# Scalable Parallel Optimisation Using Fast Messy Genetic Algorithm

Pavel Kostka, Zbynek Skvor

Department of electromagnetic field, FEE CTU in Prague, Technicka 2, 166 27 Praha 6

Phone: +420 2 2435 5966, E-mail: xkostka@fel.cvut.cz, skvor@fel.cvut.cz

*Abstract* **Fast messy genetic optimisation is found suitable for complex microwave circuit design. Increase in computation speed is achieved using several ordinary computers connected to a network. Calculations are running on background so that computers can be used for other purposes at the same time. When applied to microwave circuits, a modification to previous genetic optimisation methods proved suitable. Dynamic change of bounds has significantly improved convergence rate.**

***Key words:*** **microwave circuit optimisation, genetic, parallel**

## 1. Introduction

Circuit optimisation became one of the applications most used by microwave engineers. Due to the nature of microwave circuits, the optimisation task is far from easy. Error functions mostly suffer from local minimum problems, so that in many cases the optimisation gets unbelievably inefficient.

During larger microwave structure optimisation local iteration methods tend to fall into local minimums. Hence, the new - global - methods like simultaneous annealing, taboo method or genetic algorithms are being used lately. The more variables we try to tune, the bigger state area gets and the optimisation on a single machine becomes rather slow. For overall speedup, several computers connected to each other are used.

The genetic algorithm (GA) is a stochastic global search method that mimics the metaphor of natural biological evolution. GAs operate on a population of potential solutions applying the principle of survival of the fittest to produce (hopefully) better and better approximations to a solution.

## 2. Fast Messy Genetic Algorithm (fmGA)

Fast messy genetic algorithm is a special clone of common simple genetic algorithm (GA). This type represents new, more powerful kind in the GA branch. It resists premature local-minimum fall and solves problems in shorter time.

A gene is represented by a pair *(allele locus, allele value)* in messy algorithms, so that – for instance – chromosome (2,0),(0,1),(1,1) represents 3 bit-long chromosome 110. Operation cut and splice are used to bread new offspring. Incomplete specification of the chromosome (underspecification) or redundant specification (overspecification) could occur during evolution. Overspecification is solved by right-to-left scan, i.e. the only first occurence of an appropriate pair is taken into account. For example: ((0,0), (2,1), (1,0), (2,0)) represents chromosome 001. On the other hand, underspecification is harder to deal with. We must complete the chromosome in order to be able to evaluate its fitness function (FF). Hence a template is used – see Fig. 1.

The algorithm works in two main iteration cycles – in the inner and outer loops. Each new start of outer loop is called as an era. The change of building-block order together with inner cycle launching is the main aim of this cycle.

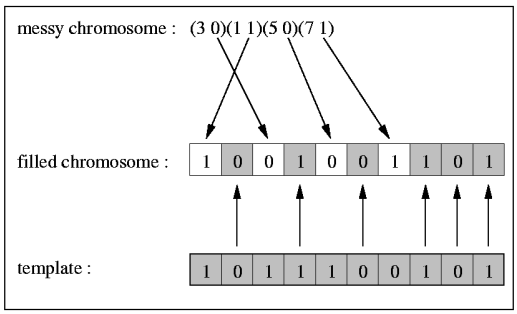


**Fig. 1** Representation of a solution – chromosome.

Inner cycle consists of three phases:

1) initialisation
2) building block filtering phase
3) juxtapositional phase

Detailed description of the algorithm could be found in [1], [3] and [4].

## 3. Improvements

We have decided to use this algorithm for microwave circuit optimisation. In order to save time, circuit analysis and Fitness Function (FF) evaluation has been carried out using the source code of a microwave CAD program called MIDE (see [2]). Implemented algorithm is based on [1] with the following important modifications:

*Parallel multitasking*

The bottleneck of any common algorithm is in the time consumption of FF evaluation – even a simplest circuit gets evaluated hundreds to thousand times per second. If one uses almost any global search method (and GAs specially) hundreds of thousands evaluations are required. The local area network has been used and the task has been divided to many common workstations. The application has been running under multitasking environment so that computers could be used for ordinary tasks like writing reports or playing games. At no extra cost, performance exceeding any single PC has been made available.

*Implementation*
Parallel computation machine was implemented using common client-server model. The first idea led to launching single fmGA algorithm on the server and then distribute appropriate variables together with optimising circuit to the clients. Clients then perform evaluation (compute the solution) and return result to the server. In order to minimize the total network overhead we suggested batch processing (a bundle of waiting states are together send to the client and then – after evaluation – the same bundle of solutions are returned).

*Dynamical change of bounds*
As mentioned, GA is global search method. Its success and convergence rate to global (or at least usable local) minimum depends on suitable bounds. Logically – the wider bounds are set, the lower is the probability of right minimum discovery. From experiments results that while the approaching to a neighbourhood of a minimum takes relatively short time, falling into minimum takes at least same time or longer. This is mostly seen as a bottleneck of GA – but we have used this fact to improve robustness of global optimum search. To achieve that, we have implemented dynamical change of bounds depending on the actual position in the space state.

*A sketch of the procedure follows:*
a. look for a solution in the state space
b. if the FF doesn't improve for longer time, try to adjust bounds so that the so-far found solution would lie in the centre; modify the bounds in order to be the half of the original range
c. perform operations a-b until
    i. Solution is found
    ii. So-far found solution is out of new set range
d. if no solution in a-b-c was found, return to point a and look for another solution

## 4. Results
As an example that is easy to understand but not rather easy to optimise, we chose a textbook optimisation problem, see Fig. 2.

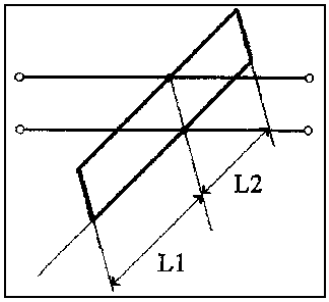


**Fig. 2** Test case – two shorted stubs in parallel to a line.

*The task follows:*
Find electrical lengths L1 and L2 of two short-ended stubs, connected to a transmission line in parallel, so that the transmission coefficient would be lower than –25 dB between frequencies 6 GHz and 6.2 GHz. Simultaneously we want to achieve transmission better than -0.5 dB at 3 GHz. These demands are schematically shown in Fig. 4.
In the case the lengths are to be found in a range between 1 and 180 degrees for both stubs, the task is quite easy and any optimisation method works. To obtain a good test case, we try to find the optimum in a wider range, say 1 – 1000 degrees. FF shape then exhibits a number of local minima. The reason is obvious – electrical character of both stubs is repeated every 180 deg. Global minimum discovery in such selected area is almost impossible for common optimisation methods – see Fig. 3. Typical simplex methods will mostly find a local minimum, however, FF is always bigger than 0 (not all goals are met). A few examples of such solutions are shown in Fig.3.

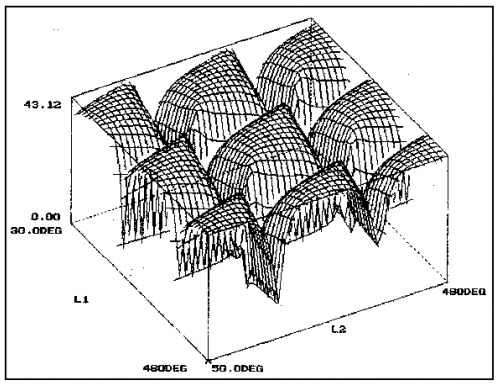


**Fig. 3** Fitness function surface plot shows a number of local optima.

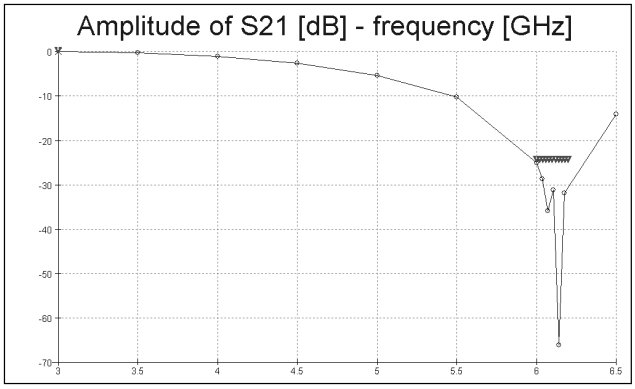


**Fig. 4:** Forward transmission for correct solution (approx 175 and 178 degrees @ 6GHz).

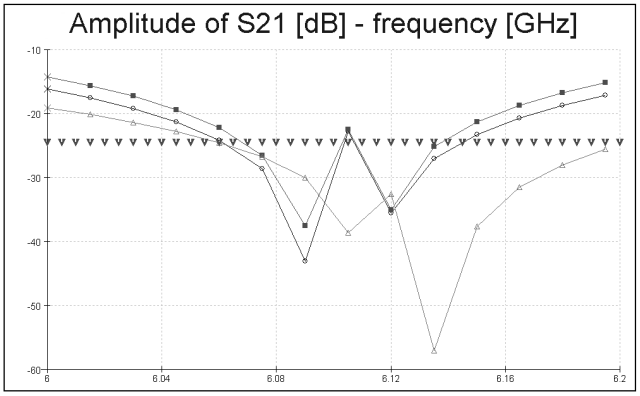


**Fig. 5** Some of the false solutions produced by ordinary optimisation methods.

Using the new method, we started from the same point (we filled up variables with the same start values similarly to the above mentioned optimisation method) and carried out several optimisation runs. In all runs the global minimum has been found.

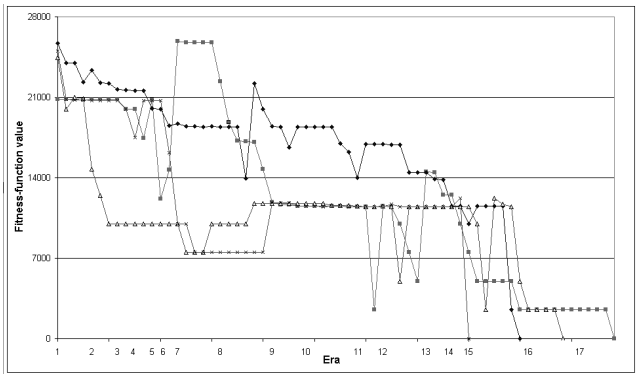


**Fig. 6** Fitness function plotted against eras.

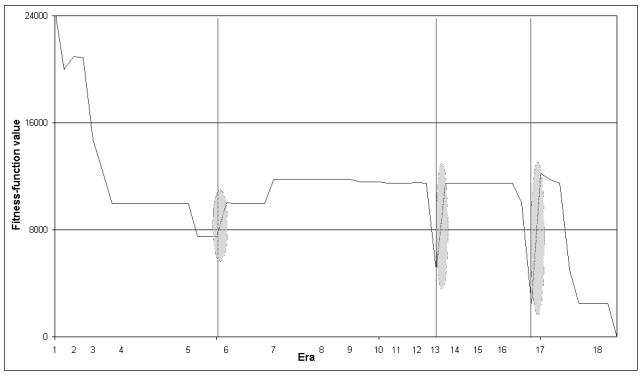


**Fig. 7** A detailed sketch of solution evolution describing method behaviour in local optima.

The process is illustrated at Fig. 6. Notice one important feature: during the launching of a new era, the best so-far found chromosome is copied into template. It results in temporary degradation of FF (you can see bumps in the graph), however, in a few moments a new – often better - solution is found. Such iterative progress is typical for messy algorithms.

Figure 7 shows effect of dynamical change of bounds. It arises at the end of 6th, 12th and 17th era (vertical red lines). Please note, that there are significant temporary FF degradations, but the local minima are left (red ellipses).

## 5. Outlook

Unfortunately, even the best found ratio between block size (which was sent to clients for processing) and time spent for comunication shown that interconnected communication overhead took approximately 15-20% of computational time. That was the reason why a new method was proposed.

*Parallel fmGA server*

The new method combines both mentioned improvements together: clients no more behave only as silly slaves (whose only task is fitness function evaluation) but each of them runs its own fmGA algorithm. Server side must solve the area division problem. Server can than send to each client only some sub-bounds of original problem. Each client receives different bounds and thus totally independently tries to discover solution(s) in its own sub-area.

*Area division problem*

Division of an 1-D area is obviously the simplest task – see Fig. 8. Appropriate areas are a little bit overlapped. It was found useful in cases when the right solution lay near the border of a sub-range.

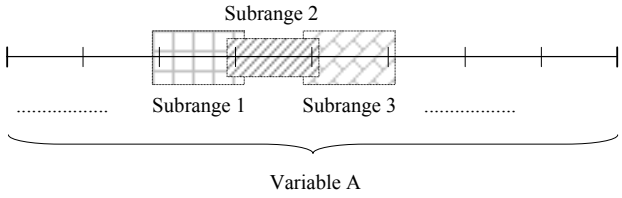


**Fig. 8** Example of division and overlapping in 1-D area.

Situation appears rather harder for n-dimensional area (where N>1). Of course, we can perform the same division, but the less clients we have the coarser the division is (especially in case of many variables with large bound limits). A right solution can be discovered with much less probability. Keeping the uniform division step for all variables is a better technique, however we must cope with the situation that we create more intervals than cooperating clients. The above mentioned reasons made us to implement so-called *computational stack*, i.e. a structure that temporarily saves areas which wait for exploration – see Fig. 8 and 9.

As soon as the stack is empty (and no suitable solution has been found), results from all sub-areas are compared together and area bounds are changed so that they more tight surround so-far best found solution (i.e. dynamical change of bounds). If there are more than one area, where partial results are comparable, all of these areas are processed – one

is explored and the others are saved into stack for later exploration.

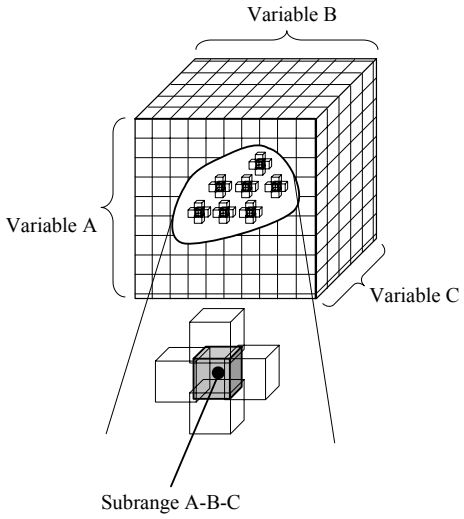


**Fig. 9** Example of division and overlapping in 3-D area.

*Gradient method incorporation*

First experiments of suggested method shown that time spent on discovering near neighbourhood of FF minimum and time spent on falling into the minimum are for algorithm fmGA almost comparable. I.e. the algorithm is capable of discovering the surrounding of solution in quite large search space, but at the present there exist better algorithms (typically gradient methods) for quick discovering of *FF - peek*.

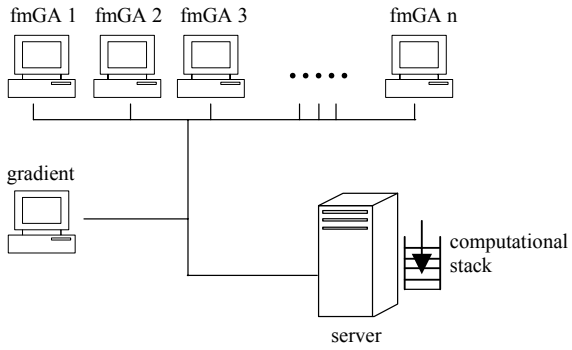


**Fig. 10** Sketch of parallel computation engine.

## 6. Conclusions

Fast messy genetic optimisation is a promissive method found suitable for complex microwave circuit design. High computational load can be overcome at nearly no cost using several ordinary computers connected to a network. Running calculations on background enables other users to do their jobs at the same time.

A modification to previous genetic optimisation methods, suitable for microwave circuits, proved to work and significantly improved convergence rate.

**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] Knjazew, D.: Application of the Fast Messy Genetic Algorithm to Permutation and Scheduling Problems, Illigal Report 2000022, University of Illinois, May 2000, available also at http://gal4.ge.uiuc.edu/pub/

[2] Skvor, Z.: CAD pro vf. techniku, textbook, Czech Technical University, Prague 1998, http://www.mide.cz

[3] Goldberg, E. – Deb, K. – Kargupta, H. – Harik, G.: Rapid, Accurate Optimisation of Difficult Problems Using Fast Messy Genetic Algorithms, Illigal Report 93004, University of Illinois, February 1993

[4] Pelikan M. – Sastry, K. – Goldberg, D.: Evolutionary Algorithms + Graphical Models = Scalable Black-box Optimisation, Illigal Report 2001029, University of Illinois, November 2001