J Glob Optim
DOI 10.1007/s10898-008-9307-9

DESA: a new hybrid global optimization method and its application to analog integrated circuit sizing

Jernej Olenšek · Árpád Bűrmen · Janez Puhan · Tadej Tuma

Received: 1 June 2007 / Accepted: 4 April 2008

Abstract This paper presents a new hybrid global optimization method referred to as DESA. The algorithm exploits random sampling and the metropolis criterion from simulated annealing to perform global search. The population of points and efficient search strategy of differential evolution are used to speed up the convergence. The algorithm is easy to implement and has only a few parameters. The theoretical global convergence is established for the hybrid method. Numerical experiments on 23 mathematical test functions have shown promising results. The method was also integrated into SPICE OPUS circuit simulator to evaluate its practical applicability in the area of analog integrated circuit sizing. Comparison was made with basic simulated annealing, differential evolution, and a multistart version of the constrained simplex method. The latter was already a part of SPICE OPUS and produced good results in past research.

Keywords Optimization · Simulated annealing · Differential evolution · Analog integrated circuit sizing

1 Introduction

Global optimization has received a lot of attention in the recent years due to the fact that many real-world problems can be treated as global optimization problems of the following form:

J. Olenšek (✉) · Á. Bűrmen · J. Puhan · T. Tuma
Faculty of electrical engineering, University of Ljubljana, Tržaška 25, 1000 Ljubljana, Slovenija
e-mail: jernej.olensek@fe.uni-lj.si

Á. Bűrmen
e-mail: arpadb@fides.uni-lj.si

J. Puhan
e-mail: janez.puhan@fe.uni-lj.si

T. Tuma
e-mail: tadej.tuma@fe.uni-lj.si

$$
\begin{aligned}
x^* &= \arg\min_{x \in S} f(x) \\
f &: S \to \mathbb{R} \\
S &= \left\{ x, x \in \mathbb{R}^N, l(i) \leq x(i) \leq u(i), i = 1, \ldots, N \right\}
\end{aligned} \tag{1}
$$

where $f(x)$ is the so called cost function (CF), x is a N-dimensional vector of optimization variables, and $l(i)$ and $u(i)$ are the lower and upper bounds for the ith variable, respectively.

Unfortunately problem (1) usually cannot be solved analytically. Many different classes of optimization methods have been developed to solve the problem numerically. Gradient methods are the fastest, but they require the information on the derivatives of the CF and they work only on differentiable functions. This reduces their suitability for many practical applications. The alternative to gradient methods are the direct search methods [1–3]. They do not require gradients of the CF and can handle noisy and multimodal functions. Another possible way is to classify optimization methods as local or global. The former are designed to find the minimum very fast, even though it is not the true global minimum. The latter are usually slower but can find the global minimum with high probability. There are also many different hybrid methods that exploit the fast convergence of the local methods and good global search capabilities of the global methods [4–7].

In this paper we present a new optimization method referred to as DESA. The method exploits the key features of the simulated annealing algorithm (SA) [8], namely the random sampling and the so called metropolis criterion. In order to improve the convergence rate, elements from differential evolution (DE) [9] are incorporated. DESA works with a population of points and also exploits the efficient search mechanism of DE. Since the method uses random sampling and the metropolis criterion when accepting trial points, we can expect slower convergence rates when compared with basic DE, that uses a greedy selection strategy. But on the other hand it allows us to theoretically analyze the global convergence properties of DESA and also allows the method to work with a small population. DE relies on a very large population to perform global search and lacks the global convergence property since it has no means of escaping from suboptimal solutions after all population members have converged to it. In comparison with basic SA we expect considerable improvement in terms of the convergence speed and the final solution quality since inefficient pure random sampling of SA is combined with the DE operator.

We tested the performance of DESA on a set of 23 mathematical test functions and also in the area of device sizing in analog integrated circuit (IC) design. Most modern ICs consist of both digital signal processing components and analog interfaces to the outside world. Many efficient cell-based tools exist to design the digital portion of the system, but the design of analog components is more difficult. The process usually consists of an optimization tool that selects trial circuits and a circuit simulator that evaluates its performance. The main issue in this process is the fact that the evaluation of a single circuit usually takes a considerable amount of time. In the past several ways to tackle this problem have been developed. In [10] an extensive survey of analog synthesis strategies is presented. In [11] for example equation based subcircuits were composed to simplify and speed up the evaluation of the circuit performance. In [12] the circuit equations were also modified during the synthesis process. While equation based synthesis provides the means for a very fast circuit evaluation, it is very difficult to obtain the equations. The accuracy of such evaluators is also limited and the equations must be constructed for every manufacturing technology separately. In [13] the authors use the polynomial interpolation method to allow efficient and fast repetitive evaluations of the circuits. They theoretically analyze the method and propose modifications to improve its accuracy and efficiency.

In order to automatically obtain the reduced order symbolic model for the linear transfer functions of the circuit, symbolic analysis was used in [14–18], but such models cannot be applied to highly nonlinear characteristics of the circuit. In [10,19] the authors used two different modules to design analog circuits. The first one compiles the SPICE netlist describing the circuit and design specifications into a C-style cost function and the second module then uses simulated annealing to find its minimum. In [20] the authors use a combination of a special DC simulator and a set of equations either provided by the user or obtained through SPICE simulations and neural network training. They use a hybrid between evolution strategies and simulated annealing to solve their optimization problem.

In [21] a more extensive survey of the automated device sizing methods is presented, while in [22,23] the authors present an overview of recent design methodologies for the design of large systems on chip (SoC). The methods involve the extraction of nonlinear macromodels of the circuits which are then used to evaluate circuit performance.

In this paper we use an approach similar to [24]. The authors in [24] developed a new optimization procedure as a combination of a population based approach and pattern search. But unlike the approaches described above they use full device models circuit evaluation. This has two main consequences. First the use of full device models ensures the maximal accuracy for the evaluation of circuit's performance. On the other hand this approach takes much more time than the simulation of simplified reduced order models, which means it can only be used for medium sized circuits.

In our experiments we used the SPICE OPUS circuit simulator [25] to evaluate the circuit's performance. Unlike most programs based on the original SPICE code from Berkley, SPICE OPUS includes an internal optimizer. This reduces the overhead present when the simulator is completely separated from the optimization engine and needs to be invoked for every CF evaluation separately.

Device sizing in analog ICs is inherently a multiobjective optimization problem, since the designer is usually interested in several circuit properties (e.g. gain, bandwidth, power consumption, area, etc.). One way to deal with multiple objectives is to combine all objectives into a scalar real-valued function. This approach is implemented in SPICE OPUS. First a separate penalty function is constructed for every design goal, and the final CF is obtained as a weighted sum of these penalty functions. Varying environmental conditions (e.g. temperature, model parameters, etc.) are considered by simulating the circuit across several corners, where every corner represents a different set of values for environmental parameters. The evaluation of circuit properties is conducted for every corner and the worst values are used in the construction of the corresponding penalty function and the calculation of the final CF value [26].

We should mention that in our experiments we ignored some very important aspects of analog IC design, such as worst case and mismatch issues, yield estimation, and layout issues, etc [27,28]. These can be included in the existing framework as separate analyses and measurements but are not the subject of this paper. Our main goal is to find methods that are capable of finding the global minimum of complex cost functions, and hopefully have a theoretical background to backup its practical performance.

This paper is organized as follows. In Sects. 2 and 3 the basic SA and DE algorithms are described. Section 4 gives a detailed description of the DESA algorithm. In Sect. 5 the theoretical global convergence of the algorithm is established. Section 6 contains the experimental setup and the optimization results for the mathematical test functions and real-world IC design problems. Section 7 contains the concluding remarks.

Notation M, N, $U[0,1]$, $x(i)$, and $\mu(A)$ denote the population size (the number of samplers), the dimensionality of the problem, an uniformly distributed random variable

from the [0,1] interval, the ith component of vector x, and the Lebesgue measure of set A, respectively. Superscripts denote different points in the search space and indices denote different iterations.

2 Simulated annealing

Simulated annealing (SA) is a stochastic global optimization algorithm that performs random sampling of the search space [8]. Its key feature is the mechanism controlling the transition from the current point (x) to a new point (x^n), generated by a random perturbation (δx) of point x. The transition mechanism is known as the Metropolis criterion and is defined as

$$P = \min(1, e^{-\frac{f(x^n)-f(x)}{T}}) \tag{2}$$

where P is the probability of making the transition from x to x^n (i.e. x^n replaces x). $f(x^n)$ and $f(x)$ are the CF values at x^n and x, respectively. Parameter T is referred to as the temperature. This mechanism always accepts downhill moves (i.e. if $f(x^n) < f(x)$), but it also allows uphill transitions with positive probability. This probability is controlled by the current value of the temperature parameter (T). Initially T is set to a very high value which means that most transitions (including the uphill) are accepted. During the annealing process T is reduced according to the specified cooling schedule. Therefore the probability of making an uphill transition becomes smaller. At the end of the optimization T has a very low value and the algorithm behaves almost like a descent method. The annealing mechanism allows the algorithm to escape from local minima when T is high and fine-tune the solution when T is low.

One of the attractive features of the SA algorithm is the fact that its theoretical convergence to a global minimum can be guaranteed under certain assumptions. This requires a very careful selection of the cooling schedule and the mechanism for generating δx. In [29] for instance there are several convergence results for various SA algorithms. SA algorithms that can guarantee convergence often do not perform well in practice because they require a very slow cooling schedule and thus have very slow convergence. Pure random sampling is also a drawback of SA since it does not use any knowledge gathered during the search. In order to speed up the process, modifications to the sampling mechanism and the cooling schedule are often used. This means that the convergence results no longer apply but the algorithm may still perform reasonably well. Selection of SA parameters is however a very delicate process. It depends on the optimization problem at hand and can greatly influence the success rates of SA.

3 Differential evolution

Differential evolution (DE) is a parallel direct search method that uses a population of M points to search for a global minimum of a function over a continuous search space [9]. In every generation the entire population is used to generate new trial points. For every point (target point x^{it}) in the current population a so called mutated point (x^m) is generated by adding a weighted difference of two randomly selected points (x^{ic1} and x^{ic2}) from the current population to the third point (x^{ic3}). We denote the weight factor by w and is given by the user. Usually the value of w is in the (0,1] interval. Then crossover between the current target point and the mutated point is applied to generate a trial point (x^g). Several crossover schemes

have been reported in the literature. Binomial crossover where the crossover is applied independently to every variable with probability P_c, is often used. If the CF value at x^g is lower than at x^{it}, x^g replaces x^{it} in the next generation. The process is repeated until the maximal number of generations is reached.

4 DESA algorithm

With DESA we hope to improve the search efficiency and convergence speed of SA by incorporating some features of DE. The method uses a population of M samplers to guide the search process. Along with the random sampling the DE operator is used to consider the knowledge already accumulated by the population. Instead of the greedy selection strategy of DE the original metropolis criterion is used to allow the acceptance of non-improving solutions into the population. However since the annealing schedule is one of the most problematic aspects of SA, we use a somewhat different approach. Instead of having a single sampler and decreasing the temperature with time we have multiple samplers operating at different but constant temperatures. Temperature changes are achieved by exchanging the points between different samplers. There have already been several attempts to use such an approach to avoid the difficulties of selecting the appropriate cooling schedule [30,31]. Like the temperature, we assign to every sampler different but fixed crossover probability and an additional parameter called the sampling radius, which controls the length of random steps. So the ith sampler g^i (where $i = 1, 2, \ldots, M$) can be fully defined by the following features:

1. Temperature T^i, which is used in the Metropolis criterion
2. Radius R^i, which is used for random step generation
3. Crossover probability P_c^i, which is used in DE operator
4. A point x^i in the search space S

Since different optimization parameters in practical applications can have values that can differ by several orders of magnitude, we normalize all optimization variables to the [0,1] interval. The pseudo code of the DESA algorithm is given by Algorithm 1. A detailed description of the method is given in the following subsections.

Algorithm 1 DESA algorithm

Require: $M, T^M, R^M, P_c^1, P_c^M, D_{stop}$
1. Initialize population {generate M points}
2. Initialize method {set T^i, R^i, P_c^i, $i = 1, 2, \ldots, M$}
3. $k = 0$ {iteration counter}
4. **repeat**
5. $k = k + 1$
6. trial point generation
7. replacement {Metropolis criterion}
8. acceleration
9. transition between samplers
10. **until** termination conditions are met

4.1 User defined input parameters

The method uses some parameters that must be set by the user. They are the number of samplers $M \geq 4$ (population size), minimal temperature $T^M > 0$ (temperature for the last

sampler), minimal random sampling radius $R^M > 0$ (radius parameter for the last sampler), crossover probabilities for the first and the last sampler $P_c^1, P_c^M \in [0, 1]$, and the stopping distance $D_{stop} > 0$. Default values for these parameters are $M = 20$, $T^M = 10^{-6}$, $R^M = 10^{-6}$, $P_c^1 = 0.1$, $P_c^M = 0.5$ and $D_{stop} = 10^{-4}$.

4.2 Initialization of population

The initial population can be generated randomly but in our method we use the Latin hypercube approach that allows more thorough exploration of the search space. Every optimization variable interval is first divided into M equal subintervals. Then M points are randomly generated so that every subinterval for every optimization variable is included in the initial population. This is very important in algorithms that use crossover operators. The values of parameters inside subintervals are chosen randomly.

4.3 Initialization of method parameters

At the beginning of the optimization run some additional method parameters must be set. These parameters are the temperature, the sampling radius parameter and the crossover probability for every sampler. All of them are initialized in the same way. We use the values of the parameters for the first and the last sampler and an exponential function to calculate the values for the remaining samplers.

$$\begin{aligned} c_t &= \frac{1}{M-1} \cdot \log\left(\frac{T^1}{T^M}\right) \\ T^i &= T^1 \cdot e^{-c_t \cdot (i-1)}, \quad i = 1, 2, \ldots, M \end{aligned} \tag{3}$$

T^1 is the maximum temperature and is set to the CF difference between the worst and the best point in the initial population. The same procedure is then repeated for the sampling radius parameter R^i with $R^1 = 1$, and for the crossover probabilities P_c^i.

$$\begin{aligned} c_r &= \frac{1}{M-1} \cdot \log\left(\frac{R^1}{R^M}\right) \\ R^i &= R^1 \cdot e^{-c_r \cdot (i-1)}, \quad i = 1, 2, \ldots, M \end{aligned} \tag{4}$$

$$\begin{aligned} c_p &= \frac{1}{M-1} \cdot \log\left(\frac{P_c^M}{P_c^1}\right) \\ P_c^i &= P_c^1 \cdot e^{c_p \cdot (i-1)}, \quad i = 1, 2, \ldots, M \end{aligned} \tag{5}$$

4.4 Trial point generation

In every iteration a single point is selected for improvement. We select the worst point in the current population but any point that is not the best point can be selected here. We denote the sampler that holds this target point with the superscript it. A new trial point is generated using a combination of an operator similar to the original DE operator and a random move. The procedure is given by Algorithm 2.

x^m is the mutated point, x^g is the generated trial point, and r is a random step generated according to the Cauchy probability distribution. Since the sampling radius parameter R^i has different values for different samplers, the mechanism acts almost as the original DE operator when R^{it} is small and becomes very much like random search operator when R^{it} is

Algorithm 2 Trial point generation

1. select target point (denoted by x^{it})
2. select randomly $ic1, ic2, ic3 \in \{1, 2, \ldots, M\}$ where $ic1 \neq ic2 \neq ic3 \neq it$
3. $w = U[0, 1] \cdot 2$
4. $x^m = x^{ic1} + (x^{ic2} - x^{ic3}) \cdot w$
5. **for** $i = 1, 2, \ldots, N$ **do**
6. $r = R^{it} \cdot tan(\pi \cdot (U[0, 1] - 0.5))$
7. **if** $U[0, 1] < P_c^{it}$ **then**
8. $x^g(i) = x^m(i) + r$
9. **else**
10. $x^g(i) = x^{it}(i) + r$
11. **end if**
12. **if** $x^g(i) > 1$ (upper bound constraint violated) **then**
13. $x^g(i) = x^{it}(i) + (1 - x^{it}(i)) \cdot U[0, 1]$
14. **else if** $x^g(i) < 0$ (lower bound constraint violated) **then**
15. $x^g(i) = x^{it}(i) + (0 - x^{it}(i)) \cdot U[0, 1]$
16. **end if**
17. **end for**

large. Different samplers can be initialized with different crossover probabilities (P_c^i) by the user (via the input parameters P_c^1 and P_c^M) to fine tune the trial point generation mechanism. Large values of P_c^{it} mean that along with the random step r a DE operator is used for many variables which speeds up the convergence while low values of P_c^{it} emphasize random search with only an occasional use of DE operator.

4.5 Replacement

In this phase of the algorithm the generated trial point x^g is submitted to the Metropolis criterion (2) with temperature T^{it}. If the criterion is satisfied, x^g replaces x^{it} in the next iteration. Better points are always accepted. If the trial point x^g is worse than the current target point, the transition depends on the sampler that holds the target point. If the target point x^{it} is located at the sampler with high temperature, x^g will have a high probability of being accepted. If the target point is located at a sampler with low temperature, this probability will be low. With this mechanism the chances for the algorithm to escape from a local minimum are increased.

4.6 Acceleration

The method can use many different mechanisms to speed up the convergence. In our case we used a very simple procedure. Every time a new best point is found, we apply this mechanism. We construct a quadratic model function based on three collinear points in the search space. The first point is a randomly selected point from the current population. Points with higher CF values have higher probability of being selected. The probability of selecting the point il is given by (6).

$$P(il) = \frac{\frac{f(x^{il}) - f(x^{best})}{f(x^{worst}) - f(x^{best})}}{\sum_{j=1}^{M} \frac{f(x^j) - f(x^{best})}{f(x^{worst}) - f(x^{best})}} \tag{6}$$

x^{best} and x^{worst} denote the points from the current population with the lowest and highest CF values, respectively.

The second point is the centroid of the population points (c) and is calculated according to (7).

$$c = \frac{1}{M} \sum_{i=1}^{M} x^i \tag{7}$$

These two points define a search direction $d = c - x^{il}$. The third point p^3 is obtained by making a random move from x^{il} in the direction d. If the obtained quadratic model function is not convex, the best of these three points is returned. For the convex case the minimum of the model function is returned. If the minimum of the model function violates box constraints, it is first contracted towards x^{il} until the violation is removed. The returned point replaces x^{il} if it has lower CF value. The procedure is given by Algorithm 3.

Algorithm 3 Acceleration mechanism

1. select a point from the population {denoted by x^{il}}
2. calculate the centroid $c = \frac{1}{M} \sum_{i=1}^{M} x^i$
3. set $p^1 = x^{il}$, $p^2 = c$
4. set $w = 1 + U[0, 1]$
5. set $p^3 = p^1 + (p^2 - p^1) \cdot w$
6. **if** p^3 violates box constraints **then**
7. **repeat**
8. $p^3 = 1/2 \cdot (p^3 + p^1)$
9. **until** p^3 satisfies box constraints
10. **end if**
11. construct quadratic model function through p^1, p^2, p^3
12. **if** model function convex **then**
13. find minimum of the model function p^m
14. **if** p^m violates box constraints **then**
15. **repeat**
16. $p^m = 1/2 \cdot (p^m + p^1)$
17. **until** p^m satisfies box constraints
18. **end if**
19. **else**
20. set p^m as the best among p^1, p^2, p^3
21. **end if**
22. **if** $f(p^m) < f(x^{il})$ **then**
23. replace x^{il} with p^m
24. **end if**

4.7 Transition between samplers

One of the main problems of the original SA algorithm is the selection of the appropriate cooling schedule. This means the selection of the initial temperature, the number of steps at every temperature stage and the temperature reduction mechanism. If the cooling is too fast the algorithm can get trapped in a local minimum and if the cooling is too slow the optimization takes too long to be of any use for practical purposes. In DESA the cooling schedule is not needed because temperature changes are achieved by simply exchanging points between samplers which operate at different but fixed temperatures. After every trial point generation, replacement, and acceleration phase we randomly select a sampler g^{is} from the population. Then samplers g^{it} and g^{is} exchange their points in search space with probability given by (8).

$$P = \min\left(1, e^{-(\frac{1}{T^{is}} - \frac{1}{T^{it}})\cdot(f(x^{is}) - f(x^{it}))}\right) \tag{8}$$

This mechanism is quite different from the original idea of SA. Here the idea is to always send bad solutions to samplers with low T and R where the trial point will be generated with the DE operator, random step will be small and most uphill transitions will be rejected. But occasionally we also allow the transition of bad points to samplers with high T and R where random steps are large and uphill transitions are accepted with higher probability. So if the algorithm cannot find good solutions, the target point will eventually end up at samplers with low T and R, where trial points are generated in a way similar to DE. When good solution is found, the next target point is likely to be at a sampler with higher T and R so the algorithm will run at least for a while like a random search allowing longer jumps through the search space and making uphill transitions with higher probability. If an acceptable solution is not found the target point will eventually end up at samplers with small T and R and the whole process is repeated. This scheme also performs a kind of reannealing and further improves the chances of escaping from a local minimum.

4.8 Termination criteria

Several termination criteria can be used in our method. In practice the time available for the optimization is always limited so the maximal number of function evaluations is a logical choice for termination. The maximal distance between points in the population and the current best point is also used in the termination condition. When this distance falls below a user-defined stopping distance D_{stop} the algorithm is terminated. The third termination criterion is the CF value difference between the best and the worst point in the current population. When this difference becomes smaller than the user-defined minimal temperature (T^M) the algorithm is terminated. In analog IC design with SPICE OPUS all the design goals are satisfied when the CF value reaches zero (see [26]). This can also be used in the termination condition.

5 Convergence

In this section we establish the convergence in probability for the proposed DESA method. We prove that if the number of iterations is large enough, DESA converges to the set of ϵ-optimal solutions. The proof is based on the fact that a global optimization algorithm must have a positive probability of reaching an arbitrary subset of points in the search space in a finite number of iterations. The acceleration mechanism described in the previous section affects a single point in the population and does not affect the convergence of the algorithm, therefore it is excluded from the theoretical analysis.

5.1 Notation

We define the set of global optimal solutions S^* and the set of ϵ-optimal solutions S^*_ϵ as:

$$\begin{aligned} S^* &= \{x^*, x^* \in S,\ f(x^*) = f^*\} \\ S^*_\epsilon &= \{x, x \in S,\ f(x) \leq f^* + \epsilon\} \end{aligned} \tag{9}$$

where f^* is the global minimum of the CF in S.

The algorithm runs as a stochastic process which depends on a population of points in the search space. We define the state space of this process as

$$\mathbb{X} = \left\{ \mathbf{x}, \mathbf{x} = (x^1, \ldots, x^M), x^i \in S, i = 1, 2, \ldots, M \right\} \tag{10}$$

where $\mathbf{x}$ denotes a state of the process (a population of points). We also define a random variable X_k which corresponds to the population of points at the kth iteration. Since the population at every iteration depends only on the population at the previous iteration, the stochastic process can be modeled by a time homogeneous Markov chain, for which we define a transition probability $p(\mathbf{x}, A)$ (the probability of making a transition from state $\mathbf{x}$ to a state with at least one point in set A):

$$p(\mathbf{x}, A) = P(\nu_A(X_{k+1}) \neq 0 | X_k = \mathbf{x}) \tag{11}$$

where $\nu_A(\mathbf{x})$ denotes the number of points from state $\mathbf{x}$ that reside in set A.

We also define

$$p_{min}(A) = \inf_{\mathbf{x} \in \mathbb{X}} (p(\mathbf{x}, A)) \tag{12}$$

and a set of states (populations) which are not ϵ-optimal:

$$C_\epsilon = \left\{ \mathbf{x} \in \mathbb{X}, \nu_{S_\epsilon^*}(\mathbf{x}) = 0 \right\} \tag{13}$$

We must also define the concept of convergence. The method is said to converge to the set of ϵ-optimal solutions if the following condition holds:

$$\exists \epsilon > 0, K(\epsilon) \geq 0 \text{ such that } \nu_{S_\epsilon^*}(X_k) \neq 0 \quad \forall k > K(\epsilon) \tag{14}$$

5.2 Assumptions

For DESA we make the following assumptions:

(1) The search space is bounded, i.e.

$$-\infty < l(i) \leq x(i) \leq u(i) < \infty, \quad i = 1, 2, \ldots, N \tag{15}$$

which means that after normalization of variables all points in the population will belong to the set $S = [0, 1]^N$.

(2) The CF is bounded, i.e.

$$-\infty < f(x) < \infty, \quad \forall x \in S \tag{16}$$

which ensures that the maximum temperature T_{max} (calculated in the method initialization step) is always bounded.

(3) S_ϵ^* must have a positive Lebesgue measure i.e.

$$\mu(S_\epsilon^*) > 0, \quad \forall \epsilon > 0 \tag{17}$$

5.3 Convergence proof

For the convergence analysis we need two lemmas.

Lemma 1 *The following condition holds:*

$$\text{if } k \geq 0 \quad \text{and} \quad \nu_{S_\epsilon^*}(X_k) \neq 0 \text{ then } P(\nu_{S_\epsilon^*}(X_{k+1}) \neq 0 | X_k) = 1 \tag{18}$$

This means that when the process reaches an ϵ-optimal state, the subsequent states will also be ϵ-optimal.

Proof The method ensures that the best point in the current population is never selected as a target point for replacement. Hence if the population at iteration k is ϵ-optimal (i.e. $\nu_{S_\epsilon^*}(X_k) \neq 0$), then the current best point resides in S_ϵ^*. This point can only be selected for replacement if another sampler has found a new point that has even lower CF value. Therefore this new point also resides in S_ϵ^*. So

$$\nu_{S_\epsilon^*}(X_{k+1}) \neq 0 \tag{19}$$

which completes the proof. □

With (12) we define the minimal probability of making a transition form an arbitrary state $\mathbf{x} \in \mathbb{X}$ to a state with at least one point residing in set A.

Lemma 2 *Suppose that for any $A \subset S$ with positive Lebesgue measure (i.e. $\mu(A) > 0$) the assumptions from the subsection 5.2 hold. Then the following condition also holds:*

$$p_{min}(A) = \beta^* > 0 \tag{20}$$

Proof First we divide the transition into two steps. First is the generation of the trial point x^g inside the set A, given the current target point x^{it}. We denote this probability with

$$P_G(x) = P(x^g \in A | x^{it} = x) \tag{21}$$

The second step is the replacement of the current target point x^{it} with x^g. In this step we use the Metropolis acceptance criterion and we denote this probability with

$$P_M(x^{it}, x^g) = e^{-\frac{f(x^g) - f(x^{it})}{T^{it}}} \tag{22}$$

We now compute the lower bound for $P_G(x)$. Since the variables are perturbed independently, it is sufficient to consider only the one dimensional case. We denote the current point, the mutated point and the generated trial point (now in one dimension) with x^t, x^m and y, respectively. Due to assumption 1, normalization of variables, and using the mechanism described by Algorithm 2 the following conditions hold

$$\begin{aligned} 0 \leq x^t \leq 1 \\ -w \leq x^m \leq (1 + w) \end{aligned} \tag{23}$$

In one dimension the set A becomes a subinterval of [0,1] i.e.

$$A = \{x, x \in \mathbb{R}, 0 \leq a_L \leq x \leq a_H \leq 1\} \tag{24}$$

A trial point y is generated by adding a random step r to either x^t or x^m, depending on the crossover probability P_c. The random step is generated according to the Cauchy probability distribution which is defined by

$$p(r) = \frac{R}{\pi(R^2 + r^2)} \tag{25}$$

where R is the random sampling radius parameter for the sampler that holds the current target point x^t.

We now compute two probabilities. The first one (P_m) is the probability of generating a random step r that will move the mutated point x^m into A and the second one (P_t) is the probability of generating a random step r that will move the target point x^t into A.

P_m can be expressed as:

$$P_m = P_m(y \in A) \geq \int_{a_L}^{a_H} \frac{R}{\pi(R^2 + (y - x^m)^2)} dy \tag{26}$$

The inequality comes from the fact that some points can end up violating box constraints after the random step. The mechanism described by lines 12–16 of the Algorithm 2 removes the violations and increases the probability of making a transition into set A.

Since the generated trial point y always obeys box constraints (i.e. $0 \leq y \leq 1$) and due to (23), the following always holds

$$(y - x^m)^2 \leq (1 + w)^2 \tag{27}$$

Thus (26) becomes

$$P_m \geq \frac{R}{\pi(R^2 + (1 + w)^2)} \int_{a_L}^{a_H} dy = \frac{R}{\pi(R^2 + (1 + w)^2)} \cdot (a_H - a_L) \tag{28}$$

The positive Lebesgue measure of A in one dimension means that

$$a_H - a_L > 0 \tag{29}$$

and since the sampling radius parameters and weight factor in the trial point generation mechanism are bounded and non zero we can write

$$\beta_m = \min(P_m) > 0 \tag{30}$$

Similarly we can calculate the second probability (P_t) which is the probability of generating a random step r that will move the current target point x^t into set A.

$$P_t = P_t(y \in A) \geq \int_{a_L}^{a_H} \frac{R}{\pi(R^2 + (y - x^t)^2)} dy \tag{31}$$

The inequality here comes from the same fact as before. Box constraints violations are handled by lines 12–16 of the Algorithm 2, which increases the probability of making a transition into set A.

Following the same procedure as before we get

$$(y - x^t)^2 \leq 1 \tag{32}$$

$$P_t \geq \frac{R}{\pi(R^2 + 1)} \int_{a_L}^{a_H} dy = \frac{R}{\pi(R^2 + 1)} \cdot (a_H - a_L) \tag{33}$$

Using the same reasoning as before we can write

$$\beta_t = \min(P_t) > 0 \tag{34}$$

We now use (30) and (34) to compute

$$\begin{aligned} P(y \in A) &= P_c \cdot P_m + (1 - P_c) \cdot P_t \\ &\geq P_c \cdot \beta_m + (1 - P_c) \cdot \beta_t = \beta > 0 \end{aligned} \tag{35}$$

where P_c is the crossover probability.

Since all variables are perturbed independently we can now compute the lower bound for $P_G(x)$

$$\beta_G = \min_{x \in S}(P_G(x)) = \beta^N > 0 \tag{36}$$

Next we need to compute the lower bound for replacement probability $P_M(x^{it}, x^g)$. Due to assumption 2 and non zero user defined minimum temperature T_{min} we can write

$$\begin{aligned} F_M &= \max_{x,y \in S} |f(x) - f(y)| < \infty \\ 0 &\leq |f(x^g) - f(x^{it})| \leq F_M \\ T_{min} &> 0 \\ \beta_M &= \min_{x,y \in S}(P_M(x, y)) = e^{-\frac{F_M}{T_{min}}} > 0 \end{aligned} \tag{37}$$

Now we can express

$$p_{min}(A) = \min(P_G(x)) \cdot \min(P_M(x^{it}, x^g)) = \beta_G \cdot \beta_M = \beta^* > 0 \tag{38}$$

which completes the proof of the lemma. □

Theorem 1 *If the assumptions from the subsection 5.2 hold, the algorithm converges to S_ϵ^* with probability one, i.e.*

$$\lim_{k \to \infty} (P(\nu_{S_\epsilon^*}(X_k) \neq 0) = 1 \tag{39}$$

Proof Since the process is a time homogeneous Markov chain and due to Lemma 1 we can write

$$\begin{aligned} P(\nu_{S_\epsilon^*}(X_k) = 0) &= P(\nu_{S_\epsilon^*}(X_1) = 0, \nu_{S_\epsilon^*}(X_2) = 0, \ldots \nu_{S_\epsilon^*}(X_k) = 0) \\ &= P(\nu_{S_\epsilon^*}(X_1) = 0) \prod_{j=2}^{k} P(\nu_{S_\epsilon^*}(X_j) = 0 | \nu_{S_\epsilon^*}(X_{j-1}) = 0) \end{aligned} \tag{40}$$

We assume that the initial population is not optimal, i.e.

$$P(\nu_{S_\epsilon^*}(X_1) = 0) = 1 \tag{41}$$

We now use (13) to compute

$$\begin{aligned} P(\nu_{S_\epsilon^*}(X_j) = 0 | \nu_{S_\epsilon^*}(X_{j-1}) = 0) &= \frac{P(\nu_{S_\epsilon^*}(X_j) = 0, \nu_{S_\epsilon^*}(X_{j-1}) = 0)}{P(\nu_{S_\epsilon^*}(X_{j-1}) = 0)} \\ &= \frac{\int_{C_\epsilon} P(\nu_{S_\epsilon^*}(X_j) = 0 | X_{j-1} = \mathbf{x}) P(X_{j-1} = \mathbf{x}) \mu(d\mathbf{x})}{\int_{C_\epsilon} P(X_{j-1} = \mathbf{x}) \mu(d\mathbf{x})} \end{aligned} \tag{42}$$

Basically (42) states the probability of making a transition from non-optimal state (population) to non-optimal state. Now using (12) and Lemma 2 let

$$p^* = p_{min}(S_\epsilon^*) \geq \beta^* > 0 \tag{43}$$

and thus

$$P(\nu_{S_\epsilon^*}(X_j) = 0 | X_{j-1} = \mathbf{x}) \leq (1 - p^*), \quad \forall \mathbf{x} \in C_\epsilon \tag{44}$$

We insert (44) into (42) and the result is

$$P(\nu_{S_\epsilon^*}(X_j) = 0 | \nu_{S_\epsilon^*}(X_{j-1}) = 0) \leq \frac{(1-p^*)\int_{C_\epsilon} P(X_{j-1} = \mathbf{x})\mu(d\mathbf{x})}{\int_{C_\epsilon} P(X_{j-1} = \mathbf{x})\mu(d\mathbf{x})} = (1-p^*) \quad (45)$$

Inserting (45) and (41) into (40) yields

$$P(\nu_{S_\epsilon^*}(X_k) = 0) \leq (1-p^*)^{k-1} \quad (46)$$

Now we can compute

$$\lim_{k\to\infty} (P(\nu_{S_\epsilon^*}(X_k) = 0) \leq \lim_{k\to\infty} (1-p^*)^{k-1} = 0 \quad (47)$$

Finally it follows

$$\lim_{k\to\infty} (P(\nu_{S_\epsilon^*}(X_k) \neq 0) = 1 - \lim_{k\to\infty} (P(\nu_{S_\epsilon^*}(X_k) = 0) = 1 \quad (48)$$

which completes the proof of the theorem. □

6 Testing the algorithm's performance

DESA was tested on 23 mathematical test functions and on seven real-world cases of analog IC design. Comparison was made with basic SA and DE, and also with the method that produced good results for IC design problems in SPICE OPUS. The latter is a multistart version of the simplex method referred to as the constrained simplex method (COMPLEX). The original COMPLEX method [1] has some global search capabilities but it is still local in nature and the final result depends greatly on the choice of the initial point. Therefore the multistart concept was implemented in SPICE OPUS to improve the method's global search capabilities [32]. The concept proved to be fairly successful in IC design [26,32,33] but it is sometimes slow and unreliable. So our goal is to find methods that are capable of producing better and more consistent results.

DESA, DE and SA were implemented in C language and integrated into SPICE OPUS. All methods were tested using the same algorithm parameter values for all test cases. The population size for DE was set to 100, crossover probability was 0.9, and weight factor 0.5. SA parameters were set in the following way. First 50 random points were evaluated and the initial temperature was set to accept the largest uphill jump with probability 0.9. The best of these 50 points was selected as the starting point for the algorithm. Temperature and random sampling radius were annealed with factor 0.99, and the number of steps in every temperature stage was set to twice the number of optimization variables. The trial points were generated according to the Cauchy probability distribution which allows longer jumps through the search space than the original Gaussian distribution. These settings do not exactly meet the requirements for guaranteed global convergence but were selected to get at least some results with the given number of CF evaluations. SA was terminated when the temperature reached 10^{-6}.

6.1 Optimization of mathematical functions

A total of 23 mathematical test functions were used with dimensionality ranging from 2 to 30. The definitions of the test functions can be found in [34]. For every function 30 independent

optimization runs were conducted with different seeds for the random number generator. The results are given in Table 1.

The table contains the average of the final CF values over 30 runs (mean CF) and the average number of CF evaluations (CFE) after which the methods were terminated (mean CFE). For the multistart COMPLEX method the (rounded) average of finished restarts is also given along with the number of CFE. The CFE limit was set to 100,000 for $f_1 - f_{13}$, 20,000 for f_{14}, $f_{16} - f_{19}$, 30,000 for f_{15}, $f_{21} - f_{23}$ and 40,000 for f_{20}. The multistart COMPLEX method was allowed to finish the last restart even if the CFE limit was reached.

We can see that SA achieved the worst performance on all functions so we will not include SA in our discussion. $f_1 - f_5$ are unimodal. For these functions DE and the multistart COMPLEX are expected to outperform DESA, however except for f_3, where DESA was the worst, DESA produced results that were better or comparable with the multistart COMPLEX, but was outperformed by DE. For discontinuous f_6 DE was the most successful. DESA was prematurely terminated in one run but multistart COMPLEX failed in all 30 runs. On the noisy f_7 the methods achieved similar performance.

$f_8 - f_{13}$ are high dimensional functions with a large number of local minima. These functions represent the most difficult class of optimization problems and are the main focus of our work. It appears that for such functions DE is not the best choice. DESA produced better results than the multistart COMPLEX for all functions except f_9, and was better than DE on three problems. For the remaining three problems it ended up in the vicinity of the global minimum, while DE produced the best results for these functions. DE was the worst method on f_9 and failed badly on f_8 and f_{11}.

The remaining functions are low dimensional functions with only a few local minima. DESA worked well for most of these functions. For f_{15} the minimal random steps prohibited DESA to fine-tune the solution and on f_{21}, and f_{22} it was terminated before it could reach the global minimum in three runs and one run, respectively. DE missed the global minimum twice for f_{20}. The multistart COMPLEX found the solutions in the vicinity of the true global minima for these functions but was unable to fine-tune them better than DE for f_{15}, f_{18}, and $f_{21} - f_{23}$.

We can say that SA is the least successful method while DE produced the best average results for this set of functions. The results also show that while DE has very good fine-tunning abilities, it is not the best choice for high dimensional functions with many local minima. For these functions DESA appears to be better suited since it was able to find a better solution on three functions and ended up in the close proximity of the minimum for the other three. Multistart COMLPEX method was the best only for f_9.

We should mention that a very strict limit in terms of maximal number of CF evaluations was imposed in our experiments on mathematical functions. We wanted to test how well different methods perform with the given budget of CF evaluations. When imposing different termination criteria (e.g. CFE limit, distance between solutions, etc), there is always the risk of missing the true global minimum of the CF. Many methods have been developed that can guarantee theoretical global convergence (e.g. [29,35,36]) but they usually assume that they are allowed to run forever, which is not possible in practice. For DESA we also established the global convergence but due to fairly strict termination conditions, the method missed the global minimum or was unable to fine-tune the solution in some optimization runs.

6.2 Optimization of integrated circuits

The performance of the described methods was tested on seven real-world cases of analog IC design. We will describe in detail only the first case (damp1) which is a differential amplifier circuit. The circuit topology is depicted in Fig. 1.

Table 1 Optimization results for mathematical test functions

	Multistart COMPLEX		DESA		DE		SA	
	Mean CF	Mean CFE /restarts	Mean CF	Mean CFE	Mean CF	Mean CFE	Mean CF	Mean CFE
f_1	$9.1042 \cdot 10^{-02}$	102,608/21	$2.6273 \cdot 10^{-04}$	74,587	$4.2527 \cdot 10^{-08}$	100,000	$7.5072 \cdot 10^{-01}$	100,000
f_2	$5.0819 \cdot 10^{-01}$	103,133/17	$1.0812 \cdot 10^{-02}$	100,001	$2.6623 \cdot 10^{-04}$	100,000	$2.6075 \cdot 10^{+05}$	100,000
f_3	7.9217	110,425/7	$8.0047 \cdot 10^{+03}$	100,001	$3.1111 \cdot 10^{+01}$	100,000	$3.2012 \cdot 10^{+04}$	100,000
f_4	3.2181	102,339/22	4.4407	100,001	$3.9301 \cdot 10^{-01}$	100,000	$1.1039 \cdot 10^{-04}$	100,000
f_5	$2.6262 \cdot 10^{+01}$	103,488/15	$2.6844 \cdot 10^{+01}$	100,001	$2.2072 \cdot 10^{+01}$	100,000	$7.9585 \cdot 10^{+04}$	100,000
f_6	5.0000	101,291/32	$3.3333 \cdot 10^{-02}$	100,001	0	100,000	4.8667	100,000
f_7	$1.7628 \cdot 10^{-02}$	101,238/47	$3.7535 \cdot 10^{-02}$	100,001	$1.2941 \cdot 10^{-2}$	100,000	$8.7052 \cdot 10^{-01}$	100,000
f_8	$-8.7872 \cdot 10^{+03}$	104,604/10	$-1.2558 \cdot 10^{+04}$	76,556	$-7.3130 \cdot 10^{+03}$	100,000	$-1.1896 \cdot 10^{+04}$	100,000
f_9	$1.2761 \cdot 10^{+01}$	103,217/16	$1.3044 \cdot 10^{+02}$	100,001	$1.8613 \cdot 10^{+02}$	100,000	$1.1112 \cdot 10^{+01}$	100,000
f_{10}	$9.4101 \cdot 10^{-01}$	102,138/23	$3.3060 \cdot 10^{-03}$	71,610	$6.3523 \cdot 10^{-05}$	100,000	$4.5263 \cdot 10^{-04}$	96,558
f_{11}	$2.2638 \cdot 10^{-01}$	102,203/22	$9.6958 \cdot 10^{-03}$	91,008	$9.9427 \cdot 10^{-01}$	100,000	$9.8152 \cdot 10^{-01}$	36,338
f_{12}	$1.4471 \cdot 10^{-02}$	106,962/9	$1.0265 \cdot 10^{-04}$	100,001	$4.0636 \cdot 10^{-09}$	100,000	$5.3172 \cdot 10^{+04}$	100,000
f_{13}	$3.2698 \cdot 10^{-02}$	102,221/26	$2.1767 \cdot 10^{-04}$	100,001	$2.5585 \cdot 10^{-08}$	100,000	$2.0167 \cdot 10^{+05}$	100,000
f_{14}	$9.9800 \cdot 10^{-01}$	20,036/237	$9.9800 \cdot 10^{-01}$	2,423	$9.9800 \cdot 10^{-01}$	20,000	2.3099	8,944
f_{15}	$3.0806 \cdot 10^{-04}$	30,203/82	$3.0839 \cdot 10^{-04}$	30,001	$3.07486 \cdot 10^{-04}$	30,000	$5.2450 \cdot 10^{-03}$	24,523
f_{16}	-1.0316	20,042/248	-1.0316	3,037	-1.03163	20,000	-1.0071	9,867
f_{17}	$3.9789 \cdot 10^{-01}$	20,042/252	$3.9789 \cdot 10^{-01}$	3,055	$3.9789 \cdot 10^{-01}$	20,000	$4.0068 \cdot 10^{-01}$	8,262
f_{18}	3.0001	20,041/245	3.0000	1,830	3	20,000	4.2923	11,861
f_{19}	-3.8628	20,072/101	-3.8628	4,337	-3.86278	20,000	-3.8613	10,450

Table 1 continued

	Multistart COMPLEX		DESA		DE		SA	
	Mean CF	Mean CFE /restarts	Mean CF	Mean CFE	Mean CF	Mean CFE	Mean CF	Mean CFE
f_{20}	-3.3220	40,243/66	-3.3220	11,610	-3.3141	40,000	-3.2544	20,149
f_{21}	$-1.0151 \cdot 10^{+01}$	30,142/118	-9.4868	5,685	$-1.0153 \cdot 10^{+01}$	30,000	-6.2774	12,507
f_{22}	$-1.0401 \cdot 10^{+01}$	30,122/118	-9.7351	5,085	$-1.0403 \cdot 10^{+01}$	30,000	-6.8488	12,678
f_{23}	$-1.0535 \cdot 10^{+01}$	30,124/118	$-1.0313 \cdot 10^{+01}$	5,134	$-1.0536 \cdot 10^{+01}$	30,000	-6.0873	12,854

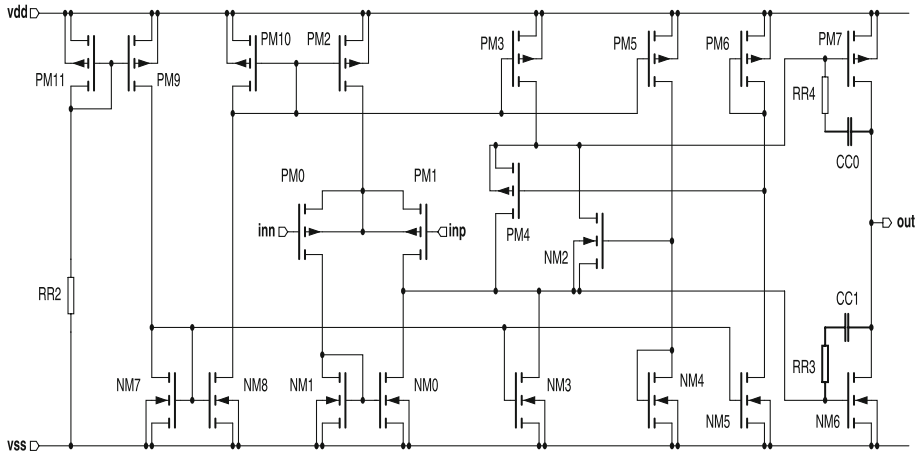

Fig. 1 Topology of the damp1 case

There are 27 optimization variables:

- 3 resistors → 3 optimization variables
- 2 capacitors → 2 optimization variables
- Transistors NM0 and NM1 should be identical → 2 optimization variables (width and length)
- Transistors NM3, NM5, NM7, and NM8 should be identical → 2 optimization variables (width and length)
- Transistors PM0 and PM1 should be identical → 2 optimization variables (width and length)
- Transistors PM2, PM3, PM5, and PM10 should be identical → 2 optimization variables (width and length)
- Transistors PM9 and PM11 should be identical → 2 optimization variables (width and length)
- Transistors NM2, NM4, NM6, PM4, PM6, and PM7 → $6 \cdot 2 = 12$ optimization variables (widths and lengths)

In this case we do not optimize transistor multipliers.

The properties which we are interested in are: circuit area, current consumption, AC gain, unity gain bandwidth, bandwidth, phase margin, gain margin, maximal derivative of gain magnitude, output voltage swing, DC gain, settling time, overshoot, slew rate, rise time, and fall time.

In order to measure these properties we need to perform the following analyses: an operating point analysis, a DC analysis, an AC analysis, and a transient analysis.

We only consider a single corner point with typical model parameters and ambient temperature of 27°C.

The remaining cases will be described more briefly. The second case (damp1-5c) optimizes the same circuit as the first case, with the same optimization parameters and design goals, but considers five different corner points to account for varying environmental conditions.

The third case (lfbuffer) is a circuit with 36 optimization parameters (32 transistors, 1 capacitor and 1 resistor). It requires an OP, an AC, a DC, and a transient analysis to measure 13 circuit properties chosen as design goals. The case considers a single corner point.

The fourth case (lfbuffer-5c) is the same as the third one but considers five different corner points.

The next case (nand) is a simple NAND gate element with only three optimization parameters (four transistors) and considers three corners. The case requires an OP analysis and two transient analyses to measure nine circuit properties.

The delay case (delay) has 12 optimization parameters (six transistors) and a single corner point. It requires an OP analysis and a transient analysis to obtain six considered circuit properties.

The last case (damp2) is another amplifier circuit with 15 optimization parameters (nine transistors, one capacitor and one resistor) and 14 corner points. We perform an OP, a DC, two AC, a transient, and a noise analysis to measure 13 circuit properties.

All considered test cases were optimized using the same algorithm parameter values as for the optimization of mathematical functions. Since the optimization is very time consuming, every circuit was optimized only once.

Optimization results are given in Table 2. For every case the number of design variables (VARS), the number of design goals (GOALS), and the number of corner points (CORNERS) is given. The table shows the number of CFE needed to reach a solution with the given CF value, the CF value of the final solution, the number of CFE needed to find the final solution, and the number of CFE when the methods were terminated. For the multistart COMPLEX method the number of restarts needed to perform the given number of CFE is also given (restarts). Circuits for which the final CF value is zero have satisfied all the design goals and the optimization was stopped at that time even though the termination criteria or the CFE limit were not reached yet. Table 3 shows the detailed results for the damp1 case.

We can see from Table 2 that SA exhibits the worst performance as was the case with mathematical functions. Since we did not experiment to determine the appropriate values for SA parameters, the performance is quite bad as expected. Multistart COMPLEX method performed reasonably well but despite several restarts it was unable to find better solutions than DE or DESA. So the main comparison will be between DE and DESA. We can safely state that DE and DESA both found the global minimum for damp1, lfbuff, and delay case, since we know that zero is the minimal possible CF value in our experiments. For the remaining cases however we can make no such claims. Sometimes the design goals can be too ambitious and cannot be achieved with the given topology or parameter bounds. In such cases we have no knowledge of the minimal CF value nor of the location of the global minimum. This is in fact a major problem in most practical applications. Despite this we can still conclude a few things about DE and DESA. One can see that DESA exhibits fast initial progress. For the first five cases DESA approached solutions with low CF values considerably faster than DE. In the later stages the fine tunning abilities of DE began to stand out. For all cases 100 population members of DE contained enough information to fine tune the solution better and faster than the simple acceleration mechanism of DESA. For more complex cases however (damp1-5c and lfbuff-5c) DE was able to find a slightly better solution then DESA but it also required more CFE. Random sampling allows DESA to work with a small population and still maintain adequate diversity of solutions to successfully find global minimum of difficult problems. But it also affects the convergence speed and fine tunning abilities of the method. This problem however is much easier to solve than improving the methods global search capabilities. DESA appears to have good global search capabilities but the replacement of the current acceleration procedure with a more sophisticated local search could greatly improve the convergence speed and make the method even more competitive.

Table 2 IC optimization results

Case		DESA	Multistart COMPLEX	DE	SA
damp1	CFE for CF < 1	5,843	5,208 (2)	11,185	–
	CFE for CF < 0.1	7,818	–	11,185	–
VARS $= 27$	Final CF	0	0.087	0	1.697
GOALS $= 15$	CFE for minimal CF	63,863	68,471 (15)	38,922	30,028
CORNERS $= 1$	Final CFE	63,863	100,000 (24)	38,922	100,000
damp1-5c	CFE for CF < 10	2,490	1,356 (1)	3,549	–
	CFE for CF < 5	3,598	21,281 (5)	8,980	–
VARS $= 27$	Final CF	1.806	3.425	1.741	14.448
GOALS $= 15$	CFE for minimal CF	250,021	231,312 (55)	294,646	30,304
CORNERS $= 5$	Final CFE	300,000	300,000 (73)	300,000	300,000
lfbuffer	CFE for CF < 10	1,781	414 (1)	4,797	–
	CFE for CF < 1	9,523	1,339 (1)	25,652	–
VARS $= 36$	Final CF	0	0.512	0	12.909
GOALS $= 13$	CFE for minimal CF	79,377	3,310 (1)	43,284	51,597
CORNERS $= 1$	Final CFE	79,377	100,000 (24)	43,284	100,000
lfbuffer-5c	CFE for CF < 10	1,941	989 (1)	5,584	–
	CFE for CF < 5	5,852	13,523 (4)	18,877	–
VARS $= 36$	Final CF	2.330	4.313	1.950	18.294
GOALS $= 13$	CFE for minimal CF	229,658	157,612 (41)	289,889	16,155
CORNERS $= 5$	Final CFE	300,000	300,000 (78)	300,000	300,000
nand	CFE for CF < 500	282	40 (1)	485	2,037
	CFE for CF < 200	507	94 (1)	1,023	4,798
VARS $= 3$	Final CF	166.641	166.686	166.640	166.693
GOALS $= 9$	CFE for minimal CF	2,208	5,818 (47)	9,564	7,543
CORNERS $= 3$	Final CFE	2,986	10,000 (81)	10,000	10,000
delay	CFE for CF $< 20 \cdot 10^3$	38,698	660 (1)	8,063	–
	CFE for CF $< 10 \cdot 10^3$	84,302	21,101 (28)	20,832	–
VARS $= 12$	Final CF	0	6,183.500	0	59,217.8
GOALS $= 6$	CFE for minimal CF	183,687	114,794 (143)	33,141	9,562
CORNERS $= 1$	Final CFE	183,687	200,000 (250)	33,141	200,000
damp2	CFE for CF < 20	989	420 (1)	1,261	–
	CFE for CF < 10	11,282	3,926 (3)	10,095	–
VARS $= 15$	Final CF	5.926	7.487	5.926	21.394
GOALS $= 13$	CFE for minimal CF	251,244	326,011 (231)	98,190	7,282
CORNERS $= 14$	Final CFE	365,343	500,000 (352)	500,000	500,000

To demonstrate the difficulties of IC optimization we also calculated the profile of the CF for the damp1 case. We performed a sweep through all optimization parameters starting from some initial solution and from the final solution found by DESA. Since there are 27 optimization variables the resulting profile is difficult to plot. Therefore Fig. 2 shows the CF profile for a subset of the optimization parameters. The curves intersect at a point with x-axis value zero representing the final parameter values. Eventhough the figure does not

Table 3 Results for the damp1 case

Measurement	Goal	DESA	Multistart COMPLEX	DE	SA
Circuit area	$<10^{-8}\,\mathrm{m}^2$	$8 \cdot 10^{-9}\,\mathrm{m}^2$	$5.95 \cdot 10^{-9}\,\mathrm{m}^2$	$8.01 \cdot 10^{-9}\,\mathrm{m}^2$	$7.64 \cdot 10^{-9}\,\mathrm{m}^2$
Current consumption	<1 mA	$437 \cdot 10^{-6}$ A	$530 \cdot 10^{-6}$ A	$420 \cdot 10^{-6}$ A	$607 \cdot 10^{-6}$ A
AC gain	>70 dB	70.4 dB	70 dB	71.2 dB	92.5 dB
Unity gain bandwidth	>5 MHz	17.2 MHz	15.7 MHz	9.25 MHz	20.1 MHz
Bandwidth	>500 Hz	1.38 kHz	2.23 kHz	1.37 kHz	0.169 kHz
Phase margin	$>60°$	65.9°	90.3°	96.6°	56.4°
Gain margin	$>10°$	33.4°	15.6°	51.7°	13.2°
Max.derivative of gain magnitude	<0	$-104 \cdot 10^{-9}$	$-39.9 \cdot 10^{-9}$	$-105 \cdot 10^{-9}$	$6.95 \cdot 10^{-6}$
Output voltage swing	>1.6 V	1.6 V	1.57 V	1.6 V	1.37 V
DC gain	>60 dB	69.3 dB	66.7 dB	69.9 dB	86.9 dB
Settling time	<300 ns	174 ns	168 ns	168 ns	75.6 ns
Overshoot	$<1\%$	$696 \cdot 10^{-3}\%$	$885 \cdot 10^{-3}\%$	$45.5 \cdot 10^{-3}\%$	$3.94 \cdot 10^{-3}\%$
Slew rate	$>5 \cdot 10^6$ V/s	$7.44 \cdot 10^6$ V/s	$7.48 \cdot 10^6$ V/s	$6.68 \cdot 10^6$ V/s	$14.7 \cdot 10^6$ V/s
Rise time	<200 ns	64.4 ns	64.1 ns	71.8 ns	32.8 ns
Fall time	<200 ns	57.5 ns	81.1 ns	187 ns	53.1 ns

show the profile for all 27 variables, it clearly shows why IC optimization is so difficult. The CF is highly nonlinear and contains several local minima. The sensitivity of the CF to different parameters varies considerably. Noise is also clearly visible. All these facts make fast gradient descent methods inefficient and the entire optimization task extremely difficult. When there are several corner points to consider, the task becomes even more challenging. The quality of the final solution (point with x-axis value zero) can also be confirmed from the profile of the CF, since the CF value increases or remains zero when the parameters are varied.

We should mention that a very large number of CFE was allowed in our experiments. This was done for comparison purposes only. In practice the optimization time is always limited so there would be a very strict limit on the maximal number of CFE. In our experiment we have seen that DESA exhibits faster initial progress but DE can fine tune the solution faster and better. This means that DESA could be further improved by using a fast local search method in the second stage of the optimization run.

7 Conclusions

Design of integrated circuits is a very challenging task. Once the appropriate circuit topology has been selected, the parameter values for every electronic component must be determined in order to obtain a circuit that meets the design specification. The task can be formulated as an optimization problem and solved with the help of computer software. A very large number

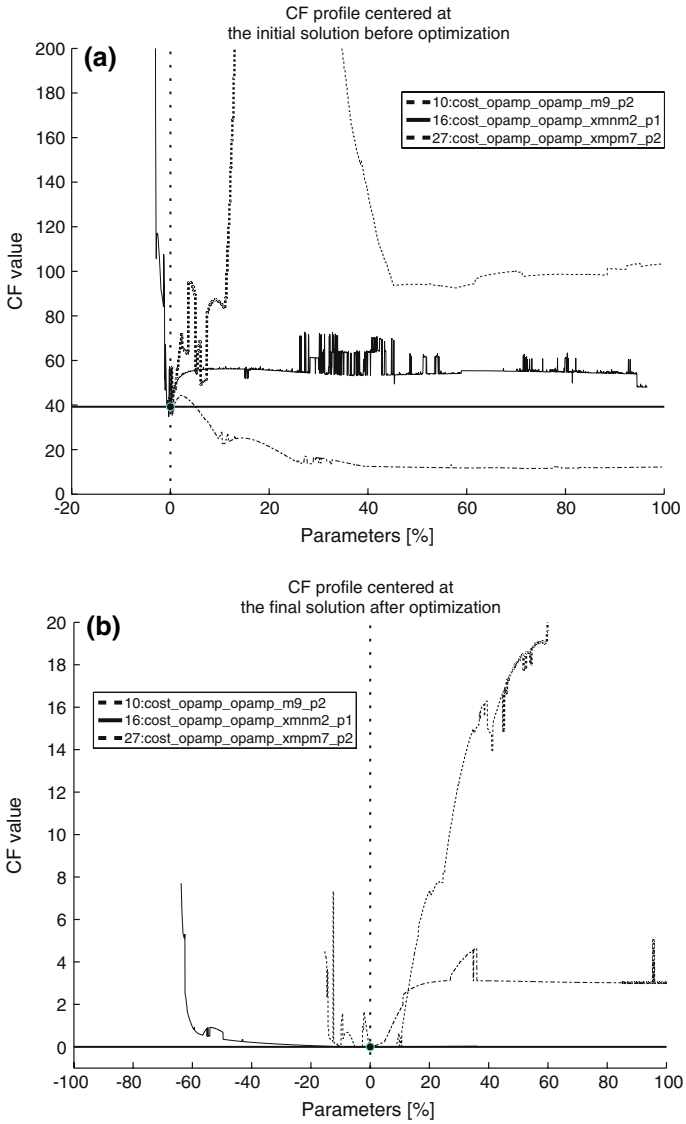

Fig. 2 Cost profile of case damp1—(**a**) at the initial point and (**b**) at the final solution found by DESA. Every curve represents a sweep of one of the optimization parameters

of different optimization methods have been developed in the past. Unfortunately many of them are not suitable for IC optimization due to a very complex structure of the CF. Due to numerical noise and several local optima the fastest gradient methods are not the best choice. Global direct search methods must be used that can handle noisy functions and are capable of locating the global minimum with high probability.

DE and SA are two very popular optimization methods. SA has good global search capabilities and can easily escape from low quality local minima, but it has very slow convergence. DE on the other hand has fast convergence but no means of escaping from local minima once

the population members get too close to each other. Large populations are the only chance for DE to find the global minimum. DESA is a combination of SA and DE and is expected to be able to escape from local minima due to random sampling and the metropolis criterion. DE operator on the other hand allows the use of knowledge from the entire population for trial point generation which in turn speeds up the convergence. Since basic DE does not use random sampling and uses a greedy selection strategy it is expected to run faster than DESA, but SA features allow DESA to guarantee global convergence under mild assumptions and still run much faster than SA.

We tested the performance of DESA, DE, SA and multistart COMPLEX method on 23 mathematical test functions, with strict limits on the maximal number of CFE. Simple SA produced the worst results for all functions. Pure random sampling is too inefficient and the selection of the appropriate method parameters depends greatly on the given problem. The multistart COMPLEX method performed fairly well on several unimodal and low dimensional functions but for high dimensional cases with many local minima the lack of global search capabilities becomes apparent. It produced the best result for only one of these difficult problems. DE exhibited efficient performance on unimodal and low dimensional functions. It produced the best results for most of these cases. However when optimizing high dimensional functions with many local minima it became clear that DE depends on a large population to maintain diversity of the solutions. Such a large population requires much more time to converge to a single solution, especially when local minima are spread across the entire search area. Except for only a few of these functions DE performed better than the multistart COMPLEX method. The proposed hybrid method (DESA) produced fairly good results for unimodal and low dimensional functions. For some of the functions the random steps reduced its fine-tunning abilities, and its strict termination conditions caused premature termination in a few runs for some functions. On high dimensional functions with many local minima good global search capabilities of DESA became apparent. On three out of six such difficult functions DESA produced much better results that DE and the multistart COMPLEX. For the other three DESA was unable to fine tune the solutions to the same extent as DE and ended up in the close proximity of the global minimum.

We also tested the methods on seven real-world cases of analog IC design. The results again confirmed that basic SA is too inefficient and that without parameter tunning it is very difficult to predict its performance for a specific problem. Our experiments have shown that the multistart COMPLEX method achieves fast progress towards the local minima but it requires too many restarts before the method finds local minima with low CF values. DE proved to be a very good method for IC optimization. While its initial progress was rather slow it exhibited very good fine tunning capabilities. DESA also performed well for IC optimization. While random steps allow DESA to work with a small population they also affect its local search capabilities and convergence speed. Incorporation of a more sophisticated local search strategy could greatly improve the performance of DESA and is a good direction for future work.

We should mention that in our experiments we ignored some very important aspects of analog IC design, such as mismatch and yield estimation, layout issues, etc. They are not the main purpose of this paper but can be incorporated in the existing framework. Our goal is to find methods that are capable of finding the minimum of complex CFs. The main purpose of this paper was to introduce a new hybrid global optimization method. The experiments shoved that while global convergence was established for DESA, it comes at a price of reduced convergence speed and weaker local search capabilities. These drawbacks however can be addressed by implementing a local search procedure that is more sophisticated and efficient than the one currently used.

Acknowledgements The research has been supported by the Ministry of Higher Education, Science and Technology of the Republic of Slovenia within programme P2-0246—Algorithms and optimization methods in telecommunications.

References

1. Box, M.J.: A new method of constrained optimization and a comparison with other methods. Comput. J. **8**, 42–52 (1965)
2. Powell, M.J.: Direct search algorithms for optimization calculations. Acta Numer. **7**, 287–336 (1998)
3. Torczon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**(1), 1–25 (1997)
4. Preux, P., Talbi, E.G.: Towards hybrid evolutionary algorithms. Int. Trans. Oper. Res. **6**, 557–570 (1999)
5. Garcia-Palomares, U.M., Gonzales-Castaño, F.J., Burguillo-Rial, J.C.: A combined global & local search (CGLS) approach to global optimization. J. Glob. Optim. **34**, 409–426 (2006)
6. Schmidt, H., Thierauf, G.: A combined heuristic optimization technique. Adv. Eng. Software **36**, 11–19 (2005)
7. Ho, S.L., Yang, S., Ni, G., Mochado, J.M.: A modified ant colony optimization algorithm medeled on tabu-search methods. IEEE Trans. Magn. **42**(4), 1195–1198 (2006)
8. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**, 1277–1292 (1983)
9. Storn, R., Price, K.: Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**, 341–359 (1997)
10. Ochotta, E., Mukherjee, T., Rutenbar, R., Carley, L.: Practical Synthesis of High-Performance Analog Circuits. Kluwer Academic, Norwell, MA (1998)
11. Harlani, R., Rutenbar, R., Carley, L.: OASYS: a framework for analog circuit synthesis. IEEE Trans. Comp. Aided Des. **8**(12), 1247–1266 (1989)
12. Gielen, G. et al.: Analog circuit design optimization based on symbolic simulation and simulated annealing. IEEE J. Solid-State Circuits **25**, 707–713 (1990)
13. Fernández, F.V., Guerra, O., Rodríguez-García, J.D., Rodríguez-Vázquez, A.: Symbolic analysis of large analog integrated circuits: the numerical reference generation problem. IEEE Trans. Circuits Systems II: Analog Digital Signal Process. **45**(10), 1351–1361 (1998)
14. Gielen, G., Wambacq, P., Sansen, W.: Symbolic analysis methods and applications for analog circuits: A tutorial overview. Proc. IEEE **82**, 287–304 (1990)
15. Shi, C.J., Tan, X.: Symbolic analysis of large analog circuits with determinant decision diagrams. In: Proceedings, ACM/IEEE ICCAD, pp. 366–373 (1997)
16. Yu, Q., Sechen, C.: A unified approach to the approximate symbolic analysis of large analog integrated circuits. IEEE Trans. Circuits Syst. **43**, 656–669 (1996)
17. Daems, W., Gielen, G., Sansen, W.: Circuit complexity reduction for symbolic analysis of analog integrated circuits. In: Proceedings, ACM/IEEE, Design Automation Conf., 958–963, Jun. 1999
18. Wambacq, P., Vanthienen, J., Gielen, G., Sansen, W.: A design tool for weakly nonlinear analog integrated circuits with multiple inputs (mixers, multipliers). In: Proceedings, IEEE CICC, San Diego, CA, pp. 5.1.1–5.1.4, May 1991
19. Ochotta, E., Rutenbar, R., Carley, L.: Synthesis of high-performance analog circuits and ASTRX/OBLX. IEEE Trans. Comput. Aided Des. **15**, 237–294 (1996)
20. Alpaydın, G., Balkır, S., Dündar, G.: An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. IEEE Trans. Evol. Comput. **7**(3), 240–252 (2003)
21. Gielen, G.G.E., Rutenbar, R.A.: Computer-aided design of analog and mixed-signal integrated circuits. Proc. IEEE **88**(12), 1825–1849 (2000)
22. Gielen, G.G.E.: CAD tools for embedded analogue circuits in mixedsignal integrated systems on chip. Proc. IEEE Comput. Digit. Tech. **152**(3), 317–332 (2005)
23. Rutenbar, R.A., Gielen, G.G.E., Roychowdhury, J.: Hierarchical modeling, optimization, and synthesis for system-level analog and RF designs. Proc. IEEE **95**(3), 640–669 (2007)
24. Phelps, R., Krasnicki, M., Rutenbar, R., Carley, R., Hellums, J.: Simulation-based synthesis of analog circuits via stochastic pattern search. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **19**(6), 703–717 (2000)
25. SPICE OPUS circuit simulator homepage: http://www.fe.uni-lj.si/spice/. Accessed 1 Dec. 2005 (2005)
26. Bűrmen, Á., Strle, D., Bratkovič, F., Puhan, J., Fajfar, I., Tuma, T.: Automated robust design and optimization of integrated circuits by means of penalty functions. AEU-Int. J. Electron. C. **57**(1), 47–56 (2003)

27. Antreich, K.J., Graeb, H.E., Wieser, C.U.: Circuit analysis and optimization driven by worst-case distances. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **13**(1), 703–717 (1994)
28. Schwencker, R., Schenkel, F., Pronath, M., Graeb, H.: Analog circuit sizing using adaptive worst-case parameter sets. In: Proceedings, Design, Automation and Test (2002)
29. Yang, R.L.: Convergence of the simulated annealing algorithm for continuous global optimization. J. Optim. Theor. Appl. **104**(3), 691–716 (2000)
30. Bilbro, G.L.: Fast stochastic global optimization. IEEE Trans. Syst. Man. Cybern. **24**(4), 684–689 (1994)
31. Thompson, D.R., Bilbro, G.L.: Sample-sort simulated annealing. IEEE Trans. Syst. Man. Cybern. B Cybern. **35**(3), 625–632 (2005)
32. Puhan, J., Bűrmen, Á., Tuma, T.: Analogue integrated circuit sizing with several optimization runs using heuristics for setting initial points. Can. J. Electr. Comput. Eng. **28**(3–4), 105–111 (2003)
33. Puhan, J., Tuma, T., Fajfar, I.: Optimisation methods in SPICE, a comparison. In: Proceedings, European Conference on Circuit Theory and Design. Stresa, Italy, Vol. 2, pp. 1279–1282 (1999)
34. Yao, X., Liu, Y., Lin, G.: Evolutionary programming made faster. IEEE Trans. Evol. Comput. **3**(2), 82–102 (1999)
35. Birbil, Şİ., Fang, S.C., Sheu, C.L.: On the convergence of a population-based global optimization algorithm. J. Glob. Optim. **30**, 301–318 (2004)
36. He, J., Yu, X.: Conditions for the convergence of evolutionary algorithms. J. Syst. Architect. **47**, 601–612 (2001)