Analog circuit optimization system based on hybrid evolutionary algorithms

Bo Liu a, Yan Wang a,*, Zhiping Yu a, Leibo Liu a, Miao Li a, Zheng Wang a, Jing Lu a, Francisco V. Fernández b

a Institute of Microelectronics, Tsinghua University, China
b IMSE, CSIC and University of Seville, Spain

1. Introduction

Nowadays, VLSI technology progresses towards the integration of mixed analog–digital circuits as a complete system-on-a-chip. Though the analog part is a small fraction of the entire circuit, it is much more difficult to design due to the complex and knowledge-intensive nature of analog circuits. Without an automated synthesis methodology, analog circuit design suffers from long design time, high complexity, high cost and requires highly skilled designers. Consequently, automated synthesis methodologies for analog circuits have received much attention. The analog design procedure consists of topological-level design and parameter-level design (also called circuit sizing) [1,2]. This paper concentrates on the latter, aiming at parameter selection and optimization to improve the performances for a given circuit topology.

There are two main purposes of a synthesis system: first, replace tedious and ad-hoc manual trade-offs by automatic design of parameters; second, solve problems that are hard to design by hand. Accuracy, ease of use, generality, robustness, and reasonable run-time are necessary for a circuit synthesis system to gain acceptance [3]. Other than those requirements, ability to deal with large-scale problems, closely meet the designer’s requirements, even for highly-constrained problems, and ability to achieve optimum results are significant objectives of the proposed system. Many parameter-level design strategies, methods, and tools have been published in recent years [1–23], and some have even reached commercialization [24].

Most analog circuit sizing problems can be naturally expressed as the minimization of an objective 1 (e.g., power consumption), usually subject to some constraints (e.g., DC gain larger than a certain value). They can be formulated as follows:

\[
\min_{x} f(x)
\]

subject to

\[
\begin{align*}
g(x) & \geq 0 \\
h(x) & = 0 \\
L_{x} & < x < U_{x}
\end{align*}
\]

(1)

In this equation, the objective function \(f(x)\) is the performance function to be minimized and \(h(x)\) are the equality constraints. In analog circuit design, the equality constraints mainly refer to Kirchhoff’s current law (KCL) and Kirchhoff’s voltage law (KVL) equations. Vector \(x\) corresponds to the design variables, and \(L_{x}\) and \(U_{x}\) are their lower and upper bounds, respectively. The vector \(g(x)\geq 0\) corresponds to user-defined constraints.

The proposed system uses the formulation of the analog circuit design problem in Eq. (1) as a constrained optimization problem and, then, solves it by evolutionary algorithms. A new algorithm, 

---

1 The maximization of a design objective can easily be transformed into a minimization problem by just inverting its sign.
called competitive co-evolutionary differential evolution (CODE) algorithm, is proposed to deal with this constrained optimization problem. The algorithm has several novel features, which enable it to deal with large-scale and highly-constrained problems in an acceptable computation time with high robustness.

The evolutionary algorithm has been implemented in MATLAB [25]. Evaluation of the objective function and constraints (values of \( f(x) \) and \( g(x) \) in Eq. (1)) is performed by using an electrical simulator, HSPICE [26], for which an appropriate link with MATLAB has been implemented.

The structure of the paper is as follows. Section 2 reviews related work and motivates the strategy of our optimization approach. The evolutionary algorithm used in this approach, differential evolution, and its implementation, are discussed in Section 3. Section 4 formulates the competitive co-evolution approach to handle constraints in the differential evolution algorithm. Section 5 provides practical examples and benchmark tests to show the efficiency, effectiveness and advantages of the proposed approach. Comparisons with other common methods are also carried out. Finally, some concluding remarks are given.

2. Related work

Synthesis can be carried out by the following two different approaches: knowledge- and optimization-based. The basic idea of knowledge-based synthesis is to formulate design equations in such a way that given the performance characteristics, the design parameters can be calculated [4–6]. In these tools, the quality of the solutions in terms of both accuracy and robustness is not acceptable since the very concept of knowledge-based sizing forces the design equations to be simple. Other drawbacks are the large preparatory time/effort required to develop design plans or equations, the difficulty in using them in a different technology, and the limitation to a fixed set of circuits.

In optimization-based synthesis, the problem is translated into function minimization problems that can be solved through numerical methods. Essentially, they are based on the introduction of a performance evaluator within an iterative optimization loop. The system is called equation-based when the performance evaluator is based on equations capturing the behavior of a circuit topology [7–11]. However, creating the equations often consumes much more time than manually designing the circuit. In addition, the simplifications required in the closed form analytical equations cause low accuracy and incompleteness. On the contrary, simulation-based methods do not rely on analytical equations but on SPICE-like simulations to evaluate the circuit performances in the optimization process, which result in superior accuracy, generality, and ease of use [12–15]. Therefore, our system is simulation-based. Through the link between HSPICE and MATLAB, the candidate parameters are transmitted from the optimization system to the simulation engine, and the circuit performances obtained by the electrical simulator are returned to the optimization system. The penalty to pay is a relatively long computation time (compared to other methods), although, as the experimental results in Section 5 demonstrate, it can be kept within acceptable limits.

Techniques for analog circuit optimization that appeared in literature can be broadly classified into two main categories: deterministic optimization algorithms and stochastic search algorithms (evolutionary computation algorithms, simulated annealing, etc.). The traditional deterministic optimization methods mainly include steepest-descent algorithm and downhill algorithm. These techniques are available in some commercial electrical simulators [26]. The drawbacks of deterministic optimization algorithms are mainly in the following three aspects: (1) they require a good starting point, (2) an unsatisfactory local minimum may be reached in many cases, and (3) they often require continuity and differentiability of the objective function. Some researchers have tried to address these difficulties, such as in Ref. [16], where a method to determine the initial point is presented. Another approach is the application of geometric programming methods, which guarantee the convergence to a global minimum [11]. However, they require a special formulation of design equations, which make them share many of the disadvantages of equation-based systems. Research efforts on stochastic search algorithms, especially evolutionary computation (EC) algorithms (genetic algorithms, differential evolution, genetic programming, etc.) have begun to appear in literature in recent years [1,17–23]. Due to the ability and efficiency to find a satisfactory solution, genetic algorithms (GA) have been employed as optimization routines for analog circuits in both, industry and academia. For problems with practical design specifications, most reported approaches use the penalty function method to handle constraints. Though these works have made a significant progress, the optimization algorithms for analog circuit design automation remain an active research area because of the following reasons:

1. GA is the most popular evolutionary algorithm, but its search ability and convergence rate have been criticized. It has also been proved that canonical GA cannot converge to the global optimum [27]. GA with elitism converges to the global optimum theoretically, but it is not always the case in practice. On the other hand, some other population-based metaheuristics (PBMH) methods, such as swarm intelligence [28] and differential evolution [29] are attracting much attention in the community of operations research because of their advantages over GAs. Their potentials in analog circuit design automation still need to be exploited.

2. The constraint handling problem is very important in analog circuit design, especially for high performance circuits, which are always highly constrained. Most reported synthesis methods use penalty functions to handle constraints, and few of them investigated solution algorithms for high performance design problems. In these methods, the constrained optimization problem is transformed into an unconstrained one by minimizing the following function:

\[
f(x) = f(x) + \sum_{i=1}^{n} w_i g_i(x),
\]

where the parameters \( w_i \) are the penalty coefficients and \( g_i(x) \) returns the absolute value of \( g_i(x) \) if it is negative, and zero otherwise. The results of the methods based on penalty functions are very sensitive to the penalty coefficients, and may not meet the designer’s specifications in many cases. Small values of penalty coefficients drive the search outside the feasible region and often produce infeasible solutions, while imposing very severe penalties make it difficult to drive the population to the optimum solution. Usually, exact solutions are hard to find without tuning the penalty coefficients for many times. Although several penalty strategies have been developed [30,31], there has been no general rule for designing penalty coefficients till now.

3. Ability to handle large-scale design problems is still under investigation. Most of the available methods can deal with about 10–20 variables simultaneously, but analog circuits with 30 or more unknown variables are common.
3. Differential evolution and its implementation

The differential evolution (DE) algorithm and its implementation are introduced briefly in this section. The DE algorithm is suitable for unconstrained problems, and it is also a basic component in CODE.

Differential evolution is a population-based evolutionary computation technique, which uses a simple differential operator to create new candidate solutions, and a one-to-one competition scheme to greedily select new candidates. Recently, DE has attracted much attention in various technical fields [32,33].

The flow diagram of the DE algorithm is summarized in Fig. 1. The DE algorithm starts with the random initialization of a population of individuals in the search space and works on the cooperative behaviors of the individuals in the population. At each generation, the mutation and crossover operators are applied to the individuals, and a new population arises. Then, selection takes place, and the corresponding individuals from both populations compete to build the next generation. The algorithm tries to find the globally optimal solution by utilizing the distribution of solutions in the search space and differences between pairs of solutions as search directions. However, the searching behavior of each individual in the search space is adjusted by dynamically altering the direction and step length in which the search is performed.

The ith individual in the d-dimensional search space at generation t can be represented as

\[ X(t) = [x_{i1}, x_{i2}, \ldots , x_{id}], \quad i = 1, 2, \ldots , NP, \quad (3) \]

where NP denotes the size of the population.

For each target individual i, according to the mutation operator, a mutant vector:

\[ V_i(t + 1) = [v_{i1}(t + 1), \ldots , v_{id}(t + 1)] \quad (4) \]

is generated by adding the weighted difference between a pair of individuals, randomly selected from the population at generation t, to another individual, as described by the following equation:

\[ V_i(t) = X_{i0}(t) + f(X_{i1}(t) - X_{i2}(t)), \quad (5) \]

where indices \( r_1 \) and \( r_2 \) \((r_1, r_2 \in \{1, 2, \ldots , NP\})\) are randomly chosen and mutually different, and also different from the current index i.

The scaling factor \( F \) \((F \in [0, 1])\) controls the amplification of the differential variation \( (X_{i1}(t) - X_{i2}(t)) \). Although \( F \) has not an upper limit, \( F \in [0, 2] \) is commonly used. The population size \( NP \) must be at least 4, so that the mutation operator can be applied. Vector \( X_{i0}(t) \) is the base vector to be perturbed. There is a variety of mechanisms to select this base vector.

After the mutation phase, the crossover operator is applied to increase the diversity of the population. Thus, for each target individual, a trial vector \( U_i(t) = [u_{i1}(t), \ldots , u_{id}(t)] \) is generated as follows:

\[ u_{ij}(t) = \begin{cases} v_{ij}(t), & \text{if } \text{rand}(j) \leq CR \text{ or } j = \text{rand}(i), \\ X_{ij}(t), & \text{otherwise,} \end{cases} \quad (6) \]

where \( \text{rand}(j) \) is a random number uniformly distributed in the range [0, 1]. The index \( \text{rand}(i) \) is randomly chosen from the set \{1, 2, \ldots , d\}, and prevents the trial vector from being identical to the target vector. The parameter \( CR \in [0, 1] \) is a constant called crossover parameter that controls the diversity of the population.

Following the crossover operation, the selection arises to decide whether the trial vector \( U_i(t) \) will be a member of the population of the next generation \( t + 1 \). For a minimization problem, \( U_i(t) \) is compared to the initial target individual \( X_i(t) \) by the following one-to-one based greedy selection criterion:

\[ X_i(t + 1) = \begin{cases} U_i(t), & \text{if } f(U_i(t)) < f(X_i(t)), \\ X_i(t), & \text{otherwise,} \end{cases} \quad (7) \]

where \( X_i(t+1) \) is the new individual of the population of the next generation, and \( f(x) \) is the objective function.

The procedure described above is considered as the standard version of DE. Several strategies of DE have been proposed, depending on the selection of the base vector to be perturbed, the number and selection of the trial vectors and the type of crossover operators [32,33]. In our implementation, the base vector \( X_{i0}(t) \) is selected to be the best member of the current population to share its information among the individuals of the population and bias solutions towards better vectors.

Special care has been taken for handling boundaries of the parameters of the search space. Two classes of boundaries are distinguished: hard and weak. Hard boundaries are those that cannot be exceeded (e.g., a passive resistance cannot be negative or the transistor gate length cannot be below the minimum value allowed in the technological process), even if there are mathematically better solutions beyond those points. During the execution of the DE algorithm, the overstepped individuals are set to the nearest bounds. Weak boundaries are those roughly estimated to reasonably limit the search space. Although all individuals in the initial population are selected within these bounds, mutations during the evolution of the population may yield individuals beyond those limits if better solutions are found. Unlike GA and particle swarm optimization (PSO), DE can deal with this problem. DE is also more effective as the accuracy of local search is better than that of GA and PSO [29].

For some parameters it is also interesting to use logarithmic scales to favor lower values of the parameters with large spans, e.g., if a bias current spans over several decades, high values, hence high power consumption, will be favored if a linear scale is used. Other parameters must be discretized, e.g., device sizes can only change according to a given grid. In our implementation, mutant vectors are allowed to vary continuously, to promote diversity. However, the parameters are set to the nearest grid value when evaluating the fitness of the individual.
4. Constrained analog circuit optimization problem

Though DE is very effective and efficient, it is not enough for the sizing of analog circuits. All EC algorithms themselves lack a mechanism to deal with the constraints of a problem, which remains an open research area. However, there exist user-defined specifications for most analog circuit design problems, and these constraints must be appropriately handled.

The use of penalty functions is the most common method, but it is very sensitive to penalty coefficients and can hardly get satisfactory results without proper penalty coefficients. Though several penalty strategies [30,31] have been developed to improve static penalty coefficients, there is no general rule to determine proper penalty coefficients till now. Michalewicz describes the difficulties in each available penalty strategy in [34]. Moreover, Michalewicz and Schoenaer [35] concluded that the static penalty function method without any sophistication is more robust, as one such sophisticated method may work well on some problems but may not work well on another problem.

In this paper, constraints are handled by using the augmented Lagrangian method, which transforms the constrained optimization problem into a problem amenable to the DE algorithm described in Section 3. Penalty parameters in the augmented Lagrangian formulation are automatically updated during execution of the algorithm to reach the optimum point, hence avoiding the problems related to inappropriate settings of the penalty parameters. Parameters are updated based on a co-evolution methodology.

In recent years, co-evolution methodologies, including cooperative and competitive mode, have attracted much attention. Methods based on cooperative mode mainly aim at unconstrained optimization problem [36,37]. Cooperative co-evolution for constrained optimization problems has been proposed in Ref. [38]. In this paper, the competitive co-evolution concept [39–41] and the modified DE algorithm based on the augmented Lagrangian method are combined to formulate a hybrid algorithm, CODE, for constrained optimization problems in analog IC synthesis. We will begin by introducing augmented Lagrangians, and then discuss the combination of augmented Lagrangians with competitive co-evolution methodology.

4.1. Augmented Lagrangians

A constrained non-linear optimization problem can be expressed as

\[
\begin{align*}
\text{minimize } & f(x) \\
\text{subject to } & g_i(x) \leq 0, \quad i = 1, \ldots, m \\
& g_j(x) = 0, \quad j = m + 1, \ldots, n.
\end{align*}
\] (8)

These functions can be combined into a single transformation function \( \Phi \), called the augmented Lagrangians [42]:

\[
\Phi(x, r, \theta) = f(x) + \frac{1}{2} \sum_{i=1}^{m} r_i (g_i + \theta_i)^2 - \theta_i^2 + \frac{1}{2} \sum_{j=m+1}^{n} r_j (g_j + \theta_j)^2 - \theta_j^2.
\] (9)

In analog circuit design, equality constraints are limited to current and voltage relationships imposed by Kirchhoff’s laws: KCL and KVL. Kirchhoff’s laws are automatically included in the circuit equations in electrical simulators, so only the inequality constraints defined by design specifications have to be taken into account in the optimization process. Therefore, the augmented Lagrangian formulation is reduced to:

\[
\Phi(x, r, \theta) = f(x) + \frac{1}{2} \sum_{i=1}^{m} r_i (g_i + \theta_i)^2 - \theta_i^2.
\] (10)

Here, \( x \) is the vector of decision variables, \( r \) is the vector of penalty parameters, and \( \mu_i = r_i \theta_i \) is the Lagrangian multiplier associated with the \( i \)th constraint. This process is repeated until convergence. The optimization objective is to find the saddle point \((x^*, \mu^*)\), such that:

\[
\Phi(x^*, \mu^*) \leq \Phi(x', \mu').
\] (11)

In the Lagrangian dual method, there is not such a saddle point for non-convex problems. However, augmented Lagrangians addresses the problem by convexifying the objective function with quadratic penalty terms associated with the constraints [43]. For most practical problems, a saddle point always exists, and \( x^* \) is the optimal solution of the optimization problem.

4.2. Combination of co-evolutionary methodology and augmented Lagrangians

The main problem of the augmented Lagrangian method is how to update the Lagrange multipliers so that it converges to the saddle point to avoid local optimization. A predetermined updating scheme may work well on some problems but may not work well on another problem. Co-evolution methodology, relying on the current evolution result of decision variables, solves this problem.

Competitive co-evolution method was inspired by observing predator–prey relationship, where organisms adapt to each other in a dynamic environment. Groups are rewarded if they defeat individuals that compete with them. Competitive co-evolution strategy can be viewed as arm races of two groups [39,40]. To arouse competition, two populations, whose values of fitness functions are opposite to each other, must be generated.

In order to find the saddle point in augmented Lagrangians, the problem can be formulated as

\[
\min_{x, \mu} \max_{r, \theta} \Phi(x, r, \theta) = f(x) + \frac{1}{2} \sum_{i=1}^{m} r_i (g_i + \theta_i)^2 - \theta_i^2,
\] (12)

where \( x \) is vector of design variables, and \( \mu \) is the vector of Lagrangian multipliers. The purpose of the first population is to minimize \( \Phi(x, r, \theta) \) with individuals in this population encoding values of \( x \) and using a fixed \( \mu \) generated from the second population. The evolution of this population tries to satisfy the following property of the saddle point: \( \Phi(x^*, \mu^*) \leq \Phi(x', \mu') \) in Eq. (11). The second population aims to maximize \( \Phi(x, r, \theta) \) with its individuals encoding values of \( \mu \) and using a fixed \( x \) generated from the first population. The reason of this operation is \( \Phi(x'^*, \mu'^*) \leq \Phi(x', \mu') \) in Eq. (11). This process establishes arm races of the two populations. Once the first population achieves a solution \( x \) with a previous value of \( \mu \), the second population gets a better \( \mu \) based on \( x \) to defeat it. Then the first population generates a better \( x \) to defeat the second population. At last, the saddle point in Eq. (11) can be reached, which is the optimal point of \( x \). Values of \( r \) and \( \theta \) are initialized at the beginning of the optimization process, and are updated at the end of each cycle of DE-based optimization. The penalty coefficients should increase as \( r_{i+1} = r_i \times 0.1 \) after each cycle, where \( r_0 \) and \( a \) should be initialized first. The flow diagram of CODE is summarized in Fig. 2.

In the first iteration of the DE-based optimization cycle, an individual is randomly selected from the second population. Its purpose is that the first population needs a fixed Lagrangian multiplier from the second population in the first generation, but
the second population does not start its evolution operations at
the same time.

It can be seen that the CODE algorithm fully inherits the
advantages of the augmented Lagrangian method. The exact
solution can be achieved by augmented Lagrangian method,
whereas this is not true for the static penalty function method
[43]. In addition, the advantages of the differential evolution
algorithm, as described in Section 3, are also inherited. As any
other stochastic optimization algorithm, it cannot be guaranteed
that the global optimum solution is found for every problem, but
the experimental results demonstrate that better solutions than
previous methods are obtained in all cases.

5. Experimental results

In this section, the developed algorithm will be applied to three
practical analog circuit sizing problems and four mathematical
benchmark problems. The three circuit sizing problems corre-
spond to three amplifiers of increasing complexity. The purpose of
these examples is to test the capability of CODE to handle highly-
constrained optimization problems, the ability to handle large
search spaces (large number of design parameters), its compari-
ton to other optimization algorithms and its low sensitivity to the
initial values of the optimization parameters. Finally, benchmark
tests of the evolutionary computation field for constrained
optimization are shown.

In all the examples, the DE step size $F$ is 0.8 and the crossover
probability $C_R$ is 0.8. The inner generations of population
encoding values of $x$ is 80 for problems with less than 20
variables, and 100 for other problems, and the generations of
population encoding values of $u$ is 80. The above parameters are
commonly used in DE based algorithms. We used $r_0 = 1$ and
$a = 1.5$ for all the problems, except in the specific experiments
which demonstrate the low sensitivity of the results to these
parameter values. The design parameter search space is quite
wide in all cases. Transistor lengths were allowed to vary between
the minimum value allowed by the technological process to
10 $\mu$m. Transistor widths were changed between the minimum
technology value to several hundreds of micrometers. Capacitor
values and bias currents and voltages also had broad (yet
reasonable) ranges.

The inputs to the system are a SPICE net list file containing the
structure, and user defined specifications. All the examples are
run on a 2.4 GHz PC with 1 GB RAM, in the MATLAB environment.
Reported computation times include processing time in MATLAB,
the communication time between HSPICE and MATLAB, and the
simulation time of HSPICE.

5.1. Example 1: Design of a two-stage amplifier

The main purpose of this example is to test the capability of
CODE to handle constraints. A typical Miller-compensated two-
stage amplifier, shown in Fig. 3, is chosen first to test the
algorithm. The technology used is a 0.25 $\mu$m CMOS process and
the load capacitance $C_L$ is 30 pF. The design parameters are:
transistor widths and lengths, compensation capacitor and bias
currents.

The first experiment tries to achieve the design objectives and
constraints shown in Table 1. Appropriate matching constraints
were established and appropriate operating region was ensured
by imposing constraints like $V_{DS}(V_{GS}-V_{TH}) > 1$ for NMOS transis-
tors. The same experiment was tried with the standard genetic
differential evolution algorithms and using the static penalty
function method to handle constraints (denoted GA+PF and
DE+PF, respectively). We tried to manually improve the penalty

![Flow diagram of CODE.](image)
coefficients through five runs of the GA+PF and DE+PF algorithms. At each new run, the penalty coefficients were updated trying to increase the relative importance of the constraints not met in the previous run. Table 1 shows the best result from these five runs. It can be seen that the GA+PF algorithm slightly violates the phase margin specification and gets a considerably higher power consumption. Both, the DE+PF and CODE algorithms, meet the requirements, CODE and the static penalty methods work relatively well (the latter usually needs several sets of penalty coefficients before an acceptable value is found). However, for problems with many and restrictive constraints, such as the case above, the drawbacks of both algorithms based on static penalty methods become obvious, whereas CODE is still able to meet the specifications. However, neither the GA+PF algorithm nor the DE+PF algorithm are able to, even though five different sets of penalty coefficients were tried. The design parameters obtained by CODE for this case are shown in Table 3.

An important advantage of CODE is that the algorithm has a low sensitivity to the initial settings of the penalty parameters. To illustrate this, Table 4 shows the results of the application of the CODE algorithm when different initial values of the optimization parameters are used. By comparing this table with Table 2, it can be checked that all constraints are met independently of the initial values of the optimization parameters. Moreover, variations in the final value of the objective function keep below 3%.

From these experiments, we can conclude that for low requirements, CODE and the static penalty methods work relatively well (the latter usually needs several sets of penalty coefficients before an acceptable value is found). However, for problems with many and restrictive constraints, such as the case above, the drawbacks of both algorithms based on static penalty methods, GA+PF and DE+PF, become obvious, whereas CODE consistently performs well.

5.2. Example 2: Design of a TCFC amplifier

The second example will use an amplifier based on the transconductance with capacitance feedback compensation.

---

Table 1
Specifications and results of CODE, GA+PF and DE+PF

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Constraints</th>
<th>CODE</th>
<th>GA+PF</th>
<th>DE+PF</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC gain (dB)</td>
<td>≥ 70</td>
<td>2.068</td>
<td>2.040</td>
<td></td>
</tr>
<tr>
<td>GBW (MHz)</td>
<td>≥ 2</td>
<td>60</td>
<td>54</td>
<td></td>
</tr>
<tr>
<td>Phase margin (°)</td>
<td>≥ 55</td>
<td>2.0965</td>
<td>2.2274</td>
<td></td>
</tr>
<tr>
<td>Output swing (V)</td>
<td>≥ 2</td>
<td>1.9709</td>
<td>1.9249</td>
<td></td>
</tr>
<tr>
<td>CMRR (dB)</td>
<td>≥ 70</td>
<td>59.708</td>
<td>57.402</td>
<td></td>
</tr>
<tr>
<td>PSRR (dB)</td>
<td>≥ 80</td>
<td>5.2156</td>
<td>4.523</td>
<td></td>
</tr>
<tr>
<td>Noise (nV/√Hz)</td>
<td>≤ 20</td>
<td>2.0929</td>
<td>2.07</td>
<td></td>
</tr>
<tr>
<td>Slow rate (V/μs)</td>
<td>≥ 1.5</td>
<td>1.9555</td>
<td>2.0359</td>
<td></td>
</tr>
<tr>
<td>Power (mW)</td>
<td>Minimize</td>
<td>1.0535</td>
<td>1.0217</td>
<td></td>
</tr>
<tr>
<td>Total run time (s)</td>
<td>10097</td>
<td>10126</td>
<td>9865</td>
<td></td>
</tr>
</tbody>
</table>

Table 2
Specifications and results of CODE, GA+PF and DE+PF

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Constraints</th>
<th>CODE</th>
<th>GA+PF</th>
<th>DE+PF</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC gain (dB)</td>
<td>≥ 85</td>
<td>86.1</td>
<td>78.436</td>
<td></td>
</tr>
<tr>
<td>GBW (MHz)</td>
<td>≥ 2.5</td>
<td>2.5052</td>
<td>6.6431</td>
<td></td>
</tr>
<tr>
<td>Phase margin (°)</td>
<td>≥ 55</td>
<td>57.746</td>
<td>54.958</td>
<td></td>
</tr>
<tr>
<td>Output swing (V)</td>
<td>≥ 2</td>
<td>2.0965</td>
<td>2.2274</td>
<td></td>
</tr>
<tr>
<td>CMRR (dB)</td>
<td>≥ 80</td>
<td>80.452</td>
<td>77.553</td>
<td></td>
</tr>
<tr>
<td>PSRR (dB)</td>
<td>≥ 85</td>
<td>86.13</td>
<td>78.641</td>
<td></td>
</tr>
<tr>
<td>Noise (nV/√Hz)</td>
<td>≤ 20</td>
<td>18.537</td>
<td>7.979</td>
<td></td>
</tr>
<tr>
<td>Slow rate (V/μs)</td>
<td>≥ 1.8</td>
<td>1.9709</td>
<td>1.9249</td>
<td></td>
</tr>
<tr>
<td>Power (mW)</td>
<td>Minimize</td>
<td>1.0143</td>
<td>2.4344</td>
<td></td>
</tr>
<tr>
<td>Total run time (s)</td>
<td>11206</td>
<td>10533</td>
<td>11037</td>
<td></td>
</tr>
</tbody>
</table>

Table 3
Parameters of the two-stage amplifier

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3</td>
<td>85.66</td>
<td>99.48</td>
<td>99.48</td>
<td>99.48</td>
<td>99.48</td>
<td>99.48</td>
<td>99.48</td>
<td>4.67</td>
<td>4.67</td>
<td>0.7</td>
<td>0.7</td>
<td>3.88</td>
<td>0.2173</td>
<td>0.2173</td>
<td>0.2173</td>
<td></td>
</tr>
</tbody>
</table>

Table 4
Experiments with different initial values of $r_0$ and $\alpha$

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Constraints</th>
<th>$r_0 = 2$, $\alpha = 2$</th>
<th>$r_0 = 1$, $\alpha = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC gain (dB)</td>
<td>≥ 85</td>
<td>93.603</td>
<td>86.117</td>
</tr>
<tr>
<td>GBW (MHz)</td>
<td>≥ 2.5</td>
<td>2.5072</td>
<td>3.3719</td>
</tr>
<tr>
<td>Phase margin (°)</td>
<td>≥ 55</td>
<td>59.708</td>
<td>57.402</td>
</tr>
<tr>
<td>Output swing (V)</td>
<td>≥ 2</td>
<td>2.029</td>
<td>2.07</td>
</tr>
<tr>
<td>CMRR (dB)</td>
<td>≥ 80</td>
<td>82.9</td>
<td>80.873</td>
</tr>
<tr>
<td>PSRR (dB)</td>
<td>≥ 85</td>
<td>93.622</td>
<td>86.154</td>
</tr>
<tr>
<td>Noise (nV/√Hz)</td>
<td>≤ 20</td>
<td>16.204</td>
<td>10.088</td>
</tr>
<tr>
<td>Slow rate (V/μs)</td>
<td>≥ 1.8</td>
<td>1.9555</td>
<td>2.0359</td>
</tr>
<tr>
<td>Power (mW)</td>
<td>Minimize</td>
<td>0.88</td>
<td>0.0047</td>
</tr>
<tr>
<td>Total run time (s)</td>
<td>10957</td>
<td>11031</td>
<td>11031</td>
</tr>
</tbody>
</table>

---

Fig. 3. The Miller-compensated two-stage amplifier.
The TCFC amplifier is shown in Fig. 4 and the target technology is a 0.35 \textmu m CMOS process. The optimization problem contains 36 design parameters, hence, it is considerably more complex than the previous example.

Table 5 shows the design specifications and the results of the CODE algorithm for the TCFC amplifier. The specifications include DC gain (dB), GBW (MHz), Phase margin (°), Slow rate (V/\mu s), Power (mW), and Total run time (s). The constraints are met within the specified ranges, and the results show that the best solution obtained is 82.3830 dB for DC gain, 2.2186 MHz for GBW, and 54.4970 ° for Phase margin.

Table 6 shows the results of the GA+PF algorithm (same specifications than Table 5). The algorithm uses a set of penalty coefficients: 20, 50, 5, 50, and 100 for the design constraints and objectives. The results in the first column show the best results obtained, with DC gain ranging from 67.794 to 83.401 dB, GBW ranging from 2.5645 to 2.0853 MHz, and Phase margin ranging from 36.5 to 54.98 °. The Power consumption is also varied, with the best result being 0.1516 mW.

Table 7 shows the results of the DE+PF algorithm (same specifications than Table 5). The algorithm uses a different set of penalty coefficients, and the results show that the best solution obtained is 76.901 dB for DC gain, 1.8462 MHz for GBW, and 53.66 ° for Phase margin. The Power consumption is also varied, with the best result being 0.7234 mW.

Table 8 shows the specifications for the gain-boosted folded-cascode amplifier. The specifications include DC gain (dB), GBW (MHz), Phase margin (°), Slow rate (V/\mu s), Power (mW), and Total run time (s). The target technology is a 0.25 \textmu m CMOS process. The best solution obtained is 80.217 dB for DC gain, 1.8462 MHz for GBW, and 53.66 ° for Phase margin. The Power consumption is also varied, with the best result being 0.7234 mW.

Finally, we will use the gain-boosted folded-cascode amplifier to design a high-performance amplifier. The TCFC technique is applied to the amplifier design, and the target technology is a 0.35 \textmu m CMOS process. The optimization problem contains 36 design parameters, and the constraints are met within the specified ranges. The best solution obtained is 82.3830 dB for DC gain, 2.2186 MHz for GBW, and 54.4970 ° for Phase margin.
algorithm is not able to meet the performance specifications and several transistors are not in the correct operating region.

As stated above, an important advantage of the CODE algorithm, is that the competitive co-evolution adjusts the penalty coefficients to the appropriate values. As an illustration, Table 9 shows the evolution of the Lagrange multipliers for all constraints along several cycles of the co-evolutionary algorithm. It can be seen that in a few cycles, the multipliers converge to the right values to achieve a proper solution.

5.4. Benchmark problems for constrained optimization

In computer science, especially in evolutionary computation, benchmark problems are of great importance to evaluate and compare different algorithms. Benchmark problems are tough, and if an algorithm works well in benchmark problems, it is often regarded as very effective for medium-sized optimization problems. Three highly-constrained benchmark problems [45], described in Appendix are tested first. The results for three experiments with different number of generations are shown in Tables 10 and 11. Best, worst and average results were extracted from 15 runs of the algorithm. The results show that the proposed algorithm for constrained optimization problems, CODE, is quite effective.

To show the advantages of CODE, the same benchmark problems were tried with the DE+PF and GA+PF algorithms. Different penalty coefficients were used in each of the 15 runs and the best result among them is shown in Table 12. The comparison of computation times is shown in Table 13.

From the comparison, we can conclude that methods based on penalty functions are worse than the methods based on co-evolution methods. In particular, unlike many previous works, in all cases, we use the same optimization parameters introduced above, which have not undergone any specific calculation in view

---

**Fig. 5.** (a) Gain-boosted folded-cascode amplifier; (b) P amplifier and (c) N amplifier.

---

2 This is the most favorable comparison for the GA+PF and DE+PF algorithms, since bad sets of penalty coefficients tend to decrease the mean value.
of different problems; that is, the algorithm can achieve good result without detailed parameters studies.

In addition, in order to test the ability of CODE to deal with active constraints in optimization problems, the benchmark problem in Ref. [46] was selected (Test problem 4 in Appendix). In this problem, both constraints are active at the optimum. The best results of CODE, GA+PF and DE+PF after 15 runs are $5.5080$, $5.6833$ and $5.5538$, respectively.

6. Conclusions

This paper presents CODE: an evolutionary-based system for parameter-level design of analog integrated circuits. The basic elements of the system are a co-evolutionary methodology based on a differential evolution algorithm and the use of augmented Lagrangians to represent the constrained non-linear optimization problem. CODE achieves the following three novel features: (1) it avoids the tedious tuning of penalty coefficients, (2) it can closely meet the designer’s specifications even for highly-constrained problems, and (3) it is suitable for medium or large-scale problems. Moreover, CODE is efficient.

Acknowledgments

This work has been supported by National Natural Science Foundation of China grant no. 60676012 and Special Funds for
Appendix. Description of benchmark test problems

Test problem 1

Minimize

\[ G_1(x) = 5x_1 + 5x_2 + 5x_3 + 5x_4 - 5 \sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \]

subject to:

\[ \begin{align*}
2x_1 + 2x_2 + x_{10} + x_{11} - 10 & \leq 0, \\
2x_1 + 3x_3 + x_{10} + x_{12} - 10 & \leq 0, \\
2x_2 + 2x_3 + x_{11} + x_{12} - 10 & \leq 0, \\
-2x_4 - x_5 + x_{10} & \leq 0, \\
-2x_6 - x_7 + x_{11} & \leq 0, \\
-2x_8 - x_9 + x_{12} & \leq 0, \\
-8x_1 + x_{10} & \leq 0, \\
-8x_2 + x_{11} & \leq 0, \\
-8x_3 + x_{12} & \leq 0,
\end{align*} \]

with

\[ \begin{align*}
x_i & \leq 1, \quad i = 1, \ldots, 9 \\
x_i & \leq 100, \quad i = 10, 11, 12 \\
x_{13} & \leq 1
\end{align*} \]

The optimum solution is \( x^* = (1, 1, 11, 1, 1, 1, 1, 1, 3, 3, 1) \) and the function value is \( G_1(x^*) = -15 \).

Test problem 2

Minimize

\[ G_2(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^1 + 7x_6^2 + x_7^1 - 4x_8x_9 - 10x_9 - 8x_10 \]

subject to:

\[ \begin{align*}
2x_1^2 + 3x_2^2 + x_3 + 4x_4 + 5x_5 & \leq 127, \\
7x_1 + 3x_2 + 10x_3 + x_4 - x_5 & \leq 282, \\
23x_1 + x_2^2 + 6x_3^2 - 8x_4 - 196 & \leq 0, \\
4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 & \leq 0,
\end{align*} \]

with

\[ \begin{align*}
-10 \leq x_i & \leq 10, \quad i = 1, \ldots, 7
\end{align*} \]

The optimum solution is

\[ x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227) \]

and the function value is \( G_2(x^*) = 680.6300573 \).

Test problem 3

Minimize

\[ G_3(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45 \]

subject to

\[ \begin{align*}
105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 & \geq 0, \\
-3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^1 + 7x_4 + 120 & \geq 0, \\
-10x_1 + 8x_2 + 17x_7 - 2x_8 & \geq 0, \\
-x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 & \geq 0, \\
x_1 - x_2 - 5x_9 + 2x_{10} & \geq 12, \\
5x_2^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 + 40 & \geq 0, \\
3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} & \geq 0, \\
-0.5(x_1 - 8)^2 - 2(x_2 - 4) - 3x_3^2 + x_6 + 30 & \geq 0,
\end{align*} \]

with

\[ \begin{align*}
-10 \leq x_i & \leq 10, \quad i = 1, \ldots, 10
\end{align*} \]

The optimum solution is

\[ x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927) \]

and the function value is: \( G_3(x^*) = 24.3062091 \).

Test problem 4

Minimize

\[ G_0(x) = -x_1 + x_2 \]

subject to:

\[ \begin{align*}
x_2 - 2x_1^2 + 8x_1^3 - 8x_2 - 2 & \leq 0, \\
x_2 - 4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 - 36 & \leq 0,
\end{align*} \]

with

\[ \begin{align*}
0 \leq x_1 & \leq 3, 0 \leq x_2 \leq 4
\end{align*} \]

The optimum solution is

\[ x^* = (2.32952024, 3.17849288) \]

and the function value is: \( G_0(x^*) = -5.508013271 \).

References


B. Liu et al. / INTEGRATION, the VLSI journal 42 (2009) 137–148. Zhiping Yu graduated from Tsinghua University, Beijing, China, in 1967 with B.S. degree. He received his M.S. and Ph.D degrees from Stanford University, Stanford, CA, USA in 1980 and 1985, respectively. He is presently the professor in the Institute of Microelectronics, Tsinghua University, Beijing, China. From 1989 to 2002, he has been a senior research scientist in the Dept. of Electrical Engineering in Stanford University, USA, while serving as the faculty member in Tsinghua. He returned to Tsinghua full time since September 2002 and holds Pericom Microelectronics Professorship (2002–2004) established by Pericom Semiconductor Corp. in San Jose, USA. His research interests include device simulation for nano-scale MOSFETs, quantum transport in nanoelectronic devices, compact circuit modeling of passive and active components in RF CMOS, and numerical analysis techniques.

Leibo Liu received the B.S. and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1999 and 2004, respectively. He is currently an associate professor with the Institute of Microelectronics, Tsinghua University, Beijing, China. His research interests include reconfigurable processor design and design methodologies, and system-level energy-aware design approaches for portable applications.

Miao Li received the B.S. in in electronic engineering from Tsinghua University, PR China, in 2006. He was at the Institute of Microelectronics of Tsinghua University from the September of 2006 to now. His research focuses on modeling of III-V compound semiconductor materials and devices.
Zheng Wang is an undergraduate student in electronic engineering from Tsinghua University, Beijing, China.

Jing Lu received the B.S. in electronic engineering from Tsinghua University, China, in 2005. She was a graduate student at the Institute of Microelectronics of Tsinghua University from 2005 and will receive the master degree in 2008. Her research focuses on modeling of III–V compound semiconductor materials and devices.

F.V. Fernández got the Physics-Electronics degree from the University of Seville in 1988 and his Ph.D. degree in 1992. In 1993, he worked as a postdoctoral research fellow at Katholieke Universiteit Leuven (Belgium). Since 1995, he is an Associate Professor at the Department of Electronics and Electromagnetism of University of Sevilla. He is also a researcher at CSIC-IMSE-CNRI. His research interests lie in the design and design methodologies of analog and mixed-signal circuits. Dr. Fernández has authored or edited three books and has co-authored more than 100 papers in international journals and conferences. Dr. Fernández is currently the Editor-in-Chief of Integration, the VLSI Journal (Elsevier). He regularly serves at the Program Committee of several international conferences. He has also participated as researcher or main researcher in several National and European R&D projects.