INTEGRATION the VLSI journal 56 (2017) 148–166

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

A new hybrid algorithm for analog ICs optimization based on the shrinking circles technique

Maryam Dehbashian, Mohammad Maymandi-Nejad*

Department of Electrical Engineering, Ferdowsi University of Mashhad, Mashhad, Iran

ARTICLE INFO

Keywords:
Analog ICs optimization
Automated sizing tool
The shrinking circles
Advanced gravitational search algorithm and particle swarm optimization (Advanced GSA_PSO)
Corners analysis

ABSTRACT

This paper presents a novel technique named the Shrinking Circles to enhance the performance of optimization algorithms embedded in automated sizing tools of analog ICs. This technique creates a balance between the exploration and exploitation capabilities when the optimization algorithm is converging to a possible optimum point. With the help of the shrinking circles concept, we upgrade a hybridization version of Gravitational Search Algorithm with Particle Swarm Optimization (Advanced GSA_PSO). Accordingly, a developed tool for the automation of analog ICs sizing is proposed. The performance of this tool is evaluated by two cases: minimizing the power consumption of a two-stage CMOS op-amp and simultaneous minimizing the circuit area and power consumption of a folded-cascode op-amp. In this paper, the corners analysis is also incorporated into the proposed circuit sizing tool based on a straightforward procedure by which this tool not only can obtain the solutions being robust against process, voltage, and temperature (PVT) variations, but also it alleviates the computational burden. Comparisons with available methods show that the proposed tool performs much better in terms of efficiency.

1. Introduction

Although the analog block occupies a small area of the entire integrated circuit, it is one of the key parts of ICs and its optimal design is certainly as important as the digital block's design. The implementation and integration of analog, digital and RF circuits are provided through VLSI technology on the same chip and with the same technology as a complete system-on-a-chip. Therefore, in this way the latest analog circuits would be supplied to the industry. But the main problem is that the robust and optimal design of an analog circuit is a challenging process which involves complicating trade-offs between design objectives, parameters and constraints. The manual design of analog circuits is very complex, costly, and time consuming as well as requiring highly skilled designers [1]. For this reason, researchers have been investigating methods in the analog circuit design automation since more than thirty years ago [2], and their achievements have been presented as Electronic Design Automation (EDA) tools [3].

The analog circuit design procedure consists of topology selection, circuit sizing, and layout synthesis. This paper concentrates on the circuit sizing due to its significant importance with assumption that the circuit topology is already selected by the designer.

The analog ICs sizing tools include two main sections, i.e., *synthesis* and *optimization* which are linked together. *Synthesis section* creates a sized circuit by taking into account two important input factors, i.e., the desired specifications and topology of the circuit [4]. Synthesis approaches used in the analog ICs sizing tools can be classified into *knowledge-based* and *optimization-based* categories. In the knowledge-based approach, a pre-designed plan is separately created for each circuit topology based on designer experience and design equations. In contrast, the optimization-based approach applies an optimization process instead of a design plan to size the circuit components. This process is an iterative procedure in which design variables are updated at each iteration until a stop criterion is satisfied.

The optimization-based approach includes an iterative loop, an optimization engine together with an evaluation engine. The evaluation engine is typically implemented using one of the three following approaches: an equation-based optimization, a simulation-based optimization, or a modeling-based optimization approach [5]. Synthesis approaches used in analog ICs sizing tools are classified and illustrated in Fig. 1. This figure gives a general overview of advantages and disadvantages of the synthesis approaches, indicating that the simulation-based optimization approach has more advantages in comparison with above-mentioned approaches. The only deficiency of this approach is its relatively long computation time compared to other approaches. However, due to the emerging of high processing speed computers, the computational time of a circuit sizing process can be

* Corresponding author.
E-mail address: maymandi@um.ac.ir (M. Maymandi-Nejad).

http://dx.doi.org/10.1016/j.vlsi.2016.09.009
Received 27 April 2016; Received in revised form 19 August 2016; Accepted 26 September 2016
Available online 28 September 2016

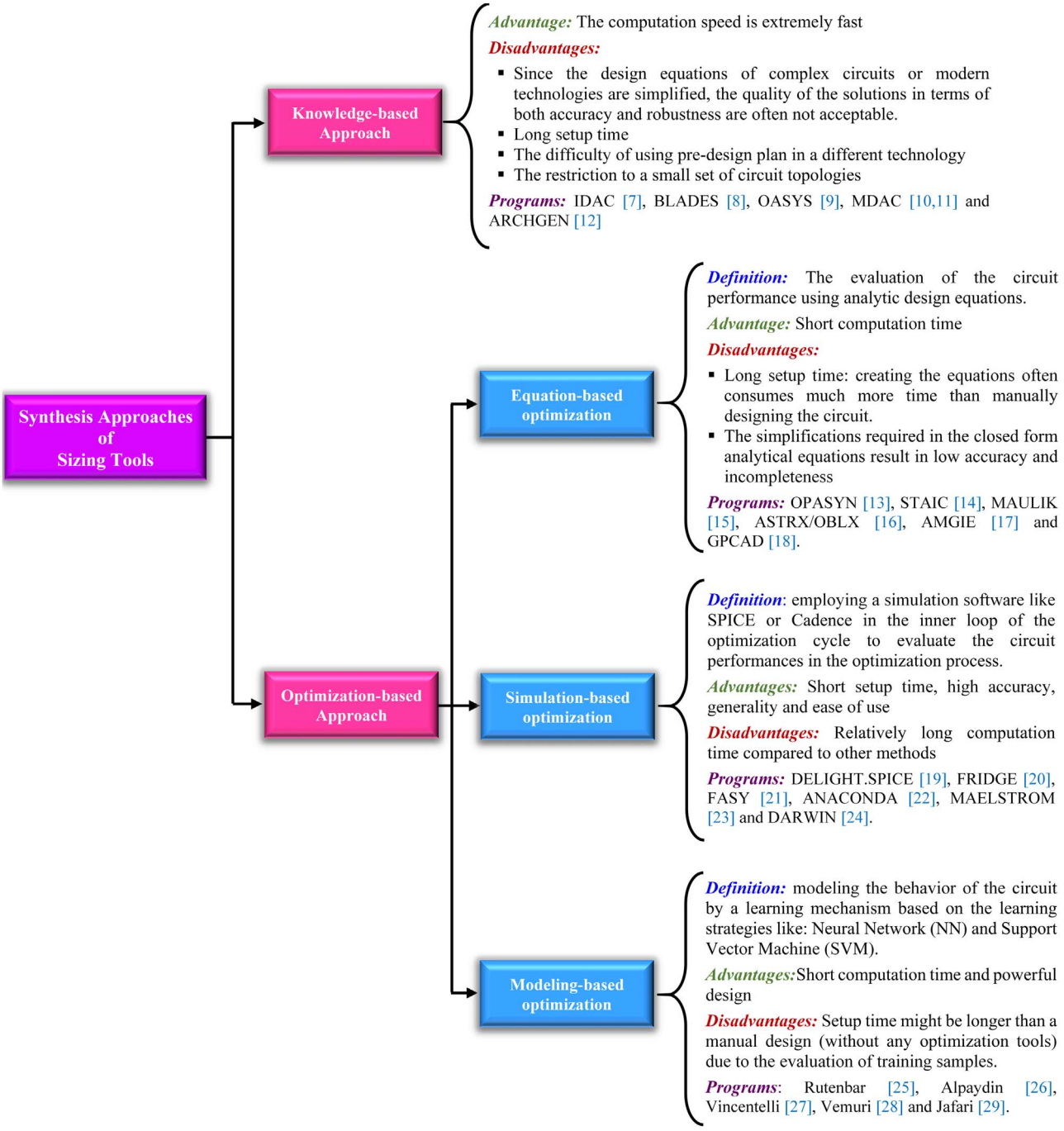


Fig. 1. Classification of synthesis approaches used in analog ICs sizing tools [7-29].

acceptable, normally about a few minutes for a circuit with nearly tens of transistors [6].

At the end of the synthesis process of a circuit, the primary solutions (i.e., the proposed sizes of the circuit components) are offered. Certainly, they are not the best solutions because a minor change may result in better solutions. In the second section of the circuit sizing tools, i.e., the optimization section, the primary solutions are optimized as much as possible using optimization techniques. Optimization techniques applied in the analog circuit sizing tools can be classified into two main categories, i.e., deterministic and meta-heuristic. Deterministic optimization algorithms entail traditional methods such as Newton and Levenberg-Marquardt techniques. The disadvantages of deterministic optimization algorithms are mainly referred to the three following issues: (1) requiring a good starting point; (2) falling into the trap of local optimum points; (3) depending on the continuity and differentiability of the objective function [30].

Metaheuristic algorithms were proposed a couple of decades ago. In contrast with deterministic approaches, they have shown the more efficiency and capability in dealing with complicated optimization problems. As illustrated in Fig. 2, metaheuristic optimization algorithms are classified into the three following main classes: evolutionary, stochastic and swarm intelligence algorithms.

According to Fig. 2, most of analog ICs sizing tools with commercial or academic applications tend to use Genetic Algorithm (GA) to optimize their design process. The ability, efficiency and robustness features to find an acceptable solution in an appropriate time are the main reasons for the tendency to use this algorithm. However, the search ability, the convergence rate and structural complexities in

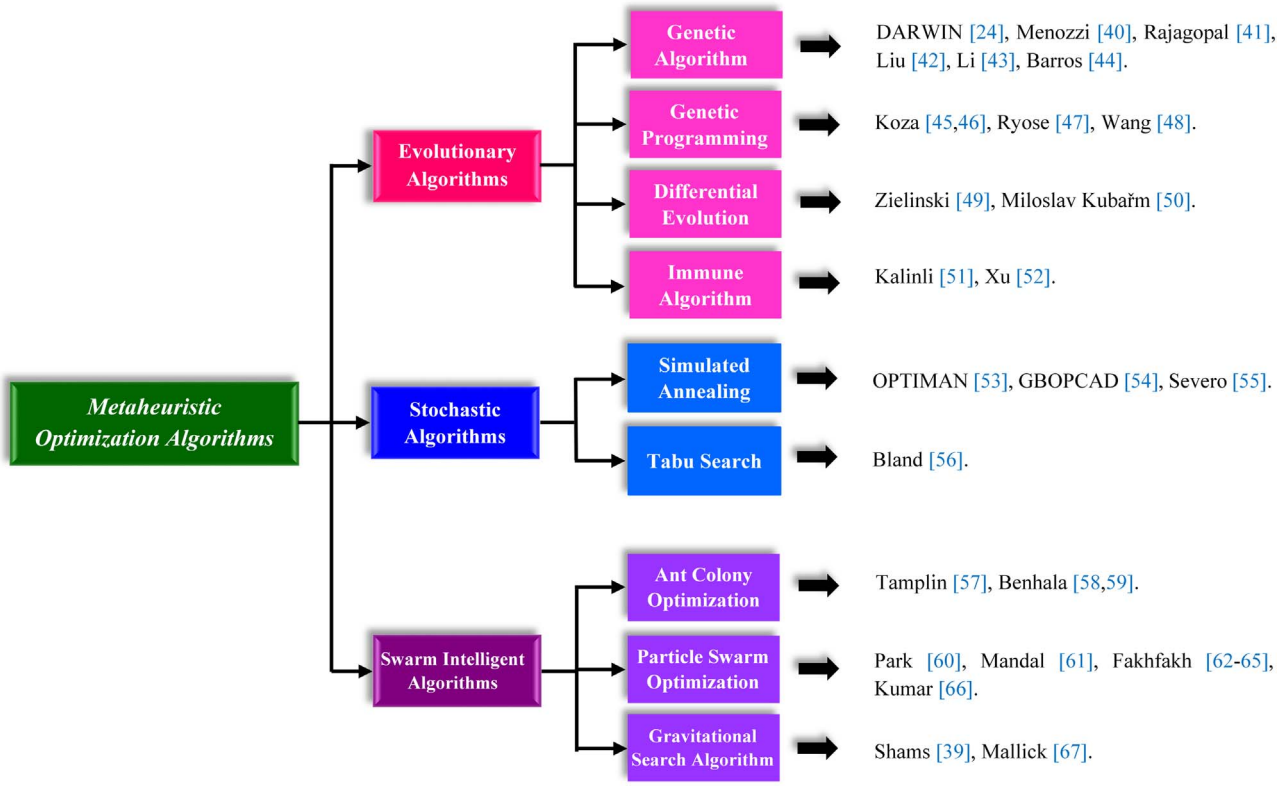


Fig. 2. Classification of metaheuristic optimization algorithms in the optimization section of circuit sizing tools [24,39-67].

setting of GA parameters have been criticized. For instance, in [31] it is proven that a canonical GA cannot be converged to the global optimum and it is concluded that in theory, GA with elitism converges to the global optimum, although practically, this is not always possible.

In recent years, swarm intelligence algorithms have highly been taken into consideration in different optimization problems especially in the development of analog ICs design tools. As can be seen in Fig. 2, after GA, swarm intelligence algorithms, particularly Particles Swarm Optimization (PSO) algorithms, have received much attention.

Among these swarm intelligence algorithms, a promising method has been introduced based on Newton's laws of gravity and motion. This method is called the Gravitational Search Algorithm (GSA) proposed by Rashedi et al. in 2009 [32].

GSA has some positive features such as easy implementation, fast convergence to global optimum and relatively low computational cost [32,33]. In most engineering applications, GSA in comparison with other optimization algorithms such as PSO and GA, has shown a better performance [34,35,39,67]. In contrast to the aforementioned abilities of GSA, there is sometimes a possibility of premature convergence. In this case, the algorithm converges to a non-optimal solution and cannot escape from this situation. In effect, when the algorithm converges to a local optimum point, it loses the ability to explore the search space in finding the optimum solution or other good near global optimums. To resolve this problem, different solutions have been presented like adding an extra operator to GSA such as disruption operator [36], chaotic operator [37], and black hole operator [38] or applying other techniques like Clustered-GSA [39]. Also, hybridization of GSA with other optimization algorithms is an efficient way to balance its exploration and exploitation abilities [68–70].

On the other hand, positive features of PSO algorithm such as simple modeling, fast convergence and great ability to exploit the search space are the main reasons for using this algorithm in hybrid methods. In [68,69], GSA and PSO have been hybridized to incorporate the exploration ability of GSA with the exploitation ability of PSO in such a way that the resulting algorithm (PSOGSA) has shown better performance in comparison with each of GSA and PSO algorithms.

In this paper, a new technique called the *shrinking circles* is proposed. This technique is used to balance the exploration and exploitation abilities when the algorithm is converging to a possible optimum point. With the help of the shrinking circles concept, we propose an upgraded version of GSA named Advanced GSA. Then, with the goal of creating a powerful algorithm for optimization of analog ICs sizing process, we hybridize Advanced GSA with PSO, hereinafter refer to as *Advanced GSA_PSO*.

Worthwhile research efforts in recent years have been conducted to promote the development of commercial tools in analog ICs sizing (e.g., Solido [71] and MunEDA [72]). Also, there are some academic tools for the circuit design which are currently in the research phase (e.g., AIDA [73]). However, it cannot be claimed that the solution provided by these tools is the best possible solution because the circuit sizing tools have to achieve the best trade-off between solutions with respect to three important factors, i.e., accuracy, robustness and run time.

In this paper, we propose a novel tool for automated sizing of analog ICs using Advanced GSA_PSO. The tool has also combined Advanced GSA_PSO with a constraint handling technique, i.e., static penalty function, to present a powerful algorithm for the optimization of analog circuit sizing process.

The rest of this paper is organized as follows: Section 2 provides a brief review of PSO, GSA and PSOGSA algorithms. The shrinking-circles technique and how it can be applied in Advanced GSA_PSO are discussed in Section 3. The performance evaluation of Advanced GSA_PSO and a comparative study on some benchmark functions are presented in Section 4. Our proposed tool for analog circuit sizing and optimization based on Advanced GSA_PSO is introduced in Section 5. Also, simulation results of the proposed tool in two study cases and compare them with other circuit sizing tools are provided in this section. Following, Section 6 carries out corners analysis and statistical studies into the obtained solutions to validate their robust-

ness. Finally, conclusions are drawn in Section 7.

2. Brief review of PSO, GSA and PSOGSA algorithms

2.1. Particle Swarm Optimization

PSO has been proposed by Kennedy and Eberhart in 1995 [74]. Currently, it is the most popular member of swarm intelligence algorithms due to its efficiency and effectiveness. The concept of PSO was inspired by the intelligent social behaviors of living creatures like flocks of birds and schools of fish. In PSO, the candidate solutions are called particles. In the search space of PSO, each particle moves to a new position to find the best particle or the best solution. However, the movement of each particle is made based on a certain velocity, so that this velocity depends on two factors: the own best position of each particle and the best position found among all particles. PSO is modelled as follows:

$$v_i^d(t+1) = w(t).v_i^d(t) + c_1.\ r_1(pbest_i^d - x_i^d(t)) + c_2.\ r_2(gbest^d - x_i^d(t)) \quad (1)$$

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1) \quad (2)$$

where $v_i^d(t)$ is the velocity of particle i in the dth dimension at iteration t; $w(t)$ is the inertia weight function; c_1 and c_2 are cognitive and social acceleration coefficients, respectively; r_1 and r_2 are random numbers between 0 and 1; $x_i^d(t)$ is the current position of particle i in the dth dimension at iteration t; $pbest_i^d$ is the personal best position of particle i in the dth dimension at iteration t, and $gbest^d$ is the global best position i in the dth dimension over all the particles.

In (1), the first term, $w.v_i^d(t)$ provides exploration ability for PSO. The second and third terms, $c_1.\ r_1(pbest_i^d - x_i^d(t))$ and $c_2.\ r_2(gbest^d - x_i^d(t))$, represent private thinking and collaboration of particles respectively. In each iteration, the velocity of the particles is calculated using (1) and then the positions of the particles are calculated using (2). The position of particles is changed in every iteration until stopping criterion is met.

2.2. Gravitational Search Algorithm

GSA is one of the widely used algorithms of the swarm intelligence family that is inspired by the Newton's laws of gravity and motion [32]. In the search space of GSA, there is a collection of N agents (candidate solutions). In GSA, each agent has a certain mass, so that the performance of which is evaluated by the value of its fitness function. Moreover, it attracts other agents using the mutual gravitational force between them. This force of gravity is directly proportional to the product of their mass values and inversely proportional to the square of the distance between them. The gravitational force moves all the agents towards the agent with heavier mass according to the Newtonian law of motion. By lapse of time, the heaviest mass attracts the other agents gradually. Once the convergence criterion is met, this mass is considered as the optimum solution. Optimization steps of GSA are outlined below.

- *Step 1: Initialization*
  At the beginning of the algorithm, N masses are assumed to be in the search space of the problem. The positions of the N masses are initialized randomly as below:

$$X_i = (x_i^1, \ldots, x_i^d, \ldots, x_i^n) \quad for \quad i = 1, 2, \ldots, N. \quad (3)$$

where x_i^d presents the position of ith mass in the dth dimension.

- *Step 2: Fitness evaluation*
  According to the objective function, the fitness values of all masses at iteration t are calculated. Then, the best and worst fitness values are defined as below (for a minimization problem):

$$best(t) = \min_{j\in\{1,\ldots,N\}} fit_j(t) \quad (4)$$

$$worst(t) = \max_{j\in\{1,\ldots,N\}} fit_j(t) \quad (5)$$

where, $fit_j(t)$ represents the fitness value of the mass i at iteration t.

- *Step 3: Normalization*
  The normalized mass is calculated at iteration t by the following equations:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \quad (6)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \quad (7)$$

where $M_i(t)$ represents the normalized mass i at iteration t.

- *Step 4: Gravitational constant*
  The gravitational constant $G(t)$ is computed at iteration t using the following equation:

$$G(t) = G_0 e^{-\alpha t/t_{max}} \quad (8)$$

where G_0 is the initial value, α is a descending coefficient, t is the current iteration and t_{max} is the maximum number of iterations. The gravitational constant is initialized with a large value but according to (8), the more the time passes, the more it is reduced. The reason is that the large values of this parameter increase the exploration ability of the algorithm while the small values enhance its exploitation ability.

- *Step 5: Acceleration*
  The acceleration of each mass is calculated as below based on the Newtonian gravity and motion laws:

$$a_i^d(t) = \sum_{\substack{j\in Kbest \\ j\neq i}}^{N} rand_j.\ G(t).\ \frac{M_j(t)}{R_{ij}(t) + \varepsilon}(x_j^d(t) - x_i^d(t)) \quad (9)$$

where $a_i^d(t)$ is the acceleration of mass i in the dth dimension at iteration t, $rand_j$ is a uniform random variable in the interval [0,1] which is used to give a stochastic behavior to the space searching. ε is also an arbitrary small constant, and $R_{ij}(t)$ is the Euclidean distance between two masses i and j. Moreover, $Kbest$ is the set of first K agents with the best fitness value and biggest mass. It is a linear function of time with a negative slope and the intercept of K_O. Here, K_O is set to the total number of agents and is decreased linearly to 1.

- *Step 6: Updating*
  The velocity and the position of each mass at next iteration (t+1) are updated using the following equations:

$$V_i^d(t+1) = rand_i \times V_i^d(t) + a_i^d(t) \quad (10)$$

$$x_i^d(t+1) = rand_i \times x_i^d(t) + V_i^d(t+1) \quad (11)$$

- *Step 7: Repeat steps 2–6*

Steps 2–6 are repeated in an iterative loop until the number of iterations reaches to its maximum limit or the stopping criterion is met. The global fitness is the optimal value of $best(t)$ computed at the final iteration and its corresponding position is presented as the global solution.

2.3. The hybrid PSOGSA algorithm

Mirjalili [68,69] hybridized PSO with GSA and called it the hybrid PSOGSA algorithm. The basic idea of PSOGSA is a combination of the capability of the social thinking ($gbest$) in PSO with the local search capability of GSA. This hybridization is modelled as follows:

$$v_i^d(t+1) = rand_i.v_i^d(t) + c_1'.\ r_1.\ a_i^d(t) + c_2'.\ r_2.\ (gbest^d - x_i^d(t)) \quad (12)$$

where $v_i^d(t)$ is the velocity of agent i in the dth dimension at iteration t,

c_1' and c_2' are acceleration coefficients, r_1 and r_2 are random numbers between 0 and 1, $a_i^d(t)$ is the acceleration of agent i in the dth dimension at iteration t, and $gbest^d$ is the best solution found over all the agents. In each iteration, the positions of agents are updated as follows:

$$x_i^d(t+1)=x_i^d(t)+v_i^d(t+1) \tag{13}$$

In contrast to PSO, GSA is a memory-less algorithm but the hybrid PSOGSA requires memory to save the global best position at each iteration. In other words, the only difference between GSA and hybrid PSOGSA is the third term of (12) which records the global best position of all agents, and hence it affects the calculation of the velocity at each iteration.

3. Advanced GSA_PSO

When a premature convergence happens for GSA, the algorithm converges to a non-optimal solution and it cannot escape from this situation. In other words, the algorithm is trapped in a local optimum point and therefore, it loses its ability to explore other possible optimum points and only exploits the last obtained solutions.

In this paper, a new technique named the *shrinking circles* is proposed to create a balance between the exploration and exploitation capabilities of the GSA when it is converging to a possible optimum point. In this technique, another search process in parallel with the conventional search method of GSA is executed in such a way that it creates an irregularity in the convergence trend of the algorithm. Since this irregularity is quite controllable, it leads the algorithm to be exited from the premature convergence situation, or at least helps the algorithm to find better solutions.

As already mentioned, (4) calculates the best fitness value for all masses at each iteration. Therefore, *best(t)* represents the fitness value of the best mass. At each iteration, the *shrinking circles* technique identifies the position of the best mass, M_{best}, in the search space of the problem. Then, a circle with the center M_{best} and the radius R is considered. R is characterized by the following equation:

$$R = \mathrm{R}_0 * \left(1-\frac{t}{t_{max}}\right) + \mathrm{R}_{min} \tag{14}$$

where t is the current iteration, t_{max} is the maximum number of iterations, R_0 and R_{min} are the initial and minimum values of the radius R, respectively.

Afterwards, a new mass (masses) is (are) generated randomly inside this circle. Note that the new mass or masses is/are different from the existing masses of GSA. The number of these new masses is determined using (15) where N is the number of the existing masses of the algorithm.

$$Number_{New\_Mass}=1 + \mathrm{round}((N-1)*\frac{t}{t_{max}}) \tag{15}$$

It is important to emphasize that the new masses are accepted if and only if they are located in the search space of the problem.

In Fig. 3, the search process proposed by the *shrinking circles* technique is illustrated. It shows clearly how this technique finds a better position around M_{best} at each iteration. As can be seen, in each iteration of the algorithm, a circle with the center M_{best} and the radius calculated by (14) is drawn. Then, a new mass (masses) is (are) generated randomly inside this circle. If the fitness value of the new mass or each of new masses is better than the fitness value of M_{best}, M_{best} will be replaced by the best new mass; otherwise, the new mass will be ignored..

In the next iterations of the algorithm, the considered circle is shrunk gradually. In other words, the radius of the circle according to (14) becomes smaller and smaller and subsequently, the number of the generated masses according to (15) becomes more. In fact, the decreasing of the radius simultaneously with the increasing of new masses reinforces the exploration and exploitation abilities of GSA.

It is noteworthy that in the primary iterations the radius of the circle is considered large and thus, GSA has the opportunity to explore a wide space around M_{best}. As a result, the probability of premature convergence of GSA is significantly reduced. In the last iterations when GSA is being converged to a global optimum point, the number of new masses is increased so that the algorithm can exploit an optimal point with higher accuracy.

We have applied the shrinking circles technique to GSA to enhance its performance and efficiency. This is an upgraded version of GSA and for convenience, hereinafter, we refer to this version as *Advanced GSA*. Then, we have hybridized Advanced GSA with PSO, denoted by *Advanced GSA_PSO*, in order to merge their capabilities and create a high-performance optimization algorithm for complex problems.

Notably, however, the *shrinking circles* technique discussed in this paper is not restricted to GSA algorithm, and thus, it would be capable of being applied to other swarm intelligence algorithms as well.

4. Comparative study of Advanced GSA_PSO

4.1. Benchmark functions

In order to use Advanced GSA_PSO in the optimization section of IC sizing tools, at first, the performance of this algorithm in comparison with other optimization methods is required to be evaluated by a set of standard benchmark functions This set includes 23 benchmark functions [32,75] which is classified into three categories:

1. Unimodal functions (F_1 to F_7): in these functions, the convergence rates of the algorithm in comparison with the final results are more interesting.
2. Multimodal high-dimensional functions (F_8 to F_{13}): these functions have many local minimum points, hence the algorithm must not only be able to find the optimum (or at least a good near-global optimum) solution, but also it should not be trapped in any local optimum points.
3. Multimodal low-dimensional functions (F_{14} to F_{23}): In contrast to multimodal high-dimensional functions, these functions do not have a large number of local minimum points.

These functions are given in Tables 1–3 respectively, where n is the dimension of the function and $S \subseteq R^n$ defines the search space.

4.2. Advanced GSA_PSO vs. GSA, PSO, PSOGSA and Clustered-GSA

In this section, Advanced GSA_PSO is applied to the benchmark functions introduced in the previous section and is compared with the other algorithms. This paper compares Advanced GSA_PSO with GSA (the standard version) [32], PSO (the standard version) [74], PSOGSA [68] and Clustered-GSA (the improved version used in the circuit sizing) [39] and demonstrates its capabilities in solving more complex problems. It should be noted that in Clustered-GSA, the number of agents is reduced during iterations using the clustering method of the k-Means algorithm and so this algorithm improves the efficiency of GSA by reducing the number of fitness function evaluation.

For illustrative purposes, the number of agents is set to 50 in all algorithms. The dimension is 30 (n=30) for functions of Tables 1–2 and is set to values of Table 3 for other functions. Maximum iteration (t_{max}) is 1000 and 500 for functions of Tables 1–2 and Table 3, respectively. In GSA, G_O and α in (8) are assumed to be 100, 20 [32]; in PSO, c_1 and c_2 in (1) are set to 2 [74]; in PSOGSA c_1' and c_2' in (12) are set to 0.5 and 1.5 [68], respectively; also in Advanced GSA_PSO, the parameters G_O, α, c_1' and c_2' are set to 100, 25, 0.5 and 1.5 as well as R_0 and R_{min} in (14) are assumed to be 10 and 0.1, respectively.

It is worthy of mentioning that the number of fitness function

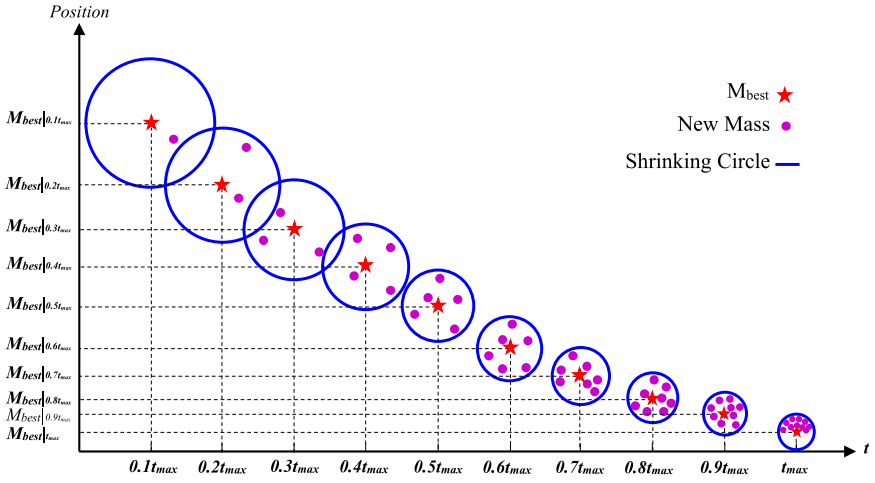


Fig. 3. The search process to find a better position around M_{best} by the shrinking circles.

Table 1
Unimodal test functions.

Test function	S
$F_1(X) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$F_2(X) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
$F_3(X) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^n$
$F_4(X) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	$[-100, 100]^n$
$F_5(X) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]^n$
$F_6(X) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	$[-100, 100]^n$
$F_7(X) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	$[-1.28, 1.28]^n$

Table 2
Multimodal test functions with high dimension.

Test function	S
$F_8(X) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$[-500, 500]^n$
$F_9(X) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^n$
$F_{10}(X) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right)$ $-exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e$	$[-32, 32]^n$
$F_{11}(X) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$
$F_{12}(X) = \frac{\pi}{n}\{10 sin^2(\pi y_1)$ $+ \sum_{i=1}^{m-1}(y_i - 1)^2[1 + 10 sin^2(\pi y_{i+1})]$ $+ (y_n - 1)^2\} + \sum_{i=1}^{m} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$, $u(x_i, a, k, m) = f(x) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	$[-50, 50]^n$
$F_{13}(X) = 0.1\{sin^2(3\pi x_1)$ $+ \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)]$ $+ (x_n - 1)^2[1 + sin^2(2\pi x_n)]\}$ $+ \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$

Table 3
Multimodal test functions with low dimension.

Test function	S
$F_{14}(X) = \left(\frac{1}{500} + \sum_{j=1}^{25}\frac{1}{j + \sum_{i=1}^{2}(x_i - a_{ij})^6}\right)^{-1}$	$[-65.53, 65.53]^2$
$F_{15}(X) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	$[-5, 5]^4$
$F_{16}(X) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 + 4x_2^2 + 4x_2^4$	$[-5, 5]^2$
$F_{17}(X) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2$ $+10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$[-5, 10] \times [0, 15]$
$F_{18}(X) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2$ $-14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2$ $\times(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2$ $+27x_2^2)]$	$[-5, 5]^2$
$F_{19}(X) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^3$
$F_{20}(X) = -\sum_{i=1}^{4} c_i exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^6$
$F_{21}(X) = -\sum_{i=1}^{5}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$
$F_{22}(X) = -\sum_{i=1}^{7}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$
$F_{23}(X) = -\sum_{i=1}^{10}[(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]^4$

evaluations for all algorithms except Advanced GSA_PSO is equal to 50,000 (i.e., 50×1000) for functions of Tables 1, 2 and 25,000 (i.e., 50×500) for functions of Table 3. On the other hand, the number of fitness function evaluations for Advanced GSA_PSO is 75,525 (i.e., 50×1000+25,525; where the value 25,525 is equal to the number of new masses according to (15)) for functions of Tables 1, 2 and 37,775 (i.e., 50×500+12,775; where the value 12,775 is equal to the number of new masses according to (15)) for functions of Table 3. For a fair comparison between the aforementioned algorithms in the same conditions, the number of fitness function evaluations of all algorithms must be equal. To this end, we decrease the number of iterations of Advanced GSA_PSO as far as the sum of evaluations of its masses and the new masses produced by the *shrinking circles* technique becomes equal to the evaluations of other algorithms. Therefore, we considered the number of iterations for functions of Tables 1–2 as 662 so that the number of fitness function evaluations becomes equal to 50,006 (i.e., 662×50+16,906; where the value 16,906 is equal to the number of new masses according to (15)) and also we considered the number of iterations for functions of Table 3 as 331 so that the number of fitness function evaluations becomes equal to 25,015 (i.e., 331×50+8465; where the value 8465 is equal to the number of new masses according to (15)). In this way, the number of fitness function evaluations of all considered algorithms are nearly equivalent.

The optimization results of unimodal functions, Multimodal high-

dimensional functions and Multimodal low-dimensional functions are provided in Tables 4–6. Because of the random nature of these algorithms, the presented results are averaged over 30 independent runs of each algorithm. *Average best-so-far* is the average of the best solutions, *Median best-so-far* is the median of the best solutions, *Best best-so-far* is the best of the best solutions and *Std best-so-far* is the standard deviation of the best solutions achieved during all runs. Furthermore, in the second column of Tables 4–6, f_{opt} represents the

Table 4
Optimization results of benchmark functions in Table 1 with n=30 and t_{max}=1000.

Functions	f_{opt}	Statistical Indicators	GSA	PSO	PSOGSA	Clustered-GSA	Advanced GSA_PSO
F_1	0	Average best-so-far	1.93×10^{-17}	2.73×10^{-6}	2.79×10^{-18}	3.48×10^{-22}	4.58×10^{-31}
		Best best so far	1.19×10^{-17}	1.38×10^{-11}	1.15×10^{-18}	N.A.*	5.87×10^{-35}
		Std best so far	6.22×10^{-18}	9.09×10^{-6}	2.85×10^{-18}	N.A.	1.39×10^{-30}
F_2	0	Average best so far	2.44×10^{-8}	5.32×10^{-2}	7.25×10^{-9}	4.75×10^{-11}	1.16×10^{-15}
		Best best so far	1.82×10^{-8}	9.71×10^{-6}	5.06×10^{-9}	N.A.	1.41×10^{-17}
		Std best so far	3.92×10^{-9}	1.44×10^{-1}	1.19×10^{-9}	N.A.	1.29×10^{-15}
F_3	0	Average best so far	260. 89	108. 38	2973. 46	2.25×10^{-18}	2.02×10^{-30}
		Best best so far	85. 24	0. 216	173. 51	N.A.	3.98×10^{-34}
		Std best so far	92. 43	213. 61	1869. 27	N.A.	2.74×10^{-30}
F_4	0	Average best so far	2.06×10^{-3}	5. 14	30. 04	2.51×10^{-11}	4.25×10^{-16}
		Best best so far	2.19×10^{-9}	2. 13	11. 85		5.11×10^{-18}
		Std best so far	1.11×10^{-2}	2. 54	6. 59	N.A.	6.04×10^{-16}
F_5	0	Average best so far	26. 23	550. 70	64. 26	26. 21	22. 79
		Best best so far	25. 74	11. 03	12. 02	N.A.	21. 06
		Std best so far	0. 427	2535. 91	50. 98	N.A.	1. 029
F_6	0	Average best so far	0. 00	25. 13	289. 16	0. 00	0. 00
		Best best so far	0. 00	0. 00	2	N.A.	0. 00
		Std best so far	0. 00	29. 03	315. 47	N.A.	0. 00
F_7	0	Average best so far	2.20×10^{-2}	1.65×10^{-1}	4.17×10^{-2}	0. 07	8.39×10^{-5}
		Best best so far	7.70×10^{-3}	5.00×10^{-2}	1.64×10^{-2}	N.A.	8.36×10^{-6}
		Std best so far	7.87×10^{-3}	8.99×10^{-2}	1.80×10^{-2}	N.A.	8.45×10^{-5}

N.A.=Not Available.

Table 5
Optimization results of benchmark functions in Table 2 with n=30 and t_{max}=1000.

Functions	f_{opt}	Statistical Indicators	GSA	PSO	PSOGSA	Clustered-GSA	Advanced GSA_PSO
F_8	$-12.56\times10^{+3}$	Average best so far	$-2.70\times10^{+3}$	$-5.76\times10^{+3}$	$-7.02\times10^{+3}$	$-1.05\times10^{+3}$	$-7.22\times10^{+3}$
		Best best so far	$-3.77\times10^{+3}$	$-6.85\times10^{+3}$	$-7.97\times10^{+3}$	N.A.*	$-8.27\times10^{+3}$
		Std best so far	$4.34\times10^{+2}$	$5.20\times10^{+2}$	$5.46\times10^{+2}$	N.A.	$5.03\times10^{+2}$
F_9	0	Average best so far	14. 89	37. 78	46. 60	0. 00	0. 00
		Best best so far	7. 96	23. 88	29. 85	N.A.	0. 00
		Std best so far	3. 75	6. 64	10. 92	N.A.	0. 00
F_{10}	0	Average best so far	3.56×10^{-9}	2. 002	1. 58	1.87×10^{-13}	1.01×10^{-15}
		Best best so far	2.51×10^{-9}	3.14×10^{-6}	8.88×10^{-16}	N.A.	8.88×10^{-16}
		Std best so far	5.75×10^{-10}	1. 12	1. 82	N.A.	6.38×10^{-16}
F_{11}	0	Average best so far	4. 86	7.53×10^{-2}	1. 83	0. 00	0. 00
		Best best so far	1. 69	4.25×10^{-11}	1.87×10^{-1}	N.A.	0. 00
		Std best so far	2. 16	1.58×10^{-1}	1. 66	N.A.	0. 00
F_{12}	0	Average best so far	3.19×10^{-2}	3.11×10^{-1}	3.03	0.0176	3.46×10^{-3}
		Best best so far	9.68×10^{-21}	1.15×10^{-19}	2.15×10^{-4}	N.A.	1.52×10^{-24}
		Std best so far	8.78×10^{-2}	3.80×10^{-1}	1.98	N.A.	1.86×10^{-2}
F_{13}	0	Average best so far	7.33×10^{-4}	4.17×10^{-1}	6.14	3.66×10^{-4}	1.10×10^{-2}
		Best best so far	1.02×10^{-18}	7.78×10^{-11}	2.10×10^{-19}	N.A.	3.47×10^{-23}
		Std best so far	2.74×10^{-3}	8.50×10^{-1}	5.53	N.A.	3.18×10^{-1}

N.A.=Not Available

Table 6
Optimization results of benchmark functions in Table 3 with t_{max}=500.

Functions	f_{opt}	Statistical Indicators	GSA	PSO	PSOGSA	Clustered-GSA	Advanced GSA_PSO
F_{14}	1	Average best so far	5. 790	0. 998	2. 176	9. 81	1. 791
		Best best so far	0. 998	0. 998	0. 998	N.A.*	0. 998
		Std best so far	3. 530	4.44×10^{-16}	2. 617	N.A.	1. 098
F_{15}	0.00030	Average best so far	4.32×10^{-3}	5.81×10^{-4}	5.49×10^{-4}	4.90×10^{-3}	4.50×10^{-4}
		Best best so far	1.40×10^{-3}	3.07×10^{-4}	3.07×10^{-4}	N.A.	3.07×10^{-4}
		Std best so far	2.34×10^{-3}	4.25×10^{-4}	3.19×10^{-4}	N.A.	2.27×10^{-4}
F_{16}	−1.0316	Average best so far	−1. 0316	−1. 0316	−1. 0316	−1. 0316	−1. 0316
$n = 2$		Best best so far	−1. 0316	−1. 0316	−1. 0316	N.A.	−1. 0316
		Std best so far	6.66×10^{-16}	6.66×10^{-16}	6.66×10^{-16}	N.A.	6.66×10^{-16}
F_{17}	0.398	Average best so far	0. 3979	0. 3979	0. 3979	0. 3979	0. 3979
$n = 2$		Best best so far	0. 3979	0. 3979	0. 3979	N.A.	0. 3979
		Std best so far	0. 00	0. 00	0. 00	N.A.	1.66×10^{-16}
F_{18}	3	Average best so far	3.00	3.00	3.00	3.00	3.00
$n = 2$		Best best so far	3.00	3.00	3.00	N.A.	3.00
		Std best so far	0. 00	0. 00	0. 00	N.A.	0. 00
F_{19}	−3.86	Average best so far	−3.8628	−3.8628	−3.8628	−3.8628	−3.8628
$n = 3$		Best best so far	−3.8628	−3.8628	−3.8628	N.A.	−3.8628
		Std best so far	3.11×10^{-15}	2.49×10^{-5}	3.11×10^{-15}	N.A.	3.10×10^{-15}
F_{20}	−3.32	Average best so far	−3.322	−3.255	−3.242	−3.3180	−3.2626
$n = 6$		Best best so far	−3.322	−3.322	−3.322	N.A.	−3.3220
		Std best so far	1.77×10^{-15}	6.27×10^{-2}	5.60×10^{-2}	N.A.	5.95×10^{-2}
F_{21}	−10.1532	Average best so far	−7.546	−6.228	−7.564	−7.4141	−10.1530
$n = 4$		Best best so far	−10.1532	−10.1532	−10.1532	N.A.	−10.1530
		Std best so far	3.458	−3.331	3.258	N.A.	0. 00
F_{22}	−10.4028	Average best so far	−10.4029	−7.15	−8.456	−10.4029	−10.1803
$n = 4$		Best best so far	−10.4029	−10.4029	−10.4029	N.A.	−10.4029
		Std best so far	7.10×10^{-15}	3.337	3.035	N.A.	1.1988
F_{23}	−10.5363	Average best so far	−10.5364	−6.589	−8.113	−10.5364	−10.0897
$n = 4$		Best best so far	−10.5364	−10.5364	−10.5364	N.A.	−10.5364
		Std best so far	8.88×10^{-15}	3.725	3.470	N.A.	1.6715

N.A. = Not Available

optimum solution of each function.

As can be observed in Table 4, in Functions F_1 to F_5 and F_7, Advanced GSA_PSO has demonstrated the higher ability to explore and exploit in comparison with GSA, PSO and PSOGSA; as a result, Advanced GSA_PSO provides much better solutions.

In Function F_6, two algorithms GSA and Advanced GSA_PSO could find the optimum solution although Advanced GSA_PSO has a faster convergence rate. The process of convergence of GSA, PSO, PSOGSA and Advanced GSA_PSO in finding the optimized solution for F_1, F_3, F_5 and F_6, are shown in Fig. 4.

The optimization results of multimodal high-dimensional functions are given in Table 5. As explained above, Functions F_8 to F_{13} have many local minima so that the number of local minima increases exponentially as the dimension of the function increases. As can be seen, in Functions F_8 to F_{12}, the performance of Advanced GSA_PSO is better than others, especially in Function F_{10}, Advanced GSA_PSO performs significantly better in exploring and exploiting of the search space. In Function F_{13}, although *Average best so far* of Advanced GSA_PSO is less than its corresponding values in GSA and Clustered-GSA, it has obtained a better result for *Best best so far*. The process of convergence of GSA, PSO, PSOGSA and Advanced GSA_PSO to reach the optimized solution for F_8, F_9, F_{10} and F_{12}, are illustrated in Fig. 5.

The optimization results of multimodal low-dimensional functions are provided in Table 6. In Functions F_{16} to F_{19}, all algorithms could converge to the optimum solution. It is obvious that in Functions F_{14}, F_{15} and F_{21} the exploration ability of Advanced GSA_PSO performs better than others because it is more robust. In Functions F_{22} and F_{23}, GSA, Clustered GSA and Advanced GSA_PSO perform similar but convergence rate of Advanced GSA_PSO is faster. In Fig. 6, the process of convergence of GSA, PSO, PSOGSA and Advanced GSA_PSO in looking for the optimized solution for F_{14}, F_{15}, F_{20} and F_{21}, are shown.

According to the results, it can be inferred that, Advanced

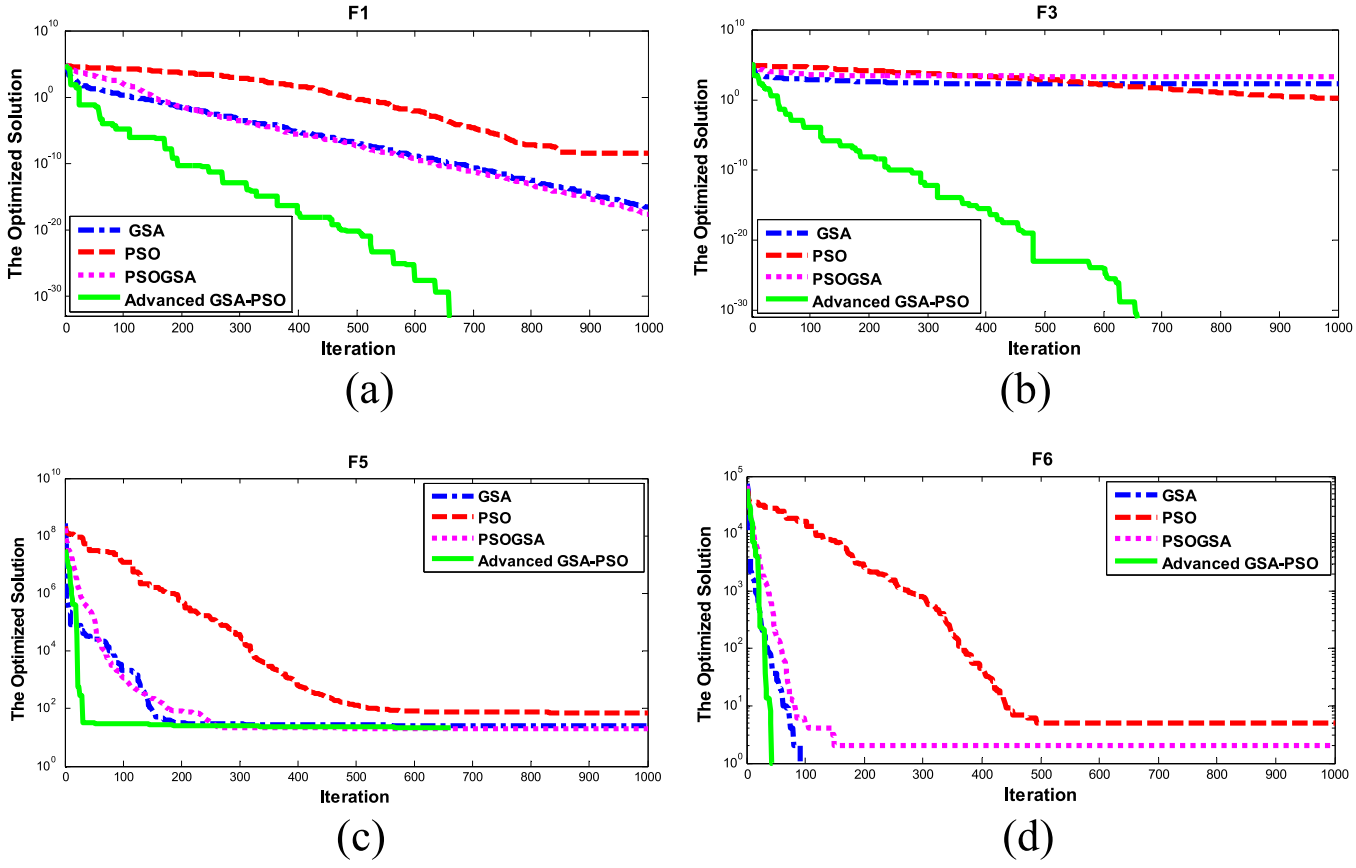


Fig. 4. Comparison of the convergence process of GSA, PSO, PSOGSA and Advanced GSA_PSO for unimodal functions: (a) F_1, (b) F_3, (c) F_5 and (d) F_6.

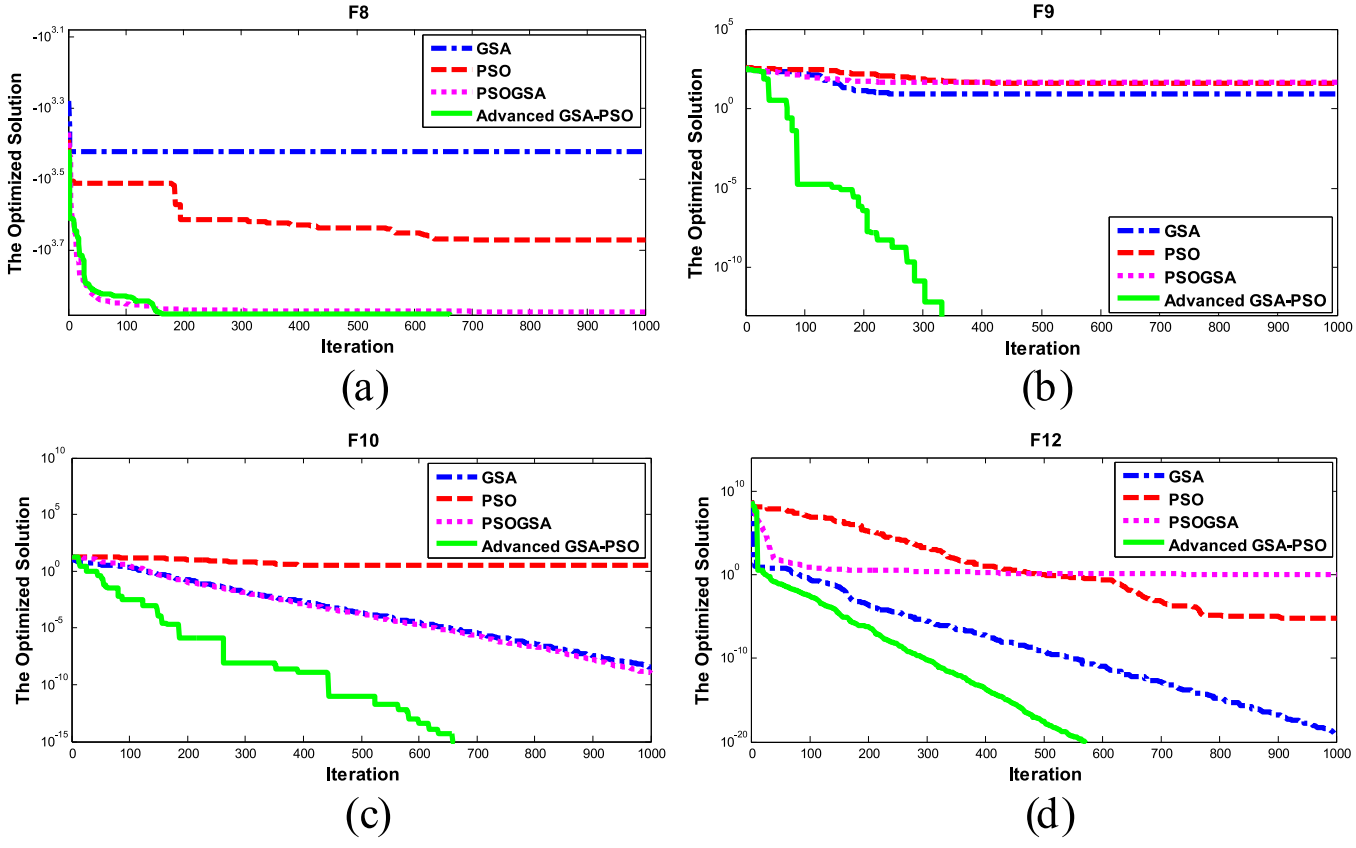


Fig. 5. Comparison of the convergence process of GSA, PSO, PSOGSA and Advanced GSA_PSO for multimodal functions with high dimension: (a) F_8, (b) F_9, (c) F_{10} and (d) F_{12}.

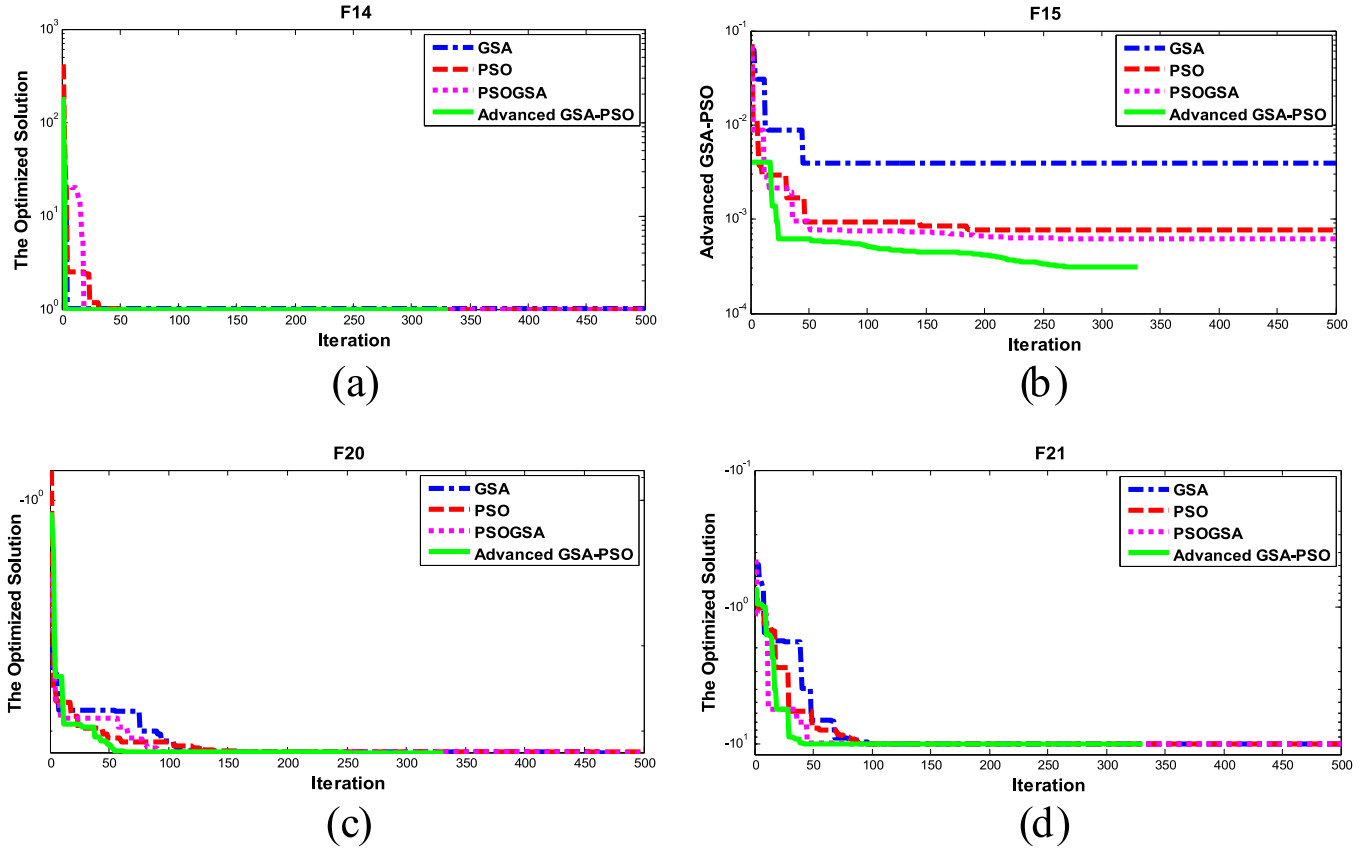


Fig. 6. Comparison of the convergence process of GSA, PSO, PSOGSA and Advanced GSA_PSO for multimodal functions with low dimension: F_{14}, (b) F_{15}, (c) F_{20} and (d) F_{21}.

GSA_PSO is able to demonstrate better capability in the exploration and exploitation of the search space of the most of benchmark functions and hence, it would be introduced as a promising optimization algorithm for circuit sizing purposes as we will discuss in detail in the next section.

5. A Novel tool for analog IC sizing based on Advanced GSA_PSO

In this section, a novel ICs sizing tool is proposed for automated design of analog ICs. As mentioned in Section 1, basically, circuit sizing tools consist of two main sections, i.e., synthesis and optimization which are linked together. Hence, our proposed tool includes two sections: the synthesis section which uses the simulation-based optimization method, and the optimization section which employs the proposed algorithm, Advanced GSA_PSO. It should be noted that analog circuits are simulated by HSPICE simulator and Advanced GSA_PSO is implemented in MATLAB. The synthesis and optimization sections of this tool are connected together through the link between MATLAB and HSPICE software.

5.1. Constrained-Advanced GSA_PSO

The circuit sizing problems would be defined as either constrained single- or multi-objective optimization problems. Aside from this, metaheuristic algorithms are only able to solve unconstrained optimization problems. Hence, constraint handling techniques should be considered as complementary methods to deal with constrained problems. *Static penalty function* is a constraint handling technique that is used by most of the circuit sizing methods [6,18,19,76]. Consequently, we have combined Advanced GSA_PSO with *static penalty function* in order to employ a powerful method for solving constrained optimization problems. In constrained-Advanced GSA_PSO, the original constrained optimization problem is transformed into an unconstrained one as follows:

$$f'(x) = f(x) + \sum_{i=1}^{n} \omega_i \times \langle g_i(x) \rangle \tag{16}$$

where *f(x)* is the objective function to be minimized, and ω_i are the penalty weighting coefficients. $\langle g_i(x) \rangle$ returns the absolute value of $g_i(x)$ if it is positive, and zero otherwise assuming that $g_i(x) \leq 0$, i=1,2,..., n. Note that, the penalty weighting coefficients,w_i, are determined in advance and can be kept constant during the optimization process [6].

5.2. The arhitecture of the proposed tool

The functionality of our proposed tool in analog circuits sizing and optimization is described in this section. The architecture of this tool is illustrated in Fig. 7. The figure indicates that at the beginning, design parameters and constraints are determined by the designer, while a reasonable pre-defined range is also taken into account for each design parameter. Note that design parameters consist of the length and width of the MOS transistors, capacitor values, and biasing currents or voltages..

Then, an initial population of the circuit parameters (N masses) is randomly generated and given to the synthesis section as a starting point of the design process. In synthesis section, the initial design parameters are sent to the input file of HSPICE (i.e., *.sp file*). Next, HSPICE simulates the circuit considering the received inputs and saves the output values including DC gain, unity-gain frequency, slew rate, phase margin, power consumption, etc., in the output file of HSPISE (i.e., *.lis file*). The proposed tool reads.*lis file* through the link between MATLAB and HSPICE and calculates the violation of each constraint and accordingly, the fitness function is evaluated using (16).

In the next step, the proposed tool creates new design parameters for the next iteration by running the optimization process of Advanced

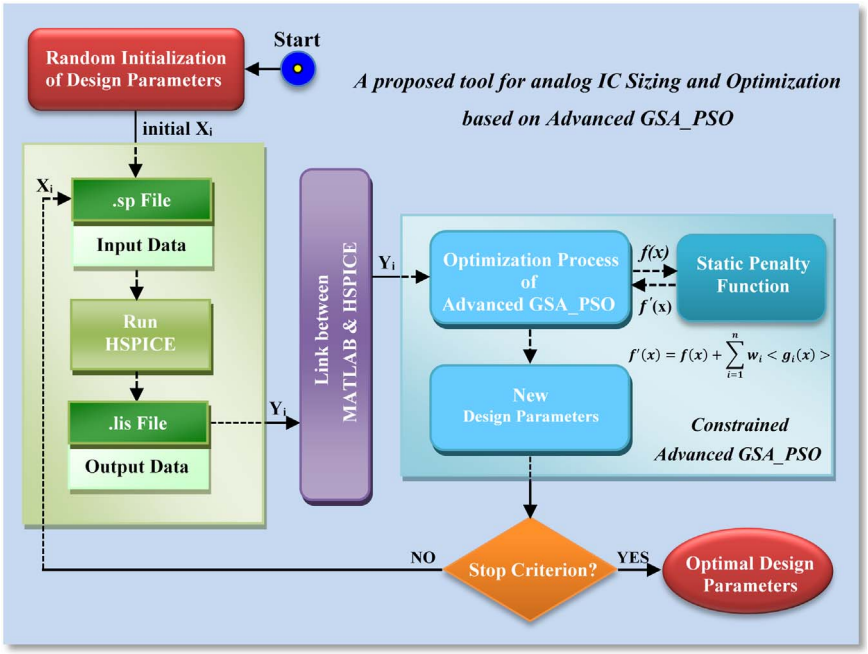


Fig. 7. The proposed tool architecture overview.

GSA_PSO. This process is continued until the stop criterion is satisfied. Finally, the optimal circuit sizing is found and reported for the analog circuit.

5.3. Case studies

To validate the performance of our proposed tool in the analog circuit sizing, we consider two cases. The first case is a two-stage CMOS op-amp and the second one is a single ended folded-cascode op-amp. The main purpose of the former case is to test the capability of our proposed tool to handle constraints in both typical and severe conditions, while the performance of this tool is further illustrated through the latter case which has more design specifications, parameters and objectives.

Note that in Advanced GSA_PSO, the number of masses is considered to be 10 (N=10), G_O, α and t_{max} in (8) are set to 100, 2 and 100, respectively. R_0 and R_{min} in (14) are also assumed to be 10 and 0.01, respectively. In addition, c_1' and c_2' in (12) are set to 1. ω_i in (16) are set to 50,000 (normal conditions) and 80,000 (severe constraints) in Case 1 and 500 in Case 2.

The experiments are executed on an Intel(R) Core(TM) i7-5500UCPU at 2.40 GHz with 8 GB of RAM. These cases are discussed as follows.

5.3.1. Case1: two-stage CMOS operational amplifier

A two-stage CMOS op-amp with Miller compensation is shown in Fig. 8. In this case, design parameters are the compensation capacitance (C_c), biasing current (I_{bias}), the total values of transistor widths (W_i, $i=1,\ldots,8$) and lengths (L_i, $i=1,\ldots,8$). Here, the appropriate matching relations are imposed: $M_1 \equiv M_2$, $M_3 \equiv M_4$ and $M_5 \equiv M_8$. It means that the number of independent design parameters is 12. Therefore, the vector of design parameters is defined as follows which has to be optimally determined by the proposed tool.

$$X_i = [W_1\ W_3\ W_5\ W_6\ W_7\ L_1\ L_3\ L_5\ L_6\ L_7\ C_C\ I_b] \quad (17)$$

.

In addition, the positive power supply (V_{DD}), the negative power supply (V_{SS}) and the load capacitance (C_L) are equal to 2.5 V, −2.5 V and 30 pF, respectively.

For the automated sizing of the above-mentioned two-stage CMOS op-amp, our proposed tool aims to minimize the power consumption (as the optimization goal) with constraints on DC gain, gain-bandwidth product (GBW), phase margin, slew rate, the total CMOS transistor area as well as the functional constraints that all the transistors should be in the saturation region. The design objective and constraints are indicated in the second column of Table 7.

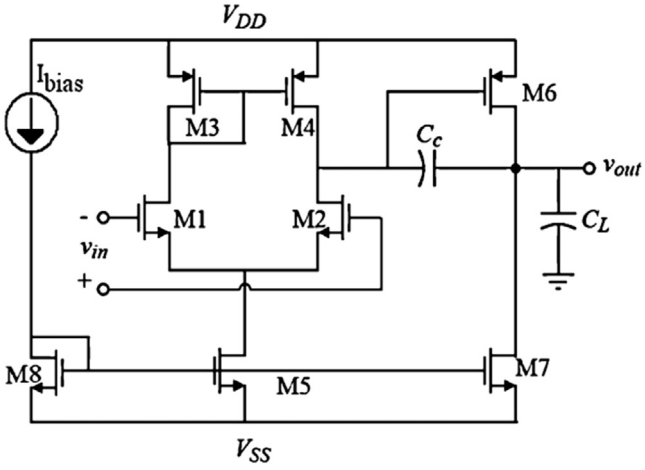


Fig. 8. Two-stage CMOS op-amp with 0.25 μm Technology [77].

The capability of the proposed tool in the optimal sizing of the two-stage CMOS op-amp is compared with other ICs sizing tools; GA+PF (genetic algorithm and the static penalty function to handle constraints), DE+PF (differential evolution algorithm and the static penalty function method to handle constraints) and CODE (co-evolutionary differential evolution and the augmented Lagrangian method to handle constraints) [42]. The synthesis section of all these tools is based on simulation in which HSPICE simulator has been employed.

For a fair comparison between the aforementioned algorithms in the same conditions, the same technology 0.25 μm CMOS process with 5 V power supply is used.

Table 7 shows the results of the proposed tool in comparison with three other methods. The obtained results clearly demonstrate that our tool not only satisfies all specifications and design constraints, but also minimizes the power consumption significantly in comparison with GA

Table 7
Specifications and results of the proposed tool based on Advanced GSA_PSO in comparison with other tools (Case 1).

Specification	Constraints	GA+PF	DE+PF	CODE	Advanced GSA_PSO
DC gain (dB)	≥ 70	72.601	80.659	76.48	70.441
GBW (MHz)	≥ 2	4.523	2.041	2.068	2.017
Phase Margin (°)	≥ 50	49.876	55.641	55.946	50.181
Slew Rate (V/μs)	≥ 1.5	1.649	1.503	1.521	2.231
Area (μm^2)	≤ 300	N.A.*	N.A.	N.A.	210.003
Output Swing (V)	≥ 2	2.126	1.918	2.202	2.415
CMRR (dB)	≥ 70	70.99	70.018	90.01	88.187
$PSRR^+$ (dB)	≥ 70	74.658	80.802	76.571	72.675
$PSRR^-$ (dB)	≥ 70	N.A.	N.A.	N.A.	131.91
M_1	Saturation	Met	Met	Met	Met
M_3	Saturation	Met	Met	Met	Met
M_5	Saturation	Met	Met	Met	Met
M_6	Saturation	Met	Met	Met	Met
M_7	Saturation	Met	Met	Met	Met
Power Consumption (mW)	Objective	2.215	1.116	0.731	0.400
Run Time (s)	–	10,126	9,865	10,097	280.15
Technology	–	0.25 μm	0.25 μm	0.25 μm	0.25 μm

N.A.: Not Available.

+PF, DE+PF, and CODE. It is obvious that the circuit performance of the proposed tool is quite preferable to its rivals because its total run time is impressively lower than other methods.

The optimal design parameters of the two-stage CMOS op-amp obtained by our proposed tool are provided in Table 8. Moreover, the optimization trend of our tool in minimizing the power consumption has been illustrated in Fig. 9. As can be seen, it yields different feasible solutions at each iteration but the solution with the minimum power consumption (the best solution) is only presented in Table 7. Therefore, it corroborates that, this tool has the ability to suggest various solutions considering the performance requirements ordered by the ICs designer.

Fig. 10(a) shows the voltage-transfer curve of the two-stage CMOS op-amp with the optimal design parameters as reported in Table 8. The figure indicates that the input systematic offset voltage (*Vos*) is equal to 503.95 μV. Fig. 10(b) also shows the voltage-transfer curve when *Vos* has been cancelled by the technique presented in [77]. The design specifications of the two-stage CMOS op-amp including ICMR, slew rate, gain and phase margins, CMRR, and PSRR are plotted by our proposed tool in Figs. 11–15, respectively.

To evaluate the performance of our proposed tool in dealing with severe constraints, we consider now a set of more severe constraints in comparison with the ones mentioned in the second column of Table 7. These constraints together with the obtained results of the tools GA +PF, DE+PF, CODE, and Advanced GSA_PSO are shown in Table 9. As can be seen, among these tools, only the algorithms CODE and Advanced GSA_PSO are able to satisfy such severe constraints while the power consumption presented by Advanced GSA_PSO is nearly half of the value obtained by CODE. Furthermore, the comparison between the design parameters of these two algorithms shows that the

Table 8
Optimum design parameters obtained by the proposed tool (Case 1).

Design Parameters	Advanced GSA_PSO
W_1/L_1,W_2/L_2 (μm/μm)	16.5646/2.1758
W_3/L_3,W_4/L_4 (μm/μm)	46.9744/0.6127
W_5/L_5, W_8/L_8 (μm/μm)	8.0200/2.8223
W_6/L_6 (μm/μm)	84.4042/0.2895
W_7/L_7 (μm/μm)	6.5805/1.6189
C_C (pF)	13.4488
I_{bias}(μA)	30.0000

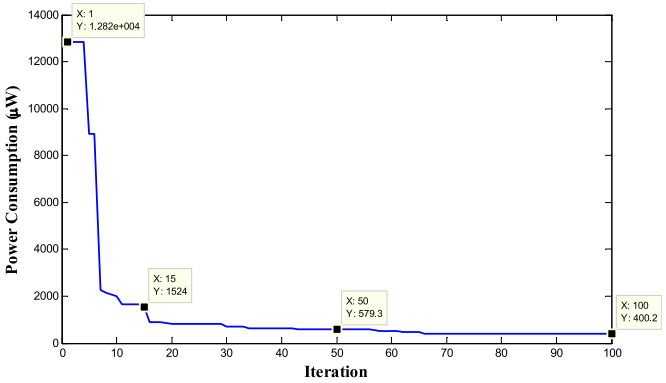


Fig. 9. Optimization trend of the power consumption using the proposed tool (Case 1).

total circuit area obtained by Advanced GSA_PSO is less. Table 10 provides the optimum design parameters obtained by CODE and Advanced GSA_PSO algorithms for two-stage CMOS op-amp in severe constraints.

5.3.2. Case 2: single ended folded-cascode Op-Amp

A single ended folded-cascode amplifier is shown in Fig. 16. In this case, design parameters are the biasing current (I_{bias}), the biasing voltages (V_{cm1}, V_{cm2}), the total values of transistor widths (W_i, *i=1, ...,13*) and lengths (L_i, *i=1,...,13*). Here, the appropriate matching relations are imposed: $M_1 \equiv M_2$, $M_3 \equiv M_4 \equiv M_{bp}$, $M_5 \equiv M_{bn}$, $M_6 \equiv M_7$, $M_8 \equiv M_9$ and $M_{10} \equiv M_{11}$. It means that the number of independent design parameters is 15. Therefore, the vector of design parameters is defined as follows which has to be optimally determined by the proposed tool.

$$X_i=[W_1W_3W_5W_6W_8W_{10}L_3L_5L_6L_8L_{10}I_{bias}V_{cm1}V_{cm2}] \quad (18)$$

.

The allowed ranges of the design parameters are assumed as follows: transistor widths are allowed to vary between [0.24, 200] μm, transistor lengths can be changed between [0.18, 5] μm, the biasing current is allowed to vary between [30,400] μA, and the biasing voltages V_{cm1} and V_{cm2} can be varied between [−0.4, 0] V and [0, 0.4] V, respectively. In addition, the positive power supply (V_{DD}), the negative power supply (V_{SS}) and the load capacitance (C_L) are equal to 0.9 V, −0.9 V and 5 pF, respectively.

For the automated sizing of the folded-cascode op-amp, our proposed tool aims to minimize both the power consumption and the total circuit area (as the optimization goals). The design constraints are classified to two groups, first, the constraint defined by designer including DC gain, gain-bandwidth product (GBW), phase margin, slew rate and output swing, and second, the functional constraints according to which all the transistors must be in the saturation region. The design specifications, constraints and objectives are listed in the first and second columns of Table 11.

The capability of the proposed tool in the optimal sizing of the folded-cascode op-amp is compared with two important circuit design tools FRIDGE [80] and GENOM [44]. Note that the synthesis section of all these tools is based on simulation in which HSPICE simulator has been employed. Moreover, for a fair comparison between the aforementioned tools in the same conditions, the same technology 0.18 μm CMOS process with 1.8 V power supply is used.

It is worth mentioning that decomposition approaches [78] are generally used to transform a multi-objective problem into many single-objective problems, among them the *weighted aggregation* and *ε-constraint* methods are the most widely used. To solve two-objective optimization problem, FRIDGE and GENOM have used the weighted aggregation method but in this paper, we have applied the ε-constraint method due to its several advantages over the weighted aggregation method mentioned in [79]. Accordingly, the minimization

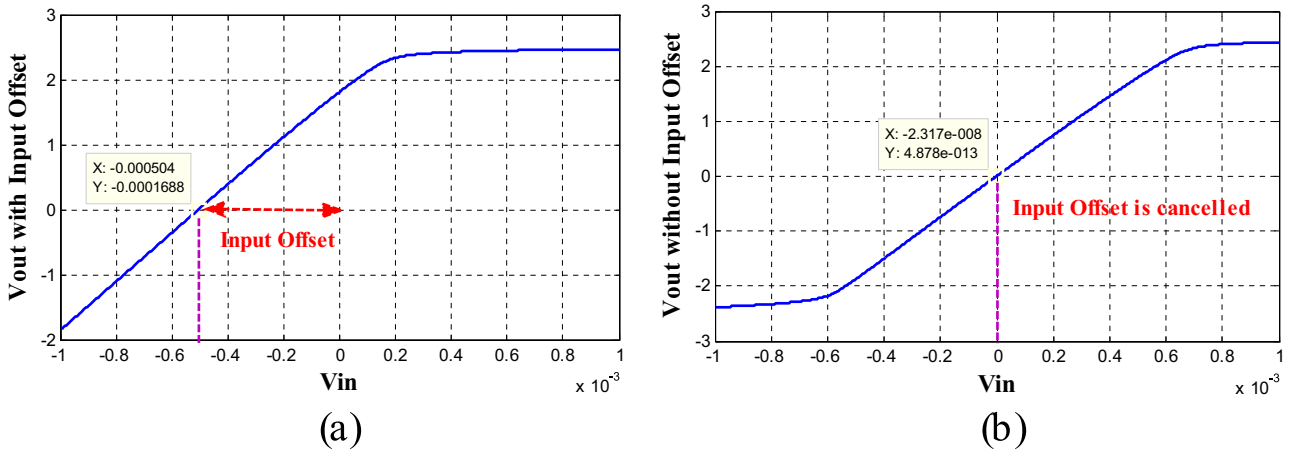


Fig. 10. Voltage-transfer curve (Case 1): (a) with input systematic offset, (b) after offset cancellation.

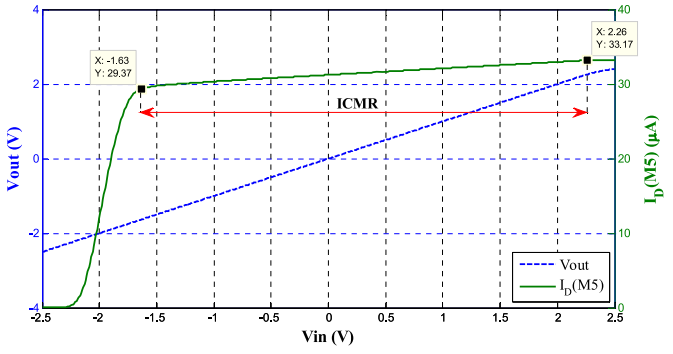


Fig. 11. ICMR plotted by the proposed tool (Case 1).

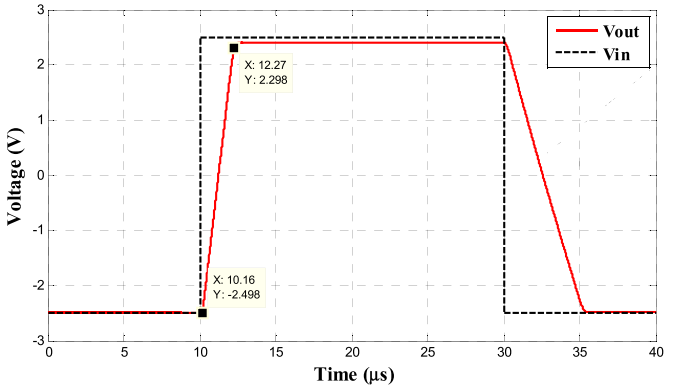
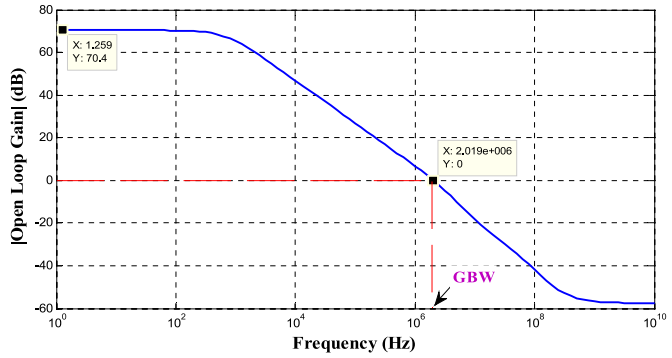


Fig. 12. Slew Rate behavior plotted by the proposed tool (Case 1).

of the power consumption is considered as the objective function and the second goal (i.e., circuit area minimization) is incorporated in the constraint part of the model.

Table 11 shows the results of the proposed tool in comparison with two other tools. The obtained results clearly demonstrate that our tool not only satisfies all specifications and design constraints, but also it exhibits a relatively better performance than FRIDGE and GENOM. The area obtained by our proposed tool, 16.961 μm^2, is less than the area presented by FRIDGE, 23.71 μm^2, and nearly equal with the area presented by GENOM, 16.87 μm^2. On the other hand, the power obtained by our tool, 222.1 μW, is less than the power presented by FRIDGE, 233.3 μW, and GENOM, 244.6 μW.

The optimal design parameters of the folded-cascode op-amp obtained by our proposed tool, FRIDGE and GENOM are provided in Table 12. Moreover, the optimization trend of our tool in minimizing the power consumption is also illustrated in Fig. 17.



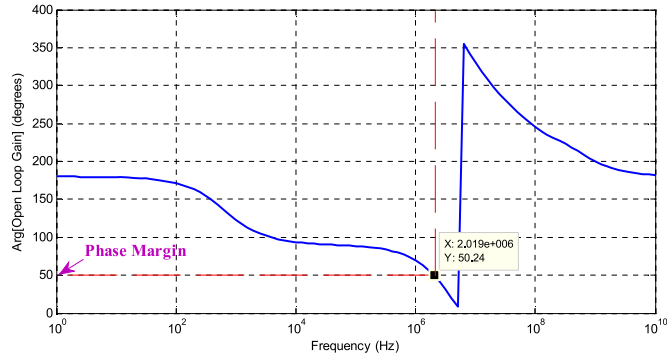


Fig. 13. Bode curve plotted by the proposed tool (Case 1): Gain margin and Phase margin.

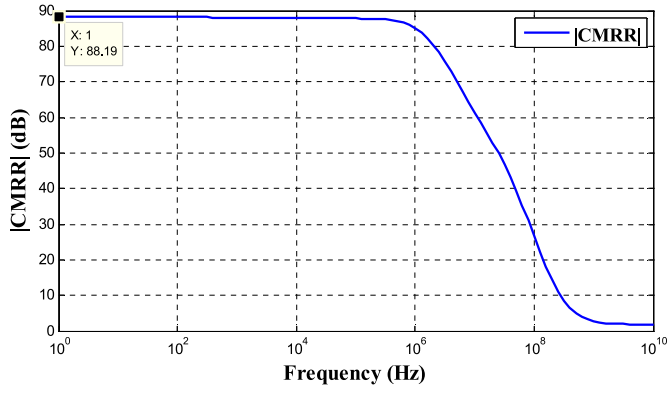


Fig. 14. CMRR plotted by the proposed tool (Case 1).

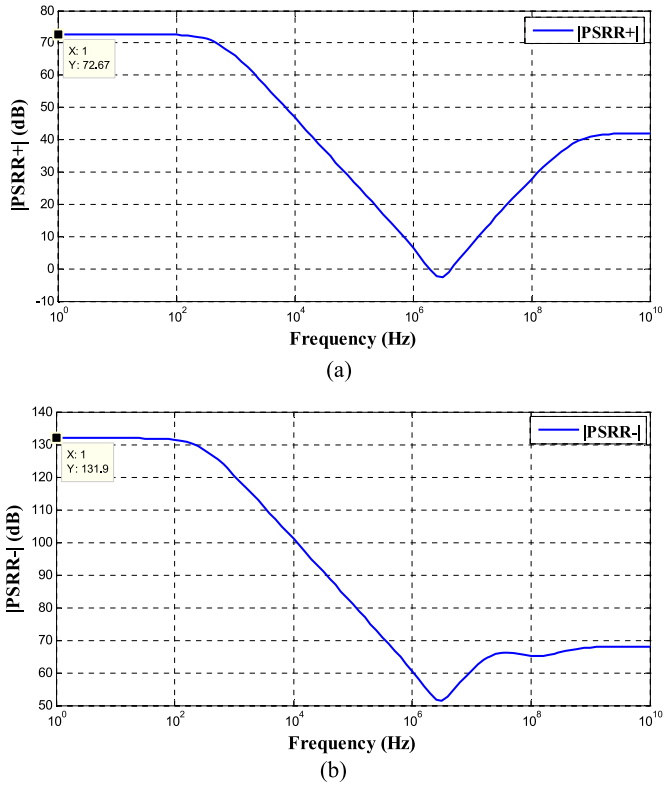


Fig. 15. PSRR plotted by the proposed tool (Case 1): (a) $PSRR^{+}$, (b) $PSRR^{-}$.

Table 9
Specifications and results of the proposed tool based on Advanced GSA_PSO in comparison with other tools (with severe constraints in Case 1).

Specification	Cons.	GA+PF	DE+PF	CODE	Advanced GSA_PSO
DC gain (dB)	≥ 85	78.436	85.92	86.1	85.111
GBW (MHz)	≥ 2.5	6.643	2.768	2.505	2.502
Phase Margin (°)	≥ 55	54.96	54.785	57.746	55.005
Slew Rate (V/μs)	≥ 1.8	1.925	1.863	1.971	2.753
Output Swing (V)	≥ 2	2.227	2.045	2.096	2.393
CMRR (dB)	≥ 80	77.553	75.667	80.452	103.908
$PSRR^{+}$ (dB)	≥ 85	78.641	81.02	86.13	86.560
$PSRR^{-}$ (dB)	≥ 85	N.A.*	N.A.	N.A.	123.360
Power Consumption (mW)	Obj	2.434	2.129	1.014	0.575
Run Time (s)	–	10,533	11,037	11,206	268.772
Technology	–	0.25 μm	0.25 μm	0.25 μm	0.25 μm

N.A.: Not Available.

Table 10
Optimum design parameters obtained in severe constraints (Case 1).

Design Parameters	CODE	Advanced GSA_PSO
W_1/L_1,W_2/L_2 (μm/μm)	10.3/4.34	16.5646/2.1758
W_3/L_3,W_4/L_4 (μm/μm)	99.48/0.7	46.9744/0.6127
W_5/L_5, W_8/L_8 (μm/μm)	99.48/4.77	8.0200/2.8223
W_6/L_6 (μm/μm)	85.66/0.59	84.4042/0.2895
W_7/L_7 (μm/μm)	46.27/3.88	6.5805/1.6189
C_C (pF)	40.023	13.4488
I_{bias}(μA)	217.3	30.0000
Area (μm^2)	1,407.8	796.508

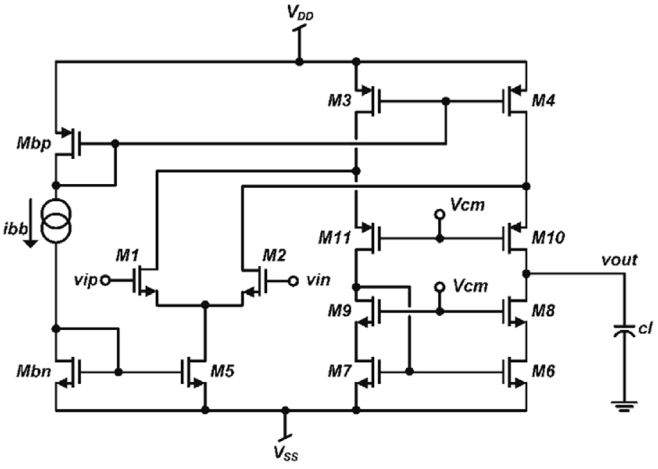


Fig. 16. Single ended folded-cascode Op-Amp with 0.18 μm technology [5].

Table 11
Specifications and results of the proposed tool based on Advanced GSA_PSO in comparison with other tools (Case 2).

Specification	Constraints	FRIDGE	GENOM	Advanced GSA_PSO
DC gain (dB)	> 70	70	70.61	70.427
GBW (MHz)	> 12	16	15.35	15.505
Phase Margin (°)	> 55	80.6	79.60	83.574
Slew Rate (V/μs)	> 10	15.3	15.36	10.001
dm1[a]	> 1.2	9.78	9.245	15.067
dm3	> 1.2	5.20	1.568	2.419
dm5	> 1.2	2.21	1.836	5.540
dm6	> 1.2	10.5	8.171	2.217
dm8	> 1.2	3.05	2.807	2.220
dm10	> 1.2	1.95	1.653	11.709
M1	Saturation	Met	Met	Met
M3	Saturation	Met	Met	Met
M5	Saturation	Met	Met	Met
M6	Saturation	Met	Met	Met
M8	Saturation	Met	Met	Met
M10	Saturation	Met	Met	Met
OSP[b] (V)	> 0.5	0.625	0.566	0.695
OSN[c] (V)	< −0.5	−0.502	−0.5057	−0.641
Area (μm^2)	Objective	23.71	16.87	16.961
Power Consumption (μW)	Objective	233 .3	244.6	222.1
Run Time (s)	–	N.A.[d]	53.68	299.67
Technology	–	0.18 μm	0.18 μm	0.18 μm

[a] dm1=VDS (M1)/VDS_{sat} (M1).
[b] OSP (Output Swing Positive)=VDD-abs (VDS_{sat} (M10))-abs (VDS_{sat} (M4)).
[c] OSN (Output Swing Negative)=VSS-abs (VDS_{sat} (M6))-abs (VDS_{sat} (M8)).
[d] N.A.=Not Available.

Table 12
Optimum design parameters obtained (Case 2).

Design parameters	FRIDGE	GENOM	Advanced GSA_PSO
W_1 (μm)/L_1 (μm)	19.51/1.56	14.91/1.38	33.211/0.568
W_3 (μm)/L_3 (μm)	30.34/0.47	6.99/1.94	35.802/0.334
W_5 (μm)/L_5 (μm)	71.31/0.38	36.78/0.37	57.331 0.523
W_6 (μm)/L_6 (μm)	104.53/0.76	63.04/0.91	5.305/1.382
W_8 (μm)/L_8 (μm)	65.62/2.06	31.45/0.89	48.808/0.873
W_{10} (μm)/L_{10} (μm)	3.08/0.60	7.32/2.19	73.429/0.367
Ibias (μA)	48.42	48.51	50.005
Vcm1 (mV)	−87.554	−44.90	0
Vcm2 (mV)	62.47	1	0

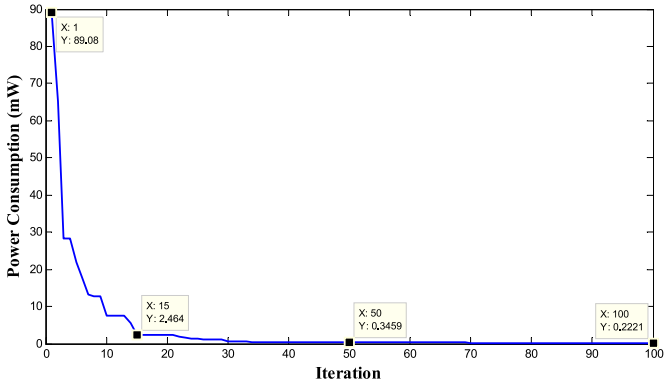


Fig. 17. Optimization trend of the power consumption using the proposed tool (Case 2).

Table 13
Results of statistical study of the proposed tool (Case 1).

Design Specifications	Average	Best	STD
DC gain (dB)	71.38	76.24	1.8657
GBW (MHz)	2.36	8.29	1.3653
Phase Margin (°)	52.25	65.54	5.0238
Slew Rate (V/μs)	14.51	136.36	32.6054
Area (μm^2)	223.75	52.51	65.2124
Output Swing (V)	2.37	2.42	0.0458
Power Consumption (mW)	0.437	0.184	0.098
Feasible Solutions	17.5	29	5.894
Infeasible Solutions	None	None	None
Time(s)	276.18	260.24	6.588

Table 14
Results of statistical study of the proposed tool (Case 2).

Specification	Average	Best	STD
DC gain (dB)	70.64	72.04	0.6243
GBW (MHz)	16.53	36.37	6.1749
Phase Margin (°)	77.39	83.57	3.1309
Slew Rate (V/μs)	12.55	25.73	4.2843
dm1	15.92	17.78	1.3530
dm3	1.97	2.98	0.4882
dm5	4.08	5.88	1.2500
dm6	1.96	2.90	0.4120
dm8	2.73	3.90	0.6025
dm10	7.50	12.87	2.6150
OSP (V)	0.64	0.72	0.0466
OSN (V)	−0.63	−0.73	0.0552
Area (μm^2)	18.37	15.64	0.8718
Power Consumption (μW)	0.26	0.19	0.0935
Feasible Solutions	26.30	49	9.6752
Infeasible Solutions	None	None	None
Run Time (s)	292.93	276.35	6.7233

Table 15
Comparison between the best feasible and robust solutions of the proposed tool (Case 1).

Design Specifications	The Best Feasible Solution	The Best Robust Solution
DC gain (dB)	70.441	73.620
GBW (MHz)	2.016	2.528
Phase Margin (°)	50.181	52.245
Slew Rate (V/μs)	2.231	2.449
Area (μm^2)	210.003	246.549
Output Swing (V)	2.415	2.407
Power Consumption (mW)	0.400	0.488
Run Time (s)	268.772	23.74

Table 16
Comparison between the best feasible and robust solutions of the proposed tool (Case 2).

Specification	The Best Feasible Solution	The Best Robust Solution	GENOM
DC gain (dB)	70.427	71.398	68.719
GBW (MHz)	15.505	15.402	18.450
Phase Margin (°)	83.574	83.221	74.353
Slew Rate (V/μs)	10.001	10.001	21.030
dm1	15.067	15.023	8.352
dm3	2.419	2.411	2.588
dm5	5.540	4.929	1.803
dm6	2.217	1.695	10.086
dm8	2.220	3.348	2.828
dm10	11.709	10.601	1.695
OSP (V)	70.427	0.695	0.575
OSN (V)	15.505	−0.582	−0.618
Area (μm^2)	16.961	16.360	24.509
Power Consumption (μW)	222.1	223.3	328.600
Run Time (s)	299.67	82.833	411.18

6. The robustness validation of the proposed tool

6.1. Statistical study

Due to significant capabilities of metaheuristic algorithms, these algorithms are welcomed to be used in the optimization section of ICs sizing tools. However, the random nature of these algorithms will result in a different solution by each time running the ICs sizing tool. It is thus essential to note that by executing these tools for only once, one cannot conclude whether their performance is good or not. In the tools introduced by [42] and [44], it is not stated that, on average, by running how many times of their optimization algorithms, the results have been obtained. Hence, it can be deduced that [42] and [44], have reported the best results of their ICs sizing tools. For this reason, we also reported the best results of our proposed tool compared with them in Tables 7 and 11.

Therefore, to validate the actual performance of our proposed tool and to test its robustness appropriately, a statistical study is needed. We executed 20 runs of the proposed tool starting from 20 different initializations. The results of our tool for Case 1 and Case 2 are provided in Tables 13, 14, respectively. In fact, these tables show the real performance of this tool in design of the two-stage CMOS op-amp and folded-cascode op-amp circuits. In both tables, four columns on the right show the values of the average, median, best and standard deviation for the design specifications obtained in these 20 runs, respectively.

It is easily observed that all 20 runs result in feasible solutions; extensively the search space and exploit fully the best feasible solutions.

6.2. Corners analysis

To design an efficient and robust circuit, the corners analysis is usually performed during the optimization process, but the main drawback of this approach is, by far, the computational burden involved. This is due to the fact that each candidate solution must not only satisfy all the design specifications and constraints, but also it has to be assessed in each of process, voltage, and temperature (PVT) variations. On the other hand, if the obtained solutions are not desirable or even no feasible solution is found then the designer must rerun the optimization process again which can be time consuming. To save computing time in the procedure of corners analysis, for instance, the authors of [81] have proposed a method in which the circuit is first simulated in the typical process, then the feasible solutions generated by the primary optimization loop are delivered to a secondary optimization loop to guarantee the final feasible solutions under PVT variations. However, the computation time is not reported in their

work.

In this paper, we propose another straightforward strategy with the aim of alleviating such computational burden. At the first stage, the circuit transistors are simulated in typical conditions with respect to the nominal values of voltage supplies and temperature. In this stage, the solutions that have satisfied all the design specifications and constraints, while the objective function has been also optimized, are introduced as *feasible solutions*. In the next stage, to guarantee the robustness of the feasible solutions, they are evaluated under PVT variations. Lastly, the final solutions are those feasible solutions that can tolerate the PVT variations and still cope with all constraints, i.e., their corresponding violations is equal to zero. Note that our proposed circuit sizing tool reports these solutions as *robust solutions*.

Here, it is important to point out that the types of process corners are usually named as two-letter designators including TT, FF, SS, FS, and SF meaning the typical/typical, fast/fast, slow/slow, fast/slow, and slow/fast process, respectively. Note also that the first letter refers to the NMOS corner and the second letter refers to the PMOS corner.

In fact, if an algorithm has a high capability in the exploration and exploitation of the search space, it can definitely find more and very accurate solutions. Thereby, we can only perform corners analysis for a finite set of feasible solutions instead of applying to each of candidate solution during optimization process. Hence, this strategy allows a further decrease in the computational burden of the circuit sizing process.

To validate the robustness and efficiency of our proposed tool for process variations and different operational conditions, we have considered all process corners, supply voltage and temperature variations. It should be noted that in Case 1, each of the final solution obtained by our proposed tool is examined for 81 states resulting from the cross combination of three process corners (SF, FS and TT), supply voltage variations (±10% VDD/VSS) and temperature variations (−40° C, +25° C and +85 °C). On the other hand, for the sake of comparison with [44], supply voltage variations are neglected in Case 2, and hence only 9 states are examined.

The best feasible and robust solutions proposed by our tool for the sizing of the two-stage CMOS op-amp (Case 1) and folded-cascode op-amp (Case 2) circuits are provided in Tables 15, 16, respectively. Note that *the best feasible solution* and *the best robust solution* infer to the optimum value obtained from the sets of feasible and robust solutions, respectively.

At the first step, by running the proposed tool, it found 15 (in Case 1) and 49 (in Case 2) feasible solutions that the best of which are reported in the second column of Tables 15 and 16, respectively. Then, the corners analysis is applied to all of these feasible solutions and in this condition, 2 (in Case 1) and 8 (in Case 2) robust solutions are only obtained that the best of which are given in the third column of Tables 15 and 16, respectively.

In Case1, the comparison of robust solutions with the results of [42] is not possible because it has not performed corners analysis. It is also

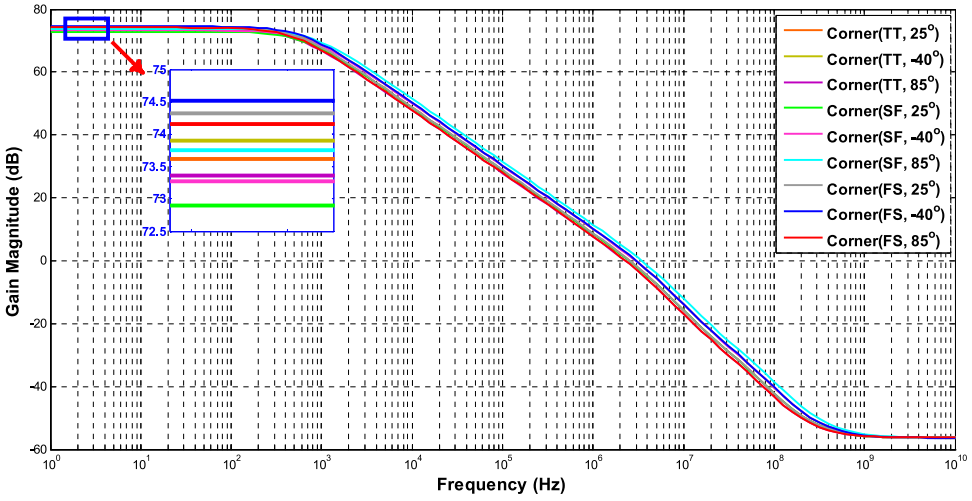


Fig. 18. Gain magnitude of the best robust solution in the corners analysis (Case 1).

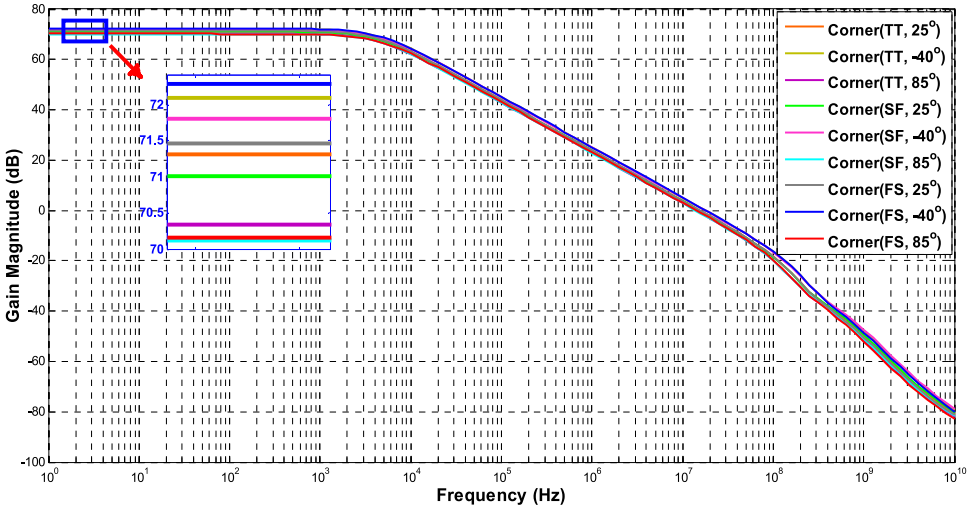


Fig. 19. Gain magnitude of the best robust solution in the corners analysis (Case 2).

Table 17
Numerical results of corners analysis for the best robust solution (Case 1).

Corner	Process	Temperature	Specification			
			Gain (dB) > 70	GBW (MHz) > 2	PM (°) > 50	Output Swing (V) > 2
1	TT	25°	73.620	2.528	52.245	2.407
2	TT	−40°	73.894	2.977	50.955	2.423
3	TT	85°	73.362	2.284	52.727	2.394
4	SF	25°	72.896	2.508	52.835	2.408
5	SF	−40°	73.266	2.952	51.607	2.424
6	SF	85°	73.746	3.473	50.534	2.436
7	FS	25°	74.315	2.544	51.656	2.406
8	FS	−40°	74.506	2.996	50.316	2.422
9	FS	85°	74.156	2.296	52.215	2.392

Table 18
Numerical results of corners analysis for the best robust solution (Case 2).

Corner	Process	Temperature	Specification				
			Gain (dB) > 70	GBW (MHz) > 12	PM (°) > 55	OSP (V) > 0.5	OSN (V) > −0.5
1	TT	25°	71.398	15.402	83.221	0.695	−0.582
2	TT	−40°	72.086	17.941	83.448	0.726	−0.645
3	TT	85°	70.332	14.034	82.839	0.668	−0.522
4	SF	25°	71.013	15.258	83.444	0.697	−0.574
5	SF	−40°	71.801	17.752	83.668	0.727	−0.640
6	SF	85°	70.120	13.866	83.076	0.671	−0.514
7	FS	25°	71.461	15.566	83.004	0.693	−0.588
8	FS	−40°	72.288	18.114	83.224	0.723	−0.651
9	FS	85°	70.154	14.192	82.587	0.660	−0.530

true for FRIDGE [81] in Case 2 while GENOM [5] has employed a two-stage strategy: typical and corner optimization. Recall that the fourth column of Table 11 provides the optimization results of the typical stage. The results of the corners analysis of GENOM are also provided in Table 16. It is noteworthy here, GENOM has relaxed one of design specifications (i.e., DC gain > 67 dB instead of DC gain > 70 dB) in the corner optimization stage so as to meet the worst-case corner.

Table 16 shows the results of the proposed tool in the corners analysis (i.e., the best robust solution) in comparison with GENOM. Through the observation of this table the predominance of our proposed strategy in the corners analysis is obvious, especially when the computation time becomes of paramount importance. As can be observed, our proposed tool has attained better results with respect to the values of the power consumption and area. Furthermore, the total run time of our proposed tool is 382.503 s (i.e., 299.67 s for computation of the feasible solutions and 82.833 s for computation of the robust solutions) which is less than that of GENOM (i.e., 411.18 s). It should be emphasized that computation time of the feasible solutions highly depends on the simulation time of HSPICE while computation time of the robust solutions only depends on the number of the feasible solution.

Figs. 18 and 19 show the graphical results of the corners analysis for the best robust solution of the Case 1 and Case 2, respectively. Moreover, numerical results of the corners analysis for the best robust solution of the Case 1 and Case 2 are provided in Tables 17, 18, respectively. As can be seen, the obtained robust solution meets the required specifications related to DC gain, GBW, PM and Output Swing in all corners.

7. Conclusion

In this paper, we have proposed a novel automated design tool for analog ICs sizing and optimization. We have also devised a new technique named the *shrinking circles* to balance the exploration and exploitation capabilities of an optimization algorithm when it is converging to a possible optimum point. With this idea in mind, *Advanced GSA* was introduced as an upgraded version of GSA in which the shrinking circles technique is innovatively incorporated. Then, we hybridized Advanced GSA with PSO, denoted by *Advanced GSA_PSO*, in order to merge their capabilities and create a high-performance optimization algorithm for complex circuit's designs.

For verification purposes, the performance of Advanced GSA_PSO was tested over 23 benchmark functions and the results were compared with GSA, PSO, PSOGSA and Clustered-GSA algorithms. The proposed algorithm provided much better results than others due to its high ability in the exploration and exploitation of the search space.

Furthermore, to validate our proposed tool, two case studies were considered. In Case 1, the power consumption of a two-stage op-amp in 0.25 μm CMOS technology was minimized (a single objective problem), while in Case 2, the power consumption and circuit area of a folded-cascode op-amp in 0.18 μm CMOS technology were minimized simultaneously (a multi-objective problem). In comparison with available tools, the achieved results demonstrated that not only this tool met all design specifications, but also it could minimize the power consumption in Case 1 as well as the power consumption and circuit area in Case 2 in a significant manner. Then, a straightforward strategy with the aim of alleviating the computational burden of the corners analysis was employed to accelerate the robustness verification of the attained solutions against more realistic variations in the process. The final results indicate that our proposed tool can be considered as a promising tool in the analog IC sizing procedure because it is able to attain accurate and robust solutions at a reasonable computation time.

Acknowledgements

We would like to express our appreciation to Dr. Morteza

Shabanzadeh from Tarbiat Modares University and Dr. Bo Liu from Glyndwr University for their inspiring guidance and generous advice throughout our research. We would also like to thank three anonymous reviewers and the Associate Editor *Professor Nuno Horta* for their insightful comments and relevant observations.

References

[1] G.G. Gielen, CAD tools for embedded analogue circuits in mixed-signal integrated systems on chip, in: Proceedings of IEE Computers and Digital Techniques, IET, 2005, pp. 317–332.
[2] S.W. Director, R. Rohrer, Automated network design-the frequency-domain case, IEEE Trans. Circuit Theory 16 (1969) 330–337.
[3] E. Martens, G. Gielen, Classification of analog synthesis tools based on their architecture selection mechanisms, Integr. VLSI J. 41 (2008) 238–252.
[4] G.G. Gielen, R. Rutenbar, Computer-aided design of analog and mixed-signal integrated circuits, Proc. IEEE 88 (2000) 1825–1854.
[5] M. Barros, M.F. Barros, J. Guilherme, N. Horta, Analog Circuits and Systems Optimization Based on Evolutionary Computation Techniques, Springer, 2010.
[6] B. Liu, F.V. Fernández, G. Gielen, R. Castro-López, E. Roca, A memetic approach to the automatic design of high-performance analog integrated circuits, ACM Trans. Des. Autom. Electron. Syst. (TODAES) 14 (2009) 42.
[7] M.G. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B.L. Goffart, E. Vittoz, S. Cserveny, C. Meixenberger, G. Van Der Stappen, IDAC: an interactive design tool for analog CMOS circuits, IEEE J. Solid-State Circuits 22 (1987) 1106–1116.
[8] F. El-Turky, E.E. Perry, BLADES: an artificial intelligence approach to analog circuit design, Comput.-aided Des. Integr. Circuits Syst. IEEE Trans. 8 (1989) 680–692.
[9] R. Harjani, R.A. Rutenbar, L.R. Carley, OASYS: a framework for analog circuit synthesis, Comput.-aided Des. Integr. Circuits Syst. IEEE Trans. 8 (1989) 1247–1266.
[10] C. Leme, A. Yufera, L. Paris, N. Horta, A. Rueda, T. Oses, J. Franca, J. Huertas, Flexible silicon compilation of charge redistribution data conversion systems, in: Proceedings of the 34th Midwest Symposium on Circuits and Systems, IEEE, 1991, pp. 403–406.
[11] N. Horta, J. Franca, C. Leme, Framework for architecture synthesis of data conversion systems employing binary-weighted capacitor arrays, in: Proceedings of the IEEE International Sympoisum on Circuits and Systems, IEEE, 1991, pp. 1789–1792.
[12] B.A. Antao, A.J. Brodersen, ARCHGEN: automated synthesis of analog systems, IEEE Trans. Very Large Scale Integr. 3 (1995) 231–244.
[13] H.Y. Koh, C.H. Sequin, P.R. Gray, OPASYN: a compiler for CMOS operational amplifiers, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 9 (1990) 113–125.
[14] J.P. Harvey, M.I. Elmasry, B. Leung, STAIC: an interactive framework for synthesizing CMOS and BiCMOS analog circuits, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 11 (1992) 1402–1417.
[15] P.C. Maulik, L.R. Carley, Automating analog circuit design using constrained optimization techniques, in: 1991 IEEE International Conference on Computer-Aided Design, 1991. ICCAD-91. Digest of Technical Papers, IEEE, 1991, pp. 390–393.
[16] E.S. Ochotta, R.A. Rutenbar, L.R. Carley, Synthesis high-performance analog circuits ASTRX/OBLX, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 15 (1996) 273–294.
[17] G. Gielen, G. Debyser, K. Lampaert, F. Leyn, K. Swings, V. Plas, G. Der, W. Sansen, D. Leenaerts, P. Veselinovic, An analogue module generator for mixed analogue/digital ASIC design, Int. J. Circuit Theory Appl. 23 (1995) 269–283.
[18] B.A. Antao, G.G. Gielen, R.A. Rutenbar, Computer-aided Design of Analog Integrated Circuits and Systems, John Wiley & Sons, Inc, 2002.
[19] W. Nye, D.C. Riley, A. Sangiovanni-Vincentelli, A.L. Tits, DELIGHT. SPICE: an optimization-based system for the design of integrated circuits, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 7 (1988) 501–519.
[20] M. McKay, R. Beckman, W. Conover, A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, Technometrics 42 (2000) 55–61.
[21] A. Torralba, J. Chavez, L.G. Franquelo, FASY: a fuzzy-logic based tool analog synthesis, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 15 (1996) 705–715.
[22] R. Phelps, M. Krasnicki, R.A. Rutenbar, L.R. Carley, J.R. Hellums, Anaconda: simulation-based synthesis of analog circuits via stochastic pattern search, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 19 (2000) 703–717.
[23] M. Krasnicki, R. Phelps, R.A. Rutenbar, L.R. Carley, MAELSTROM: efficient simulation-based synthesis for custom analog cells, in: Proceedings of the 36th annual ACM/IEEE Design Automation Conference, ACM, 1999, pp. 945–950.
[24] W. Kruiskamp, D. Leenaerts, DARWIN: CMOS opamp synthesis by means of a genetic algorithm, in: Proceedings of the 32nd annual ACM/IEEE design automation conference, ACM, 1995, pp. 433–438.
[25] H. Liu, A. Singhee, R.A. Rutenbar, L.R. Carley, Remembrance of circuits past: macromodeling by data mining in large analog design spaces, in: Proceedings of the 39th, IEEE Design Automation Conference, 2002, pp. 437–442.
[26] G. Alpaydin, S. Balkir, G. Dundar, An evolutionary approach to automatic synthesis of high-performance analog integrated circuits, Evolut. Comput. IEEE Trans. 7 (2003) 240–252.
[27] F. De Bernardinis, M.I. Jordan, A. SangiovanniVincentelli, Support vector machines for analog circuit performance representation, in: Proceedings, IEEE Design Automation Conference, 2003, pp. 964–969.
[28] G.A. Wolfe, Performance macro-modeling techniques for fast analog circuit synthesis, in: Electrical and Computer Engineering and Computer Science, College of Engineering, University of Cincinnati, USA, 1999.
[29] A. Jafari, S. Sadri, M. Zekri, Design optimization of analog integrated circuits by using artificial neural networks, in: Soft Computing and Pattern Recognition (SoCPaR), 2010 International Conference of, IEEE, 2010, pp. 385–388.
[30] B. Liu, F. Fernández, G. Gielen, Automated design of analog and high frequency circuits, A computational intelligence approach [ISBN], Springer, Berlin, Heidelberg, 2014 [978-973].
[31] G. Rudolph, Convergence analysis of canonical genetic algorithms, Neural Netw. IEEE Trans. 5 (1994) 96–101.
[32] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[33] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, BGSA: binary gravitational search algorithm, Nat. Comput. 9 (2010) 727–745.
[34] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, Filter modeling using gravitational search algorithm, Eng. Appl. Artif. Intell. 24 (2011) 117–122.
[35] H. Hemmatian, A. Fereidoon, E. Assareh, Optimization of hybrid laminated composites using the multi-objective gravitational search algorithm (MOGSA), Eng. Optim. 46 (2014) 1169–1182.
[36] S. Sarafrazi, H. Nezamabadi-Pour, S. Saryazdi, Disruption: a new operator in gravitational search algorithm, Sci. Iran. 18 (2011) 539–548.
[37] X. Han, X. Chang, A chaotic digital secure communication based on a modified gravitational search algorithm filter, Inf. Sci. 208 (2012) 14–27.
[38] M. Doraghinejad, H. Nezamabadi-pour, A. Hashempour Sadeghian, M. Maghfoori, A hybrid algorithm based on gravitational search algorithm for unimodal optimization, in: Proceedings of the 2012 2nd International Conference on Computer Knowledge Engineering (ICCKE), IEEE (2012) 129–132.
[39] M. Shams, E. Rashedi, A. Hakimi, Clustered-gravitational search algorithm and its application in parameter optimization of a low noise amplifier, Appl. Math. Comput. 258 (2015) 436–453.
[40] R. Menozzi, A. Piazzi, F. Contini, Small-signal modeling for microwave FET linear circuits based on a genetic algorithm, Circuits System I: Fundamental Theory and Applications, IEEE Trans. 43 (1996) 839–847.
[41] C. Rajagopal, K. Sridhar, A. Nunez, RF CMOS circuit optimizing procedure and synthesis tool, in: Proceedings of the 13th ACM Great Lakes symposium on VLSI, ACM, 2003, pp. 100–103.
[42] B. Liu, Y. Wang, Z. Yu, L. Liu, M. Li, Z. Wang, J. Lu, F.V. Fernández, Analog circuit optimization system based on hybrid evolutionary algorithms, Integr. VLSI J. 42 (2009) 137–148.
[43] Y. Li, A simulation-based evolutionary approach to LNA circuit design optimization, Appl. Math. Comput. 209 (2009) 57–67.
[44] M. Barros, J. Guilherme, N. Horta, Analog circuits optimization based on evolutionary computation techniques, Integr. VLSI J. 43 (2010) 136–155.
[45] J.R. Koza, F.H. Bennett III, D. Andre, M.A. Keane, F. Dunlap, Automated synthesis of analog electrical circuits by means of genetic programming, Evolut. Comput. IEEE Trans. 1 (1997) 109–128.
[46] J.R. Koza, S.H. Al-Sakran, L.W. Jones, Cross-domain features of runs of genetic programming used to evolve designs for analog circuits, optical lens systems, controllers, antennas, mechanical systems, and quantum computing circuits, in: Proceedings of the Conference on Evolvable Hardware, 2005 NASA/DoD, IEEE, 2005, pp. 205–212.
[47] H. Ryose, K. Jin'no, H. Hirose, Automated synthesis of simple nonlinear analog circuits by means of genetic algorithm, J. Signal Process. 8 (2004) 529–535.
[48] F. Wang, Y. Li, L. Li, K. Li, Automated analog circuit design using two-layer genetic programming, Appl. Math. Comput. 185 (2007) 1087–1097.
[49] K. Zielinski, R. Laur, Constrained single-objective optimization using differential evolution, in: evolutionary computation, 2006. CEC 2006, IEEE Congr. (2006) 223–230.
[50] I.M. Kubař, Novel Optimization Tool for Analog Integrated Circuits Design, in: Department of Microelectronics, Faculty of Electrical Engineering Prague Czech Technical University, 2013.
[51] A. Kalinli, Optimal Circuit Design using Immune Algorithm, Artificial Immune Systems, Springer, 2004, pp. 42–52.
[52] H. Xu, Y. Ding, Optimizing method for analog circuit design using adaptive immune genetic algorithm, in: Frontier of Computer Science and Technology, 2009. FCST'09. Fourth International Conference on, IEEE, 2009, pp. 359–363.
[53] G.G. Gielen, H.C. Walscharts, W. Sansen, Analog circuit design optimization based on symbolic simulation and simulated annealing, Solid-State Circuits, IEEE, J. Of. 25 (1990) 707–713.
[54] J. Yuan, N. Farhat, J. Van der Spiegel, GBOPCAD: a synthesis tool for high-performance gain-boosted OPAMP design, Circuits and Systems I: regular Papers, IEEE Trans. on 52 (2005) 1535–1544.
[55] L.C. Severo, A. Girardi, An optimization-based tool for circuit level synthesis analog integrated circuits, in: I.X. Microelectronics (Ed.)Students Forum, Chip on the Dunes, Natal, 2009.
[56] J. Bland, G. Dawson, Tabu search and design optimization, Comput.-Aided Des. 23 (1991) 195–201.
[57] M.R. Tamplin, A. Hamilton, Ant Circuit World: an Ant Algorithm MATLAB™ Toolbox for the Design, Visualisation and Analysis of Analogue CircuitsEvolvable Systems: From Biology to Hardware, Springer, 2001, pp. 151–158.
[58] B. Benhala, A. Ahaitouf, A. Mechaqrane, B. Benlahbib, F. Abdi, E.-H. Abarkan, M. Fakhfakh, Sizing of current conveyors by means of an ant colony optimization

technique, in: Multimedia Computing and Systems (ICMCS), 2011 International Conference on, IEEE, 2011, pp. 1–6.
[59] B. Benhala, A. Ahaitouf, A. Mechaqrane, B. Benlahbib, Multi-objective optimization of second generation current conveyors by the ACO technique, in: Multimedia Computing and Systems (ICMCS), 2012 International Conference on, IEEE, 2012, pp. 1147–1151.
[60] J. Park, K. Choi, D.J. Allstot, Parasitic-aware RF circuit design and optimization, Circuits and Systems I: regular Papers, IEEE Trans. on 51 (2004) 1953–1966.
[61] S.K. Mandal, S. Sural, A. Patra, ANN-and PSO-based synthesis of on-chip spiral inductors for, RF ICs, Comput.-Aided Des. Integr. Circuits Syst., IEEE Trans. on 27 (2008) 188–192.
[62] M. Fakhfakh, S. Masmoudi, Y. Cooren, M. Loulou, P. Siarry, Improving switched current sigma delta modulators' performances via the particle swarm optimization technique, Int. J. Appl. Metaheuristic Comput. (IJAMC) 1 (2010) 18–33.
[63] A. Sallem, B. Benhala, M. Kotti, M. Fakhfakh, A. Ahaitouf, M. Loulou, Application of swarm intelligence techniques to the design of analog circuits: evaluation and comparison, Analog Integr. Circuits Signal Process. 75 (2013) 499–516.
[64] A. El Dor, M. Fakhfakh, P. Siarry, Performance optimization of CMOS second generation current conveyors using a multi-swarm algorithm, AEU-Int. J. Electron. Commun. 68 (2014) 496–503.
[65] M. Fakhfakh, Y. Cooren, A. Sallem, M. Loulou, P. Siarry, Analog circuit design optimization through the particle swarm optimization technique, Analog Integr. Circuits Signal Process. 63 (2010) 71–82.
[66] P.P. Kumar, K. Duraiswamy, An Optimized Device Sizing of Analog Circuits using Particle Swarm Optimization, J. Comput. Sci. 8 (2012).
[67] S. Mallick, R. Kar, D. Mandal, S. Ghoshal, Optimal sizing of CMOS analog circuits using gravitational search algorithm with particle swarm optimization, International, J. Mach. Learn. Cybern. (2015) 1–23.
[68] S. Mirjalili, S.Z.M. Hashim, A new hybrid PSOGSA algorithm for function optimization, in: Computer and information application (ICCIA), 2010 international conference on, IEEE, 2010, pp. 374–377.
[69] S. Mirjalili, S.Z.M. Hashim, H.M. Sardroudi, Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm, Appl. Math. Comput. 218 (2012) 11125–11137.
[70] S. Gao, C. Vairappan, Y. Wang, Q. Cao, Z. Tang, Gravitational search algorithm combined with chaos for unconstrained numerical optimization, Appl. Math. Comput. 231 (2014) 48–62.
[71] Design Solido Automation, ⟨http://www.solidodesign.com⟩.
[72] MunEDA, ⟨http://www.muneda.com⟩.
[73] AIDAsoft project, ⟨http://www.aidasoft.com⟩.
[74] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[75] X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, Evolut. Comput., IEEE Trans. on 3 (1999) 82–102.
[76] E.S. Ochotta, R.A. Rutenbar, L.R. Carley, Synthesis High-Performance Analog Circuits ASTRX/OBLX, Comput.-Aided Des. Integr. Circuits Syst. IEEE Trans. 15 (1996) 273–294.
[77] P.E. Allen, D.R. Holberg, CMOS Analog Circuit Design, Oxford Univ. Press, 2002.
[78] A. Santiago, H.J.F. Huacuja, B. Dorronsoro, J.E. Pecero, C.G. Santillan, J.J.G. Barbosa, J.C.S. Monterrubio, A survey of decomposition methods for multi-objective optimizationRecent Advances on Hybrid Approaches for Designing Intelligent Systems, Springer, 2014, pp. 453–465.
[79] G. Mavrotas, Effective implementation of the ε-constraint method in multi-objective mathematical programming problems, Appl. Math. Comput. 213 (2009) 455–465.
[80] F. Medeiro, F.V. Fernández, R. Domínguez-Castro, and A. Rodríguez-Vázquez, A statistical optimization-based approach for automated sizing of analog cells, in Proceedings of the 1994 IEEE/ACM international conference on Computer-aided design 1994, pp. 594–597.
[81] E. Tlelo-Cuautle, A. Sanabria-Borbon, Optimising operational amplifiers by evolutionary algorithms and gm/Id method, Int. J. Electron. (2016) 1–20.

Maryam Dehbashian received the B.S. and M.S. degrees from Shariaty Technical College, Tehran, Iran and University of Birjand, Birjand, Iran in 2006 and 2011, respectively, all in electronics engineering. She is currently pursuing the Ph.D. degree at Ferdowsi University of Mashhad, Mashhad, Iran. Her current research interests include analog IC design automation, swarm intelligence and evolutionary computation techniques as well as soft computing.

Mohammad Maymandi-Nejad (M'02) received the B.S. degree from the Ferdowsi University of Mashhad, Mashhad, Iran, in 1990, and the M.S. degree from the Khajeh Nassir Tossi University of Technology, Tehran, Iran, in 1993, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2005, all in electronics engineering. He is currently an Associate Professor with the Department of Electrical Engineering at Ferdowsi University of Mashhad. His current research interests include low-voltage, low-power analog ICs, and their applications in biomedical circuits and systems.