10.2478/v10170-010-0024-5

Archives of Control Sciences
Volume 20(LVI), 2010
No. 4, *pages 417–433*

Improving energy compaction of a wavelet transform using genetic algorithm and fast neural network

JAN STOLAREK

In this paper a new method for adaptive synthesis of a smooth orthogonal wavelet, using fast neural network and genetic algorithm, is introduced. Orthogonal lattice structure is presented. A new method of supervised training of fast neural network is introduced to synthesize a wavelet with desired energy distribution between output signals from low–pass and high–pass filters on subsequent levels of a Discrete Wavelet Transform. Genetic algorithm is proposed as a global optimization method for defined objective function, while neural network is used as a local optimization method to further improve the result. Proposed approach is tested by synthesizing wavelets with expected energy distribution between low– and high–pass filters. Energy compaction of proposed method and Daubechies wavelets is compared. Tests are performed using image signals.

Key words: wavelet transform, neural networks, genetic algorithms, signal processing, lattice structure

1. Introduction

During the last two decades Discrete Wavelet Transform became one of the most popular tool in the area of signal processing. Many different wavelets have been proposed, such as Daubechies, Coiflet, Morlet, Shannon, Meyer, Battle-Lamarié or Mexican hat wavelets [3, 9]. Each family of the wavelet functions has its unique properties, which makes some wavelets more suitable for particular applications than others. This raises a problem of selecting the best wavelet for a given task, which leads to a question: does there exist a wavelet that would be the best for the given task, but has not been proposed yet? To provide a general answer to this question a method for automatic adaptive synthesis of wavelets for particular task should be developed.

First step towards adapting wavelets was parametrization of wavelet filter coefficients. Many such parametrizations have been proposed so far [7, 8, 11, 14, 24, 26]. A crucial element in these methods is optimal selection of parameters' values, which is done either using numerical or analytic approach. In [15] and [23] lattice structure for designing two–channel perfect reconstruction filters was presented. This approach was

J. Stolarek is with Institute of Information Technology, Technical University of Lodz, Wolczanska str. 215, 93-005 Lodz, Poland. E-mail: jan.stolarek@p.lodz.pl.

Received 26.10.2010. Revised 2.12.2010.

based on representing a filter bank in a form of parameterized lattice structure. Parameters were optimized using well–known numerical methods (e.g. quasi–Newton method) and the resulting values, together with the lattice structure, defined the filter. Dietl, Meerwald and Uhl introduced wavelet parametrization to improve performance in digital image watermarking security [4, 10]. They proposed random selection of filter parameters to increase watermark robustness against attacks. Shark and Yu proposed parametrization of wavelets using angles as trigonometric functions' arguments and applying genetic algorithm to synthesize shift–invariant wavelets [16].

So far, all the authors have concentrated their efforts on synthesizing wavelets that are optimal with respect to some arbitrarily selected criterion e.g. smoothness or shift invariance. However, there is no guarantee that such wavelet optimality will result in a better performance of a wavelet-based signal processing algorithm. Therefore, methods for wavelet synthesis based on rating the final result of signal processing using wavelets should be explored in more detail.

This paper presents example of such a method. Proposed algorithm of wavelet synthesis is based on a Fast Orthogonal Neural Network (FONN [17]) and Genetic Algorithm (GA). Wavelet parametrization is based on an orthogonal lattice structure [18, 21, 25]. Main contribution of this paper is introduction of a method for adaptive synthesis of a wavelet with desired energy distribution between low–pass and high–pass wavelet coefficients on subsequent levels of multilevel DWT. This approach concentrates not on the wavelet properties itself, but on the optimality of a signal resulting from processing using wavelets. In such approach optimal wavelets "emerge" as a result of fulfilling conditions imposed on the processed signal. Genetic algorithm will be applied as a global optimization method, while Fast Orthogonal Neural Network (FONN) will be used as a gradient method to further optimize defined objective function, starting from minimum found using GA. These methods allow to synthesize a new wavelet that will provide desired energy distribution between low–pass and high–pass wavelet coefficients for a particular class of signals (e.g. image or sound). Synthesizing the wavelet with better energy compaction than already existing wavelets will allow to improve the quality of compressed signals.

2. Orthogonal lattice structure

The lattice structure is – likewise the lifting scheme [22] – an alternative approach to implementation of a wavelet filter bank [2, 15, 23]. In this paper wavelet synthesis is based on an orthogonal lattice structure proposed in [25]. Properties of that structure, especially its connection with the existing wavelet theory and ability to synthesize valid orthogonal wavelet functions are discussed in [19]. Its computational complexity is discussed in [25]. Below a definition of an orthogonal lattice structure is presented.

Orthogonal lattice structure is based on two–point base operations

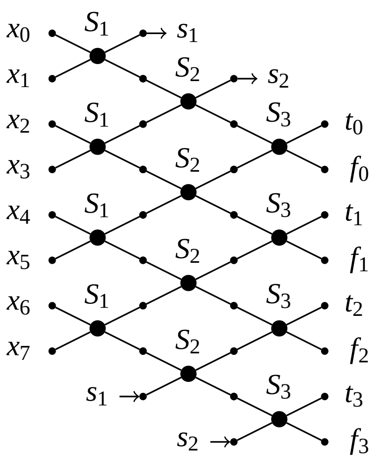

Figure 1. Lattice structure implementing 6–tap transform.

$$S_k = \begin{bmatrix} cos(\alpha_k) & sin(\alpha_k) \\ sin(\alpha_k) & -cos(\alpha_k) \end{bmatrix} , \tag{1}$$

where k stands for index of the operation (see Fig. 1). Such two–point base operation can be written in form of a matrix equation

$$\begin{bmatrix} t_0 \\ f_0 \end{bmatrix} = S_k \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} . \tag{2}$$

For S_k given by equation 1, the following equality holds true:

$$S_k^{-1} = (S_k)^T = S_k , \tag{3}$$

where S_k^{-1} is the inverse of S_k and $(S_k)^T$ is the transpose of S_k. It is therefore easy to perform inverse of the transformation given by equation 2.

Lattice structure is composed of $K/2$ stages, each containing S_k operations repeated $N/2$ times, where K and N are the lengths of the filter's impulse response and of a processed signal respectively. On each stage of the lattice structure, elements of the signal are processed in pairs by S_k base operations. After each stage, base operations are shifted down by one and a lower input of the last base operation in the current stage is connected to the upper output of the first base operation in the preceding stage (s_1 and s_2 in Fig. 1). Upper outputs of base operations in the last layer (t_0, t_1, t_2 and t_3 in Fig. 1) form the trend signal $\overrightarrow{t}$ which corresponds to the low–pass filter output, while the lower outputs (f_1, f_2, f_3 and f_4 in Fig. 1) form the fluctuation signal $\overrightarrow{f}$ which corresponds to the high-pass filter output.

3. Wavelet synthesis using fast orthogonal neural network

In this section a wavelet synthesis criterion based on energy distribution between output signals from low–pass and high–pass filters is presented. Although this algorithm can be used to synthesize wavelets with any energy distribution, its most important application is to synthesize wavelets that pack as much of the signal's energy as possible into low–pass wavelet coefficients. In a perfect situation – whole energy of a signal packed into low–pass coefficients – this would allow to remove high–pass coefficients without losing any information about the signal. Although such a situation is not possible, applying a wavelet that packs most of signal's energy into the low–pass coefficients to subsequent levels of a discrete wavelet transform, results in packing most of the signal's energy into small number of low–pass coefficients. If the synthesized wavelet compacts more energy into low–pass coefficients than other existing wavelets (e.g. Daubechies), then applying it to many levels of DWT would lead to packing more energy in the same number of wavelet coefficients, thus giving a better quality of a compressed signal. Once the wavelet is synthesized, it is applied to processing of a signal in the same manner as any other wavelets.

To synthesize a new wavelet with desired energy distribution between low–pass and high–pass wavelet coefficients Fast Orthogonal Neural Network [17] with topology based on orthogonal lattice structure is used. S_k base operations are represented using Basic Operation Orthogonal Neurons (BOON) – a special kind of neuron with two inputs and two outputs, designed to perform two–point orthogonal operations. Training signals are propagated through the network and, for each signal, error must be back-propagated. However, expected output values for training signals remain unknown, since the only thing that is defined is the output energy proportion between low–pass and high–pass wavelet coefficients. Hence existing supervised training methods have to be modified and a new objective function must be defined. In [18] the following objective function was proposed:

$$G(\overrightarrow{w}) = \frac{1}{2}\sum_{j=1}^{N/2}\left[\left(d_j^2 - t_j^2\right)^2 + \left(e_j^2 - f_j^2\right)^2\right] \quad , \tag{4}$$

where N is the number of outputs of the network (the same as length of a processed signal), j is the index of BOON in the output layer, t_j^2 is the energy of the low-pass output of a j-th BOON, d_j^2 is the expected energy on that output, f_j^2 is the energy of the high-pass output of a j-th BOON, e_j^2 is the expected energy on that output. This function measures how much does the actual energy proportion differ from the expected energy proportion. Expected energies d_j^2 and e_j^2 are calculated independently for each neuron, based on its actual inputs, using formulas

$$\begin{aligned} d_j^2 &= h_1 \cdot \left[\left(x_0^{(j)} \right)^2 + \left(x_1^{(j)} \right)^2 \right] \ , \\ e_j^2 &= h_2 \cdot \left[\left(x_0^{(j)} \right)^2 + \left(x_1^{(j)} \right)^2 \right] \ , \end{aligned} \tag{5}$$

where $x_0^{(j)}$ and $x_1^{(j)}$ are j-th neuron's inputs and h_1, h_2 are expected energy proportions between outputs, such that

$$h_1 + h_2 = 1 \ . \tag{6}$$

This method allowed to effectively synthesize a wavelet with desired energy distribution for signals of particular class. However, only first level of signal analysis was taken into account. Therefore it was possible that the synthesized wavelet had different energy proportions on subsequent levels of wavelet transform. In order to synthesize wavelets that ensure desired energy distribution on more than one level of signal analysis, multilevel DWT of a signal has to be taken into account.

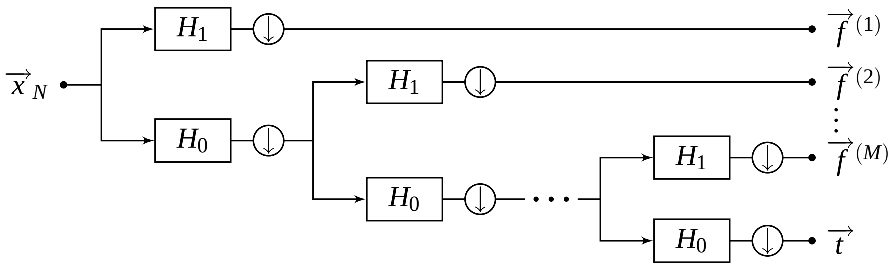

Figure 2. Multilevel discrete wavelet transform.

Fig. 2 shows a diagram of multilevel DWT. $\vec{x}_N$ is the input signal. H_0 and H_1 are low–pass and high–pass filters respectively. Together they form an orthogonal filter bank. $\textcircled{\downarrow}$ represents signal decimation (removing every other sample). $\vec{f}^{(1)}, \vec{f}^{(2)}, \ldots, \vec{f}^{(M)}$ are the fluctuation (detail) signals[1] and $\vec{t}$ is the trend (approximation) signal. In this paper the above method of adaptive wavelet synthesis using neural network is extended to further levels of DWT. The following synthesis criterion is proposed:

[1]With only one level of signal analysis we use $\vec{f}$ instead of $\vec{f}^{(1)}$ to simplify the notation.

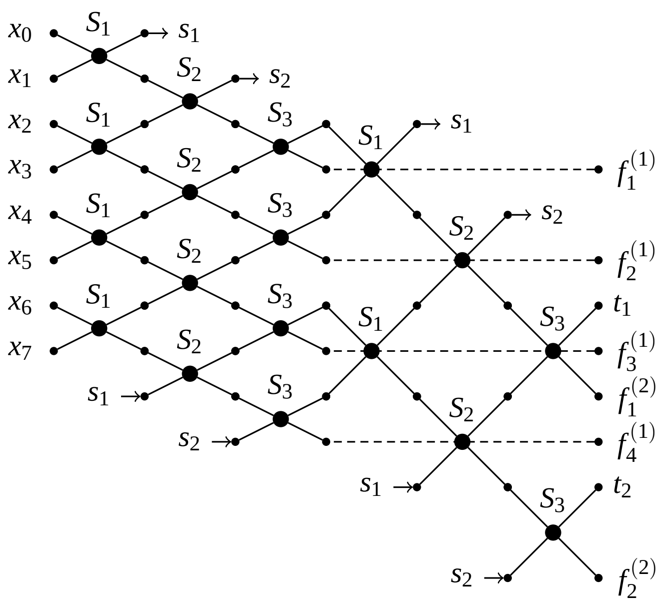

Figure 3. Lattice structure performing two levels of signal analysis.

$$
\begin{aligned}
&\frac{\mathcal{E}(\overrightarrow{f}^{(2)})+\cdots+\mathcal{E}(\overrightarrow{f}^{(M)})+\mathcal{E}(\overrightarrow{t})}{\mathcal{E}(\overrightarrow{f}^{(1)})+\cdots+\mathcal{E}(\overrightarrow{f}^{(M)})+\mathcal{E}(\overrightarrow{t})} = \\
=&\frac{\mathcal{E}(\overrightarrow{f}^{(3)})+\cdots+\mathcal{E}(\overrightarrow{f}^{(M)})+\mathcal{E}(\overrightarrow{t})}{\mathcal{E}(\overrightarrow{f}^{(2)})+\cdots+\mathcal{E}(\overrightarrow{f}^{(M)})+\mathcal{E}(\overrightarrow{t})} = \\
=&\cdots = \frac{\mathcal{E}(\overrightarrow{t})}{\mathcal{E}(\overrightarrow{f}^{(M)})+\mathcal{E}(\overrightarrow{t})} = h_1 \ ,
\end{aligned}
\tag{7}
$$

where $\mathcal{E}(\cdot)$ denotes energy of a signal, M denotes number of DWT levels and h_1 and h_2 are expected energy proportion between low–pass and high–pass outputs, as defined in equations (5) and (6). This equation means that for each level of DWT the synthesized wavelet is expected to give the same proportion between energy of low–pass wavelet coefficients and energy of the input signal. If $h_1 = 1$ is assumed then proposed method will synthesize wavelets that pack as much energy of a signal as possible into the low-pass wavelet coefficients.

It is important to notice, that it is not possible to find such a filter, that it will produce exactly the expected energy proportions for each input signal. It is however possible to determine such filter that, for a given class of signals, energy proportions will be true in a statistical sense. Therefore it is important, that the wavelet is synthesized for signals of particular class, e.g. image or sound.

Using the lattice structure approach, multilevel DWT of a signal is performed by appending another lattice structure to low–pass outputs of the lattice structure from the previous stage. Fig. 3 shows example of a lattice structure implementing 6–tap transform and performing 2 levels of DWT. Base operations for every level of DWT are the same, as denoted by identical S_k operations in corresponding layers of the network. This example network transforms signal $(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ to signal $(f_1^{(2)}, f_4^{(1)}, t_2, f_1^{(1)}, f_2^{(2)}, f_2^{(1)}, t_1, f_3^{(1)})$, where $(f_1^{(1)}, f_2^{(1)}, f_3^{(1)}, f_4^{(4)})$ is a first level detail signal ($\overrightarrow{f}^{(1)}$ in Fig. 2), $(f_1^{(2)}, f_2^{(2)})$ is a second level detail signal ($\overrightarrow{f}^{(2)}$ in Fig. 2) and (t_1, t_2) is a trend signal ($\overrightarrow{t}$ in Fig. 2). Signal periodicity is assumed.

Condition given in equation (7) means, that objective function given in equation (4) has to be applied after completing every level of DWT. This means that expected energy values are calculated after each level of DWT only for high–pass outputs of the current level. Low–pass outputs serve as inputs for the next level of analysis. After completing last level of DWT expected energy values are calculated for both low–pass and high–pass outputs according to equation (4) and all the calculated values are back-propagated through the network. For a straightforward determination of objective function's gradient in respect to the weights, Signal Flow Graphs (SFG) are used [12, 13]. Due to a non–standard form of objective function, adjustment of back-propagation algorithm is required. To calculate the value of the objective function, each output of the network is propagated through a branch that raises that value to the power of two, before comparing it to the expected value (see equation (4)). For each output it is therefore necessary to multiply back-propagated error value by $-2t_j$ (for low-pass outputs) or $-2f_j$ (for high-pass outputs). This is equal to the derivative of the quadratic function branch calculated for the value propagated forward through that branch [12].

Weights modification is performed according to the steepest descent algorithm:

$$\overrightarrow{w}_{n+1} = \overrightarrow{w}_n - \eta \, \nabla \, G(\overrightarrow{w}) \quad , \tag{8}$$

where $\overrightarrow{w}_n$ is weights vector in n–th iteration, η is the learning step and $\nabla \, G(\overrightarrow{w})$ is error function's gradient calculated in respect to the network's weights. Observations of signals in the network have shown, that values propagated through the network get bigger on subsequent levels of DWT, which is expected since the low–pass wavelet coefficients get bigger with each DWT level. This leads to a conclusion that on each level different learning step must be used to counter that effect. Using smaller learning steps for later levels prevents both skipping the minimum due to too big changes of weights and domination of the learning process by bigger derivatives on the last levels of analysis. Proper selection of learning steps may be difficult and requires many experiments combined with precise observation of values in the network. Adjustment of weights according to equation (8) leads to finding the minimum of objective function $G(\overrightarrow{w})$ (equation (4)) and therefore achieving desired distribution of energy between low–pass and high–pass wavelet coefficients, for a given set of training signals.

4. Global optimization using genetic algorithm

Steepest descent algorithm used in the training process of the network is a gradient method and, although it is effective for synthesizing wavelets for two or three levels of DWT, it is likely to get stuck in a local minimum when the number of DWT levels increases. To prevent such situation, global optimization using genetic algorithms with evolution strategies is introduced. Best solution found using the genetic algorithm is used as a starting point for network training to further improve the result.

Algorithm 1 Genetic algorithm outline

```
1: Create random population $P$ of $\mu$ individuals
2: while (stop condition not satisfied) do
3:     Calculate fitness of individuals in population $P$
4:     Create temporary population $T$ containing $\lambda$ individuals by selection from popu-
       lation $P$
5:     Perform crossover and mutation on individuals in population $T$
6:     Calculate fitness of individuals in population $T$
7:     Select $\mu$ individuals to form new population $P$
8: end while
9: Display best individual in population $P$
```

Algorithm 1 shows outline of proposed genetic algorithm. Stop condition is either reaching the maximum number of iterations (generations) denoted as gen_{max} or not getting better results in subsequent generations (minimal expected improvement of fitness will be denoted as imp_{min}). Other steps shown in the Algorithm 1 are discussed in detail in the following subsections.

4.1. Representation of individuals

Each S_k base operation is precisely defined by only one variable – angle α_k. Therefore lattice structure consisting of N layers can be represented using only N numbers $(\alpha_1, \alpha_2, \ldots, \alpha_N)$, where each $\alpha_k \in [0, 2\pi)$. Individuals in a genetic algorithm consist of N chromosomes. Each chromosome represents α_k for one layer of the lattice structure. Let us denote chromosome representing value of α_k as c_k. Each chromosome c_k is represented in a binary form using m bits. Therefore each chromosome is an integer from range $[0, 2^m)$. Values of chromosomes are mapped to the values of angles (in radians) using formula

$$\alpha_k = \frac{c_k \cdot \pi}{2^{m-1}} \ . \tag{9}$$

4.2. Fitness evaluation

To evaluate fitness of an individual its chromosomes $(c_1, c_2, \ldots, c_N)$ are converted to angles $(\alpha_1, \alpha_2, \ldots, \alpha_N)$ using equation (9). Angle values are used to calculate base operations $(S_1, S_2, \ldots, S_N)$, which are then used to perform multilevel DWT on the data in a training set using the lattice structure approach.

Individuals' fitness is calculated based on the energy distribution on the subsequent levels of DWT. For each individual the training signals are transformed and proportion between energy of low–pass wavelet coefficients and input signal energy is calculated. Low–pass coefficient are then transformed again and the same proportion is calculated for the second level of wavelet analysis (see equation (7)). This step is repeated required number of times. The actual energy distribution for each level of DWT is compared to expected energy distribution. The closer the actual value to the expected proportion, the higher the individual's fitness on that level. Fitness on a given level *i* of DWT is therefore determined using formula:

$$F_j(i) = |ME - |h_1 - AEP_{ji}|| \ , \tag{10}$$

where j is number of an individual, i is the level of DWT, ME is maximal possible error in energy distribution, h_1 is expected energy proportion and AEP_{ji} is actual energy proportion on i–th level of DWT for j–th individual, calculated using the formula

$$AEP_{ji} = \begin{cases} \dfrac{\mathcal{E}(\overrightarrow{t_j}) + \sum_{k=i+1}^{M} \mathcal{E}(\overrightarrow{f_j}^{(k)})}{\mathcal{E}(\overrightarrow{t_j}) + \sum_{k=i}^{M} \mathcal{E}(\overrightarrow{f_j}^{(k)})} & \text{for } i < M \\[2ex] \dfrac{\mathcal{E}(\overrightarrow{t_j})}{\mathcal{E}(\overrightarrow{f_j}^{(M)}) + \mathcal{E}(\overrightarrow{t_j})} & \text{for } i = M \end{cases} \ , \tag{11}$$

where $\mathcal{E}(\cdot)$ denotes energy of a signal, M denotes number of DWT levels, $\overrightarrow{t_j}$ denotes the trend (approximation) signal of the j-th individual and $\overrightarrow{f_j}^{(k)}$ denotes the fluctuation signal on k-th level of DWT for j-th individual (see Fig. 2). In equation (10), $|h_1 - AEP_{ji}|$ is an error made by an individual in the distribution of energy. The closer this error to 0, the higher the fitness. It is important to ensure that both actual energy proportion error $h_1 - AEP_{ji}$ and the fitness are positive numbers. Hence the absolute values are used.

Since energy proportion between low–pass wavelet coefficients and input signal's energy can change from 0 (whole signal energy packed into high–pass coefficients) to 1 (whole signal energy packed into low–pass coefficients), maximal possible error ME is determined using the formula:

$$ME = max(h_1, 1 - h_1) \ , \tag{12}$$

where h_1 is expected energy proportion between low–pass coefficients and input signal's energy. Equations 10, 11 and 12 ensure that fitness values fall in range $[0, 1]$.

Training set contains many signals, therefore individual's fitness for a given level of DWT is calculated as an average of fitness values $F_j(i)$ for all signals in a training set. Total fitness F of an individual is calculated by averaging partial fitness $F_j(i)$ calculated for every level of DWT:

$$F_j = \frac{\sum_{i=1}^{M} F_j(i)}{M} \quad , \tag{13}$$

where j is number of an individual and M is a number of DWT levels.

4.3. Selection, crossover and mutation

Temporary population T is created by selecting λ individuals from population P. Tournament selection is used. Each individual from population T can be selected for crossing over with other individual from that population with probability p_c. One point crossover is performed. Crossing point is chosen randomly. Each locus (bit in a chromosome) in population T can be selected for mutation with probability p_m. Mutation is performed by reversing value of selected bits.

4.4. Evolution strategies

During experiments with adaptive synthesis of wavelets using GA it was discovered, that although proposed method was able to find good solutions, individuals with very low fitness (close to 0) were not eliminated from the population. In many cases this lead to destabilization of optimization process. As a solution to the problem evolution strategies [1] were introduced. Usage of modified (μ,λ) and $(\mu+\lambda)$ strategies is proposed, as outlined in Algorithm 1. In (μ,λ) strategy a new population P is created by selecting μ fittest individuals from a temporary population T (assuming $\lambda > \mu$). In $(\mu+\lambda)$ strategy the new population P is created by selecting μ fittest individuals from both P and T populations. There are two important differences between evolution strategies proposed in literature and in presented approach: individuals have binary representation instead of real–number representation and there is no self–adaptation. As will be shown in section 5 evolution strategies allow to eliminate unfit individuals from the population, and therefore improve optimization process.

5. Experimental validation

5.1. Testing methodology

Proposed genetic algorithm and fast neural network with topology based on orthogonal lattice structure for multilevel DWT were implemented using Java and Matlab programming languages. Two different data sets were prepared. One set was used to synthesize the wavelet (training set), the other one was used for testing. Both sets contained 512 signals, each signal with length of 512. Tests were carried out using image data taken from rows of a grayscale images. Initial population in a genetic algorithm was selected randomly. Initial weights of a neural network were set to values of the best lattice structure synthesized by the genetic algorithm. Therefore neural network was used to further improve result obtained using GA.

Table 4. Wavelet synthesis with genetic algorithm using image data.

Expected energy of low–pass coefficients [%]	Actual results [%]					
	4–tap transform		6–tap transform		8–tap transform	
	training	testing	training	testing	training	testing
50	50.05	49.96	50.13	49.99	49.81	49.88
70	70.18	70.07	69.95	69.58	69.76	70.90
100	99.45	99.65	99.50	99.65	98.81	99.50
Daubechies	99.45	99.65	99.48	99.65	99.51	99.66

Table 5. Improvement of results obtained using the genetic algorithm and fast neural network.

Expected energy of low–pass coefficients [%]	Actual results [%]					
	4–tap transform		6–tap transform		8–tap transform	
	training	testing	training	testing	training	testing
50	50.03	49.94	50.05	49.91	49.77	49.84
70	70.06	69.95	69.95	69.58	69.84	70.90
100	99.46	99.64	99.50	99.65	99.28	99.59
Daubechies	99.45	99.65	99.48	99.65	99.51	99.66

Wavelet synthesis was tested using 4–tap, 6–tap and 8–tap transforms with 4 levels of signal analysis. Neural network was trained using off–line teaching with 10 training epochs. Optimal values of parameters in the neural network (e.g. learning steps for each level of analysis) may differ depending on number of layers in the network and desired energy distribution. Genetic algorithm performs equally well for different energy distributions using the same parameters. Following values were used for the experiments:

- crossover probability $p_c = 0.98$,
- mutation probability $p_m = 0.03$,
- population P size $\mu = 20$,
- temporary population T size $\lambda = 40$,
- maximum generations $gen_{max} = 10$,
- minimum expected improvement between generations $imp_{min} = 10^{-4}$.

Both proposed evolution strategies have shown to give similar results. The experiments were carried out using (μ, λ) strategy.

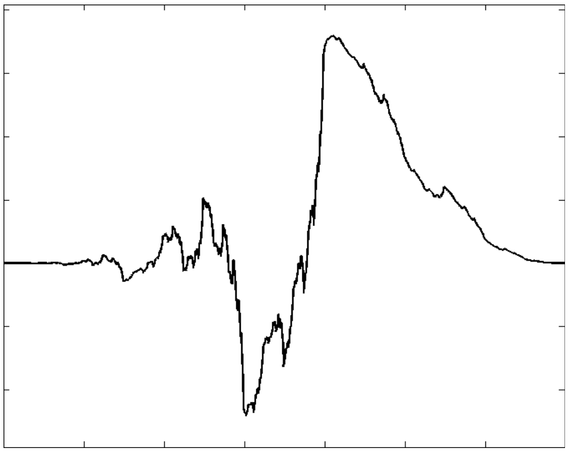

Figure 4. 4–tap wavelet synthesized using GA.

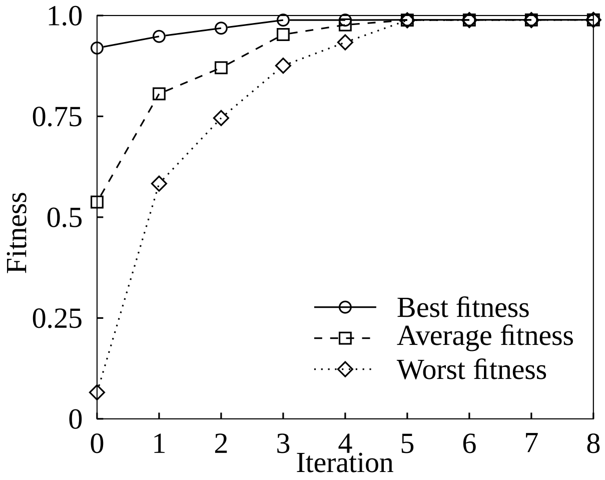

Figure 5. Change of individuals' fitness during GA optimization.

5.2. Results and discussion

Tab. 4 shows the results of wavelet synthesis using genetic algorithm. Tab. 5 shows results of improving wavelets synthesized using GA with fast neural network. In both tables the first column shows expected percentage of input energy located in the low–pass coefficients (h_1 in equation (7)) on each level of signal analysis. Remaining amount of energy is located in the high–pass coefficients, summing up to give a total of 100%. Remaining columns show testing result obtained on both training and testing sets, expressed as an average of actual percentage of energy located in low–pass coefficients on each level of signal analysis. Three different energy distributions were tested. Both tables present energy distribution for Daubechies wavelets.

Results in Tab. 4 show that proposed method of wavelet synthesis using the genetic algorithm is able to synthesize a wavelet with desired energy distribution between low–pass and high–pass wavelet coefficients with error less than 1%. For the highest requested energy compaction of 100% it synthesizes wavelets that perform similarly to Daubechies wavelets. Fig. 4 shows an example of smooth 8-tap wavelet synthesized using GA.

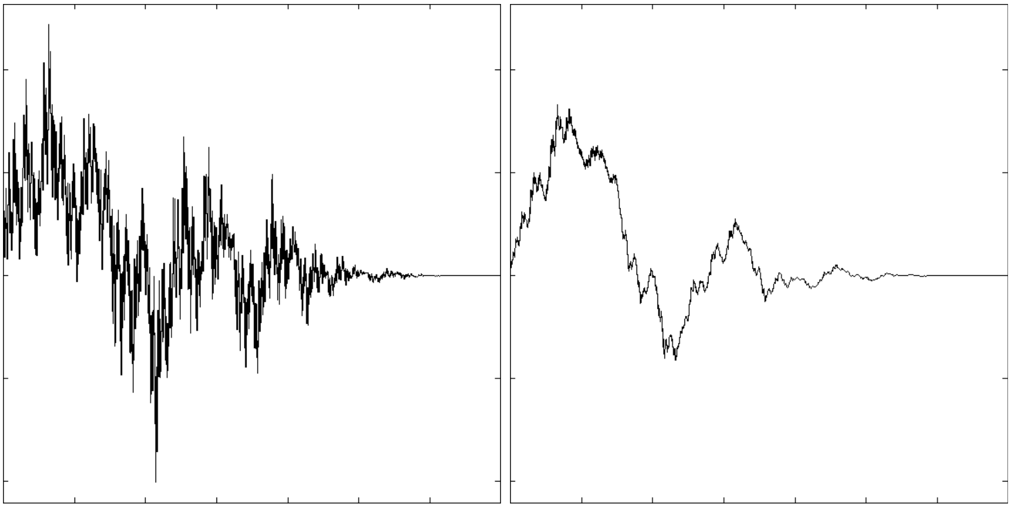

Figure 6. 8–tap wavelet synthesized using GA (left) and then improved using FONN (right).

Fig. 5 shows how fitness (vertical axis) of the individuals in the population changes between iterations (horizontal axis) of genetic algorithm. Figure shows the fitness of the best and the worst individual in the population as well as an average fitness value in the population. imp_{min} was set to 0 for this experiment and gen_{max} was set to 8. Population denoted as 0 on horizontal axis is the initial randomized population. We can see that with evolution strategies individuals with low fitness are eliminated from the population (fitness of worst individual increases in subsequent iterations of the algorithm). In 5–th iteration optimization process stabilizes – only individuals with the highest fitness remain in the population. Following generations bring improvement of the best solution by approx. 0.02%. Although elimination of unfit individuals is desired, too fast elimination of diversity in a population leads to premature convergence. This may lead to finding suboptimal solutions. Problem may be countered by increasing mutation rate, although this may destabilize optimization process by “damaging” good solutions.

As shown in Tab. 5 neural network can be used to further improve achieved results. It is expected that FONN can improve results on the training set, since it attempts to fit to the training data as much as possible. Hence there can be no guarantee as for the improvement of results on the testing set. Tables 4 and 5 demonstrate that with FONN results for the training set have improved in 6 out of 9 test. In 2 cases there was no change in the result. For the testing set 2 results have improved and 3 remained unchanged.

Fig. 6 shows example of improving the result obtained using GA. Left image shows 8–tap wavelet with desired energy distribution of 100% synthesized using GA. Right image shows further improvement of this wavelet with FONN. It can be seen that neural network produced a wavelet that is smoother than the one synthesized by the genetic algorithm. It can also be noticed from the tables that this lead to improvement of the result by 0.47% for the training set and 0.09% for the testing set.

Algorithm performance was measured on a computer with Intel Core 2 Duo T8300 processor (each of two cores operating at 2.4GHz frequency). Neural network was implemented in Java as a single threaded application. Average synthesis time was 39.2 seconds for 4–tap transforms, 44.5 seconds for 6–tap transforms and 50.5 seconds for 8–tap transforms (total time of 10 epochs). Genetic algorithm was implemented using Matlab programming language and it is capable of operating on both cores paralel. Using the Matlab Profiler, the precise time of execution was measured. The most time consuming task is the estimation of individuals' fitness. When only one processing core is used, average time needed to evaluate fitness of an individual is 1.85s. Enabling parallel computations on both processor cores reduces the average time needed to calculate individual's fitness to 0.93s. Length of the filter being synthesized doesn't influence the performance significantly. Nevertheless, in each iteration the fitness of about 40–60 individuals has to be estimated (temporary population can be smaller with $(\mu+\lambda)$ strategy). Convergence analysis (Fig. 5) has shown, that algorithm stabilizes around 5th–6th iteration, so using greater number of iterations is unnecessary. This implies, that the whole synthesis process using genetic algorithm can take approximately from 220 to 390 seconds (assuming parallel computations on two cores). Such a long time might be unacceptable for daily usage and has to be improved.

It must be noted, that neural network implementation in Java is designed as a research application and therefore it is not fully optimized. It relies heavily on design patterns utilizing polymorphism (e.g. Template Method), which is known to significantly degrade the program's performance. Genetic algorithm implementation was optimized with the aid of Matlab Profiler and, in its current form, cannot be significantly optimized. However, it must be noted that elements of the input signal are processed by the lattice structure independently in pairs. This implies, that lattice structure can be implemented using technologies with SIMD (Single Instruction, Multiple Data) capabilities. Moreover, the lattice structure itself doesn't include any conditional instructions. Therefore it seems that, for production purposes, the NVidia CUDA technology would be the most appropriate to implement the proposed algorithms. CUDA is a parallel computing architecture that makes use of the GPU and is designed to process huge streams of data with a set of identical instructions. Such an implementation should bring a significant improvement in the performance, thus making proposed algorithms suitable for daily usage on personal computers.

6. Conclusions

In this paper a new method for adaptive synthesis of a wavelet transform using genetic algorithm and Fast Orthogonal Neural Network with topology based on the orthogonal lattice structure was presented. New optimization criterion based on energy distribution on subsequent levels of DWT was introduced. This criterion puts emphasis on the final result of signal processing rather than on the wavelet itself. This is the most important contribution of this paper, since so far authors have concentrated directly on wavelet properties, while in this approach optimal wavelet "emerges" indirectly as a result of fulfilling optimality criteria by the wavelet–processed signal. Experiments have shown that proposed criterion can be used for adaptive synthesis of a new wavelet with desired energy distribution for a signal of particular class. Presented approach was compared with Daubechies wavelets in terms of energy compaction. Results have shown that the wavelets synthesized with genetic algorithm and neural network can perform better than the Daubechies wavelets.

Within further development of proposed method Hierarchical Fair Competition [5] should be considered to solve the problem of premature convergence in the genetic algorithm. It is also possible to treat fitness values $F_j(i)$ as a separate optimization criteria and view the whole problem solved by the genetic algorithm as a multi-objective optimization. This would allow to introduce evolutionary multi-objective optimization methods, e.g. Global Optimality Level [6]. Implementation can be improved by using NVidia CUDA technology.

Future research will concentrate on adjusting presented methods to allow wavelet synthesis for improving digital image watermarking fidelity. First experiments have already been carried out and the results are promising [20].

References

[1] H.-G. BEYER and H.-P. SCHWEFEL: Evolution strategies - a comprehensive introduction. *Natural Computing*, **1**(1), (2002), 3-52.

[2] T. COOKLEV: An efficient architecture for orthogonal wavelet transforms. *IEEE Signal Processing Letters*, **13**(2), (2006).

[3] I. DAUBECHIES: Ten Lectures on Wavelets. SIAM, 1992.

[4] W. DIETL, P. MEERWALD and A. UHL: Protection of wavelet-based watermarking systems using filter parametrization. *Signal Processing*, **83**(10), (2003), 2095-2116.

[5] J. J. HU and E. D. GOODMAN: The Hierarchical Fair Competition (HFC) model for parallel evolutionary algorithms. In *2002 Congress on Evolutionary Computation*, (2002).

[6] Z. KOWALCZUK and T. BIAŁASZEWSKI: *Genetic algorithms in multi-objective optimization of detection observers*, Springer-Verlag, 2004, 511-556.

[7] M. LANG and P. N. HELLER: The design of maximally smooth wavelets. In *Proc. of the Acoustics, Speech, and Signal Processing*, **3** (1996), 1463-1466.

[8] P. LIPIŃSKI and M. YATSYMIRSKYY: On synthesis of 4-tap and 6-tap reversible wavelet filters. *Przeglad Elektrotechniczny*, **12** (2008), 284-286.

[9] S. MALLAT: A wavelet tour of signal processing. Academic Press, December 2008.

[10] P. MEERWALD and A. UHL: Watermark security via wavelet filter parametrization. In *Int. Conf. on Image Processing*, **3** (2001), 1027-1030.

[11] J. E. ODEGARD and C. S. BURRUS: New class of wavelets for signal approximation. In *IEEE Int. Symp. on Circuits and Systems (ISCAS)*, (1996).

[12] S. OSOWSKI: Signal flow graphs and neural networks. *Biological Cybernetics*, **70**(4), (1994), 387-395.

[13] S. OSOWSKI and A. CICHOCKI: Application of SFG in learning algorithms of neural networks. In *Int. Workshop on Neural Networks for Identification, Control, Robotics, and Signal/Image Processing*, (1996), 75-83.

[14] G. REGENSBURGER: Parametrizing compactly supported orthonormal wavelets by discrete moments. *Applicable Algebra in Engineering, Communication and Computing*, **18**(6), (2007), 583-601.

[15] P. RIEDER, J. GOTZE, J. S. NOSSEK and C. S. BURRUS: Parameterization of orthogonal wavelet transforms and their implementation. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, **45**(2), (1998), 217-226.

[16] L.-K. SHARK and C. YU: Design of optimal shift–invariant orthonormal wavelet filter banks via genetic algorithm. *IEEE Trans. on Signal Processing*, **83** (2003), 2579-2591.

[17] B. STASIAK and M. YATSYMIRSKYY: Fast orthogonal neural networks. *Lecture Notes in Computer Science*, **4029** (2006), 142-149.

[18] J. STOLAREK: Synthesis of a wavelet transform using neural network. In *XI Int. PhD Workshop OWD, Conference Archives PTETiS*, **26** (2009).

[19] J. STOLAREK: On the properties of a lattice structure for a wavelet filter bank implementation: Part 1. *J. of Applied Computer Science*, **19**(1), (2011), to appear.

[20] J. STOLAREK and P. LIPIŃSKI: Improving digital watermarking fidelity using fast neural network for adaptive wavelet synthesis. *J. of Applied Computer Science*, **18**(1), (2010), 61-74.

[21] J. STOLAREK and M. YATSYMIRSKYY: Fast neural network for synthesis and implementation of orthogonal wavelet transform. In *Image Processing & Communications Challenges*. AOW EXIT, 2009.

[22] W. SWELDENS: The lifting scheme: A new philosophy in biorthogonal wavelet constructions. In *Wavelet Applications in Signal and Image Processing III*, (1995), 68-79.

[23] P. P. VAIDYANATHAN and P.-Q. HOANG: Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks. *IEEE Trans. on Acoustics, Speech and Signal Processing*, **36**(1), (1988), 81-94.

[24] D. WEI, A.C. BOVIK and B.L. EVANS: Generalized coiflets: a new family of orthonormal wavelets. In *Record of the Thirty-First Asilomar Conf. on Signals, Systems & Computers*, **2** (1997), 1259-1263.

[25] M. YATSYMIRSKYY: Lattice structures for synthesis and implementation of wavelet transforms. *J. of Applied Computer Science*, **17**(1), (2009), 133-141.

[26] H. ZOU and A. H. TEWFIK: Parametrization of compactly supported orthonormal wavelets. *IEEE Trans. on Signal Processing*, **41**(3), (1993), 1428-1431.