Microwave-Circuit Optimization with Parallel Enhanced Fast Messy Genetic Algorithm (pefmGA)

Pavel KOSTKA, Zbyněk ŠKVOR

Dept. of Electromagnetic Field, Czech Technical University, Technická 2, 166 27 Praha, Czech Republic

xkostka@fel.cvut.cz, skvor@fel.cvut.cz

Abstract. *Fast messy genetic optimization 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. Dynamic change of bounds, search space segmentation and gradient incorporation have significantly improved convergence rate. The new method has found global minimum in each run, while classic methods failed for some starting points.*

Keywords

Optimization, genetic optimization, messy, radiofrequency circuits.

1. Introduction

Microwave circuit design relies on Computer Aided Design (CAD) tools. Contemporary software packages allow optimization of circuit performance using so-called standard methods. However powerful, these methods require a good estimation of starting point, and provide for optimization in narrow variable bounds. What's more, the higher frequency is, the more parasitic elements must be taken into account.

As an example, consider an equivalent circuit of a microwave transistor BF405 (produced by Infineon) – see Fig. 1. Sub-circuit bounded by A-B-C points stands for just pure chip and a few parasitic inductances and capacitances that correspond to 3 bond wires between the chip and pads of transistor box.

Such circuit is fully sufficient at low frequencies and can as such be used as a model in Spice® or other simulators. Unfortunately, with increasing frequencies the model doesn't seem so ideal – one needs to include much more parasitic effects. The whole Fig. 1 shows the equivalent circuit usable for frequencies above approx. 1,07 GHz (phase error > 5°, amplitude error > 0,05). As you can see, there are 12 another parasitic elements. In order to extract equivalent circuit parameters from measured transistor characteristics, all of these are expected to be the subject of further optimization process.

Efficiency of common types of algorithms falls rapidly down with the rising size of such search space (looser variable bounds). Many of them are almost incapable to find an acceptable solution in reasonable time.

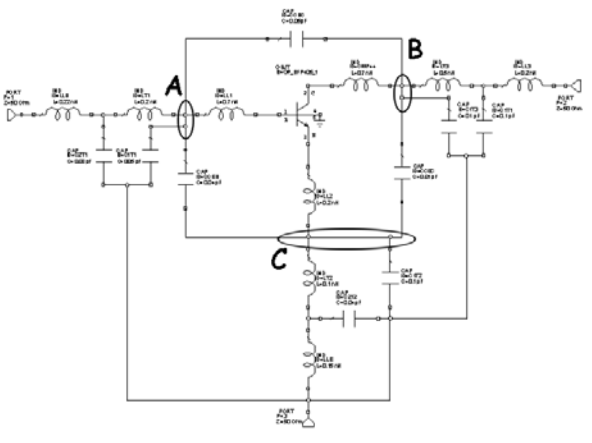

Fig. 1. Equivalent circuit of microwave transistor BF405

The above mentioned arguments led designers to the new groups of algorithms, such as neural networks, ant colony systems or genetic algorithms (GAs). The last group – GAs – went through the stormy development in the last thirty years.

Goldberg laid the foundation of this scientific field in 1989 [5]. He introduced evolutionary strategy, population breading and basic genetic cycle. All together this is called simple genetic algorithm (sGA).

sGA is nowadays widely spread and many scientists used it for solving general as well as specific types of optimization problems. However effective sGA is, it suffers from many lacks. Poor defense against local minimum fall and unconstrained number of fitness evaluation cycles are the most serious ones. Parallel enhanced fast messy genetic algorithm (pefmGA) sufficiently deals with these disadvantages and altogether with parallel processing it is capable of enlarging the search space many times in comparison with common sGA.

2. Algorithm Description

The basic idea comes from the works involving messy and fast messy GA [3,5] and from the project of searching extraterrestrial intelligence (seti@home) [6]. The algorithm has been enriched with some features that significantly improve convergence and extend the search space. Common personal computers at network have been used to distribute computation so that whole process would be faster.

2.1 Messy GAs

All messy GA (mGA) clones incorporate so-called *messy representation* of chromosomes. Gene position is no longer fixed – mGA relaxes such interconnection. In order to identify a particular gene we denote it by a pair of (allele, locus). Locus is simply the position of allele within chromosome and allele represents value (typically binary digit 0 or 1). For example, chromosome 1011 can be re-typed into messy code as four-pair string ((1,1) (2,0) (3,1) (4,1)) or equally according to the Fig. 2.

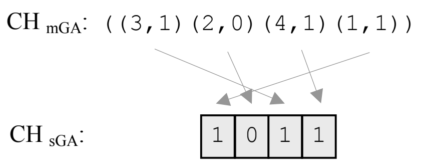

Fig. 2. Messy representation of a chromosome

Common *crossover* operation was replaced by a couple *cut* & *splice*. Even one can guess that these operations do the same as crossover does in sGA, it's a little bit more complicated. First, each operation can occur during the iteration cycle only with the some probability: each chromosome starts with a length λ, where λ<l (see section 2.3) so that at the beginning it is more probable that *splice* will occur; as soon as lengths of strings reach a constant *cut,* will be engaged. Second, these operations can produce chromosomes of different length. Fig. 3 just describes such case – *chrom A* consists of three genes, *chrom B* of nine genes. The first one is too short and let's say, that *cut* will not occur; the latter is (again – let's say) long enough and operation *cut* will occur. *Splice* operation produces offspring: the1st one consists of whole *chrom A* and the left half of *chrom B*, the second is only a copy of *chrom B*.

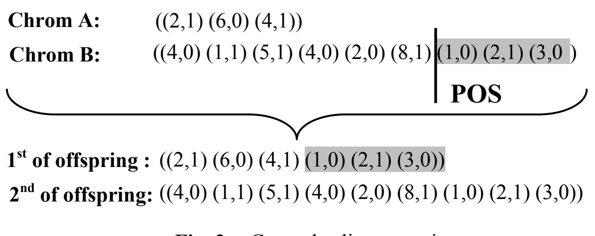
Chrom A: ((2,1) (6,0) (4,1))
Chrom B: ((4,0) (1,1) (5,1) (4,0) (2,0) (8,1) (1,0) (2,1) (3,0)
POS
1st of offspring : ((2,1) (6,0) (4,1) (1,0) (2,1) (3,0))
2nd of offspring: ((4,0) (1,1) (5,1) (4,0) (2,0) (8,1) (1,0) (2,1) (3,0))

Fig. 3. Cut and splice operation

Fig. 4 proves the prime advantage of such representation (reprinted from [3]). Let us imagine a building block [4] 1***1. Suppose that left and right ones define some good property of an individual. Comparing sGA to mGA operations one can see that: while one-point crossover disrupts the block, mGA preserves it.

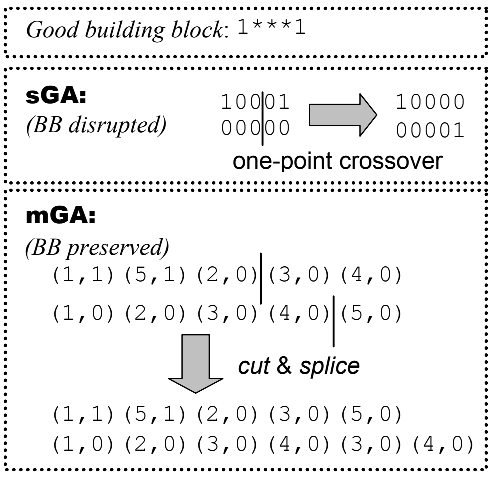

Fig. 4. Demonstration of messy representation advantage

2.2 Template Deployment

Relaxed locus brings another new feature: some genes in a chromosome can be included more than once (over-specification) and another not at all (under-specification). Over-specification is managed by left-to-right scan so that the first element is taken into account and others are ignored. Under-specification is rather worse: missing genes must be supplied from another source. We can generate it at random or – and it's much better – fill it out from a template.

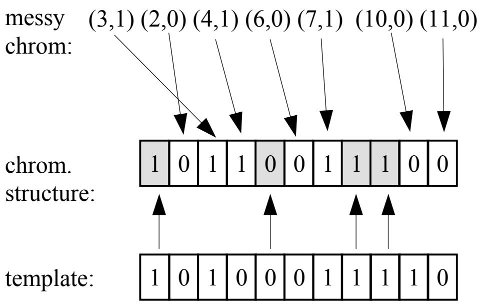

Fig. 5. Under- and over-specification

2.3 FmGA Sketch

Fig. 6 presents algorithm iteration loop. It consists of inner and outer loops. The inner loop performs two phases, so-called: primordial and juxtapositional. At the end of the primordial phase, the best-found individual is preset as the template for the next lap.

Probabilistically complete initialization [3] is done during primordial stage: first of all, a population is generated at random; gradually deletion of some genes follows so that λ-long building blocks would be constructed (of l-long problem, where l>λ). Coefficient λ is increased with the rising number of generation, thus the amount of genes filled from the template goes down. Fig. 7 presents an example of such initialization (reprinted from [4]).

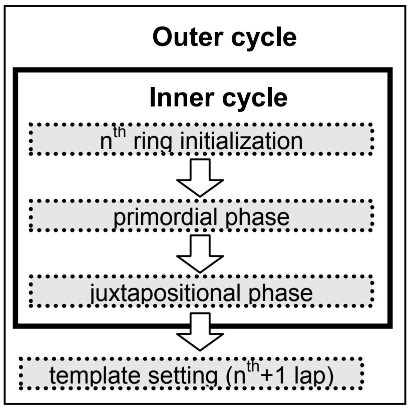

Fig. 6. Conception of mGA

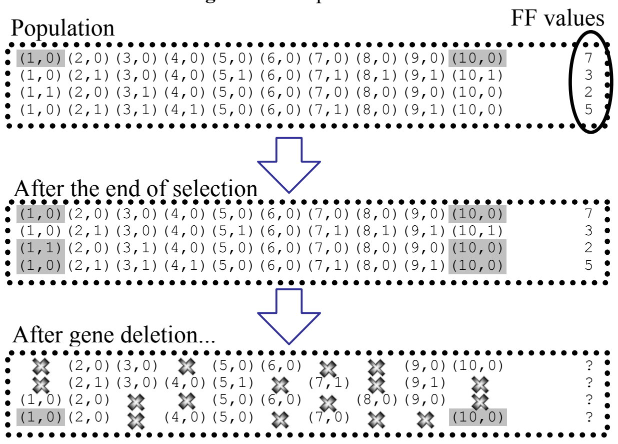

Fig. 7. Probabilistically complete initialization

Juxtapositional phase iterates with *cut* & *splice* operations, mutation (that's the same as in sGA is) and selection and fitness function (FF) evaluation. Selection (tournament) is enriched with thresholding idea: two chromosomes can fight with each other only if they have some amount of genes in common. Random selection is used otherwise.

2.4 Conception of pefmGA

Four important features have been added to fmGA algorithm to speed up and sharpen convergence:

(a) search space segmentation,
(b) mapping with dynamically changed bounds,
(c) parallel processing,
(d) gradient method incorporation.

2.5 Search Space Segmentation

Even the best algorithm has upper bounded search space so that above such limits a (global) solution can't reliably find. If the search space consists of n-dimensional (n-D) cube (where n- is number of variables) we can split the cube into some amount of smaller n-D sub-cubes so that they all together make up the original space. Fig. 8 illustrates a very easy example of 1-D area. Note that subintervals are slightly overlapped. It helps to find a solution located near their borders.

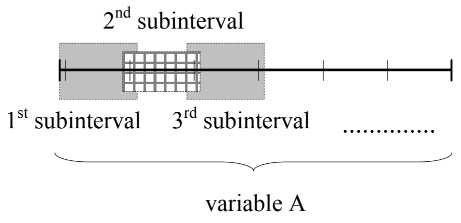

Fig. 8. 1-D search space division

The next figure (Fig. 9) then mentions the same procedure in 3-D space.

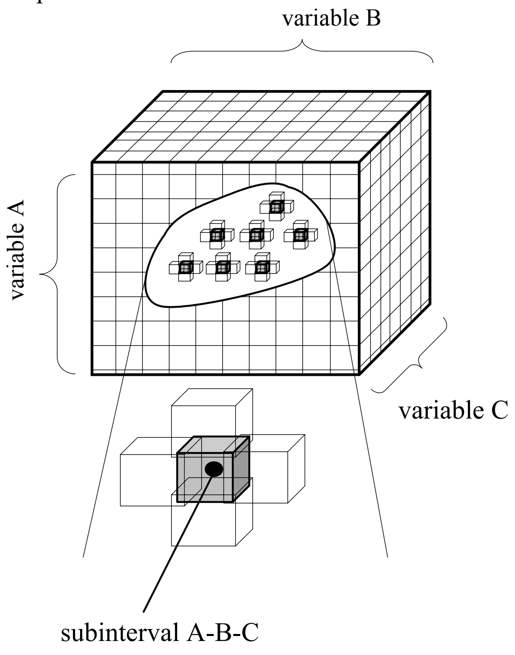

Fig. 9. 3-D search space division

2.6 Mapping & Dynamically Changed Bounds

Implementation of any genetic algorithm brings one more fundamental issue – the number of bits per one variable. It's obvious that the more bits (alleles per gene) we take, the finer granularity we obtain and the better chance to discover a solution we have. On the other hand, the greater a chromosome is, the larger the population must be and the more time its evaluation will take. What's more, however fine the granularity is, it can't ensure that the right solution might be discovered (it can simply lie between two samples).

These reasons were the main motivation to re-make the representation of variable in chromosome. Imagine some fixed length of alleles per variable independently on the number of its bound. Let's denote the minimum of the bound as MIN_G and maximum as MAX_G (watch Fig. 10). After the first iteration, a minimum "A" is discovered. The change of interval follows so that the new one (specified by MIN_N and MAX_N) is half of the original one and minimum A lies in its center.

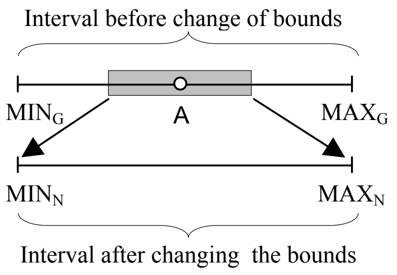
Interval before change of bounds
A
MIN_G
MAX_G
MIN_N
MAX_N
Interval after changing the bounds

Fig. 10. Change of bounds

Such technique can either converge to the solution or diverge. Typical branch and bound algorithm in case of divergence executes *rollback* operation. Nevertheless our goal is not to walk through search space in this way – that's the GA business. Such changes are performed only few times and solely in the second half of outer cycle at the moment, if no important FF improvement for longer time occurs.

2.7 Parallel Processing

Our parallel engine is based on traditional client-server architecture – see Fig.11. Server starts the whole optimization by splitting n-D search space into particular subspaces and subsequently saving them into computation stack. Then it informs efmGA clients about the beginning process and waits for their initialization. After a while, it distributes particular subspaces among clients, launches computations and collects results. Server does it until it gets stack empty. If some client doesn't respond within a timeout, its task is redistributed to another one, so that integrity is preserved.

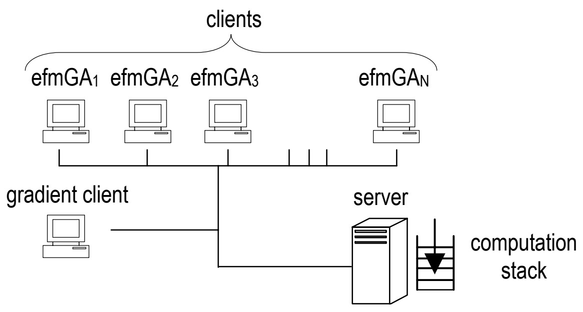

Fig. 11. Conception of pefmGA

2.8 Gradient Incorporation

Experiments with GA methods have shown an interesting fact (see Fig 12): the time needed for discovery of an estimated position of global minimum of FF (i.e. movement from point A to point B) for let's 1-D array (variable x) is fully comparable with the time, that GA needs to fall into point C (i.e. movement from point B to point C). While the coarse tuning of FF (A→B) is the typical GA task, the fine tuning (B→C) can be much more efficiently done by some gradient method (provided starting point were correctly set). This idea led to the implementation of a gradient client (see Fig. 11), which performs fine optimizations of pefmGA results.

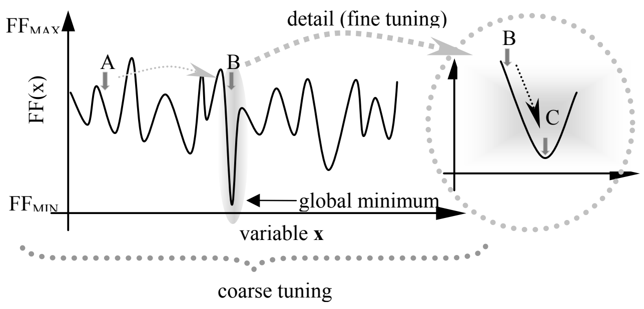

Fig. 12. Gradient incorporation

3. Results

3.1 Benchmark Task

On the contrary to common optimization, where purely defined NP-hard problems (for example TSP) are used as standardized benchmarks, microwave optimization field hasn't established such tasks yet. The presented task is one of tricky ones and (as will be demonstrated further) almost all "classic" methods are running into problems if used to find its suitable solution within wider bounds.

3-D vertical stub, positioned on a microstrip line was published in [7] – see Fig 13. Such a stub is nowadays widely used as capacitor, especially as a DC block.

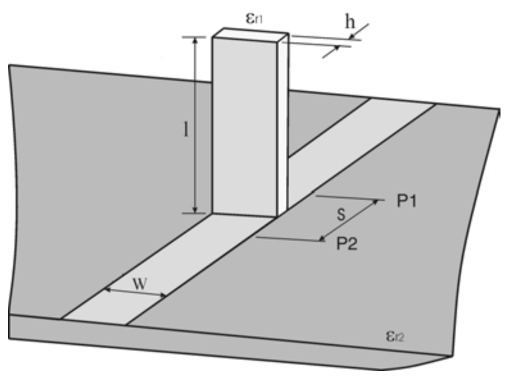

Fig. 13. Radial stub

Detail structure analysis leads to the decomposition into even and odd modes. Let's consider just the odd mode equivalent circuit, as shown in Fig 14. There are 8 parameters (even mode has up to 9 parameters), whereas TL_2 represents transmission line with nonlinear dependence of ε_R and Z_0.

The subtraction of measured and computed matrixes of circuit scattering parameters is the ordinary method of fitting values to some measured data (based on prior knowledge of equivalent circuit topology). Circuit elements are optimized in order to minimize difference between measured and calculated values. The same method has been used within this benchmark test. That means that zero amplitude (and phase) indicates exact matching. Practically used circuits don't need absolute match correspondence but it must be better than a value.

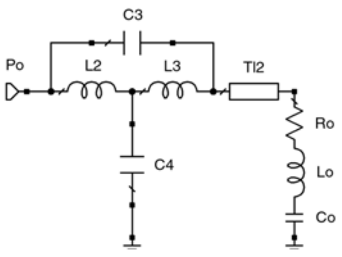

Fig. 14. Odd mode of radial stub decomposition

To better authors' published results was obviously our first goal. Fig. 15 compares the amplitude of original results (sGA optimization, Gaussian mutation – 20%, 60 individuals in population size, Microwave office®) and pefmGA results. The bounds were deliberately set very strictly to prevent from premature finish of optimization. Shadow triangles graphically present the goals.

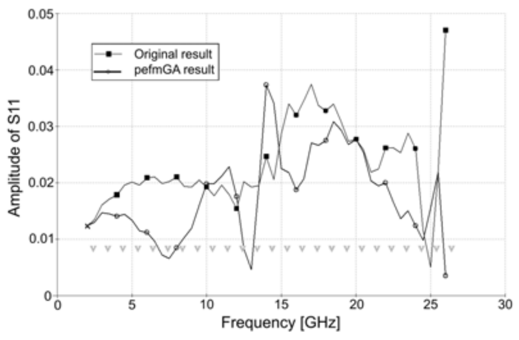

Fig. 15. Comparision of sGA and pefmGA results

3.2 Algorithm Comparison

Although the previous graph exposed that pefmGA achieved better results, one must perform a little bit more sophisticated confrontation to prove that pefmGA is more effective than others. Let's try optimization with the most often used methods (Tab. 1 sums them up).

	Abbreviation
Powell	pow
variable metrics	varMet
steepest descent	stDesc
Box	box
sGA	sGA
efmGA	efmGA
efmGA+gradient method	efmGA+grad

Tab. 1. Methods used for comparison

Almost all of these methods are described in [8]. We have selected wide optimization bounds for each variable so that many local minima are presented in search space. Tab. 2 illustrates chosen bounds.

L_2	50p ~ 500p	TL_2	2.5m ~ 4m
C_4	100f ~ 1p	R_0	10 ~ 100
C_3	10f ~ 300f	L_0	100p ~ 2n
L_3	10p ~ 300p	C_0	2f ~ 20f

Tab. 2. Bounds of variables

Each method has started five times. Tab. 3 sums summarized obtained best and average results. Time specification (specified in seconds) depends on the processor performance (these ones corresponds to AMD-KII@800MHz, 2507MIPS, 1003MFLOPS) and is mentioned rather for comparision.

	Best simulation			Average simulation		
Method	FF eval	time	FF	FF eval	time	FF
pow	405	5	14.69	405	5	14.69
varMet	181	1	14.69	181	1	14.69
stDesc	15361	90	14.67	15361	90	14.67
box	11715	52	0.050	11425	64	0.070
sGA	22135	31	0.055	25024	40	0.060
efmGA	40000	120	0.339	40000	120	0.683
efmGA+grad	42402	125	0.054	43485	127	0.065

Tab. 3. First interval results

Fig. 16 compares the results. Methods pow, varMet and stDesc are not plotted because they were not able to improve starting FF value at all. As you can see, raw efmGA results seem a little bit worse than others (although the comparison of FF (box=0.05, efmGA=0.339 versus stDesc=14.67) doesn't show out such dramatic difference). That's the reason why gradient method has been incorporated.

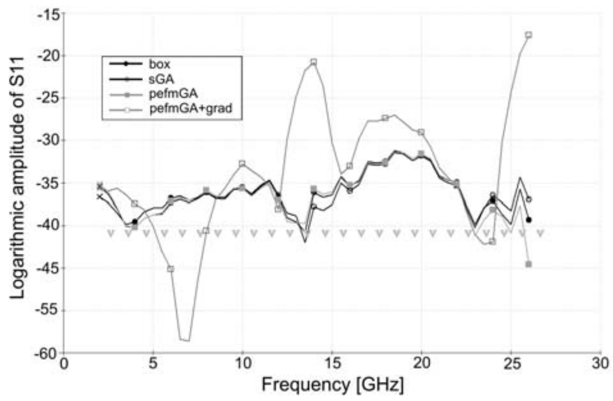

Fig. 16. Graphical comparison

To be able to demonstrate the power of the presented algorithm, substantially wider search space has been used (see Tab. 4). The first four variable bounds have been enlarged – each by two decades.

L_2	50p ~ 50000p	TL_2	2.5m ~ 4m
C_4	100f ~ 100p	R_0	10 ~ 100
C_3	10f ~ 30000f	L_0	100p ~ 2n
L_3	10p ~ 30000p	C_0	2f ~ 20f

Tab.4. Bounds of variables

Tab. 5 now shows that pefmGA discovered much better results than any other methods (box=7.24, efmGA=4.07). As you can also note, gradient influence diminished.

	Best simulation			Average simulation		
Method	FF eval	time	FF	FF eval	time	FF
box	13160	480	7.24	11446	220	9.86
sGA	22356	310	7.35	19580	130	10.19
efmGA	40000	120	4.07	40000	120	4.40
efmGA+grad	40205	129	4.07	40350	132	4.32

Tab. 5. Second interval results

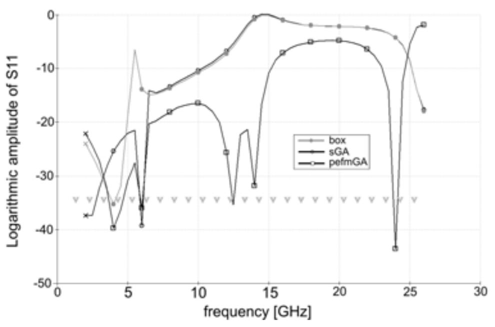

Fig. 17. Graphical comparison

3.3 Parallel Acceleration

Each parallel computation machine (regardless of utilization) can be characterized by plenty of operating parameters. Parallel acceleration is one of the most important ones (it corresponds to efficiency). Values near to one means that the incorporation for example of 10 clients will speed up equivalent sequentially process by 10 times (linear acceleration). However this value is rather theoretical because of communication expense and many others factors. Efficiency rapidly falls with lowering this value. Fig. 18 demonstrates that proposed conception and communication protocol are efficient even for huge subspace segmentations.

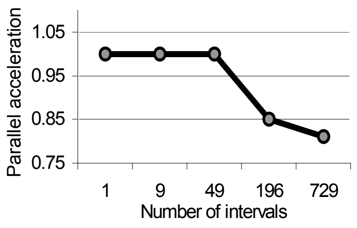

Fig. 18. Parallel acceleration

4. Conclusion

Parallel enhanced fast messy genetic optimization is a promising 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 enable other users to do their jobs at the same time.

Acknowledgements

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

[2] SKVOR, Z. *CAD pro vf. techniku.* Czech Technical University, Prague, 1998.

[3] GOLDBERG, E., DEB, K., KARGUPTA, H., HARIK, G. Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. Illigal Report 93004, University of Illinois, February 1993.

[4] GOLDBERG, D.E. *Genetic algorithms in search, optimization and machine learning,* Addison Wesley Publishing Company, January 1989.

[5] DEB, K. Binary and Floating-point Function Optimization using Messy Genetic Algorithms. Illigal Report No. 91004.

[6] The search for Extraterrestrial Intelligence, http://setiathome.ssl.berkley.edu

[7] HOFFMANN, K., SOKOL, V. Analysis of 3D Vertical Strip on Microstrip Line. In *Proceedings of the European Microwave Conference,* Vol. 2, pp. 569-572, Milan 2002.

[8] PRESS, H.W., TEUKOLSKY, S.A., VETTERLING, W.T., FLANNERY, B.T. *Numerical recepies in C.* Cambridge University Press, ISBN 0-511-43108-5.

About Authors...

Pavel KOSTKA graduated from CTU in Prague in 2000. Optimization of microwave structures, parallel computation and algorithm-migration processes are his main points of interest.

Zbyněk Škvor graduated from CTU in Prague, 1985. He is with the Department of Electromagnetic Field, CTU Prague. His field of interest includes numerical electromagnetics, CAD for radio-frequency and microwave circuits and microwave measurements.