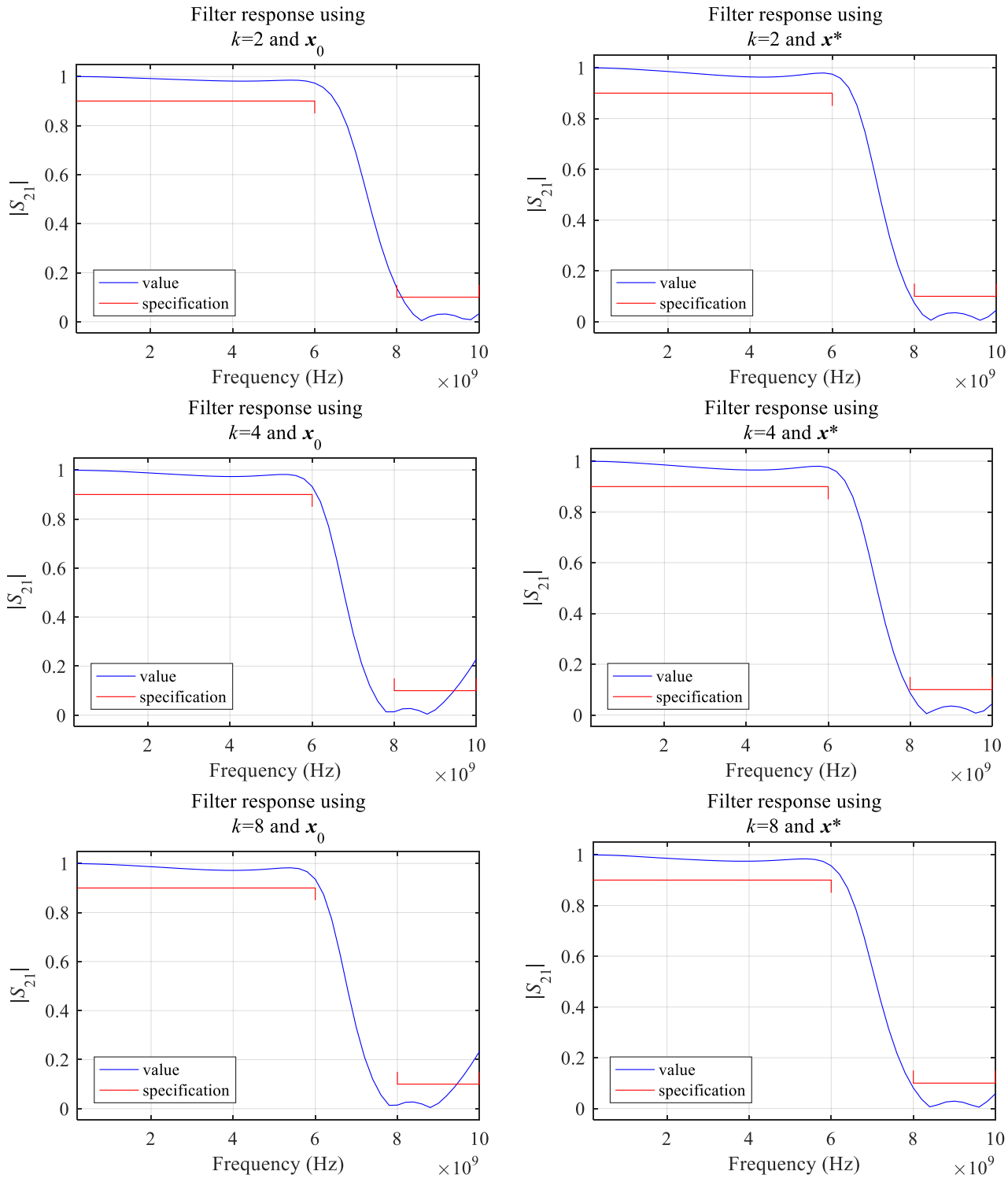


Fig. 3-4. Filter response before and after optimization for the three different resolutions, using SA configured with the SSS profile.

TABLE XII
OPTIMIZATION RESULTS FOR THE SSS CONFIGURATION FOR DIFFERENT DISCRETIZATION RESOLUTIONS (VALUES OF k)

k	$u_k(\boldsymbol{x}_k^*)$	$\boldsymbol{x}_k^*$ (mm)
2	–0.0077	$[\ 1.5242 \quad 4.9000 \quad 2.4500]^T$
4	–0.0072	$[\ 1.4564 \quad 4.9000 \quad 2.4500]^T$
8	–0.0063	$[\ 1.3750 \quad 4.9000 \quad 3.0625]^T$

4. Comparing the Performance of the Calibrated Simulated Annealing against Nelder-Mead, Sequential Quadratic Programming, and Genetic Algorithms

In this chapter, the same low pass microstrip filter used in the preceding chapter is optimized. This time, it is done by using the MATLAB implementations for Sequential Quadratic Programming (SQP), Nelder-Mead (NM), and Genetic Algorithm (GA). Then, the new results are compared against those obtained from our implementation of Simulated Annealing in the previous chapter.

SQP is a gradient/Jacobian-based algorithm. NM is classified as a heuristic direct search algorithm, and uses a pattern-based search across the objective function domain. GA is an evolutionary, population-based global optimization algorithm, inspired by the process of natural selection. Within MATLAB, SQP, NM and GA algorithms are implemented as the *fminimax*, *fminsearch*, and *ga* functions, respectively. More information about each of these algorithms can be found in [MathWorks-17-a], [MathWorks-17-b], and [MathWorks-17-c].

As explained in the previous chapter, the optimization problem was solved using three different objective functions, which differ by the resolution used for the full-wave electromagnetic simulations to get the filter response.

4.1. Algorithms Setup

As specified in Chapter 2, the presented SA implementation drives normalized objective function inputs in the range from −1 to 1. Therefore, to apply such constrains to SQP and GA, their lower/upper bounds input parameters are used to force their objective function input generation to the mentioned range (NM MATLAB implementation does not allow constrained optimization).

In the case of SQP and GA, it is mandatory to specify the inequality and equality matrices due to the propositional parameter specification in MATLAB (see [MathWorks-17-a], and [MathWorks-17-c]). In both cases, zero-matrices and vectors are used respectively to avoid any

undesired change in the behavior of the algorithms. The non-linear constrain functions were also configured to return zero with the same purpose.

Given the fact that SQP uses derivative calculations by finite differences, the differential step to is defined to be 0.001. If it is left at its default value (which is 10^{-6}), the derivative calculations are incorrect because the differential step is not large enough to provoke changes in the objective function (due to the simulation resolution).

For NM and SQP, the same seed values used in Chapter 2 are used. The seed values for GA are not specified given the fact that it uses an initial population rather than an initial point.

4.2. Results and Discussion

The objective function values at the seed are shown in Table XIII. The optimization results given by SA (from Chapter 3), NM, SQP, and GA are in tables XIV, XV, XVI, and XVII, respectively. Each table contains the final objective function value obtained by each algorithm, and the number of iterations that took each algorithm to reach such value.

NM showed a stable behavior across all the variants of the objective function, being able to reach negative values always with an average number of 110 function evaluations, as seen in Table XV.

TABLE XIII
OBJECTIVE FUNCTION VALUES AT THE SEED

k	$u_k(\boldsymbol{x}_0)$
2	0.0365
4	0.1289
8	0.1333

SQP is very sensitive to the simulation resolution due to the gradient computations. This leaded to different behaviors of the algorithm across the three versions of the objective function: both in terms of the number of function evaluations before its convergence, and the final objective function value, as seen in Table XVI. It was not even able to get a negative value for u_8.

Finally, GA got the most negative objective function values (see Table XVII), but at the cost of evaluating the objective function 1255 times (because of its population-based process). Even though SA did not get the most negative values (see Table XIV), it took the smallest number of function evaluations (50), and the found objective function value was always negative.

TABLE XIV
OPTIMIZATION RESULTS GIVEN BY SIMULATED ANNEALING (WITH SSS CONFIGURATION)

k	$u_k(\boldsymbol{x}_k^*)$	Function Evaluations
2	–0.0077	50
4	–0.0072	50
8	–0.0063	50

TABLE XV
OPTIMIZATION RESULTS GIVEN BY NELDER-MEAD

k	$u_k(\boldsymbol{x}_k^*)$	Function Evaluations
2	–0.0077	121
4	–0.0074	108
8	–0.0071	101

TABLE XVI
OPTIMIZATION RESULTS GIVEN BY SEQUENTIAL QUADRATIC PROGRAMMING

k	$u_k(\boldsymbol{x}_k^*)$	Function Evaluations
2	–0.0077	405
4	–0.0074	90
8	0.0099	23

TABLE XVII
OPTIMIZATION RESULTS GIVEN BY GENETIC ALGORITHM

k	$u_k(\boldsymbol{x}_k^*)$	Function Evaluations
2	–0.0077	1255
4	–0.0074	1255
8	–0.0075	1255

4.3. Conclusions

Once it has been configured for this objective function, our SA implementation had the least computational cost to achieve very acceptable results. By sacrificing the computational cost such calibration requires, it would be possible to optimize similar objective functions faster than any of the other three algorithms considered in this chapter.

NM is still a better option when there is no interest in solving similar optimization problems. Its results were almost as accurate as the ones obtained from GA, but they were reached with much less function evaluations.

Another benefit from our SA implementation is that it can yield acceptable results in a short number of iterations (although global optimum is not guaranteed). This means that it can be used for identifying good search regions for further optimization by other algorithms or by another SA run.

General Conclusions

In this thesis, the consistency and efficiency of SA have been improved by adding controls to its step-size, normalizing its search range, changing the control functions (for temperature, step-size, and solution acceptance probability), and applying a methodology to select their configuration. These features are added to improve SA's capacity to escape from local minima, being a global search optimization algorithm. As a result, we end up with an improved version of SA.

The enhanced SA was verified to work properly with discretized search spaces as well as with noisy objective functions. Therefore, it becomes an appropriate choice for optimizing coarsely discretized EM models, which involve both: low resolution in the design variables, and discretization noise in their simulated response. Moreover, given the fact that such models are simulated much faster than the corresponding EM models with full resolution discretization (fine models), the improved SA enables an accelerated high-frequency circuit design workflow.

This version of SA is still having the caveat of not guaranteeing the best solution or global minimum. However, it is still appropriate for exploring large design spaces, and looking for seed values to feed other optimization algorithms, such as the Nelder-Mead method. In contrast, using such algorithms directly could lead to getting stuck in local optima, or consume much more computational resources due to the size of the search space.

Future work could extend the configuration methodology by considering alternative functions for configuring the temperature, step-size, or solution acceptance probability. Moreover, further research could focus on quantifying their effects as a function of the controlled parameters. As a result, this might lead to a methodology that improves the trial-and-error based approach proposed in this work.

Appendix

A. LIST OF INTERNAL RESEARCH REPORTS

[1] J. R. Alejos-Jiménez and J. E. Rayas-Sánchez, "Basic concepts on simulated annealing and its applications," Internal Report *CAECAS-15-12-R*, ITESO, Tlaquepaque, Mexico, Aug. 2015.

[2] J. R. Alejos-Jiménez and J. E. Rayas-Sánchez, "Configuring simulated annealing for high-frequency design optimization," Internal Report *CAECAS-16-05-R*, ITESO, Tlaquepaque, Mexico, Jun. 2016.

[3] J. R. Alejos-Jiménez and J. E. Rayas-Sánchez, "Validation of the simulated annealing configuration methodology using a microstrip filter design optimization problem," Internal Report *CAECAS-16-13-R*, ITESO, Tlaquepaque, Mexico, Dec. 2016.

[4] J. R. Alejos-Jiménez and J. E. Rayas-Sánchez, "Comparing the performance of the calibrated simulated annealing against Nelder-Mead, sequential quadratic programming, and genetic algorithms," Internal Report *CAECAS-17-04-R*, ITESO, Tlaquepaque, Mexico, May 2017.

Bibliography

[D'Inzeo-79] G. D'Inzeo, F. Giannini, and R. Sorrentino, "Novel microwave integrated low-pass filters," *Electron. Lett.*, vol. 15, no. 9, pp. 258-260, Apr. 26, 1979.

[Dorigo-06] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28-39, Nov. 2006.

[Dymond-15] A. S. Dymond, A. P. Engelbrecht, S. Kok, and P. S. Heyns, "Tuning optimization algorithms under multiple objective function evaluation budgets," *IEEE Trans. Evolutionary Computation*, vol. 19, no. 3, pp. 341-358, Jun. 2015.

[Frausto-Solis-14] J. Frausto-Solis, E. Liñán-García, and G. Santamaria-Bonfil, "Tuned simulated annealing based on Boltzmann and Bose-Einstein distribution applied to MAXSAT problem," *Journal of Asian Scientific Research*, vol. 4, no. 1, pp. 14-26, Jan. 2014.

[Gall-14] J. Gall, *Computer Vision. A Reference Guide*, Tübingen, Germany: Springer, 2014.

[Henderson-03] D. Henderson, S. H. Jacobson, and A. W. Johnson, "The theory and practice of simulated annealing," in *Handbook of Metaheuristics*, F. W. Glover and G. A. Kochenberger, Ed. USA: Springer, 2003, ch. 10, pp. 287-319.

[MathWorks-17-a] MathWorks. (2017). *MATLAB Optimization Toolbox documentation: fminimax* [Online]. Available: https://www.mathworks.com/help/optim/ug/fminimax.html.

[MathWorks-17-b] MathWorks. (2017). *MATLAB documentation: fminsearch* [Online]. Available: https://www.mathworks.com/help/matlab/ref/fminsearch.html.

[MathWorks-17-c] MathWorks. (2017). *Global Optimization Toolbox documentation: ga* [Online]. Available: https://www.mathworks.com/help/gads/ga.html.

[McClave-12] J. McClave and T. Sincich, "Analysis of variance: comparing more than two means," in *Statistics*, J. McClave and T. Sincich, Ed. Harlow, UK: Pearson, 2012, ch. 10, pp. 491-568.

[Poli-07] R. Poli, J. Kennedy, and Tim Blackwell "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33-57, Jun. 2007.

[Rayas-Sánchez-12] J. E. Rayas-Sánchez and Z. Brito-Brito, "Broyden-based input space mapping optimization of a microstrip low-pass filter implemented in COMSOL," Internal Report *CAECAS-12-21-R*, ITESO, Tlaquepaque, Mexico, Oct. 2012.

[Rossmanith-11] P. Rossmanith, "Simulated Annealing," in *Algorithms Unplugged*, Vöcking, B., Alt, H., Dietzfelbinger, M., Reischuk, R., Scheideler, C., Vollmer, H., Wagner, D., Ed. Berlin, Germany: Springer, 2011, ch. 41, pp. 393-400.

[Sheen-90] D. M. Sheen, S. M. Ali, M. D. Abouzahra, and J. A. Kong, "Application of the three-dimensional finite-difference time-domain method to the analysis of planar microstrip circuits," *IEEE Trans. Microwave Theory Tech.*, vol. 38, no. 7, pp. 849-857, July 1990.

[Vanderbilt-84] D. Vanderbilt and S. G. Louie, “A Monte Carlo simulated annealing approach to optimization over continuous variables,” *Journal of Computational Physics*, vol. 56, no. 2, pp. 259-271, Nov. 1984.

[Whitley-94] Darrel Whitley, “A genetic algorithm tutorial,” *Statistics and Computing*, vol. 4, no. 2, pp 64-85, June 1994.

Index

T