BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES
Vol. 56, No. 1, 2008

Evolutionary methods to analogue electronic circuits yield optimisation

P. JANTOS* and J. RUTKOWSKI

Faculty of Automatic Control, Electronics and Computer Science, Silesian Technical University,
16 Akademicka St., 44-100 Gliwice, Poland

Abstract. Evolutionary computing and algorithms are well known tools of optimisation that are utilized for various areas of analogue electronic circuits design and diagnosis. This paper presents the possibility of using two evolutionary algorithms – genetic algorithm and evolutionary strategies – for the purpose of analogue circuits yield and cost optimisation. Terms: technologic and parametric yield are defined. Procedures of parametric yield optimisation, such as a design centring, a design tolerancing, a design centring with tolerancing, are introduced. Basics of genetic algorithm and evolutionary strategies are presented, differences between these two algorithms are highlighted, certain aspects of implementation are discussed. Effectiveness of both algorithms in parametric yield optimisation has been tested on several examples and results have been presented. A share of evolutionary algorithms computation cost in a total optimisation cost is analyzed.

Key words: evolutionary algorithms, parametric yield, production cost, optimisation, computation cost.

1. Introduction

An increasing complexity and decreasing physical size of integrated analogue electronic circuits (AEC) has prompted the necessity of careful choice of elements' values and tolerances. It has become very difficult to achieve acceptable manufacturing yield even if the nominal design fulfils all design constraints. The aim of system design is to find the values of adjustable element parameters that make the entire system satisfy design specifications. The answer to the question of whether a system with chosen values of element parameters fulfils the specification is obtained with the use of computer simulations [1, 2]. Complexity of analysis increases when tolerances and statistically distributed parameters are considered.

In this paper, a new approach to yield and cost optimisation is presented. Evolutionary algorithms such as, genetic algorithm (GA) and evolutionary strategies (ES) have been utilized as an optimisation engine. Reference [3] is among the first to present a similar idea. Author, however, have not analysed computation cost of the used evolutionary techniques.

The aim of this paper is to propose genetic algorithm and evolutionary strategies as an alternative method of yield optimisation, taking into account analysis of factors of total optimisation cost and an economical sense of such approach to yield and cost optimisation.

Section 2 of this paper is dedicated to presentation of standard methods of yield and cost optimisation. In Section 3 basics of genetic algorithm and evolutionary strategies are presented. Some implementation issues of the above algorithms for yield and cost optimisation are described in Section 4. Example circuits and their optimisation results are presented in Section 5. The computation cost of the whole process is discussed in Section 6 and conclusions are presented in Section 7.

2. Yield optimization

The manufacturing yield is composed of two factors:

– parametric yield;
– technological yield.

The former is a result of a sensitivity of circuit performance to AEC parameters deviations, caused by manufacturing conditions. The latter is a result of catastrophic technological failure [2]. This paper is orientated to parametric yield optimisation. In further parts of this paper, the term yield will be used as a synonym of parametric yield.

Many methods of yield optimisation have been presented so far. The methods may be grouped as following:

– Monte Carlo methods;
– linear approximation methods;
– quadratic approximation methods;
– simplicial approximation methods;
– statistic methods;
– yield gradient methods.

The information about parametric yield may be obtained with the use of computer simulations allowing for Monte Carlo analysis (MCa), e.g. Spice [2].

The presented method of yield optimisation may be classified as a member of Monte Carlo group. However, use of the evolutionary core has caused a great computation cost and time increase.

*e-mail: pjantos@polsl.pl

2.1. Circuit description. An optimized circuit may be given by a set of N elements **E** and a circuit structure description (e.g. a node list).

$$\mathbf{E} = \{E_1, E_2, \ldots, E_N\} \tag{1}$$

Each of the E_n may be described by number of parameters, that are elements of set **P**.

$$\mathbf{P} = \{P_1, P_2, \ldots, P_L\} \tag{2}$$

The number of parameters L, defines parameter space dimension. It has been assumed that each parameter is given by two values: nominal value and design tolerance. Hence, two vectors may be introduced. Vector of elements nominal values **NV**.

$$\mathbf{NV} = [NV_1, NV_2, \ldots NV_L] \tag{3}$$

and vector of element tolerances **T**.

$$\mathbf{T} = [tol_1, tol_2, \ldots tol_L] \tag{4}$$

In the presented research elements of **E** are resistors, capacitors and coils, characterized by a single parameters. Then, $L = N$. The circuit is described by M design specifications Q_i, that allow for defining an acceptability region in the parameter space.

$$\mathbf{Q} = \{Q_1, Q_2, \ldots, Q_M\} \tag{5}$$

One specification is considered ($M = 1$), that is a circuit amplitude response deviation from the nominal characteristic. This specification alone allows for defining the acceptability region.

The aim of optimisation process is to adjust vectors **NV** and **T**, so that all of the M specifications are fulfilled.

Two optimisation parameters ($O = 2$) are considered, that is: parametric yield O_1 and circuit cost O_2.

2.2. Design Centring and design tolerancing. Generally, there are three ways of yield and cost optimisation. These are:

- Design centring;
- Design tolerancing;
- Design centring with tolerancing.

The methodology presented in this paper is based on the design centring with tolerancing approach.

The design centring and the design tolerancing procedures are presented in the Fig. 1 (for $L = 2$, $M = 2$). An acceptability region is defined in the parameter space.

Design centring consists of adjusting **NV**. This approach does not allow for changing device production cost.

Design tolerancing consists of modifying **T**. This procedure allows for both production yield and cost adjusting.

Design centring and tolerancing is a hybrid of the above two methods. Both **NV** and **T** are an object of manipulation and this approach is definitely superior over two previous ones.

Procedures that are presented in this paper are based on design centring and tolerancing.

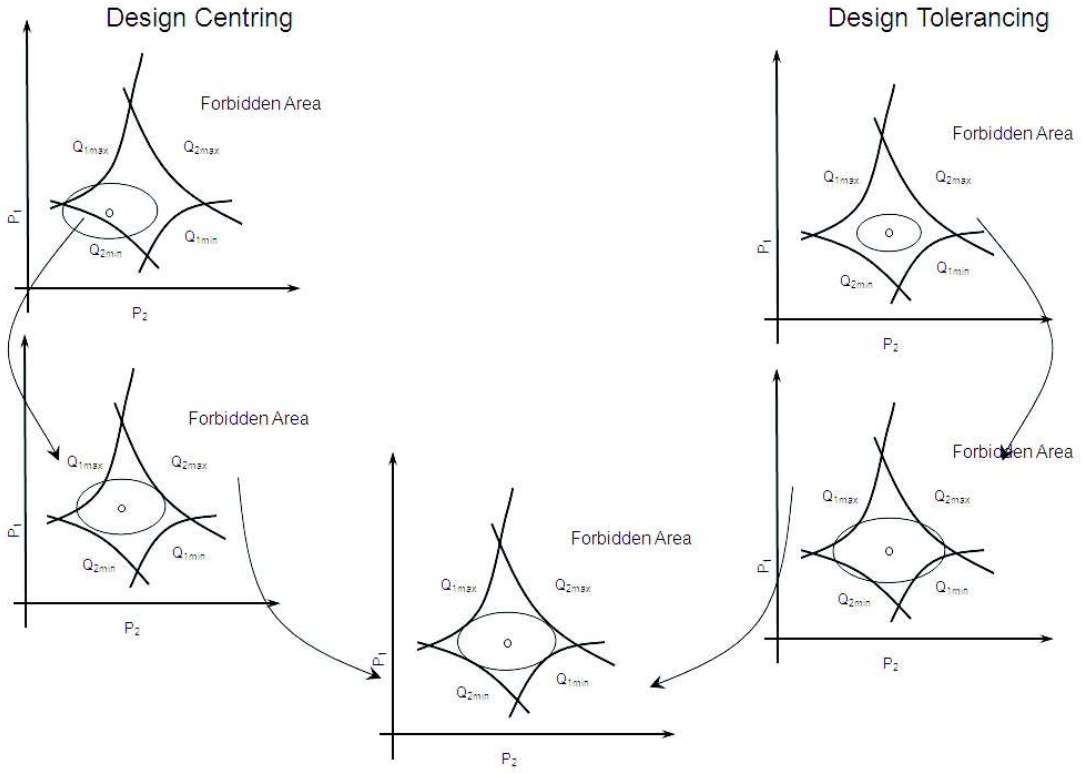

Fig. 1. Design centring and design tolerancing

3. Evolutionary algorithms

Evolution is a process of adaptation and optimisation. Scientists have imitated mechanisms of evolution to create an efficient and powerful tool of optimisation – genetic algorithms (and almost simultaneously evolutionary strategies). Evolutionary algorithms (EA) are well known and commonly used tools for optimisation, classification, approximation, and other numerical goals. The very reason to use them is an ease of implementation for different tasks. It is essential, though, to find a proper individuals coding and a fitness function. A general cycle of the EA is presented in the Fig. 2.

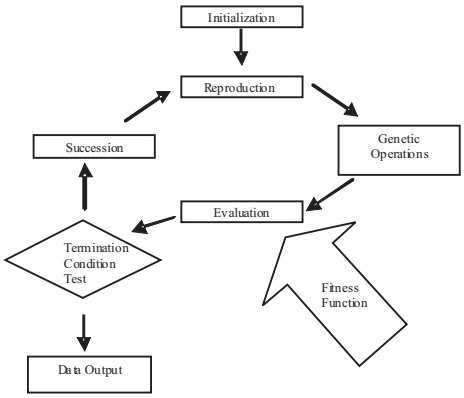

Fig. 2. General evolutionary algorithms cycle

The evolutionary algorithms are initialized with a set of randomly generated individuals which are the initial population (first parental population). The first phase of the EA cycle is a reproduction (selection and replication). Individuals are chosen and copied to the offspring population. With genetic operations such as (but not limited to) crossover and mutation, the individuals are modified.

The most important part of the EA is individuals' evaluation and assessment. Individuals' genotypes are decoded to phenotype. The phenotype is used to calculate the fitness function value. If its target value is reached, the algorithm is terminated. In the alternative case, the entire population (or certain group of individuals) is copied in the process of succession to the next parental population and the cycle begins again.

Evolutionary algorithms are based on probabilistic operators. This can result in getting as many different solutions of a task as number of times the evolutionary program is run.

The No-free-lunch Theorem [4] says that there is no one algorithm that would be applicable to solve all possible tasks. It is suggested to use EA in situations in which an existing optimisation algorithm is not satisfactory [4–8].

The cycle presented in the Fig. 2. is common for both genetic algorithm and evolutionary strategies. In the following, the most important differences between utilized evolutionary algorithms are presented.

The algorithms used in the presented research are genetic algorithm (GA) and evolutionary strategy (μ, λ) (ES). In principles, both algorithms are very similar. There are several differences, though, between these two. The most important are listed in the Table 1.

Table 1
The most important differences between Genetic Algorithm and Evolutionary Strategies

	Genetic algorithm	Evolutionary strategies
Coding	Binary (natural or Gray's Code)	Real numbers
Evolutionary operators range	Constant	Self-adapting
Most important operator	Crossover	Mutation

The coding of ES individuals is based on real numbers, while GA coding is based on binary strings. It is followed by differences in evolutionary operators, which must fit the individuals' coding. Real number coding allows for high precision, which can be very important in solving some problems. In this research, the element parameters precision is of a minor issue. Hence, this ES feature cannot be considered as an advantage over GA.

The binary coding allows GA to be used in a more flexible way and in a wider variety of tasks than ES. Unless there is the floating point coding implemented, the fixed length of the binary string causes limitations in real numbers precision. It is suggested to implement Gray's encoding to avoid convergence problem, due to crossover process. The mutation in GA may cause a very large difference between parental and offspring individual. Gray's encoding does not influence mutation in GA at all.

Real number coding allows for an easy encoding of small numbers (capacitors and coils parameters). In case of the fixed length binary coding, there are additional arithmetic operations required, which increase computation cost.

The implementation of evolutionary operators, which in both algorithms have been mutation and crossover, is related to the coding. Both crossover and mutation for GA are simple bit operations, whereas in ES these are arithmetic processes. Moreover, in ES a self adapting mutation range is used, which is the main means of finding the optimal problem solution. A mutation is the main factor of population development in ES, while in GA it is the.

In the presented work the population size for both GA and ES is 70. For evolutionary strategies the scheme (μ, λ) has been chosen, with $\mu = 10$ and $\lambda = 70$ (typical values, suggested by many authors).

For both of the utilized algorithms, a microgenetic algorithm has been implemented in the early stage of the process, if the convergence is not reached in 20 generations, the whole process is reinitialized.

4. Implementation

Successful implementation of both GA and ES requires solving of few problems, prior the EA run.

Firstly, the proper method of individuals coding presented in Section 4.1. Choosing of the selection method is the next

problem. The simple roulette method may not give useful results in a satisfactory time limit. The use of deterministic or ranking methods in combination with elitism[1] seem to be a better choice. The deterministic selection method with simple elitism has been found to be the best.

The other problem is a choice of proper fitness function (FF). An incorrectly constructed FF may slow down the process of finding the global optimum, or even cause it not to be found at all.

When defining FF, it is important to consider the fact that two optimisation parameters (criteria) are taken into account. The first is parametric yield and the second is cost of circuit. A measure of the latter may be, for example, an average tolerance of element parameters.

A masking of the aforementioned criteria may be observed, which is followed by a finding individuals that are inexpensive but do not meet specifications, or individuals that are expensive and meet specifications. The way to cope with this problem is by using penalty modifiers to the fitness function, as presented in Section 4.2.

One of EA advantages over other deterministic optimisation methods is that they allow the nominal values and tolerances of all elements to be adjusted simultaneously.

A flowgraph of the applied algorithm is presented in the Fig. 3.

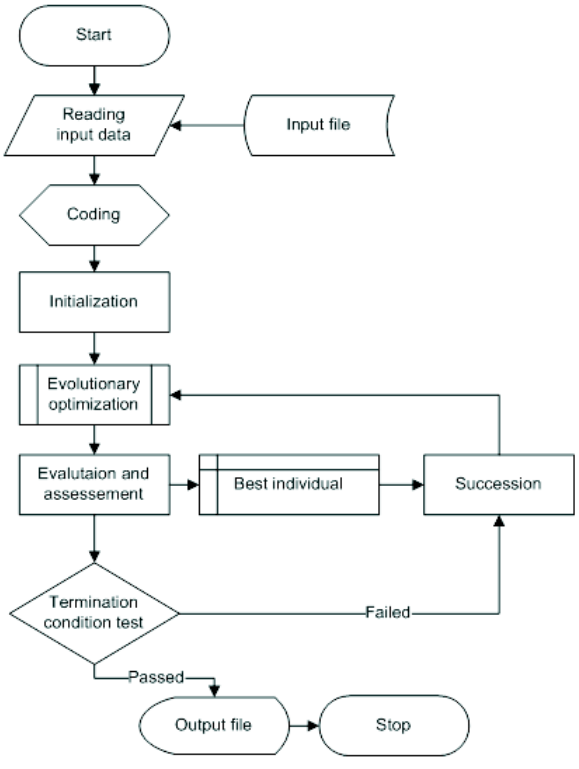

Fig. 3. Yield optimisation algorithm

4.1. Input data preparation and coding. Input data is prepared in text files. Optimized circuits are described with a net-list. Additionally, parameters listed below are predetermined:

- target parametric yield;
- target circuit cost (i.e. the average parameters tolerance);
- number of Monte Carlo analysis runs and analysis range.

The frequency characteristic of the optimized circuit is computed with the use of a built-in AEC simulator. The simulator allows for Monte Carlo analysis and is used to determine the parametric yield of the analysed circuits.

The coding process for both evolutionary algorithms, is illustrated by Fig. 4 example, 20% tolerance assumed.

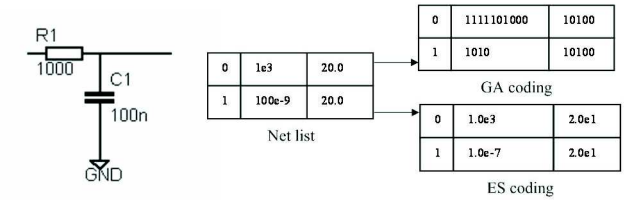

Fig. 4. Coding process

Genetic algorithm requires binary coding. To gain the satisfactory precision of capacitor and coil value, its scaling has been implemented. Coding precision for GA is as in Table 2.

Table 2
Coding precisions in GA implementation

Element	Precision
Resistor	10^{-3} Ω
Capacitor	10^{-10} F
Coil	10^{-8} H

The output data. The output data is stored in a text file. It contains the encoded net-list of the optimized circuit, the expected yield value and the average tolerance.

4.2. Criteria masking and fitness function. An evaluation stage in the process of a two-criteria optimisation is presented in the Fig. 5. In the figure, values of optimisation parameters (criteria) O_1 and O_2 are presented with bars. The fitness function value is calculated as a average value of both criteria (Fig. 5a). Some modifications to its calculation are presented in further figures (Fig. 5b–d).

The criterion O_2 is crucial and it has to have value higher than required O_2. Otherwise, or an individual codes an unacceptable solution. Criterion O_1, with a very high value, causes a very high fitness function value, even though O_2 has not reached the required value. Moreover, the fitness value is higher, than in the case presented in the right side of Fig. 5a (criterion O_2 value is higher than required). The situation presented in the Fig. 5a may be called criteria masking (O_1 is masking O_2). A few possibilities of avoiding this problem are presented in Figs. 5 b-d. At right sides of the aforementioned figures a case with criterion O_2 having value higher than the required is given for comparison.

[1]Elitism means copying the best individual (or more) to the next generation without modifying it.

An implementation of open-scale criteria values is one of the reasons for criteria masking and t should be avoided. Introduction of the criteria maximum $W_{\max}$ may not solve the problem completely (Fig. 5b.), though. If O_2 is close to the required value, but doesn't reach it, and O_1 reaches maximum value, the fitness function counted as an average may still overcome a value of better individual (at the right side of the figure).

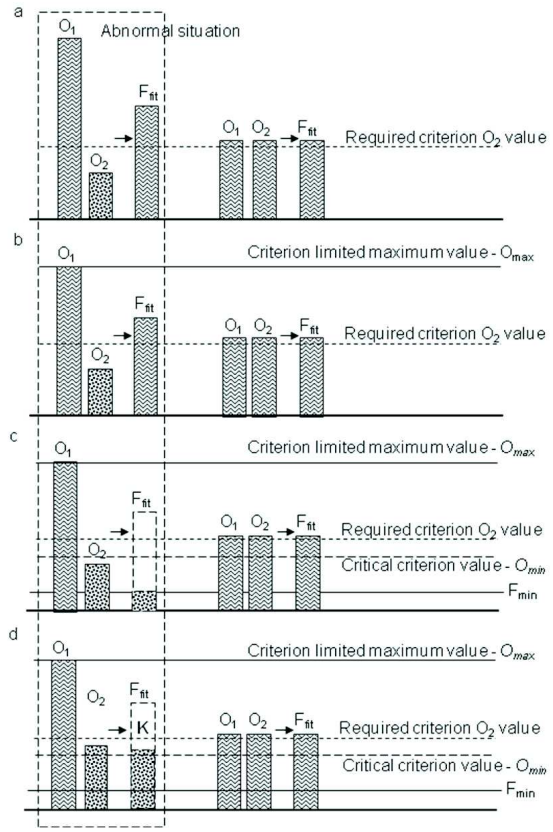

Fig. 5. Criterion masking problem (a) and three proposed solutions of avoiding it(b-d)

It is possible suggested to set a criterion critical value $-O_{\min}$. In case the criterion doesn't reach it, FF is cut to the default value $F_{\min}$ (Fig. 5c). This is a severe penalty function (cut-off penalty) and if $O_{\min}$ is very close to required value it might slow down convergence.

Small gap between $O_{\min}$ and the required value may lead to criteria masking (Fig. 5d). An unmodified FF is given with a dashed line. It is higher than one of the correct individual (in the right side of the figure). A mild penalty is subtracted to prevent this situation. This penalty does not slow down convergence. It promotes a fully acceptable individuals instead. Its value K may be calculated in several ways. With this last modification of a fitness function, a criteria masking may be successfully avoided. All of abovementioned criteria and fitness function modifications are implemented in the presented research.

In the process of evaluation of individuals, the MCa is run for each of them. In this process, information about the parametric yield is collected. Additionally, an average tolerance of all circuit elements is calculated. With this two values, the fitness function is computed.

Fitness function. For both of the researched evolutionary algorithms the same fitness function has been utilised. The aims of the function have been listed below:

- algorithm good convergence (prerequisite condition);
- gaining an acceptable parametric yield (prerequisite condition);
- lowest circuit cost (sufficient condition).

The fitness function is given with Eqs. (6–12).

$$F_{fitness} = \begin{cases} \left(\sum_{i=1}^{O=2} a_i O_i + K \right) & \text{if } \forall \left(O_i > O_{\min} \right) \\ F_{\min} \text{ if } \exists \left(O_i \leq O_{\min} \right) \end{cases} \tag{6}$$

$$\sum_{i=1}^{O=2} a_i = 1 \tag{7}$$

$$\max \left(O_i \right) = 1 \tag{8}$$

An equal weight for both optimized parameters has been assumed, $a_1 = a_2 = 0.5$. Then:

$$O_1 = 0.025 + 0.975 \left. \frac{|nMC| - |bad|}{|nMC| - |accbad|} \right|_{O_1 > 1 \rightarrow O_1 = 1} \tag{9}$$

$|nMC|$ is a number of Monte Carlo analysis series, $|bad|$ is number of results exceeding an acceptable deviation from the nominal circuit performance, $|accbad|$ is an acceptable number of incorrect Monte Carlo analysis run.
Therefore, the denominator in the Eq. 9 is acceptable parametric yield. Next,

$$O_2 = 0.025 + 0.975\ y = \left. \frac{< tol >}{preftol} \right|_{O_2 > 1 \rightarrow O_2 = 1} \tag{10}$$

$$< tol >= avg \left(tol_l \right) \quad l = 1, 2, 3 \ldots N \tag{11}$$

$preftol$ is a preferable average tolerance of all circuits elements (e.i. cost of the circuit), tol_l is a tolerance of l-th element out of N element **T** set elements.

Equations (9) and (10) allow for limiting values to range $< F_{\min} = 0.05, 1 >$ (Fig. 5b). Additionally, both:

- cut-off penalty;
- penalty function K.

have been implemented. Cut-off penalty has been used in the equation (6).

The following penalty function K has been introduced (Fig. 5d).

$$K = - \left. \frac{||dopbad| - |bad||}{100} \right|_{|bad| < |dopbad| \rightarrow K = 0} \tag{12}$$

The fitness function described with Eqs. (6) to (12) allows for efficient AEC both yield and cost optimisation. The

aims of fitness function listed before are accomplished. Saturation allows (introduced $O_{\max}$) for partial avoidance of criteria masking. The cut-off penalty function allows for removing individuals, that doesn't meet one of criterions critically. The penalty function (12) allows individuals that have been close to meet yield criterion for further evolution.

All coefficients of the fitness function have been chosen heuristically, for the best algorithms performance, based on numerous examples.

The fitness function coefficients are gathered in the Table 3.

Table 3
Fitness function coefficients

Coefficient	Value
$a_{1,2}$	0.50
$O_{\max}$	1.00
$O_{\min}$	0.20
$F_{\min}$	0.05

4.3. EA parameters. Genetic algorithm has been implemented with the following parameters (Table 4)

Table 4
Genetic algorithm parameters

Parameter	Value
Population size	70
Crossover probability	0.30 (uniform crossover)
Individual mutation probability	0.30
Gene mutation probability	0.05
Selection type	deterministic
Succession type	elitism (2 individuals)

Evolutionary strategy has been implemented with the following parameters (Table 5)

Table 5
Evolutionary strategies parameters

Parameter	Value
Strategy type	(μ, λ)
Population size	$\mu = 10$, $\lambda = 70$
Selection type	multiple draw with returning
Succession type	elitism (2 individuals)
Implemented operators	mutation, crossover

Evolutionary algorithms parameters have been chosen heuristically according to hints presented in [4–7]. It has been decided to enforce the same number of individuals in GA and ES populations, for better algorithms comparison.

5. Verification on practical examples

Five different passive circuits have been chosen with increasing number of elements. One of the exemplary circuits is shown in Fig. 6. Number of elements in testing circuits varied between 2 and 12 (2, 3, 5, 7, and 12).

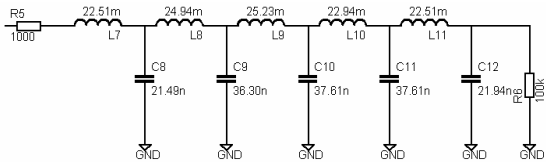

Fig. 6. Test circuit no. 5

The choice of only passive circuits does not determine a limitation of the method. The EA work only on an element's parameters, so whether or not the element is a capacitor, resistor, or an active element does not make any difference as long as it is possible to describe whole device with relationship(s) that allow to calculate parametric yield. It may be claimed that the circuit topology does not influence algorithms performance. However, proof of this thesis requires further research.

The obtained results do not allow to categorically decide which of two the two algorithms is better. Checking the computations cost dependency on AEC complication level and to determine the economic sense of using evolutionary algorithms for yield optimisation have been the main concern of the performed research. It has been decided to limit the number of generations to 200. This number of generations is enough to decide which of the two investigated algorithms is more efficient. Each process of optimisation has been repeated 5 times and the average values calculated. These are presented in figures below.

5.1. Comparison of algorithms performance. Figure 7 presents the average fitness value of the best individual in five separate and independent processes of optimisation after 200 generations. It can be observed that there is no significant difference between results of optimisation with the use of genetic algorithm and evolutionary strategy.

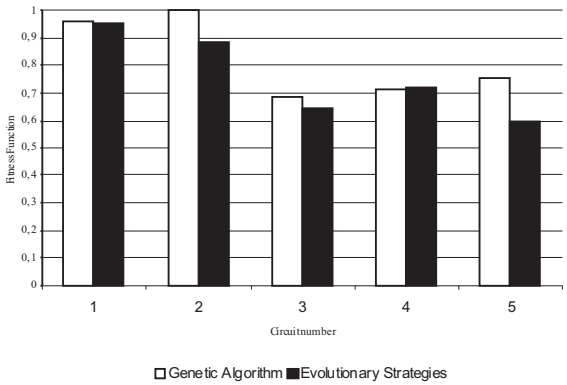

Fig. 7. Average value of the fitness function of the best individual after two hundred generations of yield optimisation with the use of genetic algorithm and evolutionary strategies

The maximum of fitness value depends on parameters of the process. If the target tolerance was set too high (i.e. much higher than the possible to achieve with the assumed yield), the fitness function did not reach the maximum value. Scaling of the fitness function have been used in more than 60%, for the best optimisation results.

In Fig. 8 the calculated tolerances are presented. Slightly better results are noticeable with the GA.

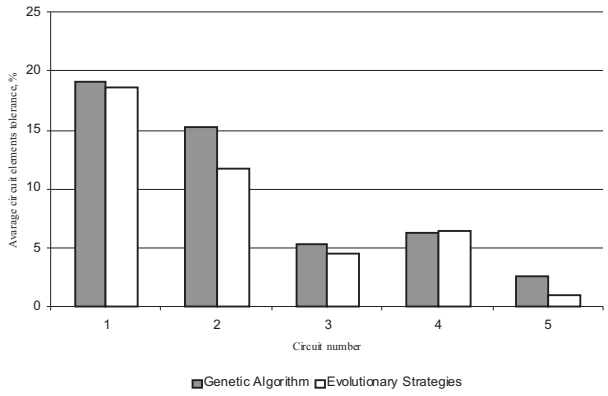

Fig. 8. Average tolerance of the best individuals after two hundred generations of yield optimisation with the use of genetic algorithm and evolutionary strategies

6. Implementation

6.1. Computer time spent. All figures in this section show dependencies between computing time and circuit complexity.

Figure 9 presents the total CTS (Computer Time Spent) of the entire optimisation process, which includes the evolutionary algorithms cost and the Monte Carlo analysis cost.

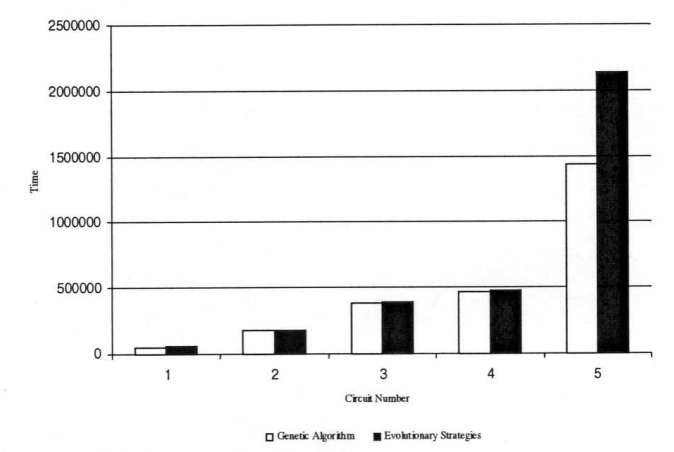

Fig. 9. Overall computing cost in the yield optimisation with the use of genetic algorithm and evolutionary strategies

In this case, processing time for both genetic algorithms and evolutionary strategy is almost the same for all the circuits, except the fifth one (Fig. 6.) – the difference has been caused by the way, the finding of the convergence centre is enforced (microgenetic algorithm). Algorithms are initialized with the starting tolerances given in an input file (20% tolerances). If the algorithm is unable to find the convergence centre in the assumed number of generations (200), the whole algorithm is reinitialized with the smaller input tolerances. The results presented in Fig. 9 may suggest that the more complex the circuit is, the longer it takes for EA to find the convergence area. However, it may be not true, as the majority of the time involved is consumed by the MCa.

6.2. Cost of evolutionary computation. Only times needed to process one generation are now taken into account and components of the overall computation time are analysed.

Figure 10 presents the total time of processing of one generation for both evolutionary algorithms. Evolutionary strategy (μ, λ) appear to be slightly slower.

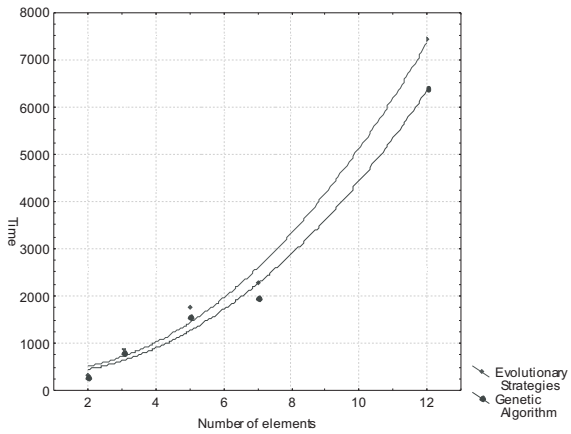

Fig. 10. Total cost of one generation processing in the yield optimisation with the use of genetic algorithm and evolutionary strategies

The total computation time of processing in one generation has two components: Monte Carlo analysis, and evolutionary algorithm.

It is noticeable that the character of the dependency between the number of elements and total computation cost is of exponential character. The question is whether it is caused by evolutionary algorithms, the Monte Carlo analysis, or both.

Evolutionary algorithms would be an efficient tool and worth further research – if and only if – the time of processing in one generation would be, at the minimum, in linear dependency to the number of elements.

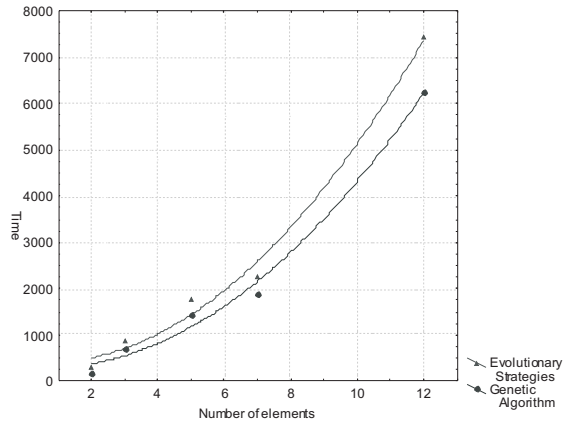

Fig. 11. Monte Carlo analysis computing cost per one generation in the yield optimisation with the use of genetic algorithm and evolutionary strategies

Relationship presented in Fig. 11 suggests that the Monte Carlo analysis computation cost increases exponentially with the number of elements.

Figure 12 shows the dependency between the evolutionary algorithms CTS in one generation, and the number of elements. The results presented in the figure are very promising. The character of dependency between computing costs of evolutionary algorithms themselves is logarithmic. The computation time rises slower than the circuit complexity. This fact is one of the most important reasons to keep developing usage of evolutionary algorithms in yield optimisation.

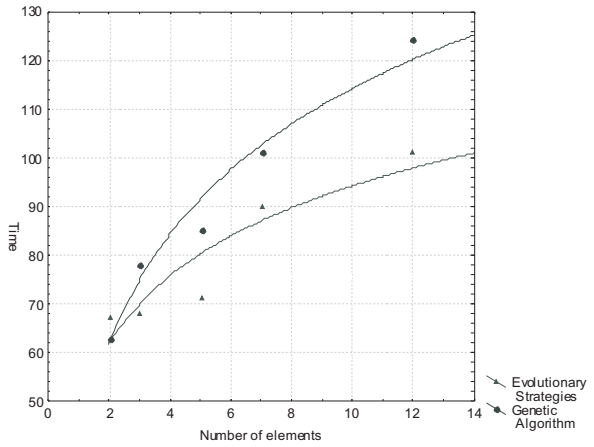

Fig. 12. Cost of evolutionary computing in the process of yield optimisation with the use of genetic algorithm and evolutionary strategies

Figure 13 shows that more complicated the circuit is the more economical is the use of evolutionary algorithms. Also, it can be concluded that ES is less time consuming than GA per one generation.

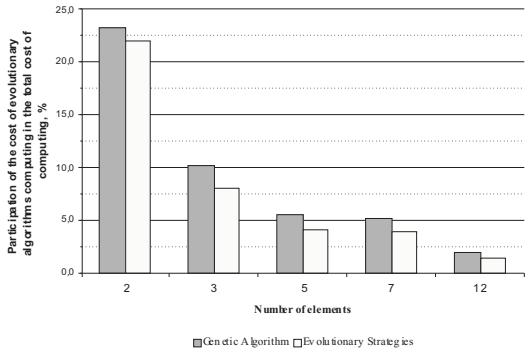

Fig. 13. Participation of the cost of evolutionary algorithms processing of one generation in the yield optimisation with the use of genetic algorithm and evolutionary strategies

7. Conclusions

A new approach to the yield and cost optimisation has been presented. This approach uses evolutionary algorithms, such as genetic algorithm and evolutionary strategy (μ, λ) in the process of optimal design centring and tolerancing. Results presented in this paper allow us to claim that evolutionary algorithms may be an alternative method of yield optimisation to the standard ones. A very important and positive feature of EA is the fact that they manipulate both tolerances and nominal values of element parameters, so the device is as inexpensive as possible with the parametric yield kept at the specified level.

Efficiency of the method has been verified on practical passive analogue circuits. It may be easily implemented for active analogue circuits with no loss of efficiency.

The economic sense of using evolutionary algorithms, in yield and cost optimisation, is discussed. The MCa is the most time consuming part of the presented method. It does not indicate, though, that it is not worthwhile to develop the EA based methods of yield optimisation. The Monte Carlo analysis is used as the tool of parametric yield calculation. Other methods of its estimation may be integrated with evolutionary optimisation engine [3].

It is proven that the EA computation cost is in logarithmic dependency to the number of circuit size. This is a good reason to keep developing EA based methods and to start research for inventing other ways of estimating parametric yield than MCa. It is not stated which of the two algorithms is better in the field of yield optimisation and it has not been the aim of the paper. Answering this question must be preceded by more detailed studies.

Acknowledgements. Piotr Jantos would like to thank Ms. S. Elizabeth Cohen and Miss Aleksandra Wysocka for their help.

REFERENCES

[1] J.M. Wojciechowski, L.J. Opalski, and K. Zamłyński, "Design centring using an approximation to the constraint region", *IEEE Transactions on Circuits and Systems – I: Regular Papers*. 5 (3), 598–607 (2004).

[2] H. Tian and L. Milor, "Yield optimisation for integrated circuits", ISR TR 92-103, http://hdl.handle.net/1903/5284 (1992).

[3] L. Zielinski and J. Rutkowski, "Design tolerancing with utilization of gene expression programming and genetic algorithm", Int. Conf. Signals and Electronic Systems ICSES, 13-15 (2004).

[4] T. Bäck, D.B. Fogel, and Z. Michalewicz, *Evolutionary Computation I – Basic Algorithms and Operators*, IOP Publishing Ltd, Bristol, 2000.

[5] T. Bäck, D.B. Fogel, and Z. Michalewicz, *Evolutionary Computation II – Advanced Algorithms and Operators*, IOP Publishing Ltd, Bristol, 2000.

[6] D. Quagliarella, J. Périaux, C. Poloni, and G. Winter, *Genetic Algorithms and Evolutionary Strategies in Engineering and Computer Science – Recent Advances and Industrial Applications*, John Wiley & Sons Ltd, London, 1998.

[7] R. L. Haupt and S.E. Haupt, *Practical Genetic Algorithms*, John Wiley & Sons Ltd, London, 1998.

[8] D. Whitley, "An overview of evolutionary algorithms: practical issues and common pitfalls", *Information and Software Technology* 43 (14), 817–831 (2001).