Multiobjective Optimization Techniques: A Study Of The Energy Minimization Method And Its Application To The Synthesis Of Ota Amplifiers

Milton Jonathan
ICA
Applied Computational Intelligence Laboratory
Electric Engineering Department
Catholic University of Rio de Janeiro
Rua Marques de S. Vicente, 225, Gavea,
Rio de Janeiro, RJ, Brazil, 22453-900
+55-21-529-9433
milton@ele.puc-rio.br

Marco Aurélio Cavalcanti Pacheco
ICA
Applied Computational Intelligence Laboratory
Electric Engineering Department
Catholic University of Rio de Janeiro
Rua Marques de S. Vicente, 225, Gavea,
Rio de Janeiro, RJ, Brazil, 22453-900
+55-21-529-9445
marco@ele.puc-rio.br

Ricardo Salem Zebulum
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91109
(818) 354-4321
ricardo@brain.jpl.nasa.gov

Marley B.R. Vellasco
ICA
Applied Computational Intelligence Laboratory
Electric Engineering Department
Catholic University of Rio de Janeiro
Rua Marques de S. Vicente, 225, Gavea,
Rio de Janeiro, RJ, Brazil, 22453-900
+55-21-529-9445
marley@ele.puc-rio.br

Abstract

This paper reviews the multiobjective fitness evaluation method called Energy Minimization [6,7,8] and presents an analysis of the method's behavior when used in a genetic algorithm applied to the synthesis of single-ended Miller operational amplifiers. A modified model is proposed in order to overcome some of the weaknesses pointed out and improve the model's performance. Finally, experimental results are presented and analyzed, leading to an overall evaluation of the benefits provided by the proposed modifications.

1 Introduction

In real-world problems, particularly in evolutionary electronics, it is often necessary to simultaneously optimize multiple performance measures, or multiple objectives. Unfortunately, some inconveniences frequently arise when known optimization techniques, both conventional and non-conventional, are applied to multiobjective problems. Indeed, these techniques are in general designed originally for single-objective problems, that is, problems in which the levels of optimality of the solutions can be given by the ordering of one single performance measure (e.g., a scalar value). The conventional *gradient descent* technique and also the widely used *simulated annealing* and *genetic algorithm* techniques are all examples of these, the latter one being the subject of this paper.

The actual problem is that, when multiple performance measures are necessary to rate a solution's optimality, there is no straightforward procedure to appropriately compare two distinct solutions. Thus, it becomes necessary to define a way in which the different evaluations can be combined so as to provide a correct decision about which solution is better, and how much better it is. Indeed, without a suitable workaround to this obstacle, none of the aforementioned techniques can work efficiently in multiobjective optimization problems.

0-7695-0762-X/00 $10.00 © 2000 IEEE

This article includes three additional sections. Section 2 reviews some multiobjective optimization techniques with emphasis on the energy minimization method, focus of this work. In section 3, we discuss the energy minimization method's behavior and propose modifications. Section 4 presents experimental results and finally in section 5 final considerations are discussed.

2 Review of Multiobjective Optimization Techniques

2.1 Linear Scalar Aggregation

The linear scalar-aggregative approach [3,4] corresponds to the most simple and direct method for combining the multiple performance measures. It consists simply of a *weighted sum* of the individual measures for each objective, with the final fitness evaluation F for a given solution being thus given by:

$$F = \sum_{i=1}^{n} w_i f_i \qquad (1)$$

where f_i corresponds to the fitness evaluation relative to objective i, and w_i corresponds to the respective weight, for a total of n objectives.

This method has the important advantage of being very easy to implement, aside from being extremely efficient computationally speaking. However, it also has a number of problems, one of the most serious of them being the great difficulty in choosing appropriate weights w_i for a given problem.

Indeed, for certain problems it can be considered that the different objectives all have the same importance and, in such cases, it becomes natural to choose equal values for the weights. However, in most situations the optimization of a particular objective may be more important than another one, or it can even be more complicated to define which objective has greater priority. Thus, in practice, the application of this method to a given problem almost always ends up in a costly and tiresome fine-tuning process, in which the algorithm is repeatedly tested with different sets of weights until satisfactory results are obtained.

2.2 Dominance and the Pareto-Optimal Set

The problem of comparing two different solutions may be partially solved by the concept of *dominance*. This concept states that a given solution v dominates another solution u only if for no objective the evaluation of v is worse than that of u. Moreover, for at least one objective the solution v must present a better evaluation than that of u [4]. Thus, formally, this dominance of v over u can be defined by:

$$\forall i \in \{1,...,n\}, v_i \geq u_i \cap \\ \exists i \in \{1,...,n\}, v_i > u_i \qquad (2)$$

where a hypothetical maximization problem of n objectives is considered, with v_i and u_i corresponding respectively to the evaluations of solutions v and u for objective i.

Obviously, we have that in particular a solution v dominates another solution u when it presents superior evaluations for all objectives. Aside from that, if a given solution is not dominated by any other, then that solution is said to belong to the *Pareto-optimal set*. Therefore, this set actually corresponds to all those solutions that, in the absence of any other information about the problem, cannot be surely stated as being inferior to anything: any other solution will always have an inferior evaluation for at least one objective.

Indeed, there are several optimization methods for multiobjective problems that seek the Pareto-optimal set, or *Pareto Frontier*. After all, it can be readily seen that the optimal solution to the problem will surely belong to this set, whatever solution that is. On the other hand, these techniques are in general quite costly computationally speaking, aside from the fact that they only partially solve the problem at hand: after all, of all solutions in the optimal set, which is the most desirable one?

2.3 Compromise Solutions and Distance-to-Target Techniques

In practice, for many real-world problems a good solution must necessarily satisfy all the objectives at hand to a minimum extent. In these situations, it is not considered acceptable for a given solution to present a spectacular performance for one objective at the same time it is a complete blunder for another. As an example of this, we can mention the typical problem of optimizing a product so as to minimize its cost and maximize its quality. In such a scenario, it is simply unacceptable to find a solution of extremely low cost but whose quality is mediocre. In other words, it is necessary to find a balanced solution, with the best possible compromise between low cost and high quality.

One way to achieve such a goal is to perform the evaluation of a given solution f by calculating the distance between the vector composed of the individual measures f_i and the target-vector *user* made of ideal evaluations for each objective [4]. Formally, such an evaluation method can be described by:

$$F = \left(\sum_{i=1}^{N} | user_i - f_i |^p \right)^{1/p} \quad p \geq 1 \qquad (3)$$

Thus, for *p=1* we have the so-called *Manhattan or metropolitan distance*, which actually consists of a simple linear aggregation of the objectives combined with a target solution [4]. With *p=2*, however, we obtain the more commonly used *Euclidean distance*. The non-linearity introduced by such an evaluation method prevents an improvement for a performance f_i from counteracting an equivalent worsening of another measure f_j. Effectively, the quadratic form causes the solution to be more penalized for a value f_j far from the target value $user_j$ than it is benefited for having another value f_i close to its target $user_i$. Thus, we can see that now there is a "pressure to compromise" so that it becomes harder for an unbalanced solution to be considered superior to a more compromising one.

In reality, the greater the value used for *p*, the greater will be the pressure exerted. In other words, larger values for *p* will increase the penalty given to those solutions with mediocre performance for an objective. Thus, priority will be given to those solutions that do not blunder in any aspect, even if they do not excel in any objective in particular. Aside from that, for the extreme case when $p \rightarrow \infty$, we obtain the technique known as *minimax* or *MinMax* [4], in which the evaluation of a given solution corresponds to the maximum distance of any of the *n* objectives relative to its target, that is:

$$F = \max(|\, user_i - f_i \,|) \qquad 1 \le i \le n \qquad (4)$$

This way, for a problem with two objectives the optimal solution will correspond to the exact intersection of the evaluation curves for each objective.

2.4 The Energy Minimization Method

The *energy minimization* method [6,7,8,9] attempts to solve the main inconvenience of most scalar aggregation techniques, which is the choice of the weights associated with each objective. Additionally, this method also incorporates the user's specifications, which is not trivially done with techniques that seek the Pareto-optimal set. The method, described below, is designed for use within a genetic algorithm. Its fundamental property is to adaptively update the weights throughout the evolutionary process so that greater priorities are constantly shifted to the objectives less satisfied by the population of solutions in general.

First, the linear scalar aggregation equation (1) is rewritten as follows:

$$F = \sum_{i=1}^{n} w_i Fnorm_i \qquad (5)$$

Here, a normalized fitness vector is used, *Fnorm*. The normalization is usually implemented by the following equation:

$$Fnorm_i = \frac{f_i}{\bar{f}_i} \qquad (6)$$

where the denominator represents the fitness average for the population of solutions relative to objective *i*.

Based on the weight updating equation used in back-propagation artificial neural networks [2], the following formula was proposed for redefining the weight values:

$$w_{i,t+1} = k_1.\alpha.w_{i,t} + k_2.(1-\alpha).e_{i,t} \qquad (7)$$

This equation uses an additional index *t*, which specifies a particular generation of the evolutionary algorithm. Thus, $w_{i,t+1}$ is the weight value associated with objective *i* for the following generation and it is based on the current weight $w_{i,t}$ and an error measure $e_{i,t}$. Here, k_1 and k_2 are normalization constants, computed by a procedure which will be described shortly. The central idea of this weight-updating scheme is to assign larger weights to the objectives with larger errors. The error measure includes the user's specification and is calculated in the following manner:

$$e_{i,t} = \left| \frac{user_i - \bar{f}_{i,t}}{user_i} \right| \qquad (8)$$

Thus, the difference between the average performance and the desired value specified by the user for objective *i* is taken into consideration for the error computation at a given time *t*. As such, the second term of equation (7) guarantees that the fitness function defined by equation (5) is dominated by those objectives with evaluations farthest from the desired values.

The effect generated by the first portion of equation (7) is analogous to the usage of *momentum* in the learning procedure of artificial neural networks, since it introduces memory to the system in a similar way. As happens in the context of neural networks, the purpose of inserting this term is to increase the system's stability. In this case, the inclusion of the previous weight value avoids drastic changes in the equation's outcome, which could make the genetic algorithm oscillate excessively. The constant α present in equation (7) is used to balance the two terms of this equation appropriately and can be assigned any value between 0 and 1.

The algorithm is initialized by choosing the starting values for the weights. The sum of these weights is defined by an integer value S_{w0} defined by the user:

$$S_{w0} = \sum_{i=1}^{n} w_{i,0} \qquad (9)$$

The value of S_{w0} is completely arbitrary and does not influence the outcome of the system.

Finally, we can determine the values for the k_1 and k_2 normalization constants. The purpose of these constants is to enable the definition of a measure of the system's convergence state based on the sum of the weights $S_{w,t}$ for a given time *t*. Establishing an analogy with hopfield neural networks, the following scalar quantity is defined:

$$E = \sum_{i=1}^{n} w_i^2 \qquad (10)$$

where *E* corresponds to the energy of the system. In fact, without considering the first term of equation (7), each weight $w_{i,t}$ is proportional to the corresponding error $e_{i,t}$. If that term is also to be taken into consideration, then it is necessary that the sum of the weights be proportional to the sum of the errors of the system at any given time *t*, that is:

$$S_{w,t} = k_3 \sum_{i=1}^{n} e_{i,t} = k_3 S_{e,t} \qquad (11)$$

where:

$$k_3 = \frac{S_{w0}}{S_{e0}} \qquad (12)$$

where S_{e0} corresponds to the sum of the errors observed for the first generation and k_3 is proportionality constant that takes into consideration the effect of the value chosen for S_{w0}. For the sum of the weights to keep obeying the aforementioned relation, the following values must be assigned to the normalization constants:

$$k_1 = \frac{S_{w,t}}{S_{w,t-1}} \therefore k_2 = \frac{S_{w,t}}{S_{e,t}} \qquad (13)$$

It should be noted that the computation of $S_{w,t}$ takes place before the calculation of the weight values themselves. Thus, the energy obtained by equation (10) is guaranteed to yield a coherent measure of the state of the evolutionary process. Therefore, the minimization of the system's energy actually corresponds to the satisfaction of multiple objectives. [6,7,8,9]

3 Analysis of the Energy Minimization Method and Proposed Modifications

3.1 Analysis of the Method's Behavior

The energy minimization method's main virtue is undoubtedly its capacity to adapt itself to the reality it faces. This way, a good diversity is always maintained in the population throughout the evolutionary process. Indeed, favoring less satisfied objectives at all times prevents any particular generation from being entirely composed of solutions that neglect a certain set of objectives so as to excel in others. Thus, a better exploration of the search space is performed, making the algorithm as a whole more robust and efficient in comparison to the traditional scalar aggregation method. Additionally, the inconvenient process of choosing the set of weight values for each objective is also avoided as was mentioned before.

On the other hand, whenever compromise solutions are desired this method doesn't necessarily converge to a satisfactory result. In fact, the constant shifting of the priorities of the objectives often produces a *speciation* of the population, with groups of solutions specializing in the satisfaction of different sets of objectives. Thus, every time the weights are updated a different group of solutions (a different *species*) begins to dominate the remainder of the population. As such, it ends up that the best solution yielded by the algorithm frequently oscillates between different solutions that excel in different sets of objectives, hardly ever converging to a final stable result. Obviously, if there is an optimal solution that dominates all the others then the algorithm will in principle converge towards it.

3.2 Proposed Modifications

The main purpose of the modifications proposed here is to prevent the algorithm from oscillating, converging instead to a definitive solution. Aside from that, it is desired that the final result yielded by the algorithm be a balanced compromise solution, with no particular objective being excessively neglected.

To do so, the final fitness evaluation function described in equation (5) will be modified. This equation originally calculates the final result simply by computing the weighted sum of the normalized evaluations *Fnorm*. Instead, the new fitness measure will consider the distance between the solution's vector of evaluations and the target vector of user-defined values for each objective (see section 2.3).

Thus, formally we rewrite equation (5) and combine it with equation (3), obtaining:

$$F = \left(\sum_{i=1}^{N} w_i . e_i^p \right)^{1/p} \quad p \geq 1 \qquad (14)$$

where e_i corresponds to the error of the solution's evaluation relative to its target value for objective *i*, and *p* defines how the vector distance is measured, as seen in section 2.3.

As such, by employing *p=2* we define a solution's final fitness as the *quadratic weighted sum of the errors* of its individual evaluations for each objective. Moreover, as

previously discussed, by increasing the value of p we also increase the "pressure to compromise" induced by the algorithm (see section 2.3). Consequently, the speciation tendency observed for the original method can be averted, preventing the evolutionary process from excessively oscillating between different species. It should also be noted that, with this modified equation, the evolutionary process will be driven more strongly towards a compromise solution, as well as being prevented from oscillating wildly (see section 2.3).

At this point, it should be noted that this fitness evaluation is only coherent when the errors e_i are normalized so that they can be appropriately compared to each other. Indeed, it must also be observed that a correct normalization should consider each of the evaluations f_i for a given objective i relative to its *search space*. Thus, for instance, the best possible evaluation for a particular objective i could yield $e_i = 0$, with the worst possible value corresponding to $e_i = 1$.

Unfortunately, in many cases this search space is not known *a priori*. Thus, it is usually necessary to *estimate* it during the evolutionary process itself and, in order to do this, values like the best and worst evaluations found so far ($best_i$ and $worst_i$) could be utilized. Also, the average evaluation avg_i of the entire population could also be employed to extract the notion of a "reasonable" evaluation for the current stage of the evolution. Optionally, the target values $user_i$ could also be used in order to achieve a more controlled estimate of the search space, although these values can somewhat distort the evolutionary process due to their arbitrary nature.

Therefore, with all these observations in mind, the following equation is proposed for error normalization:

$$e_i = \left| \frac{best_i - f_i}{best_i - avg_i} \right| \qquad (15)$$

where $best_i$ corresponds to the best evaluation found so far for objective i, while avg_i corresponds to the average evaluation obtained for the entire population relative to this objective.

4 Experimental Results

In order to measure the impact of the proposed modifications, the algorithm was tested in a complex problem of electrical circuit evolution on which the energy minimization method has already been previously applied. [8]

4.1 Electrical Circuit Evolution Problem

Although constituting only a small part of the total area of modern chips, analog circuitry is usually the limiting factor of their overall performance [5]. The current trend towards the achievement of low-power, low-area and high-speed analog cells such as operational amplifiers may increase the complexity of VLSI analog design, if hard specifications have to be met.

The Miller OTA (Operational Transconductance Amplifier) is a two-stage amplifier whose compensation capacitance introduces the Miller effect, and presents low output impedance for most of its frequency range [1,5]. The *pmos-npn* topology is employed here because of its better performance compared to the other ones, and is shown in Fig. 1.

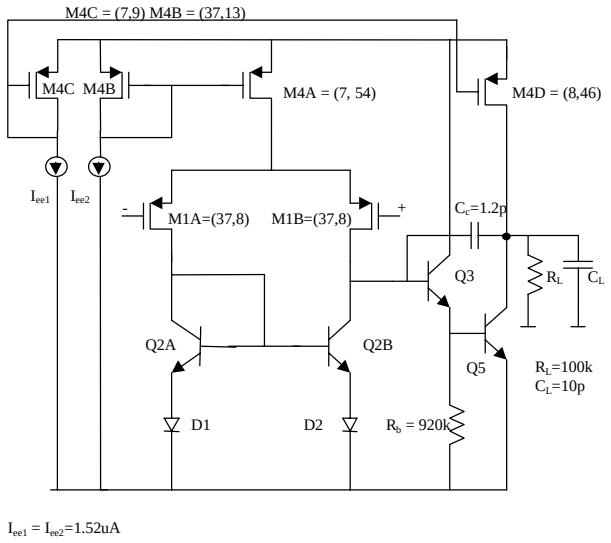


Fig. 1 – Miller OTA in BiCMOS technology: pmos-npn topology

We apply a genetic algorithm together with the multi-objective strategy defined previously so as to optimize performance for circuits of this kind. Throughout the genetic algorithm, each circuit is simulated using small signal and operating point analysis, and then has its performance estimated by a number of measures, namely *gain, GBW (gain-bandwidth product), area, power dissipation* and *phase margin*. Therefore, these measures actually correspond to the *objectives* of the optimization to be performed, and ideal values for these can be specified in order to drive the evolutionary process as desired.

The genetic algorithm manipulates the circuit specifications through the chromosome representation. An operational amplifier can be characterized by a list of real numbers representing transistor sizes, biasing current, and,

if it is the case, compensating capacitance. Each OpAmp feature is represented in the chromosome integer string so that each string element serves as a pointer to the actual OpAmp feature value. This representation is illustrated in Fig. 2.

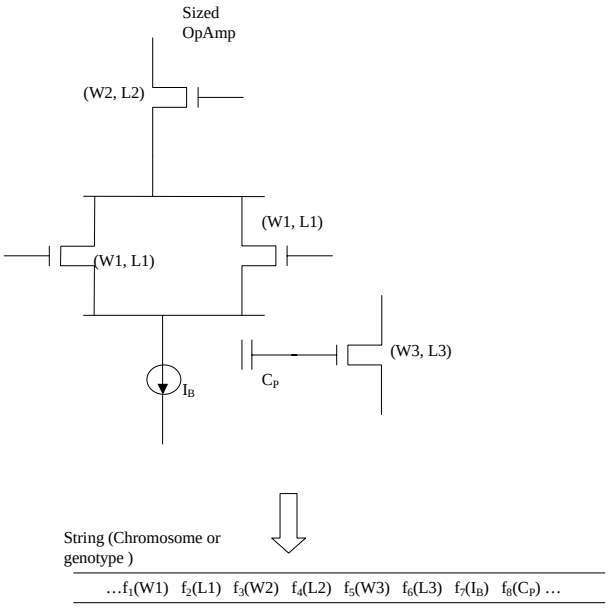


Fig. 2 – Representation of a sized OpAmp into the integer string (chromosome) processed by the genetic algorithm

The functions f_i shown in the figure above perform a simple conversion, whose general expression is given by:

$$f : \mathrm{I} \to \Re \quad y = \frac{x}{k_2} + k_1$$
$$x \in (0,1,...,N-1); \quad y \in [C_{mín}, C_{máx}] \qquad (16)$$

In the above expression, *y* and *x* are the actual value taken by the OpAmp feature and the value of the associated string position respectively. While each string position can assume *N* different integer values, each amplifier feature is constrained to values between C_{min} and C_{max}. These constraints are set according to the referred feature and to the technology being used. For instance, if the feature is a particular transistor width, C_{min} and C_{max} will stand for the minimum width allowed by the technology, W_{min}, and the maximum width chosen by the user, W_{max}. The constants k_1 and k_2 are set in order to make the conversion between *x* and *y*. A complete description of the problem can be found in [9].

The genetic algorithm employed here was based on the implementation used in [9], with the inclusion of the aforementioned changes proposed for the method.

The following parameters were used for the experiments presented here:

General Parameters	
Population:	*40*
Generations:	*300*
Rounds:	*3*
Crossover:	*70%*
Mutation:	*2,5%*
Selection Pressure:	*0,75*
Elitism:	*Yes*

Target Values (user$_i$)	
Gain:	*100 db*
GBW:	*1000000 kHz*
Dissipation:	*10μW*
Phase Margin:	*60°*
Area:	*3000μm²*

Table 1 – Parameters used for all experiments.

Performance statistics (i.e., *gain*, *GBW*, etc.) obtained for human made design can be found in [5].

Tests were performed so as to compare the original algorithm with its modified version, employing Euclidean distances (*p=2*). The results obtained were the following:

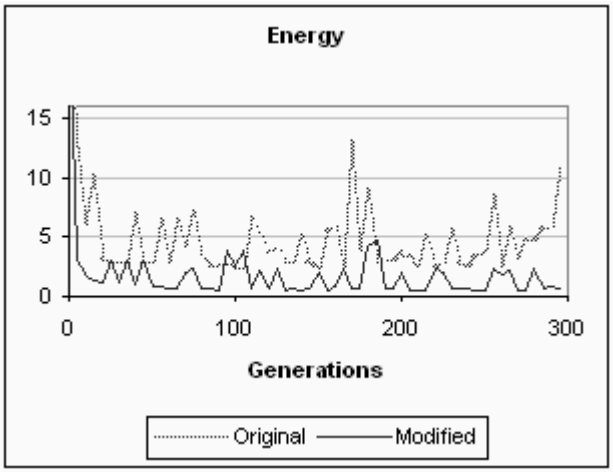


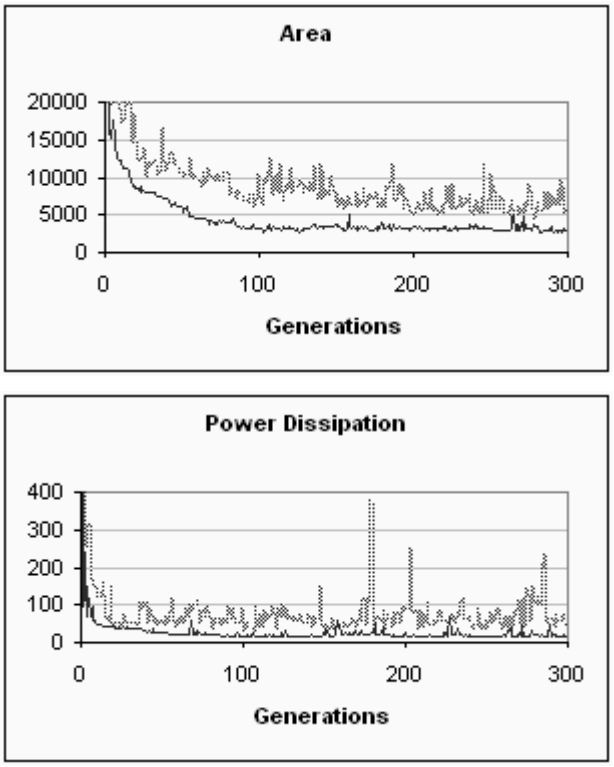
Fig. 3 – Energy graph for original and modified methods

The energy value displayed here is defined by equation (10) and represents how close to the ideal solution the system is (zero energy indicates that all the objectives have achieved the desired values). It can be readily seen here that

the modified version of the algorithm exhibits an energy level considerably lower than the original one. Also worthy of note is the fact that the modified algorithm avoids some of the excessive oscillations displayed by the original version.

Indeed, the overall evaluation is that the modified algorithm performs better than the original one. Of the five objectives, two of them were substantially better optimized by the modified version (*area* and *dissipation*) and one presented a comparable result for both versions (*gain*). Another objective achieved a better performance under the original version (*GBW*) and the last one wasn't satisfactorily optimized by any of the algorithms (*phase margin*).

All in all, it is possible to perceive that in general the modified algorithm provided a greater stability to the system, with fewer oscillations occurring throughout the evolutionary process. The graphs for these results were the following:



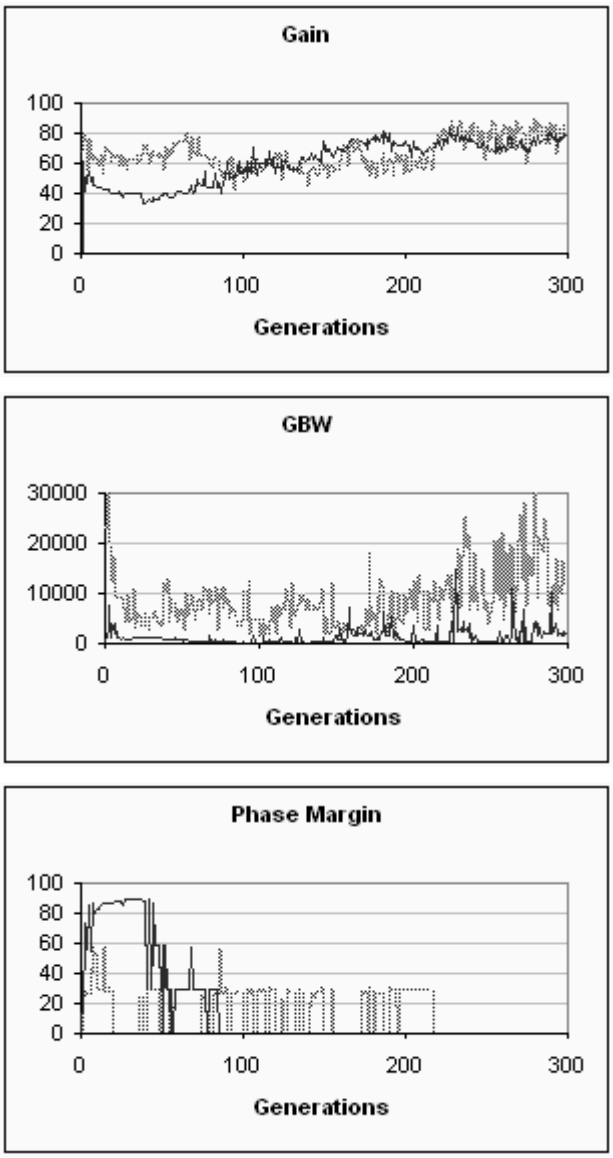


Fig. 4 – Optimization observed for each objective under the original and modified methods

5 Conclusions

After analyzing the experimental results, it can be concluded that the proposed modifications had a positive effect on the algorithm as a whole. The overall performance of the method improved and, more importantly, the excessive oscillations observed for the original version were reduced. Such oscillations can make a genetic algorithm reach a final solution that is significantly inferior to others found before, even after numerous generations. Therefore, the stability improvement provided by the

alterations should indeed be considered a very interesting result.

On the other hand, subsequent simulations proved that the modified version of the algorithm has a significant tendency to favor objectives that are to be minimized instead of maximized (*area* and *power dissipation* for the experiment presented in the previous section). In fact, the cause of this behavior can be traced to equation (8), which defines the average error calculation for each objective during the weight updating procedure. Here, for maximization objectives, the normalizing denominator $user_i$ corresponds roughly to the greatest value expected for that objective. However, when minimization is desired, this denominator corresponds to the *smallest* possible value and thus the error computed in such cases is much larger in general. As such, these larger errors end up driving the evolutionary process so as to favor the minimization objectives over the other ones. This explanation can indeed justify the modified method's poor results for *GBW* and average results for *gain* in the experiment presented previously. Nevertheless, it cannot account for the unsatisfactory performance observed for the *phase margin* objective.

In reality, a simple workaround for this problem would be to modify equation (8) so that, for objectives that are to be minimized, the average error would be given by:

$$e_{i,t} = \left| \frac{user_i - \bar{f}_{i,t}}{\bar{f}_{i,t}} \right| \qquad (17)$$

where the average fitness $\bar{f}_{i,t}$ for objective i at time t is used as the denominator instead of the target value $user_i$ for that objective. Preliminary tests show that the overall behavior of the method does become more balanced after this modification.

In fact, special attention must be given to all functions involving normalization. Not only can equation (8) be modified for the reasons already discussed, but also can equation (15), by which the individual errors e_i are computed during each evaluation. Such functions are crucial for the adequate comparison of the different objectives and the method will surely profit from a more precise normalization.

Aside from that, other modifications can be explored so as to improve the algorithm's performance even more. In particular, it might be interesting to replace equation (14) for computing the final fitness evaluation by the following function:

$$F = \left(\sum_{i=1}^{N} (w_i . e_i)^p \right)^{1/p} \quad p \geq 1 \qquad (18)$$

so that both the error e_i and the weight w_i are taken to the power of p. Additionally, the usage of other values for p can also be explored.

References

[1] Allen, P.E., Holberg, D. R., “CMOS Analog Circuit Design”, Holt, Rinehart and Winston editors, 1987.

[2] Churchland, P.S., Sejnowski, T.J., "The Computational Brain", MIT Press, 1992.

[3] Fonseca, Carlos M., Fleming, Peter J., “An Overview of Evolutionary Algorithms in Multiobjective Optimization”, Evolutionary Computation, Volume 3, Number 1, pp.1-16, MIT Press, Spring, 1995.

[4] Horn, Jeffrey, “Multicriterion Decision Making”, in Handbook of Evolutionary Computation, IOP Publishing Ltd and Oxford University Press, F1.9, 1997.

[5] Laker, K.R., Sansen, W., "Design of Analog Integrated Circuits and Systems", McGraw-Hill Inc. (eds), 1994.

[6] Zebulum, R.S., Pacheco, M.A., Vellasco, M., ”A Multi-Objective Optimisation Methodology Applied to the Synthesis of Low-Power Operational Amplifiers”, proceedings of the XIII International Conference in Microelectronics and Packaging, Vol. 1, Ivan Jorge Chueiri and Carlos Alberto dos Reis Filho (Eds), pp. 264-271, Curitiba, Brasil, August, 1998.

[7] Zebulum, R.S., Pacheco, M.A., Vellasco, M., “Analog Circuits Evolution in Extrinsic and Intrinsic Modes”, in the Proceedings of the Second International Conference on Evolvable Systems: From Biology to Hardware (ICES98), Lausanne, Switzerland, September, 23-26, 1998. M.Sipper, D.Mange and A. Pérez-Uribe (editors), vol. 1478, pp. 154-165, LNCS, Springer-Verlag, 1998.

[8] Zebulum, R.S., Pacheco, M. A., Vellasco, M., “Variable Length Representation in Evolutionary Electronics”, Evolutionary Computation, Volume 8, Number 1, pp.93-120, MIT Press, Spring, 2000.

[9] Zebulum, R.S., Pacheco, M.A., Vellasco, M., “A Novel Multi-Objective Optimisation Methodology Applied to Synthesis of CMOS Operational Amplifiers", Journal of Solid-State Devices and Circuits, Microelectronics Society - SBMICRO, 2000.

0-7695-0762-X/00 $10.00 © 2000 IEEE