J Optim Theory Appl (2015) 164:842–861
DOI 10.1007/s10957-013-0441-2

Derivative-Free Robust Optimization for Circuit Design

Angelo Ciccazzo · Vittorio Latorre · Giampaolo Liuzzi · Stefano Lucidi · Francesco Rinaldi

Received: 16 January 2013 / Accepted: 28 September 2013 / Published online: 17 October 2013

Abstract In this paper, we introduce a framework for derivative-free robust optimization based on the use of an efficient derivative-free optimization routine for mixed-integer nonlinear problems. The proposed framework is employed to find a robust optimal design of a particular integrated circuit (namely a DC–DC converter commonly used in portable electronic devices). The proposed robust optimization approach outperforms the traditional statistical approach as it is shown in the numerical results.

Keywords Robust optimization · Derivative-free methods · Circuit design

A. Ciccazzo
ST Microelectronics, Stradale Primosole 50, 95121 Catania, Italy
e-mail: angelo.ciccazzo@st.com

V. Latorre · S. Lucidi
Dipartimento di Ingegneria Informatica, Automatica e Gestionale "A. Ruberti", "Sapienza" Università di Roma, Via Ariosto 25, 00185 Rome, Italy

V. Latorre
e-mail: latorre@dis.uniroma1.it

S. Lucidi
e-mail: lucidi@dis.uniroma1.it

G. Liuzzi
Istituto di Analisi dei Sistemi ed Informatica (IASI) "A. Ruberti", CNR, Viale Manzoni 30, 00185 Rome, Italy
e-mail: giampaolo.liuzzi@iasi.cnr.it

F. Rinaldi (✉)
Dipartimento di Matematica, Università di Padova, Via Trieste 63, 35121 Padova, Italy
e-mail: rinaldi@math.unipd.it

1 Introduction

In the last decades, the complexity of digital integrated circuits (ICs) has been increasing exponentially, with the number of components or devices in a single IC more than doubling every year. Some current ICs contain over 100 million devices and a similar number of wires connecting them and have dimensions of the order of nanometers. As a consequence of the scaling of device dimensions to nanometer size, we face a multitude of challenges in designing complex circuits. Major challenges come from parametric variation such as intra- and inter-die variation of channel length, oxide thickness, and doping concentration. These phenomena have a direct and deep impact on system performance, resulting in parametric yield loss and reduced system lifetime. Numerous approaches have been proposed to address these effects and to increase parametric yield. Circuit design optimization techniques target to solve some design problems by describing them in terms of decision variables (continuous or integer), constraints on the variables (physical relationships, budget limitations, performance requirements, and so on), and one or more objective functions to be maximized or minimized (yield is an example). In this way, the design process is modeled by a complex and large-scale optimization problem, often integrated with a simulation tool. Various papers have been devoted to the use of optimization methods in the context of circuit design. We give here some examples of such papers. In [1] trust-region-type algorithms have been proposed for the optimization of circuits. Geometric programming is used in [2] for digital circuit optimization. A robust approach has been proposed in [3] for the sizing of digital circuits. In [4] a framework for optimizing dynamic control of various self-tuning parameters over lifetime in the presence of circuit aging is defined. For more details about optimization methods in circuit design, we address the interested reader to the book [5].

Uncertainty is commonly present in circuit design problems (due to the parametric variations of the circuit) and often lead to suboptimal (or even infeasible) solutions. There exist two different classes of perturbations:

(1) *implementation errors,* mainly caused by imperfect realization of the circuit variables;
(2) *parameter uncertainties,* which are due to modeling errors during the problem definition (e.g., noise).

Stochastic optimization approaches represent the classic way for dealing with uncertainty in real-world applications. In this case, the probability distributions of the uncertainties are estimated and directly incorporated into the model. In recent years, *robust optimization approaches* have become very popular, and a considerable literature has been developed both for solving convex [6–9] and nonconvex problems [10–13]. The goal in robust optimization is finding the design with the best worst-case performance. In this context, min–max approaches are usually adopted to find a robust optimal design with best worst-case performance. In almost all the existing approaches, the authors assume that objective functions and constraints are given explicitly or that there exists a tool providing objective functions and constraints of the problem at a given point as well as their gradients. On the contrary, in [13] a method is proposed that does not use any first-order information.

In this paper, we assume that

- no first-order information is available (which is often the case in simulation-based optimization problems);
- function evaluations are costly and noisy, so that finite difference approximation cannot be applied;
- there exists a subset of circuit variables that only assumes discrete values.

We then develop a derivative-free approach for robust optimization and use it to solve a hard real-world circuit design problem.

In Sect. 2, we formally introduce the problem we want to solve and give some details about it. Then, in Sect. 3, we recall the derivative-free algorithm for local mixed-integer optimization proposed in [14], and we show how it can be used to solve the robust optimization problem. In Sect. 4, we apply the proposed approach to optimally design a DC–DC converter and show its efficiency, when compared with a classic stochastic approach, both in terms of number of function evaluations and circuit yield. Finally, we draw some conclusions in Sect. 5.

2 Robust Optimization for Circuit Design

When dealing with real-world derivative-free optimization problems, we first need to specify inputs and outputs. In our case, the inputs are the so-called circuit variables. Circuit variables can be divided into three different classes:

- **Design Variables $\mathbf{x_d}$**: these variables represent transistor geometries (e.g., channel widths and lengths);
- **Operating Variables $\mathbf{x_o}$**: these variables model operating conditions (e.g., supply voltage and temperature);
- **Statistical Variables $\mathbf{x_p}$**: these variables are usually subject to uncertainty due to fluctuations in the manufacturing process (e.g., oxide thickness, threshold voltage, and channel length reduction).

Electronic devices are replicated multiple times on a wafer (a small thin circular slice of a semiconducting material, such as pure silicon, on which an integrated circuit can be formed), and different wafers are produced, but each device cannot be produced in the same way in terms of electrical performance. The main factors that make the fabrication result uncertain are: the imperfections characterizing the masks and tolerances in their positioning, various changing effects of ion plant temperature during production, tolerances in size, etc. Generally, fluctuations' processes produce fluctuations in electrical performance. Consequently, an essential tool for electronic circuit design is represented by the statistical model that formally relates the former to the latter. Typically, the statistical variables are used in the definition of these models. There exist two different classes of statistical variables:

- **Process Variables**: these variables determine the behavior of the instantiated devices, such as junction depths, sheet resistances, dielectric thickness, and doping levels.

- **Geometric Layout Variables**: device length and width, and more specific parameters related to the layout.

It is possible to use different methods for the specification of statistical variables variation (e.g., a uniform distribution with a given relative variation, a uniform distribution with a given absolute variation, and a Gaussian distribution centered at mean value with a given standard deviation).

The outputs, in circuit design problems, are the *Performance Features* of the circuit. Typical examples of performance features are delay, gain, phase, margin, slew rate, etc. Performance Features (p_i, $i = 1, \ldots, m$) usually need to satisfy some constraints (*Performance–Specification Features*):

$$l_i \leq p_i(x_d, x_o, x_p) \leq u_i, \quad i = 1, \ldots, m.$$

We say that the circuit is in *full working order* if all performance–specification features are satisfied.

We focus on the *Performance Centering Problem*. Given a certain number of performance features and the related specifications, we want to design a circuit in full working order. In practice, our goal is that of finding suitable values for the circuit variables, so that performance features meet the given specifications. Now, we give some details about the various phases of the procedure.

The optimal design of a circuit is a very challenging problem that can be characterized by three different phases:

(1) **Nominal Design Optimization**: we try to optimize the performance while keeping fixed operating and statistical variables;
(2) **Worst-Case Optimization**: we minimize the worst-case performance deviation in a given region of the operating variables;
(3) **Yield Optimization**: we finally maximize the yield, the percentage of manufactured circuits that satisfy the performance specification when varying statistical variables.

2.1 Nominal Design Optimization

In this phase, we try to optimize the performances (i.e., we try to find a solution that minimizes the violations of the given specifications), while keeping fixed operating and statistical variables (i.e., $x_o = \bar{x}_o$ and $x_p = \bar{x}_p$). This can be done by solving the following problem:

$$\min_{x_d} \sum_{i=1}^{m} \max\{0, l_i - p_i(x_d, \bar{x}_o, \bar{x}_p)\}^q + \max\{0, p_i(x_d, \bar{x}_o, \bar{x}_p) - u_i\}^q, \qquad (1)$$

where $q \geq 1$. The operating variables x_o are usually fixed to typical values (e.g., $V = 2.3$ V, $T = 27$ °C), whereas the statistical variables x_p are usually fixed to zero. The solution obtained is then used as a starting point for the second phase.

2.2 Worst-Case Optimization

In this second phase, we minimize the worst-case performance deviations from the nominal values. In other words, we want that both typical and worst-case performances satisfy the performance specifications. In order to carry out the worst-case optimization, we need to perform worst-case analysis. In practice, we compute the worst-case performances when the operating parameters can take any value within the feasible region. This is equivalent to solving the following optimization problems:

$$\min_{x_o} p_i(\bar{x}_d, x_o, \bar{x}_p), \qquad \max_{x_o} p_i(\bar{x}_d, x_o, \bar{x}_p), \quad i = 1, \ldots, m. \tag{2}$$

Once we solve these problems, we obtain the worst-case operating variables $\bar{x}_{o,j}$, $j = 1, \ldots, 2m$. In principle, one can perform this operation at anytime during the worst-case optimization phase. Since in our case this analysis is a very time-consuming task and the worst-case operating variables do not change when varying design variables within certain ranges, we decided to perform it only at the beginning.

Now, we can formally state the problem to be solved in the Worst-Case Optimization phase:

$$\min_{x_d} \sum_{j=1}^{l} \alpha_j \sum_{i=1}^{m} \max\{0, l_i - p_i(x_d, \bar{x}_{o,j}, \bar{x}_p)\}^q + \max\{0, p_i(x_d, \bar{x}_{o,j}, \bar{x}_p) - u_i\}^q, \tag{3}$$

where $q \geq 1$ and $l = 2m + 1$, so that the typical and worst cases are included in the summation. Once again, the solution is used as a starting point for the next phase (Yield Optimization).

2.3 Yield Optimization

In this last phase, the goal is maximizing the yield (i.e., the percentage of manufactured circuits that satisfy the performance specification when varying statistical variables). We can formally define the Yield as follows:

$$Y = \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} \delta(x_p) \cdot p\,df(x_p) \cdot dx_p = E\{\delta(x_p)\}$$

with $A_p = \{x_p \mid l \leq p(\tilde{x}_d, \tilde{x}_o, x_p) \leq u\}$ and

$$\delta(x_p) = \begin{cases} 1, & x_p \in A_p, \\ 0, & \text{otherwise.} \end{cases}$$

An estimator for the expectation value is

$$\hat{Y} = \hat{E}\{\delta(x_p)\} = \frac{1}{n_s} \sum_{\mu=1}^{n_s} \delta\big(x_p^{(\mu)}\big) = \frac{n_{ok}}{n_s},$$

where $x_p^{(\mu)}$, $\mu = 1, \ldots, n_s$, are n_s samples drawn from a normal distribution. The use of a normal distribution is quite common in this context (see, e.g., [5]). So, the

estimator is given by the number of the sample elements that satisfy the specifications (n_{ok}) divided by the total number of elements in the sample (n_s).

A first possibility to deal with the problem is the so called *Statistical Approach*. It consists in randomly generating a certain number of vectors, each representing the statistical variables and minimizing the violation of the specifications in both typical and worst case. The problem we want to solve is the following:

$$\min_{x_d} \sum_{\mu=1}^{n_s} \sum_{j=1}^{l} \alpha_j \sum_{i=1}^{m} \max\left\{0, l_i - p_i\left(x_d, \bar{x}_{o,j}, \bar{x}_p^{(\mu)}\right)\right\}^q + \max\left\{0, p_i\left(x_d, \bar{x}_{o,j}, \bar{x}_p^{(\mu)}\right) - u_i\right\}^q \tag{4}$$

with $q \geq 1$, $\bar{x}_p^{(\mu)}$, $\mu = 1, \ldots, n_s$ normally distributed sample elements, and $l = 2m + 1$. Of course, in this case, we need to generate a sufficiently large sample to obtain reliable results.

A different way to face the Yield Optimization problem consists in reformulating it as a robust optimization problem. The problem we want to solve in this case is

$$\min_{x_d} \max_{x_p} \sum_{j=1}^{l} \alpha_j \sum_{i=1}^{m} \max\left\{0, l_i - p_i(x_d, \bar{x}_{o,j}, x_p)\right\}^q + \max\left\{0, p_i(x_d, \bar{x}_{o,j}, x_p) - u_i\right\}^q \tag{5}$$

with $q \geq 1$ and $l = 2m + 1$. So, the new formulation comes out by considering a min–max problem that finds the best worst-case performance.

3 A Derivative-Free Algorithm for Robust Optimization

In the preceding section, we introduced robust optimization as a possible way to deal with the Yield optimization. In particular, we introduced problem (5) that, more generally, can be stated as the following problem:

$$\min_{v \in V \subset \mathbb{R}^{n_1}} f(v) = \min_{v \in V \subset \mathbb{R}^{n_1}} \max_{w \in W \subset \mathbb{R}^{n_2}} g(v, w), \tag{6}$$

where $f : \mathbb{R}^{n_1} \to \mathbb{R}$, $g : \mathbb{R}^{n_1+n_2} \to \mathbb{R}$, some of the v variables are constrained to assume integer values, whereas the w variables are estimation of uncertain data or implementation parameters. Let

$$V = \left\{v \in \mathbb{R}^{n_1} : l_v \leq v \leq u_v, v_i \in \mathbb{Z}, i \in I_z\right\},$$
$$W = \left\{w \in \mathbb{R}^{n_2} : l_w \leq w \leq u_w\right\},$$

where

$$I_z \subseteq \{1, \ldots, n_1\}$$

is a subset of indices related to the integer v variables. Furthermore, let us denote $I_c = \{1, \ldots, n_1\} \setminus I_z$ and assume that V and W are both compact.

We assume that $g(v, w)$ is computed via numerical simulations so that first-order derivatives of g and f are not available.

We note that problem (6) is a semi-infinite minimax problem that encompasses many real-world problems like, for instance, engineering design problems, e.g., motor design, ship design, circuit design problems, and many others.

Further, we remark that since $g(v, w)$ is computed via numerical simulations, the problem defined by the objective function

$$f(v) = \max_{w \in W \subset \mathbb{R}^{n_2}} g(v, w) \tag{7}$$

is a hard black-box problem itself (since no derivative information is available) and requires suitable methods to be solved. Various approaches exist for solving general black-box optimization problems (see, e.g., [14–22] and references therein). Solution methods for semi-infinite minimax problems are the subject of great research and, in the context of black-box optimization, are lacking, except for the methods proposed in [23, 24] for finite min-max problems and in [13] for bilevel continuous problems. In order to tackle problem (7), we resort to a heuristic approach that can be designed by drawing inspiration from local or global minimization methods. Both these choices present some advantages and disadvantages. In particular,

- Local approaches often require a low number of function evaluations but can get stuck in local optima;
- Global approaches, on the contrary, typically require a high number of function evaluations but can compute a good approximation of a global optimum.

Since in our case computation of the function $g(v, w)$ is very expensive in terms of CPU time, we prefer to adopt a local approach for the solution of problem (7).

3.1 The Derivative-Free Optimization Algorithm

Now we recall the derivative-free algorithm for local mixed-integer optimization introduced in [14]. Then, we show how it can be used to approximately compute the function value $f(v)$ and to solve the robust optimization problem. Finally, we carry out some considerations regarding the connection between solution of problem (6) and problem (7).

We consider the following problem:

$$\begin{aligned} &\min \varphi(x) \\ &\quad x \in X, \\ &\quad x_i \in \mathbb{Z}, \quad i \in I_z, \end{aligned} \tag{8}$$

where we assume that X is defined by bound constraints on the variables and that the variables $x_i, i \in I_z$, can only assume integer values.

Basically, the method can be thought of as a distributed algorithm in the sense that all coordinates are considered cyclically, and a different search procedure is adopted on the basis of the variable type. In order to introduce the formal description of the algorithm, we first present a simplified procedure. We will then enrich the presentation by adding more and more technical details.

Procedure 1 A Mixed-integer derivative-free optimization framework

Input: an initial point $x_0 \in X$.
Output: a stationary point of $\varphi(x)$ (as defined in reference [14])

repeat
  for $i = 1, 2, \ldots, n$ **do**
    if ith variable is continuous **then** do a continuous search along ith direction
    else do a discrete search along ith direction
    end if
  end for
  Try to (heuristically) improve the current point
until convergence

Procedure 2 A (more detailed) Mixed-integer derivative-free optimization framework

Input: an initial point $x_0 \in X$, a decrease parameter $\xi_0 > 0$.
Output: a stationary point of $\varphi(x)$ (as defined in [14])

Set $k = 0$.
repeat
  Set $y_k^1 = x_k$
  for $i = 1, 2, \ldots, n$ **do**
    if ith variable is continuous
    then compute an α continuous stepsize along the ith coordinate enforcing α^2-sufficient decrease
    else compute an α discrete stepsize along the ith coordinate enforcing ξ_k-sufficient decrease
    end if
    Produce new point y_k^{i+1}
  end for
  if $(y_k^{n+1})_z = (x_k)_z$ and the stepsizes for discrete variables are equal to 1 **then** set $\xi_{k+1} = \theta \xi_k$
  else set $\xi_{k+1} = \xi_k$.
  end if
  Find $x_{k+1} \in X \cap \mathcal{Z}$ s.t. $\varphi(x_{k+1}) \leq \varphi(y_k^{n+1})$.
  Set $k = k + 1$.
until convergence

Procedure 1 helps us to understand the main iteration loop of the algorithm we will describe later on, which basically analyzes one coordinate direction at a time and performs a different procedure depending on the type of variable under analysis.

Procedure 2 is characterized by two basic ingredients. The first one is the use of two different searches for, respectively, continuous and discrete variables. The second one is the decrease parameter ξ, which will define the sufficient decrease in the linesearch for discrete variables.

Then, we can add some details to the description of the last procedure. In particular, we specify how the decrease parameter is updated and how the new iterate x_{k+1} is obtained.

Now we are ready to present the complete MIDFO Algorithm (Algorithm 1). In this complete version, we specify all the details and, in particular, how the actual (α^i, $i = 1, \ldots, n$) and tentative ($\tilde{\alpha}^i$, $i = 1, \ldots, n$) stepsizes are computed and updated. We recall (see, e.g., [14]) that the $\tilde{\alpha}^i$s define the starting points of the extrapolations (possibly) performed by the linesearch procedures, whereas the α^is are the steps returned by the linesearch procedures. We also define the stopping condition of the outer iteration loop.

As it can be seen, for every variable, the algorithm performs either a *Continuous search* or a *Discrete search* until the stepsize and the discrete sufficient reduction parameters are small enough or the objective function value becomes smaller than a prefixed threshold. In the following, we formally define the Continuous search (Procedure 3) and Discrete search (Procedure 4) procedures.

The Continuous search procedure is defined by specifying values for parameters γ and δ, which are used, respectively, in the sufficient reduction criterion and for the expansion of the step. The main distinguishing feature of the Discrete search procedure with respect to the Continuous search consists in the sufficient decrease criterion that employs the decrease parameter ξ instead of the usual squared stepsize, which, for a discrete variable, is bounded away from zero. Indeed, we say that the

Algorithm 1 MIDFO(φ, X, n, I_z, α_{tol}, φ_{tol})

Data. $\theta \in (0,1)$, $x_0 \in X$, $\xi_0 > 0$, $\tilde{\alpha}_0^i > 0$, $i \in I_c = \{1, \ldots, n\} \setminus I_z$, $\tilde{\alpha}_0^i = 1$, $i \in I_z$, and set $d_0^i = e^i$, for $i = 1, \ldots, n$.

Repeat $k = 0, 1, \ldots$

Set $y_k^1 = x_k$.

For $i = 1, \ldots, n$

If $i \in I_c$ **then** compute α by the *Continuous search*$(\varphi, \tilde{\alpha}_k^i, y_k^i, d_k^i; \alpha)$

If $\alpha = 0$ **then** set $\alpha_k^i = 0$ and $\tilde{\alpha}_{k+1}^i = \theta \tilde{\alpha}_k^i$.
else set $\alpha_k^i = \alpha$, $\tilde{\alpha}_{k+1}^i = \alpha$.

else compute α by the *Discrete Search*$(\varphi, \tilde{\alpha}_k^i, y_k^i, d_k^i, \xi_k; \alpha)$

If $\alpha = 0$ **then** set $\alpha_k^i = 0$ and $\tilde{\alpha}_{k+1}^i = \max\{1, \lfloor \tilde{\alpha}_k^i / 2 \rfloor\}$.
else set $\alpha_k^i = \alpha$, $\tilde{\alpha}_{k+1}^i = \alpha$.

Set $y_k^{i+1} = y_k^i + \alpha_k^i d_k^i$ and $d_{k+1}^i = d_k^i$.

End For

If $(y_k^{n+1})_z = (x_k)_z$ and $\tilde{\alpha}_k^i = 1$, $i \in I_z$, **then** set $\xi_{k+1} = \theta \xi_k$ **else** set $\xi_{k+1} = \xi_k$.

Find $x_{k+1} \in X \cap \mathcal{Z}$ such that $\varphi(x_{k+1}) \leq \varphi(y_k^{n+1})$.

Until $(\max\{\xi_{k+1}, \max_{i \in I_c}\{\alpha_k^i, \tilde{\alpha}_{k+1}^i\}\} \leq \alpha_{\text{tol}}$ **or** $\varphi(x_{k+1}) \leq \varphi_{\text{tol}})$

Return $\varphi(x_{k+1})$ and x_{k+1}.

Procedure 3 Continuous search $(\varphi, \tilde{\alpha}, y, d; \alpha)$

Data. $\gamma > 0$, $\delta \in (0, 1)$.
Step 1. Compute the largest $\bar{\alpha}$ such that $y + \bar{\alpha} d \in X \cap \mathcal{Z}$. Set $\alpha = \min\{\bar{\alpha}, \tilde{\alpha}\}$.
Step 2. If $\alpha > 0$ and $\varphi(y + \alpha d) \leq \varphi(y) - \gamma \alpha^2$ **then** go to Step 6.
Step 3. Compute the largest $\bar{\alpha}$ such that $y - \bar{\alpha} d \in X \cap \mathcal{Z}$. Set $\alpha = \min\{\bar{\alpha}, \tilde{\alpha}\}$.
Step 4. If $\alpha > 0$ and $\varphi(y - \alpha d) \leq \varphi(y) - \gamma \alpha^2$ **then** set $d \leftarrow -d$ and go to Step 6.
Step 5. Set $\alpha = 0$ and **return**.
Step 6. While $(\alpha < \bar{\alpha}$ and $\varphi(y + \frac{\alpha}{\delta} d) \leq \varphi(y) - \gamma \frac{\alpha^2}{\delta^2})$
$\alpha \leftarrow \alpha / \delta$.
Step 7. Set $\alpha \leftarrow \min\{\bar{\alpha}, \alpha\}$ and **return**.

Procedure 4 Discrete search $(\varphi, \tilde{\alpha}, y, d, \xi; \alpha)$

Step 1. Compute the largest $\bar{\alpha}$ such that $y + \bar{\alpha} d \in X \cap \mathcal{Z}$. Set $\alpha = \min\{\bar{\alpha}, \tilde{\alpha}\}$.
Step 2. If $\alpha > 0$ and $\varphi(y + \alpha d) \leq \varphi(y) - \xi$ **then** go to Step 6.
Step 3. Compute the largest $\bar{\alpha}$ such that $y - \bar{\alpha} d \in X \cap \mathcal{Z}$. Set $\alpha = \min\{\bar{\alpha}, \tilde{\alpha}\}$.
Step 4. If $\alpha > 0$ and $\varphi(y - \alpha d) \leq \varphi(y) - \xi$ **then** set $d \leftarrow -d$ and go to Step 6.
Step 5. Set $\alpha = 0$ and **return**.
Step 6. While $(\alpha < \bar{\alpha}$ and $\varphi(y + 2\alpha d) \leq \varphi(y) - \xi)$
$\alpha \leftarrow 2\alpha$.
Step 7. Set $\alpha \leftarrow \min\{\bar{\alpha}, \alpha\}$ and **return**.

new trial point $(y \pm \alpha d)$ yields a sufficient reduction of the objective function value when its value is better than $f(y) - \xi$.

3.2 A Solution Approach and Approximate Calculation of $f(v)$

Solution of problem (6) is sought by applying algorithm MIDFO, namely, calling the procedure

$$\text{MIDFO}\big(f, V, n_1, I_z, 10^{-4}, -\infty\big),$$

which generates sequences $\{\xi_k\}$, $\{f(v_k)\}$, $\{\alpha_{k+1}^i\}$, and $\{\tilde{\alpha}_k^i\}$ for $i = 1, \ldots, n$. Let, for each $k \geq 0$,

$$f_k^{\min} = f(v_k)$$

be the current minimum estimate of function $f(v)$. At each iteration of algorithm MIDFO applied to the solution of problem (6), we have to compute the value of f at some given point v, that is, we should solve the following *second-level* problem

$$f(v) = \max_{w \in W \subset \mathbb{R}^m} g(v, w). \tag{9}$$

Hence, $f(v)$ is given by

$$f(v) = g\big(v, w^*(v)\big),$$

where

$$w^*(v) = \arg \max_{w \in W \subset \mathbb{R}^m} g(v, w).$$

Even though, in principle, we would be interested in a global solution $w^*(v)$ of problem (9), when dealing with computationally expensive real-world applications, we can accept a reasonable approximation of the objective function $f(v)$. Such an approximate value can be obtained by using, in place of $f(v)$, the function

$$\bar{f}(v) = g\big(v, w(v)\big),$$

where $w(v)$ is a stationary point of problem (9).

Now, during the execution of the main optimization algorithm, when it is necessary to compute the value of f corresponding to a given point v, we can make the computation even more efficient. Indeed, we can again use algorithm MIDFO to solve this "second-level" problem. Namely, we resort to the approximation

$$\tilde{f}_k(v) = g\big(v, \tilde{w}_k(v)\big),$$

where $\tilde{w}_k(v)$ is computed by

$$\text{MIDFO}\big(-g(v, \cdot), W, n_2, \emptyset, \alpha_k^{\text{target}}, f_k^{\text{target}}\big)$$

with

$$\alpha_k^{\text{target}} = \max\Big\{\xi_k, \max_{i \in I_c}\big\{\alpha_{k-1}^i, \tilde{\alpha}_k^i\big\}\Big\} \quad \text{and} \quad f_k^{\text{target}} = -f_k^{\min}.$$

It is worth noting that the quantities ξ_k, α_{k-1}^i, $\tilde{\alpha}_k^i$, $i \in I_c$, and $f_k^{\min}$ that define α_k^{target} and f_k^{target} are known at the beginning of every iteration k of Algorithm MIDFO applied to the solution of problem (6). Hence, as we can see, MIDFO stops when:

- either the stepsize parameter becomes smaller than α_k^{target}, that is, when the current point is a "sufficiently" good approximation of a stationary point with respect to the progress toward a solution of problem (6) as measured by α_k^{target} (see, e.g., [25]),
- or the second-level optimization reaches a value $\tilde{f}(v_k) \geq f_k^{\min}$, that is, when it is possible to conclude that the value of $\tilde{f}_k$ cannot improve the current minimum estimate.

4 Numerical Results

4.1 Comparison on Analytical Test Problems

In this subsection, we carry out a comparison between the proposed derivative-free method and the classical statistical method for the solution of the (semi-infinite) minimax robust optimization problem (6). To this purpose, we selected the following seven functions from the Moré, Garbow, and Killstrom test-problem collection [26]:

- Extended Powell Singular function;
- Extended Rosenbrock function;
- Penalty I function;
- Penalty II function;
- Trigonometric function;
- Variably dimensioned function;
- Watson function.

Then, we define the following problem:

$$\min_{v\in V}\max_{w\in W} g(v,w),$$

where $V=\{v\in\mathbb{R}^8 : -10\le v\le 10\}$ and $W=\{w\in\mathbb{R}^8 : -0.5\le v\le 0.5\}$.

For each problem, we run our method and the statistical one starting from the same ten different random points $v\in V$ and allowing a maximum number of 5000 function evaluations. Then, once a solution v^* has been found, we evaluate its robustness by sampling $N_p=1000$ random vectors $w^{(i)}\in W$, $i=1,\ldots,N_p$, and computing

$$\sigma=\sqrt{\frac{\sum_{i=1}^{N_p}(g(v^*,w^{(i)})-\bar{g})^2}{N_p}},$$

where

$$\bar{g}=\frac{1}{N_p}\sum_{i=1}^{N_p} g\big(v^*,w^{(i)}\big).$$

Obviously, the smaller σ, the more robust the obtained solution, and the smaller $\sigma+\bar{g}$, the better the obtained solution. We report the results by means of performance profiles using as performance indices σ and $\sigma+\bar{g}$, respectively, in Fig. 1. From these results we can conclude that the robust approach clearly outperforms the statistical one.

4.2 The Robust Circuit Design Problem

The real circuit we want to design is a DC–DC converter. This electronic circuit is commonly used to convert a source of direct current (DC) from one voltage level to another and is very common in portable electronic devices (e.g., cellular phones and laptop computers). Electronic devices contain subcircuits, each with its own voltage level requirement, which is different from that supplied by the battery. DC–DC converters offer a method to increase voltage from a partially lowered battery voltage thereby saving space instead of using multiple batteries to accomplish the same thing. We consider a DC–DC converter for AMOLED (Active-Matrix Organic Light-Emitting Diode, a display technology widely used in mobile devices and televisions). This circuit integrates two main components: a step up (output voltage is higher than the input voltage) and an inverting DC–DC converter (output voltage is of the opposite polarity as the input). We are interested in the design of a specific part of the circuit. In particular, we want to optimally design an integrated circuit formed by:

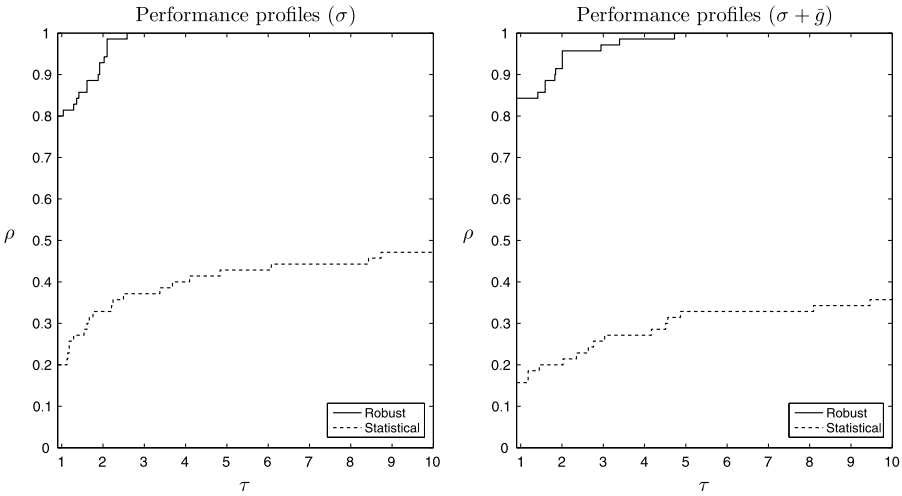

Fig. 1 Comparison between the robust and the statistical approach in terms of σ and $\sigma + \bar{g}$

- A chain of 4 CMos inverter;
- the High Side (PMos) and the Low Side (NMos) output stage;
- the driving signals (N_UP, LX1).

The device we consider is a circuit for Pulse-width Modulation (PWM), which is a commonly used technique for controlling power to electrical devices, made practical by modern electronic power switches. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast pace. The longer the switch is on compared to the off periods, the higher the power supplied to the load is. Roughly speaking, the Pulse-width Modulator is a device that breaks up a DC voltage into pulses that can be changed to our needs. When we change the width of the pulses, we are modulating them. This operation takes time that has to be kept as low as possible, avoiding, in our case, both powers on at the same time (i.e., we need to keep the signal delay low). In our case, the Step-Up PWM to LX Delay is mostly responsible for turning on the rectifier diode. As a consequence, the longer this delay is, the greater is the unwanted diode recirculating phase, hence increasing power losses. We report the scheme of the circuit in Fig. 2. In this case, the chain of 4 CMos Inverters is used as a Buffer that drives a large fan-out (the power stage composed by the low-side NMos and the high-side PMos). The increase in the load capacitance proportionally increases the propagation delay. Buffering with multiple inverter is used to maintain the speed performance of the circuit. The devices realized in an IC technology have to satisfy the design grid resolution, which means that the width and length vary in a discrete way (i.e., with a step of 10 nm). In order to avoid problems during the layout preparation (strange dimensions of devices), we have chosen to fix the design variables to integer values that all refer to the width of one device (W_M3, the width of PMos M3).

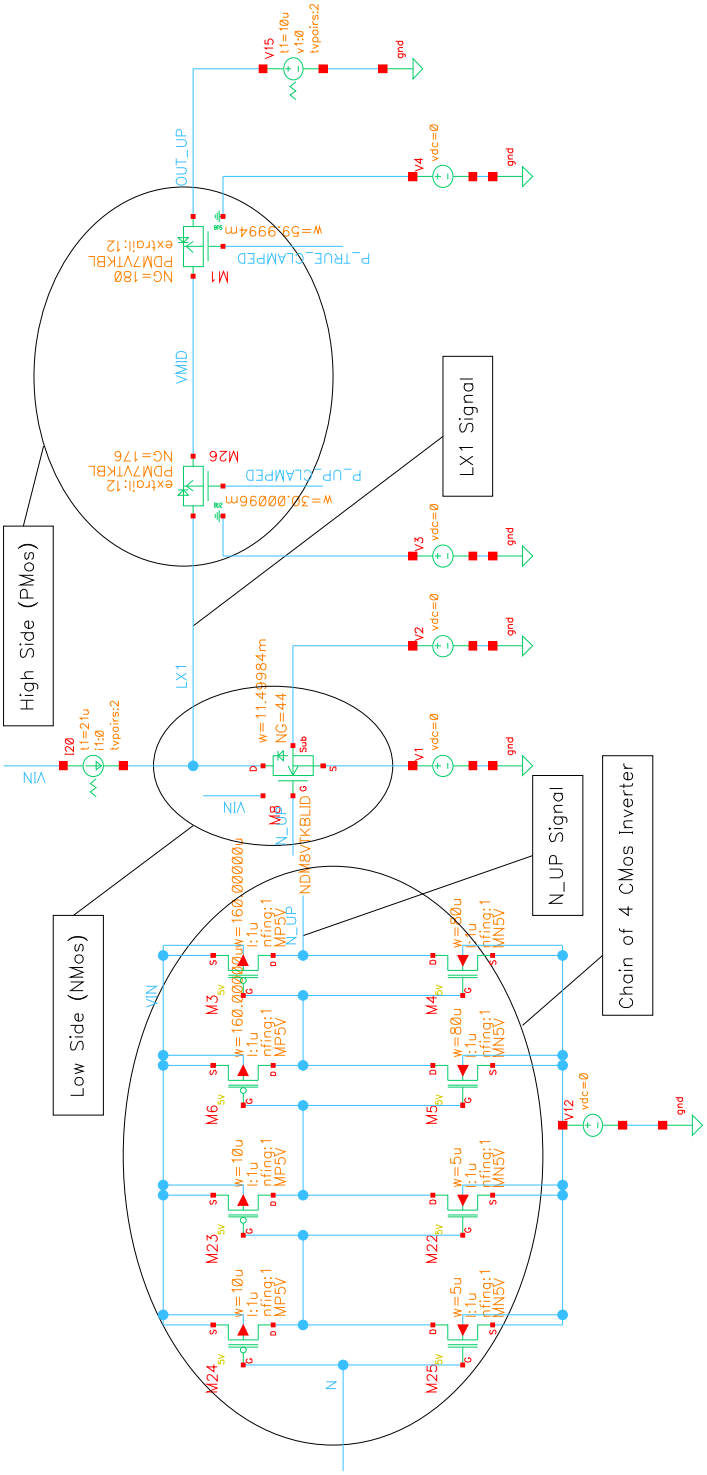

Fig. 2 Scheme of the integrated circuit

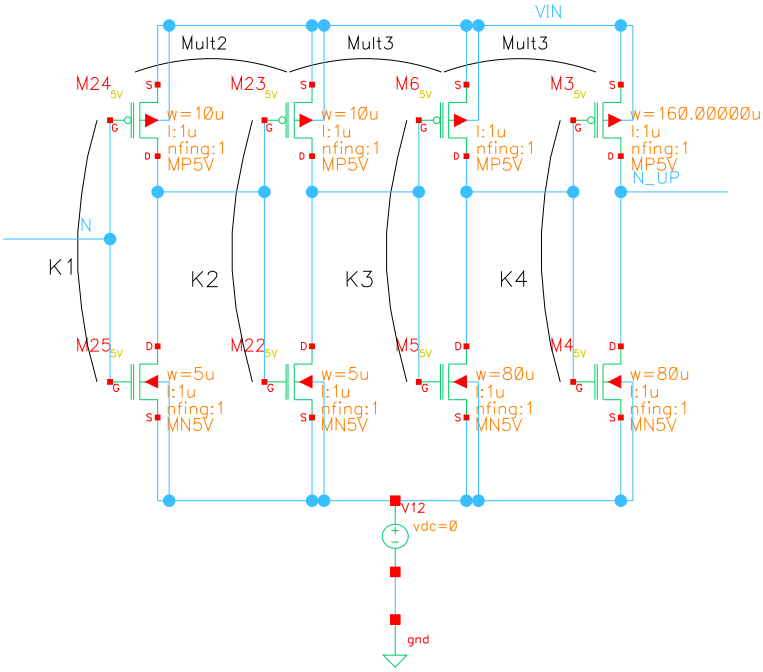

Fig. 3 Circuit variables

What concerns the variables, we have:

- **8 Design Variables**:
  - K1, K2 , K3, K4: the scale factor between PMos and NMos (discrete);
  - Mult2, Mult3, Mult4: the scale factor along the inverter chain (discrete);
  - W_M3: the width of the last PMos inverter in the chain (continuous).

  The first seven design variables (i.e., K1, K2 , K3, K4, Mult2, Mult3, Mult4) can assume only the integer values $(x_d)_i \in \{1, 2, \ldots, 10\}$, $i = 1, \ldots, 7$. The last variable, W_M3, must satisfy the bound constraint $1.0 \leq \text{W\_M3} \leq 16.0$.

  We just want to remark that the width of a specific component in the chain can be easily obtained by the width of the last PMos (e.g., W_M4 = W_M3/K4, W_M6 = W_M3/Mult4, ...).

  The design variables are reported in Fig. 3.
- **2 Operating Variables**:
  - V, T: supply voltage and temperature (continuous).
- **9 Statistical Variables** (continuous): The statistical variables involved in our example are related to 44 Nmos And Pmos devices, but after a sensitivity analysis, 35 of them were screened out. The nine remaining variables, with a Gaussian distribution centered around the mean value 0 and a standard deviation 1, are:
  - dtox5v: represents the oxide thickness relative variation;
  - dvthn5v: represents the threshold voltage of the N channel with respect to the nominal value;

Table 1 Performance specifications

Performance	LB	UB
Delay_1	0.0 n	21.0 n
Delay_2	0.0 n	21.0 n
DelaySymmetry	0.0 n	3.15 n

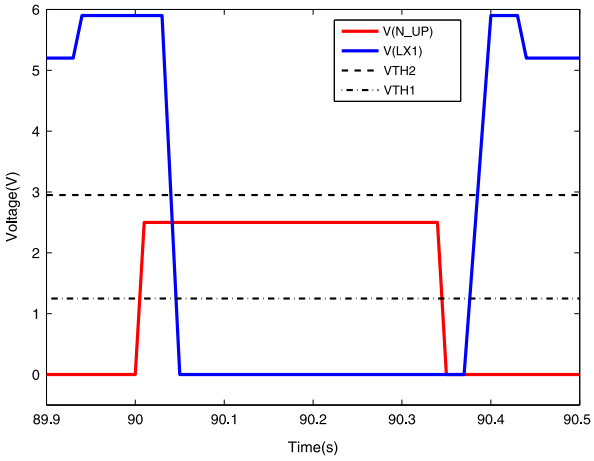

Fig. 4 Performance features

– dvthp5v: represents the threshold voltage of the P channel with respect to the nominal value;
– dmobn5v: represents the N channel mobility relative variation;
– dmobp5v: represents the P channel mobility relative variation;
– dw: represents the Poly width with respect to nominal width;
– dl: represents the Poly length with respect to nominal length;
– dvthndmos: represents the threshold voltage of the Ndmos with respect to the nominal value;
– drdhvpw: represents the DHV-PWell sheet resistance.

All of the statistical variables must satisfy the bound constraints $-3.0 \leq (x_p)_i \leq 3.0$, $i = 1, \ldots, 9$.

We then consider $m = 3$ different performance features describing the delays of our circuit (see Figs. 2 and 4):

- **Delay_1**: it represents the propagation delay between the signals V(N_UP) and V(LX1) when V(N_UP) is rising above the VTH1 threshold and V(LX1) is falling below the VTH2 threshold;
- **Delay_2**: it represents the propagation delay between the signals V(N_UP) and V(LX1) when V(N_UP) is falling below the VTH1 threshold and V(LX1) is rising above the VTH2 threshold;
- **Delay Symmetry**: it is defined as $|\text{Delay}_1 - \text{Delay}_2|$.

In Table 1, we report the performance specification features that we use.

4.3 The Optimization Process

Starting from the initial design defined by IC experts, which is

$$\tilde{x}_d = (5, \ldots, 5, 8.5)^\top,$$

we first execute the Nominal Design Optimization by solving problem (1) with $q = 1$, where $\bar{x}_o$ are fixed to the typical values (TYP), and $\bar{x}_p$ are all fixed to zero. We tackled the problem by means of the MIDFO algorithm described in [14]. Let us denote the solution of problem (1) by

$$\bar{x}_d = (9, 1, 10, 1, 10, 10, 10, 1.0)^\top.$$

Starting from this point, we perform the second phase optimization (i.e., the Worst-Case optimization). In the second phase, we first solve problems (2) and compute the worst-case operating variables $\bar{x}_{0,j}$, which are as follows:

- TYP. Typical operating conditions: Temperature = 27 °C and Voltage = 2.3 V.
- WC1. Worst-case operating condition 1, obtained by maximizing Delay_1 or Delay_2: Temperature = 120 °C and Voltage = 2.3 V.
- WC2. Worst-case operating condition 2, obtained by maximizing DelaySymmetry: Temperature = 120 °C and Voltage = 4.8 V.
- WC3. Worst-case operating condition 3, obtained by minimizing DelaySymmetry: Temperature = −40 °C and Voltage = 2.3 V.
- WC4. Worst-case operating condition 4, obtained by minimizing Delay_1 or Delay_2: Temperature = −40 °C and Voltage = 4.8 V.

Then, we solve the Worst-Case Optimization problem (3), that is,

$$\min_{x_d} \sum_{j=1}^{l} \sum_{i=1}^{m} \max\{0, l_i - p_i(x_d, \bar{x}_{o,j}, \bar{x}_p)\} + \max\{0, p_i(x_d, \bar{x}_{o,j}, \bar{x}_p) - u_i\}, \quad (10)$$

where $l = 5$, $x_{o,j}$ with $j = 1, \ldots, 4$ are fixed to the WCj operating conditions, $x_{o,5}$ is fixed to the TYP operating condition, and x_p are all fixed to zero. Once again, we tackled the problem by means of the MIDFO algorithm described in [14]. Solving problem (10) gives us the solution point

$$\hat{x}_d = (2, 2, 2, 2, 4, 4, 4, 6.4)^\top,$$

which is used as a starting point for the last phase (i.e., the Yield Optimization).

The Yield Optimization phase was carried out by means of two different approaches, namely:

(1) the Statistical Approach, described in Sect. 2, which is a reasonable and widely-used method for statistical custom IC design [5, 27];
(2) the Derivative-Free Robust Approach we developed in Sect. 2 (where we used the MIDFO algorithm to approximately solve the robust design problem).

Table 2 Comparison between statistical and robust approaches

Algorithm	n_f	T	WC1	WC2	WC3	WC4	ALL
Statistical	107000	0.869	0.700	0.654	0.999	1.0	0.578
Robust	47780	0.913	0.804	0.975	1.0	1.0	0.880
Robust*	46770	0.928	0.879	0.643	0.998	1.0	0.542

In the statistical approach, we solve problem (4) where $n_s = 200$ and $l = 5$ by means of Algorithm MIDFO. Hence, $n_s \times l = 1000$ function evaluations are needed at each iteration; this gives the final point

$$x^*_{\text{stat}} = (1, 3, 2, 2, 8, 5, 5, 15.9777)^\top.$$

In the robust approach, we solve problem (5) where we set $l = 5$, and, as for the approximate calculation of the objective function $f(v)$ (defined as in (7)), the algorithm stops when the maximum stepsize is smaller than or equal to the given tolerance 10^{-4} or the number of Black-Box evaluations reaches 20000. This approach returns the solution point

$$x^*_{\text{robust}} = (1, 2, 2, 3, 7, 5, 6, 14.8)^\top.$$

We also tried to solve problem (5) by means of the proposed algorithm but considering all of the design variables to be continuous and then rounding the first seven variables of the solution point to the nearest integer value. This alternative strategy, which we call Robust*, to deal with the integrality constraints has produced the point

$$x^*_{\text{rounded}} = (1, 4, 2, 2, 8, 4, 5, 14.5)^\top.$$

To compare the solutions obtained by means of the three approaches, we evaluated the yield at typical and worst-case operating conditions for all of them on a normally distributed sample with $n_s = 1000$ elements:

$$x_p^{(\mu)}, \quad \mu = 1, \ldots, n_s.$$

In Table 2, we report the results in terms of yield obtained using the Statistical, Robust, and Robust* approaches. We indicate with n_f the number of function evaluations obtained by summing up all the function evaluations made in the solution of the inner level optimization problem, with *TYP* the yield at typical condition (i.e., the percentage of samples that satisfy all specifications at TYP operating conditions), with WCj, $j = 1, \ldots, 4$, the yield at worst-case conditions and with *ALL* the total yield of the circuit (i.e., the percentage of samples that satisfy all specifications at all conditions).

As we can easily see from the table, the Robust approach can guarantee a result, in terms of total yield, better than the one obtained by both Statistical and Robust* approaches. In terms of the number of function evaluations, the Robust and Robust* approaches perform comparably to each other but are sensibly more efficient than the Statistical approach. We stress the fact that the result obtained by the Robust*

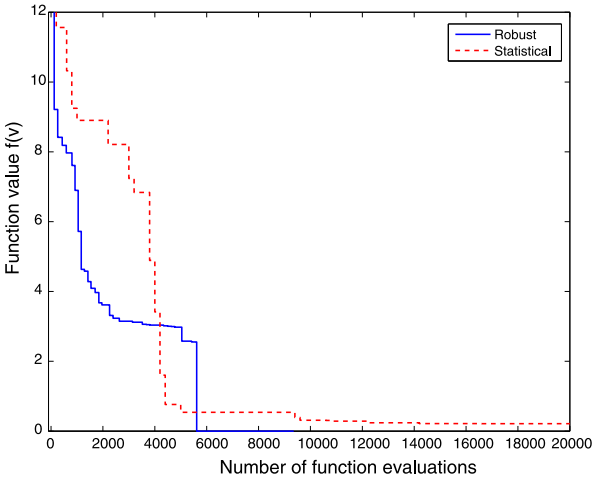

Fig. 5 Comparison between robust and statistical approaches

approach (giving a total yield of 0.542) is largely worse than that obtained by the Robust approach (which gives a total yield of 0.88).

In Fig. 5, in order to see the progress of the Robust and Statistical approaches, we further report the plots of function value $f(v)$ as functions of the number of function evaluations.

5 Conclusions

In the paper, we were concerned with the optimal design of a DC–DC converter of the kind commonly used in electronic portable devices. We discussed the three phases that compose the optimization process and in particular the Yield optimization, in which the manufacturing yield (percentage of manufactured circuits that satisfy the performance specifications when varying statistical variables) is to be optimized. We showed two alternative strategies to deal with the Yield optimization, and, in particular, we show how it can be reformulated as a robust optimization problem. Then, we introduced a novel derivative-free mixed-integer algorithmic framework for the solution of the latter robust optimization problem. The framework is based on the use of the derivative-free linesearch method for bound constrained mixed-integer problem proposed in [14].

We reported the results obtained by using the proposed algorithmic framework and compared them with those obtained by:

– a Statistical approach to solve the yield optimization problem;
– a rounding scheme to deal with integer design variables.

We showed that the proposed method outperforms both the Statistical approach and the rounding scheme in terms of the total yield.

References

1. Conn, A., Vicente, L.N., Visweswariah, C.: Two-step algorithms for nonlinear optimization with structured applications. SIAM J. Optim. **9**(4), 924–947 (1999)
2. Boyd, S.P., Kim, S.J., Patil, D.D., Horowitz, M.A.: Digital circuit optimization via geometric programming. Oper. Res. **53**(6), 899–932 (2005)
3. Patil, D., Yun, S., Kim, S.J., Cheung, A., Horowitz, M., Boyd, S.: A new method for design of robust digital circuits. In: Proceedings of the International Symposium on Quality Electronic Design, pp. 676–681 (2005)
4. Mintarno, E., Skaf, J., Zheng, R., Velamela, J., Cao, Y., Boyd, S., Dutton, R., Mitra, S.: Optimized self-tuning for circuit aging. Proc. Des. Autom. Test. Eur. **5**(7), 586–591 (2009)
5. Graeb, H.: Analog Design Centering and Sizing. Springer, Dordrecht (2007)
6. Ben-Tal, A., Nemirovski, A.: Robust convex optimization. Math. Oper. Res. **23**(4), 769–805 (1998)
7. Ben-Tal, A., Nemirovski, A.: Robust optimization. Methodology and applications. Math. Program. **92**(3), 453–480 (2003)
8. Bertsimas, D., Sim, M.: Robust discrete optimization and network flows. Math. Program. **98**(1-3), 49–71 (2003)
9. Bertsimas, D., Sim, M.: Tractable approximations to robust conic optimization problems. Math. Program. **107**(1–2), 5–36 (2006)
10. Bertsimas, D., Nohadani, O., Teo, K.: Robust optimization in electromagnetic scattering problems. J. Appl. Phys. **101**(7), 074,507 (2007)
11. Bertsimas, D., Nohadani, O., Teo, K.: Robust optimization for unconstrained simulation-based problems. Oper. Res. **58**(1), 161–178 (2010)
12. Bertsimas, D., Nohadani, O., Teo, K.: Nonconvex robust optimization for problems with constraints. INFORMS J. Comput. **22**(1), 44–58 (2010)
13. Conn, A., Vicente, L.: Bilevel derivative-free optimization and its application to robust optimization. Optim. Methods Softw. **27**, 561–577 (2012)
14. Liuzzi, G., Lucidi, S., Rinaldi, F.: Derivative-free methods for bound constrained mixed-integer optimization. Comput. Optim. Appl. **53**(2), 505–526 (2012)
15. Abramson, M., Audet, C., Chrissis, J., Walston, J.: Mesh adaptive direct search algorithms for mixed variable optimization. Optim. Lett. **3**(1), 35–47 (2009)
16. Abramson, M., Audet, C., Dennis, J. Jr.: Filter pattern search algorithms for mixed variable constrained optimization problems. Pac. J. Optim. **3**(3), 477–500 (2007)
17. Audet, C., Dennis, J. Jr.: Pattern search algorithms for mixed variable programming. SIAM J. Optim. **11**(3), 573–594 (2001)
18. Audet, C., Dennis, J. Jr.: A pattern search filter method for nonlinear programming without derivatives. SIAM J. Optim. **14**(4), 980–1010 (2004)
19. Audet, C., Dennis, J. Jr.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006)
20. Vicente, L.N., Custódio, A.L.: Analysis of direct searches for discontinuous functions. Math. Program. **133**, 299–325 (2012)
21. Lewis, R., Torczon, V.: Pattern search algorithms for bound constrained minimization. SIAM J. Optim. **9**(4), 1082–1099 (1999)
22. Torczon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**(1), 1–25 (1997)
23. Liuzzi, G., Lucidi, S., Sciandrone, M.: A derivative-free algorithm for linearly constrained finite minimax problems. SIAM J. Optim. **16**(4), 1054–1075 (2006)
24. Hare, W., Nutini, J.: A derivative-free approximate gradient sampling algorithm for finite minimax problems. Comput. Optim. Appl. (2013). doi:10.1007/s10589-013-9547-6
25. Kolda, T., Lewis, R., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev. **45**(3), 385–482 (2003)
26. Moré, J., Garbow, B., Hillstrom, K.: Testing unconstrained optimization software. ACM Trans. Math. Softw. **7**(1), 17–41 (1981)
27. Chen, W.K. (ed.): The Circuits and Filters Handbook, 2nd edn. CRC Press, Boca Raton (2003)