# EVOLUTIONARY DESIGN OF MICROWAVE CIRCUITS

**Tomáš Müller and Zbyněk Škvor**

Czech Technical University
Department of Electromagnetic Field
Technicka 2, Prague, Czech Republic
muller@kti.mff.cuni.cz

*Abstract:* **Genetic and evolutionary algorithms have been used in microwave circuit design. These algorithms proved to be capable to construct circuit topology so that design goals were met. In this paper we propose an evolutionary (single population) algorithm for designing microwave circuits and present some of its results.**

## INTRODUCTION

Nowadays microwave designers make use of CAD tools for circuit evaluation, design and optimisation. At the time being, optimisation in microwave circuit design means finding circuit element values for a fixed circuit topology, e.g. circuit topology is an input parameter.

A different approach has been chosen in this work: the design of a microwave circuit was left completely with the optimisation algorithm. Input parameters include design goals (required frequency response), number of circuit ports and a set of available circuit elements. Optimisation algorithm then sets up different circuit topologies, and the topology that satisfied design goals is an output parameter, together with circuit element values.

Repeated evolution of a population is the basic idea of all genetic algorithms. Population is typically represented by a set of elements, kinds of candidates for the desired solution. In our case, these candidates can be for example individual circuits. The genetic algorithm is trying with the help of crossing, mutation and selection of population members to gradually find out a better and better solution. [1, 6, 7]

In the following text, we present an evolutionary algorithm that always works with only a single element (solution). This can be seen as an extreme case of genetic algorithm with the cardinality of population equal to one. The algorithm works in iterations (evolutions). In each evolution step it takes the previous (parent) solution and by its mutation it creates other (children) solutions from it. Finally, one of the generated solutions is picked up to be the parent for the next iteration step.

This approach can also be considered as a variation of the local search algorithms. These algorithms also work in iterations, with one infeasible solution. In each iteration step, neighbour solutions from the previous solution are generated and evaluated. At the end of the iteration, one neighbour solution is selected into the next iteration. The purpose of these algorithms is simple: They are repeatedly trying either to decrease the number of inconsistencies in the solution and/or to get closer and closer to some optimal one. Recently, there have been developed many kinds of the neighbour search algorithms (e.g. hill climbing, steepest descant, min-conflict, genet) and their improvements (e.g. random walk, tabu-list, simulated annealing) [2, 3, 4].

In this paper, we present an evolutionary (or local search) algorithm for design of microwave circuits. The input of this algorithm is a number of gates, and desired characteristics (S-parameters) on a discrete set of frequencies. The output of the algorithm is a circuit (described as a connection list with evaluated circuit elements), which meets the required characteristics. Some practical results are also included at the end of this paper.

## ALGORITHM

As indicated above, the proposed algorithm works in iterations, with a single solution (a circuit). In each iteration step, it tries to improve the parent solution by incremental generating and evaluating child solutions until an acceptable one is found. This acceptable solution becomes a parent solution in the next iteration step. In the following section, we briefly describe the evaluation of circuits, generating of children solutions.

A circuit is represented as a graph, with vertices and edges between these vertices. Every edge represents an element (resistor, capacitor, inductor, micro-stripe line, …), every vertex represents an input/output gate, a connection between one to four elements (ideal or non-ideal) or an element with one or more than two ports (e.g. short end, open end, bounded wiring, …). The initial circuit consists only from a connection between input/output gates.

Because we need to compare two generated circuits, we need to evaluate them in order to how they met the input requirements. For computation of a circuit's S-parameters the program MIDE [5] is used. The resultant objective function is then counted as a root mean square (RMS) of the differences between the required and the computed characteristics on the given frequencies (see Fig. 1.). So, we want to minimize this objective function (RMS error). Its zero value indicates that the required circuit is found.

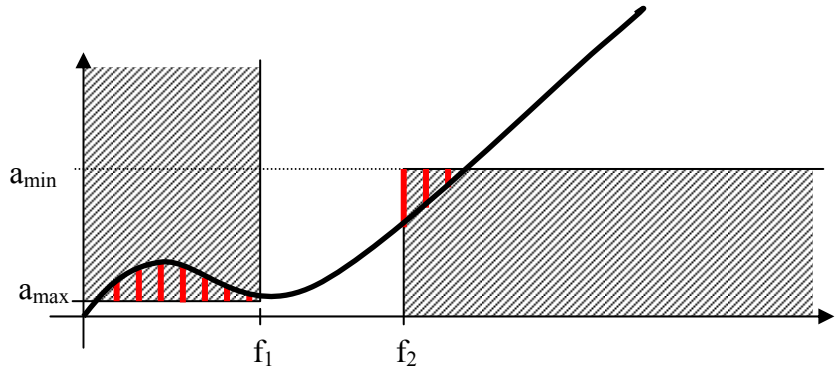


Fig. 1. RMS error for low-pass filter – only differences in shaded areas are considered.

A child circuit is generated from the parent circuit by applying one of the below listed operations. Many of them are deduced from the graph representation of a circuit (e.g. adding/removing an edge from a graph). An operation is applied only if the resultant graph is consistent – it is connected, there is no vertex with more than four edges (because there is no connection element with more than four ports in MIDE) and every vertex has an appropriate element associated (with correct number of ports).

The operations are:

- A parameter of some element is changed. (e.g. resistor's resistance is changed)
- An element (or vertex) is replaced by another with the same number of ports. (e.g. a resistor is changed to an inductor)
- An edge (2-port element) is added between two arbitrary vertices. Elements of respective vertices are changed (number of ports is increased by one).
- An edge is removed. Elements of appropriate vertices are changed. The graph has to remain connected.
- A vertex (with two or more edges) is split up into two vertices and an edge is added between them. Appropriate elements are changed.
- An edge with a single-edge vertex (e.g. short or open end) is added into a vertex. Appropriate elements are changed.
- An edge is split up into two edges and a vertex is added between them. Appropriate elements are changed.
- A vertex with one or two edges is removed and appropriate edge is removed or joined respectively. Appropriate elements are changed.

Remark, that there are also operations for simplifying the graph (e.g. by removing an edge or a vertex). There are also many other possibilities what operations to use, for example the above operations can be adopted not to violate planarity of the circuit. There can also be other circuit limitations introduced, e.g. for the maximum number of elements, elements of given type or physical feasibility of the resultant circuit.

Each circuit element, which can be used has a defined set of its parameters with minimal and maximal bounds and implicit values. When an edge or a vertex is created or changed, an appropriate element is selected randomly. The element's parameters are selected from the log-normal distribution with the mean value equal to the parameter's implicit value. When a parameter is changed (the first operation in the above

list), a new value is selected again from the log-normal distribution, but with mean value equal to the parameter's previous value.

Child circuits are generated by random applying of the above described operations on the parent circuit. Every operation has a given probability, so the most frequent operation is a change of a single element parameter or a change of an element for a single edge or vertex. After one operation is selected, appropriate vertices or edges needed by the operation are selected randomly, the operation is performed and appropriate elements are changed. The generated circuit is evaluated then and it is accepted or rejected for the following iteration. If it is rejected, the next operation is performed on the parent circuit.

A circuit is accepted with the following probability: (simulated annealing or stochastic hill-climber criterion, see [4] for details)

$$p_{accept} = \frac{1}{1 + e^{eval(s) - eval(s')/T}} \qquad (1)$$

where *eval(s')* is the RMS error of the new generated solution, *eval(s)* is the RMS error of the parent solution, *T* is a parameter called temperature. The basic idea of this criterion is to accept also not too bad solutions with changing probability according to some "cooling diagram".

Because the algorithm can struggle in some local optimal solution, where all neighbour solutions are worse, there has to be a technique to get out of this point. There are two basic methods: The first one restarts the search process after a local optimum (or the maximum number of iteration) is reached. The second one tries to avoid staying in some local optimum point by randomization of the search (e.g. random walk techniques). In our algorithm we use the first method, but we do not restart from the beginning of the search. After a certain amount of iterations, the solution is memorized and compared to the previous memorized one. If the solution is not much better (e.g. the RMS error of the solution is not 10% or more lower than the RMS error of the previous solution), the algorithm jumps back to the previous memorized solution. Each such memorized solution has also included a counter of these back jumps to it. If this counter exceeds some value, the algorithm does not jump back to it, but to its memorized ancestor. In some sense, this approach mimics a backtracking search a bit.

## PRACTICAL RESULTS

The above described algorithm was implemented in Java. It consists of several modules, describing available elements and possible operations. New operations as well as elements can be added easily.

First design example is a band pass filter with the following design goals:

- Between 1 GHz and 3 GHz transmission below -25 dB
- Passband transmission above -1.5 dB between 5.5 GHz and 6.5 GHz
- Stopband attenuation between 9.225 GHz and 12 GHz below -35 dB

A complete solution (with RMS error equal to zero) was found after 4058 seconds, measured on Pentium III 1GHz, 512 MB RAM, JRE 1.4.0. Program MIDE [5] (circuit analysis) was called 48513 times and its execution takes 88% of all time spent. Generated circuit consisted of 27 elements. Resultant characteristics are given on the following graph (Fig. 2.).

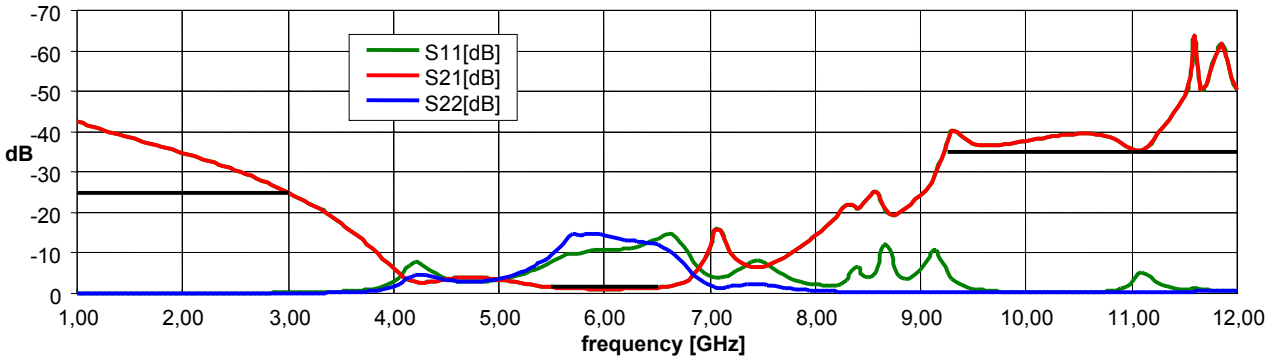


Fig.2. Characteristics of resultant circuit (band pass filter)

Second example is an amplifier with the following requirements:

- Transmission at 1.7 GHz is above 17 dB
- Reflection both on input and output gate is less then -40 dB at frequency 1.7 GHz
- Use of a MGF1801 transistor

A complete solution (with RMS error equal to zero) was found after 163 seconds, measured on Pentium III 1GHz, 512 MB RAM, JRE 1.4.0. Program MIDE (analysis of circuit) was called aprox. 5396 times and its execution takes 84% of all time spent. Generated circuit contained 23 elements. Transistor MGF1801 was used thee times. Resultant characteristics are given on the following graph (Fig. 3.).

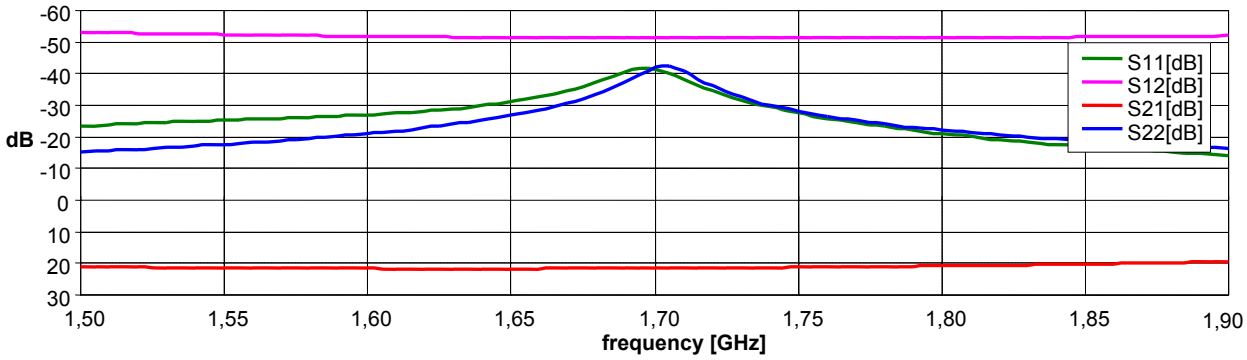


Fig.3. Characteristics of resultant circuit (amplifier)

## CONCLUSION

We presented a promising evolutionary algorithm for microwave circuit design.. The main advantage of this algorithm is to point out a possibility of use of evolutionary algorithms in the area of circuit design. The implemented program can give solutions on rather complicated requirements without any knowledge of traditional design strategies and algorithms.

Because the weakest point of the above presented algorithm is probably the high number of required circuit analysis (more than 80% of all time is spent on analysis), future improvements should go for better selection of operations, their operands (selection of vertices and edges, where the operations take place), elements and theirs parameters. For example a nice feature would be the estimation of successfulness of an operation according to its previous use.

## ACKNOWLEDGEMENTS

This research and publication have been sponsored by Czech Grant Agency, contracts no. 102/01/0571 and 102/01/0573, and by Czech Ministry of Education in the frame of project MSM 210000015.

## REFERENCES

[1] T. Bäck, F. Hoffmeister, H. P. Schwefel. *A survey of evolution strategies.* Proc. of the Fourth International Conference on Genetic Algorithms and their Applications, San Diego, California, USA, 1991.
[2] J.-M. Labat, L. Mynard, *Oscillation, Heuristic Ordering and Pruning in Neighborhood Search*. In Proceedings of CP'97, G. Smolka ed., LNCS 1330, pp. 506-518, Schloss Hagenberg, Austria, Springer, 1997.
[3] K. Marriot, P. J. Stuckey. *Programming with Constraints: An Introduction*. The MIT Press, 1998
[4] Z. Michalewicz, D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2000
[5] Z. Škvor, K. Hoffmann, J. Tomandl, Z.Medek. *Computer aided design of radiofrequency and microwave circuits.* Radioengineering, 2 (1993) 1, pp. 2-5
[6] D. Whitley. *A Genetic Algorithm Tutorial.* Technical report, Computer Science Department, Colorado State University, 1997.
[7] A. H. Wright. *Genetic algorithms for real parameter optimization.* Foundations of Genetic Algorithms, pages 205--218. Morgan Kaufmann Publishers, San Mateo, CA, 1991.