Evolutionary Design of 2-Dimensional Recursive Filters via the Computer Language GENETICA

Ioannis F. Gonos, *Member, IEEE*, Lefteris I. Virirakis, Nikos E. Mastorakis, *Senior Member, IEEE*, and M. N. S. Swamy, *Fellow, IEEE*

***Abstract*—In this paper, we present a new design method for a class of two-dimensional (2-D) recursive digital filters using an evolutionary computational system. The design of the 2-D filter is reduced to a constrained minimization problem the solution of which is achieved by the convergence of an appropriate evolutionary algorithm. In our approach, the genotypes of potential solutions have a uniform probability within the region of the search space specified by the constraints and zero probability outside this region. This approach is particularly effective as the evolutionary search considers only those potential solutions that respect the constraints. We use the computer language GENETICA, which provides the expressive power necessary to get an accurate problem formulation and supports an adjustable evolutionary computational system. Results of this procedure are illustrated by a numerical example, and compared with those of some previous designs.**

***Index Terms*—Constrained optimization, evolutionary computational system, two-dimensional (2-D) recursive filters, 2-D systems.**

I. Introduction

In RECENT years, the field of two-dimensional (2-D) digital signal processing has been growing rapidly. Processing of medical pictures, satellite photographs, radar and sonar maps, seismic data mappings, gravity waves data, magnetic records are some good examples, where 2-D signal processing is needed. An overview of 2-D signal processing is given in [1], [3]. In these applications, the design of 2-D filters plays a central role. This design is based on two different methodologies, one based on an appropriate transformation of a one-dimensional (1-D) filter [1]–[3], and the other based on appropriate optimization techniques such as linear programming, Remez exchange algorithm, nonlinear programming: gradient methods, direct search methods, Newton and Gauss–Newton Methods, Fletcher–Powell, and conjugate gradient [1], [4]–[9].

However, most of the existing methodologies (algorithms) [2]–[9] may result in an unstable filter. Some sophisticated "recipes" have been adopted in order to resolve all these instability problems, but the outcome is likely to be a system that has a very small stability margin and hence, not of much practical importance [10], [11].

Stability conditions, represented as numerical constraints place certain bounds for the potential solutions to be within specific region of the search space, referred to as the "stability region". Considering potential solutions outside the stability region reduces the effectiveness of the evolutionary search even if a hard fitness-penalty is used for such solutions [11]. However, a conventional genetic algorithm cannot restrict the search within the stability region, because this region cannot be defined to be closed with respect to the genetic operations.

The computer language GENETICA [12], [13], which we use for our application, provides the expressive power required to achieve an accurate problem formulation that specifies potential solutions having uniform probability within the stability region. Given this formulation, a focused search is performed via an evolutionary computational system, which is incorporated in to the GENETICA's programming environment.

II. Design of 2-D Recursive Filters

The design task of 2-D recursive filters amounts to finding a transfer function $H(z_1, z_2)$ as in (1) such that the function $M(\omega_1, \omega_2) = H(e^{-j\omega_1}, e^{-j\omega_2})$ approximates the desired amplitude response $M_d(\omega_1, \omega_2)$, where the frequencies $\omega_1, \omega_2 \in [-\pi, \pi]$ and $z_1 = e^{-j\omega_1}, z_2 = e^{-j\omega_2}$.

For design purposes, the function $M(\omega_1, \omega_2)$ is equivalent to a class of nonsymmetric half-plane (NSHP) filters, whose 2-D transfer function $H(z_1, z_2)$ is given by

$$H(z_1, z_2) = H_0 \frac{\sum_{i=0}^{K} \sum_{j=0}^{K} a_{ij} z_1^i z_2^j}{\prod_{k=1}^{K} (1 + b_k z_1 + c_k z_2 + d_k z_1 z_2)}, \quad a_{00} = 1 \tag{1}$$

This approximation can be achieved by minimizing [10], [11]

$$\begin{aligned} J &= J(a_{ij}, b_k, c_k, d_k, H_0) \\ &= \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{N_2} [|M(\omega_1, \omega_2) - M_d(\omega_1, \omega_2)|]^p \end{aligned} \tag{2}$$

where $\omega_1 = (\pi/N_1)n_1, \omega_2 = (\pi/N_2)n_2$ and p is a positive integer (usually $p = 1$ or $p = 2$).

Hence, the aim is to minimize the difference between the actual and the desired amplitude response of the filter at $N_1 N_2$

Manuscript received November 2, 2004; revised April 6, 2005 and August 3, 2005. This paper was recommended by Associate Editor C.-T. Lin.

I. F. Gonos is with the High Voltage Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens, GR 15780 Athens, Greece (e-mail: igonos@ieee.org).

L. I. Virirakis is with the School of Architecture, National Technical University of Athens, GR 15780 Athens, Greece (e-mail: left@genetica-informatics.org).

N. E. Mastorakis is with the Military Institution of University Education, Department of Electrical Engineering and Computer Science, Hellenic Naval Academy, Piraeus 18539, Greece also with WSEAS, Athens GR 15773, Greece (e-mail: mastor@wseas.org).

M. N. S. Swamy is with the Center for Signal Processing and Communications, Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G IM8, Canada (e-mail: swamy@ece.concordia.ca).

Digital Object Identifier 10.1109/TCSII.2005.862040

points. Since we are dealing with first-degree factors in the denominator, it is known that the stability conditions are given by [1], [2]

$$|b_k + c_k| - 1 < d_k < 1 - |b_k - c_k|, \qquad k = 1, 2, \ldots, K \quad (3)$$

where K is given positive integer.

Thus, the design of 2-D recursive filters is equivalent to the following constrained minimization problem:

$$\min J = \sum_{n_1=0}^{N_1} \sum_{n_2=0}^{N_2} \left[\left| M\left(\frac{\pi n_1}{N_1}, \frac{\pi n_2}{N_2}\right) - M_d\left(\frac{\pi n_1}{N_1}, \frac{\pi n_2}{N_2}\right) \right| \right]^p \quad (4)$$

subject to the conditions given by (3), where p is a positive integer (usually $p = 1$ or $p = 2$), and N_1 and N_2 are given positive integers.

III. Problem Formulation

For purposes of illustration, without loss of generality, we consider the case of $K = 2$. Then, $H(z_1, z_2)$ given by (1) may be written as (5) shown at the bottom of the page.

In compact form, $M(\omega_1, \omega_2)$ may be written as

$$M(\omega_1, \omega_2) = H_0 \frac{A_R - jA_I}{(B_{1R} - jB_{1I})(B_{2R} - jB_{2I})} \quad (6)$$

where

$$\begin{aligned} A_R &= a_{00} + a_{01}c_{01} + a_{02}c_{02} + a_{10}c_{10} + a_{20}c_{20} \\ &\quad + a_{11}c_{11} + a_{12}c_{12} + a_{21}c_{21} + a_{22}c_{22} \\ A_1 &= a_{01}s_{01} + a_{02}s_{02} + a_{10}s_{10} + a_{20}s_{20} + a_{11}s_{11} \\ &\quad + a_{12}s_{12} + a_{21}s_{21} + a_{22}s_{22} \\ B_{1R} &= 1 + b_1c_{10} + c_1c_{01} + d_1c_{11} \\ B_{11} &= b_1s_{10} + c_1s_{10} + d_1s_{11} \\ B_{2R} &= 1 + b_2c_{10} + c_2c_{01} + d_{21}c_{11} \\ B_{21} &= b_2s_{10} + c_2s_{10} + d_2s_{11} \end{aligned} \quad (7)$$

and

$$\begin{aligned} c_{pq} &= c_{pq}(\omega_1, \omega_2) = \cos(p\omega_1 + q\omega_2) \\ s_{pq} &= s_{pq}(\omega_1, \omega_2) = \sin(p\omega_1 + q\omega_2), \\ & \qquad p, q = 0, 1, 2. \end{aligned} \quad (8)$$

Furthermore, we proceed with the design of the 2-D recursive filter given by (1) for the case $K = 2$, as presented in (5). Let the desired amplitude response be given by

$$M_d(\omega_1, \omega_2) = \begin{cases} 1, & \text{if } \sqrt{\omega_1^2 + \omega_2^2} \le 0.08\pi \\ 0.5, & \text{if } 0.08\pi < \sqrt{\omega_1^2 + \omega_2^2} \le 0.12\pi \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

IV. Problem Solution

The aforementioned problem has been tackled by using neural networks [10] or a genetic algorithm [11]. The problem-solving application presented here is developed in the computer language GENETICA [12], [13], which is integrated in a programming environment that includes an evolutionary computational system. Given a GENETICA program that includes nondeterministic elementary decisions, such as the definition of values ranging within specific intervals, the computational system evolves these decisions with respect to either confirmation or optimization goals formulated in the program. As a consequence, no genetic algorithm needs to be developed, while all of the GENETICA's computational parameters are available.

In our case, the problem is to minimize J by defining the values $a_{ij}, b_k, c_k, d_k, H_0 (i, j = 0, 1, 2; i \cdot j > 0; k = 1, 2)$, which range in the real interval $(-3, 3)$, as it is shown in [10], [11] Table I, under the constraints $|b_k + c_k| - 1 < d_k$ and $d_k < 1 - |b_k - c_k|$.

Consider the values b_k, c_k, d_k $(k = 1, 2)$ defined on the axes B, C, D respectively (Fig. 1). The constraints define the vector (b_k, c_k, d_k) within the tetrahedron having vertices $(1, 1, 1)$,

TABLE I
Results and Comparison

	GENETICA p=1	GENETICA p=2	GENETICA p=4	Results of [10] p=2	Results of [11] p=2
a_{01}	0.3233	-0.2247	0.6187	1.8922	1.8162
a_{02}	1.1108	2.5248	1.3857	-1.2154	-1.1060
a_{10}	0.3899	-0.3498	-1.2054	0.0387	0.0712
a_{11}	-0.0437	-2.0915	0.0927	-2.5298	-2.5132
a_{12}	0.6007	0.0317	-1.6908	0.3879	0.4279
a_{20}	1.0776	2.4656	1.4947	0.6115	0.5926
a_{21}	0.5095	0.1652	-0.6703	-1.4619	-1.3690
a_{22}	0.4758	0.7713	1.4066	2.5206	2.4326
b_1	-0.9697	-0.9316	-0.8233	-0.8707	-0.8662
b_2	0.0031	-0.0249	0.1476	-0.8729	-0.8907
c_1	-0.9681	-0.9309	0.3936	-0.8705	-0.8531
c_2	-0.0225	-0.0326	-0.7094	-0.8732	-0.8388
d_1	0.9521	0.8862	-0.3846	0.7756	0.7346
d_2	-0.9162	-0.8082	-0.3043	0.7799	0.8025
H_0	0.0003	0.0010	0.0140	0.0010	0.0009
J_1	63.5489	99.7677	426.27967	140.6201	138.7508
J_2	13.9355	10.0340	34.9403	14.4182	19.1791
J_4	2.4819	0.6341	1.4061	1.2163	2.9896
J_8	0.2587	0.0089	0.0307	0.0698	0.2816

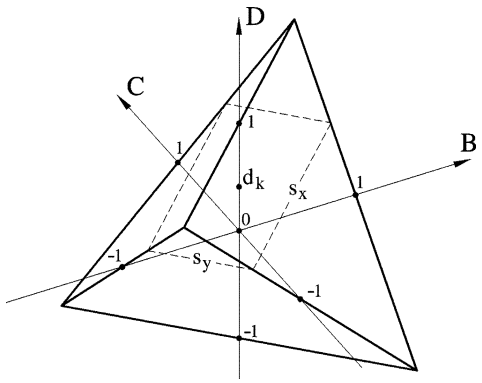

Fig. 1. Tetrahedron that represents the "stability region" in the state space of variables $b_k, c_k, d_k, (k = 1, 2)$ defined on the axes B, C, and D, respectively.

$$H(z_1, z_2) = \frac{a_{00} + a_{01}z_2 + a_{02}z_2^2 + a_{10}z_1 + a_{20} + a_{11}z_1z_2 + a_{12}z_1z_2^2 + a_{21}z_1^2 + a_{22}z_1^2z_2^2}{(1 + b_1z_1 + c_1z_2 + d_1z_1z_2)(1 + b_2z_1 + c_2z_2 + d_2z_1z_2)} \quad (5)$$

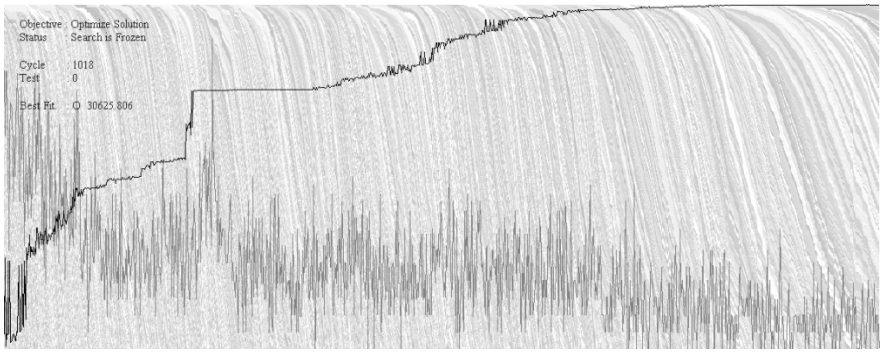

Fig. 2. Visualization of the evolutionary process minimizing the sum of absolute differences in 1018 computational cycles ($p = 1$).

$(-1,-1,1),(1,-1,-1),(-1,1,-1)$. In our application, the vector (b_k, c_k, d_k) resulting from any genetic operation has a uniform probability density within the tetrahedron and zero probability outside. This makes the evolutionary search very effective, as only unbiased potential solutions satisfying the constraints are considered.

The aforementioned probability density specifications can be achieved if the probability densities of the values b_k, c_k, d_k satisfy the following conditions.

- The probability density of the value d_k is proportional to the area of the section of the tetrahedron at a level parallel to the axes B and C, at the point $(0, 0, d_k)$ (Fig. 1, dashed line).
- The values b_k and c_k constitute respectively the B and C coordinates of a point having uniform probability density within the section.

The procedure defining the values b_k, c_k and d_k with respect to the above conditions is as follows.

1) Define d_k according to the probability density $ct \cdot A(z)$, where $-1 < z < 1$ while $A(z)$ is the area of the section of the tetrahedron at a level parallel to the axes B and C at the point $(0, 0, z)$ and ct is a normalization constant such that $ct \cdot \int_{-1}^{1} A(z) \cdot dz = 1$.
2) Calculate the dimensions s_x and s_y of the section at the point $(0, 0, d_k)$ (Fig. 1).
3) Define x and y according to uniform probability densities in the intervals $[0, s_x]$ and $[0, s_y]$ respectively
4) Calculate b_k and c_k, as the B and C coordinates respectively of the point (x, y), which is defined in the coordinate system of the section (i.e., the system having axes according to the sides marked by s_x and s_y respectively in Fig. 1).

GENETICA includes an atomic formula that allows a random selection within a real interval, according to a scalar probability density function represented as a list of real numbers (see GENETICA_Documentation.PDF in [12]: atomic formula *ipd*). We can easily formulate an arbitrarily detailed approximation of $A(z)$ as a scalar function by dividing the D interval $(-1, 1)$ in equal subintervals and calculating the area of the section of the tetrahedron at the midpoint of each subinterval. This is done in a preprocessing phase; then the list-approximation of $A(z)$ can be used in the main application, which consists of a GENETICA program. The outline of the program is given below.

1) Define $a_{ij}, H_0(i, j = 0, 1, 2; i \cdot j > 0)$ having uniform probability density in the real interval $(-3, 3)$.
2) Define b_k, c_k and $d_k (k = 1, 2)$ according to the procedure described before (Steps a to d).
3) Calculate J and define $1/(J+1)$ as the fitness value (this definition of fitness turns the problem to a maximization one—which satisfies GENETICA's optimization conventions—while assures a nonzero denominator).

GENETICA's computational system evolves the random decisions realized in both Step 1 of the main application and Steps a and c of the procedure defining the values b_k, c_k and d_k.

V. Visualization of Evolutionary Process

GENETICA's computational system evolves a population of "species" of potential solutions, while allows a visualization of the evolutionary process. A "species," which represents a set of identical genotypes, consists of a single genotype representing the generic element of the set and a number indicating the size of the set. If a genotype identical to an existing one is produced, then the new genotype is not introduced in the population. Instead, the size of the "species" of the existing genotype is increased by one. As a consequence, the population does not include different copies of identical genotypes.

Consider the bottom and the left edges of the textured area presented in Figs. 2 and 3 as the horizontal and vertical axes, respectively. The horizontal axis represents time in computational cycles, whereas the vertical axis represents the following three properties of the search state:

1) "Species" evolution, where "species" are represented by grayscale zones ordered by fitness, with the best fitness "species" at the top of the diagram. The thickness of the zones represents relative sizes of "species". Critical innovations (i.e., new best-fitness "species") emerge at the top of the diagram, while extinction takes place at the bottom of the diagram.
2) The best fitness encountered within each computational cycle is represented by a black line.
3) The ratio of the tests (i.e., fitness evaluations for different genotypes) that introduce new "species" in the population to all the tests performed during a computational cycle is represented by a gray line.

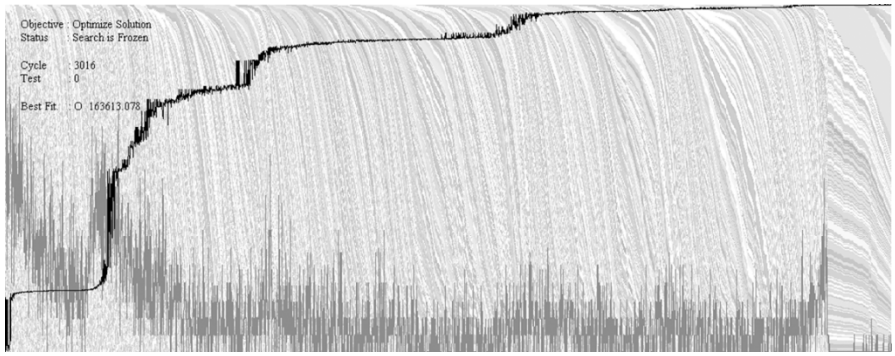

Fig. 3. Visualization of the evolutionary process minimizing the sum of square differences in 3016 computational cycles ($p = 2$).

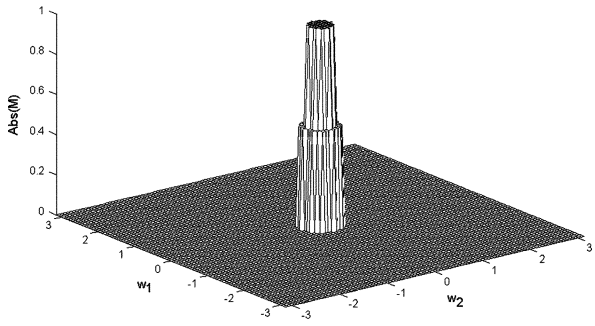

Fig. 4. Desired amplitude response.

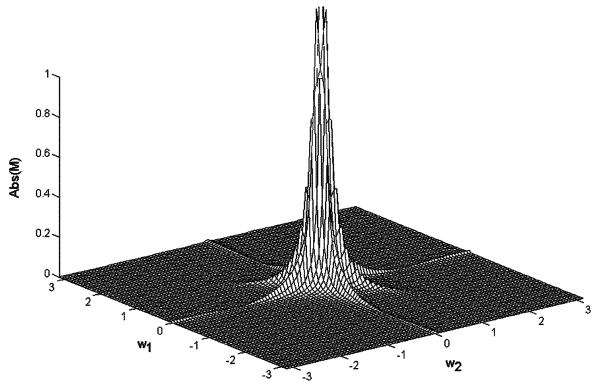

Fig. 5. GENETICA's results for $p = 1$.

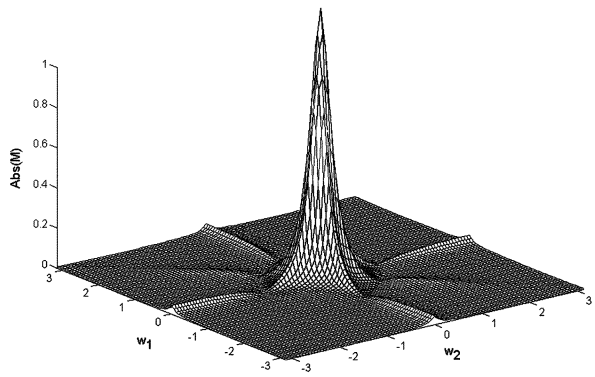

Fig. 6. GENETICA's results for $p = 2$.

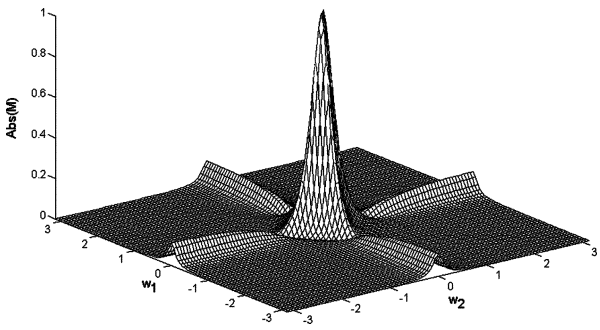

Fig. 7. Results using neural networks [10].

Sometimes, during the evolutionary process, best fitness values are temporarily stabilized in mediocre optima, as the population is trapped in isolated basins of the search space surrounded by lower fitness barriers. When mutant genotypes break these barriers jumping into a higher fitness basin, their offspring spreads searching for the new basin's optimum. In these cases best fitness values are rapidly increased, while new "species" are massively introduced in the population as the latter invades in the new basin.

VI. Results

In our application, we have calculated J by setting $N_1 = N_2 = 100$ in (2). Three different search procedures have been performed for $p = 1, 2$ and 4, respectively. For each search procedure we have used a population of 100 genotypes. Each computational cycle included 20 tests, i.e., application of genetic operations on 20 genotypes and substitution of worse-fitness genotypes of the population with better-fitness mutant genotypes. Best results are obtained for $p = 1$ and 2 in 1018 and 3016 computational cycles respectively (Figs. 2 and 3). The former search procedure took 5 min, while the latter 15 min, on a 2400-MHz Celeron CPU.

Results of the aforementioned search procedures are presented in Table I in comparison with results given by other methods. The last four rows of Table I present the J values, respective to each column, calculated with different p exponents (2). We use the notation J_p to denote the value of J calculated with exponent p.

Fig. 4 shows the graphic representation of the desired amplitude response (9) in the domain $[-\pi, \pi] \times [-\pi, \pi]$, while Figs. 5–8 show the graphic representation of (6) resulting from the coefficient values presented in the respective column of Table I.

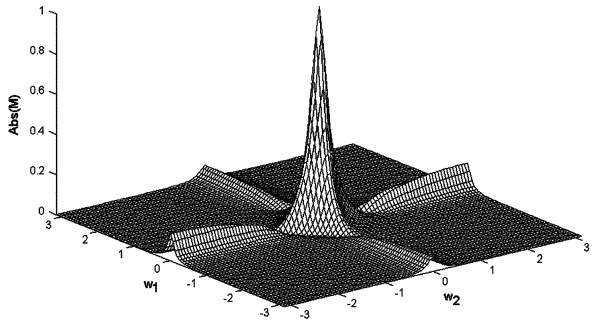

Fig. 8. Results using genetic algorithm [11].

VII. Conclusion

In this study, the design of a class of 2-D recursive filters is attempted by using an appropriate evolutionary computational scheme. The key-feature of this scheme is that genotypes' evolution is restricted in the "stability region," i.e., the region of the search space specified by the stability constraints: mutant genotypes have uniform probability within the stability region and zero probability outside. This approach gives better results than a genetic algorithm that uses fitness penalties for genotypes being outside the stability region.

Restricted evolution requires a sophisticated problem formulation, which a conventional genetic algorithm is hard to reach. This formulation was easily achieved via the computer language GENETICA, which provides the required expressive sufficiency. No application-specific genetic algorithm needed to be developed as the computational process was performed via the computational system integrated within GENETICA's programming environment.

References

[1] S. G. Tzafestas, Ed., *Multidimensional Systems, Techniques and Applications*. New York, NY: Marcel Dekker, 1986.

[2] W.-S. Lu and A. Antoniou, *Two-Dimensional Digital Filters*. New York: Marcel Dekker, 1992.

[3] T. Kaczorek, *Two-Dimensional Linear Systems*. Berlin, Germany: Springer-Verlag, 1985.

[4] G. A. Maria and M. M. Fahmy, "An LP design technique for two-dimensional digital recursive filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-22, no. 1, pp. 15–21, Feb. 1974.

[5] P. K. Rajan and M. N. S. Swamy, "Quadrantal symmetry associated with two-dimensional digital transfer functions," *IEEE Trans. Circuits Syst.*, vol. CAS-29, no. 6, pp. 340–343, Jun. 1983.

[6] T. Laasko and S. Ovaska, "Design and implementation of efficient IIR notch filters with quantization error feedback," *IEEE Trans. Instrum. Meas.*, vol. 43, no. 3, pp. 449–456, Jun. 1994.

[7] C.-H. Hsieh, C.-M. Kuo, Y.-D. Jou, and Y.-L. Han, "Design of two-dimensional FIR digital filters by a two-dimensional WLS technique," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 44, no. 5, pp. 348–412, May 1997.

[8] M. Daniel and A. Willsky, "Efficient implementations of 2-D noncausal IIR filters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 44, no. 7, pp. 549–563, Jul. 1997.

[9] W.-P. Zhu, M. O. Ahmad, and M. N. S. Swamy, "A closed-form solution to the least-square design problem of 2-D linear-phase FIR filters," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 44, no. 12, pp. 1032–1039, Dec. 1997.

[10] V. Mladenov and N. Mastorakis, "Design of two-dimensional recursive filters by using neural networks," *IEEE Trans. Neural Netw.*, vol. 12, no. 3, pp. 585–590, May 2001.

[11] N. E. Mastorakis, I. F. Gonos, and M. N. S. Swamy, "Design of 2-dimensional recursive filters using genetic algorithms," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 5, pp. 634–639, May 2003.

[12] Documentation of a prototype version, genetic evolution of novel entities through the interpretation of composite abstractions, GENETICA [Online]. Available: http://www.genetica-informatics.org

[13] L. Virirakis, "GENETICA: A computer language that supports general formal expression with evolving data structures," *IEEE Trans. Evol. Comput.*, vol. 7, no. 5, pp. 456–481, Oct. 2003.