A new Optimization Algorithm for the Design of Integrated Circuits

Jernej Olenšek*, Árpád Bűrmen*, Janez Puhan*, Tadej Tuma*
*University of Ljubljana/Faculty of Electrical Engineering, Ljubljana, Slovenia, e-mail: *jernej.olensek@fe.uni-lj.si*

***Abstract*— This paper presents a new hybrid algorithm for global optimization of integrated circuits. The algorithm exploits the efficient search mechanism of differential evolution and good global search capabilities of simulated annealing, while avoiding their weaknesses. It is easy to implement and has only a few parameters. The performance of the algorithm is verified on seven real-world cases of integrated circuit design with promising results. The proposed algorithm was implemented in SPICE OPUS simulation and optimization tool and compared with a multistart version of the constrained simplex algorithm. It outperformed the latter in terms of the final solution quality and speed.**

***Keywords*— global optimization, differential evolution, simulated annealing, design of integrated circuits**

I. 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:

$$\begin{aligned}&\min_{x\in\mathbb{R}^N} f(x) \qquad (1)\\ &f:\mathbb{R}^N\rightarrow\mathbb{R}\\ &x\in S\subset\mathbb{R}^N\\ &S=\left\{x, x\in\mathbb{R}^N, l(i)\leq x(i)\leq u(i), i=1,...N\right\}\end{aligned}$$

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 lower and upper bound for the i-th variable, respectively.

Unfortunately problem (1) usually can not be solved analytically. In most practical applications the CF is highly nonlinear, has many local minima and contains noise, especially when the value of the CF is the result of numerical simulations or measurements. Many optimization algorithms have been developed in the past. Unfortunately they are very often designed to solve very specific problems. Design of integrated circuits (IC) is an area where the shape of the CF is unknown but all the features that make an optimization process difficult are present. An algorithm is needed that is able to find good solutions in a reasonable amount of time with minimal knowledge about the shape of the CF.

In this paper we present a new global optimization method and apply it to IC design problems. It is a hybrid between differential evolution (DE) and simulated annealing (SA) algorithm. Both of these methods have received a lot of attention in the recent years and were successfully applied to many practical problems. SA and DE also have their drawbacks. A combination of both methods is expected to maintain good features of the original methods while avoiding their weaknesses. The method was tested on several real-world cases of IC design with promising results.

This paper is organized as follows. In sections 2 and 3 the basic DE and SA algorithms are briefly described. Section 4 gives a detailed description of the hybrid algorithm. In section 5 the test cases are described. Section 6 presents experimental results obtained on several real-world cases of IC design and section 7 contains the concluding remarks.

II. Differential Evolution

Differential evolution (DE) is a parallel direct search method that uses a population of points to search for a global minimum of a function over continuous search space [1]. For every point (target point) in the current population it generates a so called mutated point by adding a weighted difference of two randomly selected points from the current population to the third point. Then crossover between the current target and the mutated point is applied to generate a trial point as in (2).

$$\begin{aligned}&x^m = x^{ic1} + (x^{ic2} - x^{ic3})\cdot w \qquad (2)\\ &ic1\neq ic2\neq ic3\neq it\\ &x^g(i)=\begin{cases} x^m(i) & \text{if } \alpha < P_c^{it} \text{ or } i==I_k\\ x^{it}(i) & \text{otherwise}\\ \alpha = U & \\ i=1,2,...N & \end{cases}\end{aligned}$$

where x^m is the mutated point, $x^{ic1}, x^{ic2}, x^{ic3}$ denote three randomly selected points from the current population, x^{it} denotes the current target point, w is the weight factor (usually $w\in[0,1]$), and x^g is the generated trial point. U is an uniformly distributed random variable from $[0,1]$, and P_c is the crossover probability. I_k is a randomly chosen dimension and ensures, that the generated trial point is not the same as one of the points in the current population. The generated trial point is then evaluated. If the cost function value at the trial point is lower than that at the current target point, the trial point replaces the current target point in the subsequent generation. When this procedure is completed for every point in the current population, a new generation is started and the whole process is repeated until the maximal number of generations is reached.

III. Simulated Annealing

Simulated annealing (SA) is a stochastic global optimization algorithm that performs random sampling of the search space [2]. Its main feature is the mechanism controlling the transitions from the current point (x) to a new point (y), which is generated by random perturbation of the point x. The transition mechanism is known as the Metropolis criterion and is defined by the following expression:

$$P = \min(1, e^{-\frac{f(y)-f(x)}{T}}) \tag{3}$$

where P is the probability of making the transition from x to y (i.e. y replaces x), $f(y)$ and $f(x)$ are the CF values at x and y, respectively. Parameter T is also referred to as the temperature. This mechanism always performs downhill moves (i.e. if $f(y) < f(x)$), but it also allows uphill transitions with positive probability. This probability is controlled by the current value of the temperature parameter. Initially T is set to a very high value and most transitions (including uphill transitions) are accepted. During the annealing process T is reduced and 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. This mechanism allows the algorithm to escape from local minima and continue the search process in other parts of the search space.

One of the attractive features of the SA algorithm is the fact that its theoretical convergence to a global minimum can be proved under certain assumptions. In [3] for instance there are several convergence results for various SA algorithms. SA algorithms that can guarantee convergence usually do not perform well in practice due to their slow convergence.

IV. Hybrid Algorithm

Both algorithms from the previous two sections have their strengths but they also have some drawbacks. DE was successfully applied to many problems and has proved to be robust and fast due to its population of points and efficient mechanism of trial point generation. Due to the selection strategy (where only better solutions get into the next generation) the population can get trapped in a local minimum. SA on the other hand has the ability to escape from local minima but takes a long time to converge due to its inefficient random sampling of the search space and slow cooling schedule. In our hybrid algorithm we hope to combine the efficient search mechanism of DE and the good global search capabilities of SA. The method uses a population of M samplers, and a combination of random sampling and original DE operator. The method also uses the original Metropolis criterion to allow uphill transitions. Since the classic annealing schedule is one of the most problematic aspects of SA, we use a different approach. Instead of having a single sampler and decreasing the temperature with time we have multiple samplers operating at different 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 [4], [5]. We also include a random sampling mechanism to maintain population diversity. For this purpose every sampler has a fixed parameter called the range which is used in the generation of random moves. Now the i-th sampler g^i (where $i = 1, 2, ...M$) can be fully defined by the following features:

1. temperature T^i, which is used in the Metropolis criterion
2. range 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 IC design can have values that differ by several orders of magnitude, we also normalize all optimization variables to the [0,1] interval. The pseudo code of the hybrid method is given by Algorithm 1.

Algorithm 1

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

A detailed description of the method is given in the following subsections. Superscripts are used to denote different samplers (e.g. T^i and x^i denote the temperature parameter and the current point in the search space of the i-th sampler, respectively).

A. User Defined Input Parameters

The method uses some parameters that must be set by the user. These are the number of samplers $M \geq 4$ (population size), minimal temperature $T^M > 0$ (temperature for the last sampler), minimal range parameter $R^M > 0$ (range parameter for the last sampler), crossover probabilities for the first and the last sampler P^1, $P^M \in [0, 1]$ and stopping distance $D_{stop} > 0$. Default values for these parameters are $M = 20$, $T^M = 10^{-6}$, $R^M = 10^{-6}$, $P_c^1 = 0.1$, $P_c^M = 0.7$ and $D_{stop} = 10^{-3}$.

B. Initialization of Population

The initial population can be generated randomly but in our method we use an approach that allows more thorough sampling of the entire 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 only once in the initial population. This is very important in algorithms that use crossover operators. The values of parameters inside subintervals are chosen randomly.

C. Initialization of Method Parameters

At the beginning of the optimization run some additional method parameters must also be set. These parameters are the temperature, the range parameter and crossover probability of 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.

$$\begin{aligned} &T^M = T_{min} = T^1 \cdot e^{-c_t \cdot (M-1)} \\ &c_t = -\frac{1}{M-1} \cdot log(\frac{T^M}{T^1}) \\ &T^i = T^1 \cdot e^{-c_t \cdot (i-1)}, i = 1, 2, ...M \end{aligned} \tag{4}$$

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

$$\begin{aligned} &R^M = R_{min} = R^1 \cdot e^{-c_r \cdot (M-1)} \\ &c_r = -\frac{1}{M-1} \cdot log(\frac{R^M}{R^1}) \\ &R^i = R^1 \cdot e^{-c_r \cdot (i-1)}, i = 1, 2, ...M \end{aligned} \tag{5}$$

$$\begin{aligned} &P_c^M = P_c^1 \cdot e^{c_p \cdot (M-1)} \\ &c_p = \frac{1}{M-1} \cdot log(\frac{P_c^M}{P_c^1}) \\ &P_c^i = P_c^1 \cdot e^{c_p \cdot (i-1)}, i = 1, 2, ...M \end{aligned} \tag{6}$$

D. 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 (2) and a random move. The procedure is given by (7)

$$\begin{aligned} &x^m = x^{ic1} + (x^{ic2} - x^{ic3}) \cdot w \\ &ic1 \neq ic2 \neq ic3 \neq it \\ &x^g(i) = \begin{cases} x^m(i) + r & \text{if } \alpha < P_c^{it} \\ x^{it}(i) + r & \text{otherwise} \\ w = U \cdot 2 & \\ \alpha = U & \\ r = R^{it} \cdot tan(\pi \cdot (U - 0.5)) & \\ i = 1, 2, ...N & \end{cases} \end{aligned} \tag{7}$$

where x^m is the so-called mutated point, x^{ic1},x^{ic2} and x^{ic3} denote three randomly selected points from the current population, and x^g is the generated trial point. U is an uniformly distributed random variable from $[0, 1]$ and r is the random step generated according to the Cauchy probability distribution. If $x^g(i)$ violates box constraints it is contracted towards $x^{it}(i)$ until the violation is resolved.

Since the range parameter R 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 large. Different samplers can be initialized with different crossover probabilities (P_c^i) by the user (via the input parameters P^1 and P^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 step is used for many variables which speeds up the convergence while low values of P_c^{it} emphasize random search with only an occasional DE step.

E. Replacement

In this phase of the algorithm the generated trial point x^g is submitted to the Metropolis criterion (3) with temperature T^{it}. If the Metropolis criterion is satisfied x^g replaces x^{it} in the next generation. 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 a high temperature (i.e. T^{it} is large), x^g will have a high probability of being accepted. However if the target point is located at low temperatures, this probability will be low. With this mechanism the chances for the algorithm to escape from a local minimum are increased.

F. Transition Between Samplers

One of the main problems of the original SA algorithm is the selection of the cooling schedule. If the cooling is too fast the algorithm gets 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 our method 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 and replacement 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

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

This mechanism is quite different from the original idea of SA. Here the idea is to always send better solutions to samplers with higher T and R (i.e. if $T^{is} > T^{it}$ and $f(x^{is}) \geq f(x^{it})$) but also allow the occasional transition of a better point to a sampler with lower T. If the point x^{is} is worse than x^{it} the situation is reversed. At small values of T the method works almost as a downhill method accepting mostly points that reduce the CF value. For small values of R the random component in trial point generation is very small so the applied operator is very much like the original DE operator. As long as we are working with points at samplers with small T and R the algorithm is very similar to DE. When a good point is found the next point to be improved is likely to be at samplers with higher T and R so the algorithm runs 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 point eventually ends 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.

G. 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 a point in the population and the current best point is also used as a termination condition. When this distance falls below a user defined stopping distance D_{stop} the algorithm is terminated. The third termination condition 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 ($T_{min} = T^M$) the algorithm is terminated. In IC design when the CF value reaches zero, all the design goals are satisfied. This can also be used as a termination condition.

V. Optimization of Integrated Circuits

In IC optimization the definition of the cost function (CF) must account for all circuit properties for which the design goals are chosen. Usually there are many conflicting design goals. This makes the optimization very difficult. In addition, the circuit is expected to satisfy the design goals under different environmental conditions. This is achieved by simulating the circuit over several corner points. Every combination of environmental parameters (such as the temperature, the power supply voltage, manufacturing process variations, etc.) is represented by a corner point. For every corner point simulations are conducted resulting in real values that measure different circuit properties. The worst values of circuit properties are used to construct the penalty functions and the CF value is obtained as a weighted summ of these penalty functions [6]. Combining this process with a large number of design variables (dimensionality of the problem) and highly nonlinear characteristics of the elements in ICs makes optimization extremely time consuming. A single optimization run can often take several days or even weeks to complete.

The optimization parameters (variables) in ICs are usually CMOS transistor channel lengths, widths and multiplication factors. Resistances, capacitances and bias current values can also be included as optimization variables. Often several transistor parameters are linked by a mathematical expression (e.g. current mirrors, differential amplifiers etc.), which is why the number of transistors does not directly imply the number of optimization variables.

The proposed method was used to optimize various real-world ICs. We will describe in detail only the first case (damp1) which is an amplifier circuit. The circuit topology is given in Figure 1.

There are 27 optimization variables:

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

In this case we do not optimize multiplication factors of the transistors.

The properties which we are interested in (design goals) and their desired values are:

- circuit area ($< 10^{-8}\text{m}^2$)

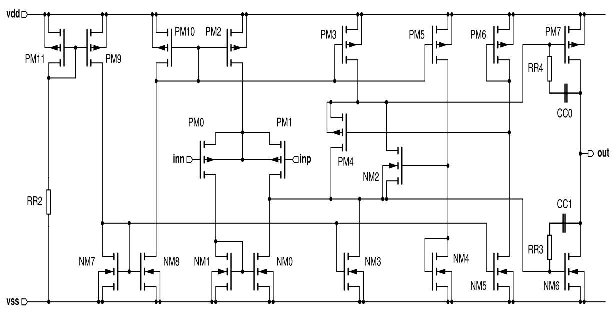

Fig. 1. Topology for the damp1 case

- current consumption ($< 1\text{mA}$)
- AC gain ($> 70\text{dB}$)
- unity gain bandwidth ($> 5\text{MHz}$)
- bandwidth ($> 500\text{Hz}$)
- phase margin ($> 60^o$)
- gain margin ($> 10^o$)
- maximal derivative of gain magnitude (< 0)
- output voltage swing ($> 1.6\text{V}$)
- DC gain ($> 60\text{dB}$)
- settling time ($< 300\text{ns}$)
- overshoot ($< 1\%$)
- slew rate ($> 5 \cdot 10^6\text{V/s}$)
- rise time ($< 200\text{ns}$)
- fall time ($< 200\text{ns}$)

In order to measure these properties we need to perform the following analyses:

- operating point analysis
- DC analysis
- AC analysis
- transient analysis

In this first case we only consider a single corner point with typical model parameters and ambient temperature of 27 degrees centigrade.

The remaining cases will only be briefly described. 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 (model parameters, temperature, circuit load, supply voltage, etc.)

The third case (lfbuffer) is a buffer 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 only 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 3 optimization parameters (4 transistors) and considers 3 corners. The case requires an OP analysis and two transient analyses to measure 9 circuit properties.

The delay case (delay) has 12 optimization parameters (6 transistors) and a single corner point. It requires an OP analysis and a transient analysis to obtain 6 considered design properties.

The last case (damp2) is another amplifier circuit with 15 optimization parameters (9 transistors, 1 capacitor and 1 resistor) and 14 corner points. We perform an OP, a DC, two AC, a transient, and a noise analysis to measure 13 circuit properties.

The summary of the main characteristics such as dimensionality, the number of design goals and the number of corner points for these problems is given in Table I. The maximal number of CF evaluations is also given.

TABLE I
TEST CASES SUMMARY

Case	design variables	design goals	corner points	maximal CF evaluations
damp1	27	15	1	500 000
damp1-5c	27	15	5	500 000
lfbuffer	36	13	1	500 000
lfbuffer-5c	36	13	5	500 000
nand	3	9	3	10 000
delay	12	6	1	500 000
damp2	15	13	14	500 000

All cases were optimized using the default values for the algorithm parameters as described in section IV/A. The method was implemented in the SPICE OPUS simulation and optimization tool [7]. It was compared with a version of the constrained simplex (COMPLEX) algorithm that is already a part of SPICE OPUS. The method uses a multi start concept to give the original local method some global search capabilities. Every time the original COMPLEX method [8] reaches its termination criteria (i.e. simplex size or maximal number of function evaluations), a special mechanism is applied to initialize a new simplex in the unexplored parts of the search space. Then the COMPLEX algorithm is restarted to find a new and hopefully better local minimum [9]. The process is repeated until the maximal number of CF evaluations is reached. This concept has proved to be fairly successful in IC design. It is however often slow and unreliable. The proposed method was compared with the multistart COMPLEX algorithm in terms of the final solution quality and the number of CF evaluations (CFE) needed to reach the solution. Since the optimization is extremely time consuming, every circuit was optimized only once.

VI. RESULTS

Optimization results are given in Table II. Circuits for which the final CF value is zero have satisfied all the design goals (global minimum was found) and the optimization was stopped at that time even though the convergence has not occurred yet. For the multistart COMPLEX method the number of restarts needed to reach a particular solution is given in parentheses.

It can be seen from the table that the proposed method outperforms the multistart COMPLEX method on all considered cases in terms of the final solution quality (minimal CF). Since the multistart COMPLEX method is designed to run without limitations on the number of CFE, the method was manually stopped once the maximal number of CFE was reached. For the proposed hybrid method the convergence can occur earlier, depending on the stopping criteria described in section IV/G.

For the first case (damp1) the multistart COMPLEX method needed 347 457 CFE (80 restarts) to find its final solution and was unable to improve that solution in all subsequent 38 restarts. Even with so many restarts and CFE the complex method was not able to find the solution that would satisfy all the design goals. The proposed

TABLE II
EXPERIMENTAL RESULTS

Case		proposed method	multistart COMPLEX
damp1	CFE for CF < 1	4 715	5 208 (2)
	CFE for CF < 0.5	4 834	22 699 (6)
	final CF	0	0.087
	CFE for minimal CF	30 965	347 457 (80)
	final CFE	30 965	500 001 (118)
damp1-5c	CFE for CF < 10	1 174	1 356 (1)
	CFE for CF < 5	8 333	21 281 (5)
	final CF	1.776	3.425
	CFE for minimal CF	497 810	231 312 (55)
	final CFE	500 001	500 001 (122)
lfbuffer	CFE for CF < 10	1 942	414 (1)
	CFE for CF < 1	12 595	1 339 (1)
	final CF	0	0.512
	CFE for minimal CF	47 744	3 310 (1)
	final CFE	47 744	500 001 (116)
lfbuffer-5c	CFE for CF < 10	1941	989 (1)
	CFE for CF < 5	5 852	13 523 (4)
	final CF	2.222	4.099
	CFE for minimal CF	430 390	394 776 (104)
	final CFE	500 001	500 001 (133)
nand	CFE for CF < 500	142	40 (1)
	CFE for CF < 200	370	94 (1)
	final CF	166.654	166.686
	CFE for minimal CF	1 185	5 818 (47)
	final CFE	1 279	10 001 (81)
delay	CFE for CF $< 20 \cdot 10^3$	71 832	21 101 (28)
	CFE for CF $< 5 \cdot 10^3$	89 019	425 548 (542)
	final CF	0	2122.000
	CFE for minimal CF	111 650	491 806 (629)
	final CFE	111 650	500 001 (640)
damp2	CFE for CF < 20	1 640	420 (1)
	CFE for CF < 10	13 806	3 926 (3)
	final CF	5.930	7.487
	CFE for minimal CF	270 060	326 011 (231)
	final CFE	395 606	500 001 (352)

hybrid method was able to satisfy the goals with less than ten percent of the maximal number of CFE.

In the second case (damp1-5c) neither method was able to satisfy all the design goals. Multiple considered corner points mean that a more robust circuit is required which may not be possible to obtain with the given topology or parameter bounds. The problem with several corners is much more complex which means that more CFE are required to find good solutions. Although the design goals were not completely satisfied, the proposed method was still able to find a considerably better solution than the multistart COMPLEX method. In this case both methods were run until the maximal number of CFE was reached.

In the third case (lfbuffer) the multistart COMPLEX converged very quickly and found a relatively good solution in the first run. But in all the remaining 115 restarts the method was unable to improve that solution. The proposed method was again able to find a global minimum with less than ten percent of the maximal number of CFE.

In the fourth case (lfbuffer-5c) both methods failed to satisfy all the design goals within the given number of CFE. Despite numerous restarts the multistart COMPLEX method was unable to find a better solution than the proposed method.

The next case (nand) was the smallest among all the considered cases. Again neither method was able to find a solution that would satisfy all the design goals. The multistart COMPLEX method converged quickly but found only local minima. It took 47 restarts to find the final solution. The proposed method required a little more than ten percent of the maximal number of CFE and was able to find a slightly better solution than the multistart COMPLEX method.

The initial progress for the delay case is much faster with the multistart COMPLEX method than with the proposed hybrid method. The multistart COMPLEX method was able to complete many restarts before the proposed method got even close to the solutions with low CF value. But slow initial progress implies that the proposed method searches more thoroughly through the search space which improves its chances of finding the global minimum. The method required about twenty percent of the total CFE to satisfy all the design goals, while the multistart COMPLEX method was not able to find a solution with $CF = 0$ even after the maximal number of CFE (81 restarts).

The last case (damp2) is more complex due to a large number of corner points. Both methods failed to satisfy all the design goals. The multistart COMPLEX method again exhibits fast initial progress but despite 352 restarts it is unable to find a better solution than the proposed method. The proposed method required about 80 percent of the maximal CFE to converge.

We will discuss the optimization results in detail for the first case only (damp1). Table III shows the measured properties at the final solution for both methods. One can see that both methods were able to satisfy most of the goals. Multistart COMPLEX method found the solution where some of the properties are even better than those found by the proposed method. This however comes at a price of not satisfying the output swing requirement. The proposed method on the other hand was able to satisfy all the design goals with a considerably lower number of CFE.

To demonstrate the difficulties of IC optimization we also calculated the profile of the CF at the final solution

TABLE III
RESULTS FOR DAMP1 CASE

Measurement	goal	proposed method	multistart COMPLEX
circuit area $[\mathrm{m}^2]$	$< 10^{-8}$	$7.33 \cdot 10^{-9}$	$5.95 \cdot 10^{-9}$
current consumption $[\mathrm{A}]$	$< 10^{-3}$	$409 \cdot 10^{-6}$	$530 \cdot 10^{-6}$
AC gain $[\mathrm{dB}]$	> 70	70.2	70
unity gain bandwidth $[\mathrm{MHz}]$	> 5	15.9	15.7
bandwidth $[\mathrm{Hz}]$	> 500	$1.11 \cdot 10^3$	$2.23 \cdot 10^3$
phase margin $[^\circ]$	> 60	73.3	90.3
gain margin$[^\circ]$	> 10	71.5	15.6
max.derivative of gain magnitude	< 0	$-161 \cdot 10^{-9}$	$-39.9 \cdot 10^{-9}$
output voltage swing $[\mathrm{V}]$	> 1.6	1.61	1.57
DC gain $[\mathrm{dB}]$	> 60	69.9	66.7
settling time $[\mathrm{ns}]$	< 300	193	168
overshoot $[\%]$	< 1	$1.29 \cdot 10^{-3}$	$885 \cdot 10^{-3}$
slew rate $[\mathrm{V/s}]$	$> 5 \cdot 10^6$	$6.36 \cdot 10^6$	$7.48 \cdot 10^6$
rise time $[\mathrm{ns}]$	< 200	75.2	64.1
fall time $[\mathrm{ns}]$	< 200	66.2	81.1

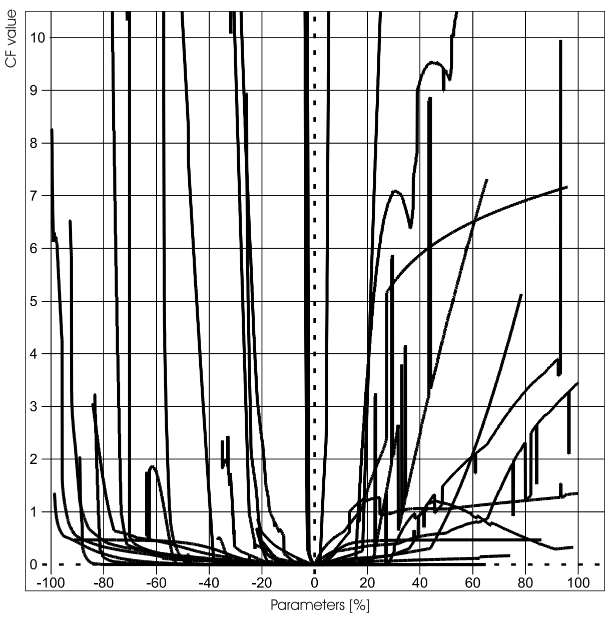

Fig. 2. Cost profile at the final solution of case damp1 found by the proposed hybrid method. Every curve represents a sweep of one of the optimization parameters.

found by the proposed hybrid method. Figure 2 shows the CF profile when sweeping through all parameters. The resulting curves intersect at the point with x axis value zero representing the final parameter values. The value of the CF remains zero or increases when the parameters are varied, which confirms the final solution quality. The figure shows that the sensitivity of the CF to different parameters varies considerably. One can also see noise and several local minima in the CF. All these facts make fast gradient descent methods inefficient and the entire optimization task extremely difficult. And when there are several corner points to consider, the task becomes even more challenging.

VII. Conclusions

A new hybrid algorithm for numerical optimization of integrated circuits was presented. Experiments were conducted on seven real-world cases of IC design to evaluate the performance of the method. The method was implemented in the SPICE OPUS simulation and optimization tool and comparison was made with a version of the simplex algorithm (multistart COMPLEX) which is already integrated as a part of SPICE OPUS. Experimental results have confirmed that the proposed hybrid method outperforms the multistart COMPLEX method in terms of global search capabilities. Since the proposed method is global in nature it requires a large number of function evaluations but practical experiments have shown that large computational burden is outweighed by the final solution quality. The proposed method is easy to implement and has only a few parameters. It would be useful to upgrade the method with a local search method to improve the convergence speed in the final stages of the optimization. Further testing is also required to confirm the efficiency of the method on more general problems.

Acknowledgment

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] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimizazation*, vol. 11, pp. 341–359, 1997.

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 1277–1292, 1983.

[3] R. L. Yang, "Convergence of the simulated annealing algorithm for continuous global optimization," *Journal of optimization theory and applications*, vol. 104, no. 3, pp. 691–716, 2000.

[4] G. L. Bilbro, "Fast stochastic global optimization," *IEEE Trans. Syst., Man, Cybern*, vol. 24, no. 4, pp. 684–689, 1994.

[5] D. R. Thompson and G. L. Bilbro, "Sample-sort simulated annealing," *IEEE Trans. Syst., Man, Cybern (B)*, vol. 35, no. 3, pp. 625–632, 2005.

[6] A. Bűrmen, D. Strle, F. Bratkovič, J. Puhan, I. Fajfar, and T. Tuma, "Automated robust design and optimization of integrated circuits by means of penalty functions," *AEU-International journal of electronics and communication*, vol. 57, no. 1, pp. 47–56, 2003.

[7] (2005) Spice opus circuit simulator homepage. [Online]. Available: http://www.fe.uni-lj.si/spice/

[8] M. J. Box, "A new method of constrained optimization and a comparison with other methods," *Computer Journal*, vol. 8, pp. 42–52, 1965.

[9] J. Puhan, A. Bűrmen, and T. Tuma, "Analogue integrated circuit sizing with several optimization runs using heuristics for setting initial points," *Canadian journal of electrical and computer engineering*, vol. 28, no. 3-4, pp. 105–111, 2003.