# Evolving Wavelets Using a Coevolutionary Genetic Algorithm and Lifting

Uli Grasemann and Risto Miikkulainen

Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
{uli, risto}@cs.utexas.edu

**Abstract.** Finding a good wavelet for a particular application and type of input data is a difficult problem. Traditional methods of wavelet design focus on abstract properties of the wavelet that can be optimized analytically but whose influence on its real-world performance are not entirely understood. In this paper, a coevolutionary genetic algorithm is developed that searches the space of biorthogonal wavelets. The lifting technique, which defines a wavelet as a sequence of digital filters, provides a compact representation and an efficient way of handling necessary constraints. The algorithm is applied to a signal compression task with good results.

## 1 Introduction

Since their introduction over two decades ago, wavelets have proven useful in an extremely broad range of applications, from data compression to numerical simulation, from signal de-noising to seismology.

Most of these applications, however, share the same problem: Although using wavelets is relatively straightforward in most cases, finding a good wavelet for a particular job or type of input data is not. Often, a good wavelet can make all the difference. The choice of a wavelet is usually made on a somewhat ad-hoc basis, by trying out several well-known wavelets and keeping the best one. Traditional methods of wavelet design rely on abstract properties like filter length or order of approximation that can be analytically optimized, but whose influence on the performance of a wavelet is not entirely understood.

In this paper, a method for adapting biorthogonal wavelet bases to a specific application and class of input data is presented. The method is based on a coevolutionary genetic algorithm closely related to the *Enforced Sub-Populations (ESP)* neuroevolution method introduced in [1]. The algorithm encodes wavelets as a sequence of *lifting steps*, i.e. finite digital filters that can be combined to form a wavelet. Lifting provides a compact representation as well as an efficient way of handling necessary constraints.

The next section gives a brief tour of wavelets, lifting and wavelet-based data compression. Section 3 discusses related work, and section 4 describes the algorithm and the evaluation function used. In section 5, the algorithm is applied to the task of compressing cubic spline curves with good results.

## 2 Background

Both wavelet theory and the application of wavelets in data compression are complex and evolving subjects. This section gives a brief high-level overview of these topics. For more details on classical wavelet theory, see [2]; [3] contains an introduction to lifting, and [4] covers the basics of wavelet-based data compression.

### 2.1 Wavelets

Wavelets are a mathematical tool for representing and approximating functions hierarchically. At the heart of wavelet theory, there is a single function $\psi$, called the *mother wavelet.* Any function can be represented by superimposing translated and dilated versions of $\psi$.

The translates and dilates of $\psi$ are denoted by $\psi_{j,i}$, where $i$ and $j$ are the translation and dilation parameter. We are focusing on the discrete case where $i$ and $j$ only take on integer values. The $\psi_{j,i}$ can be computed from the mother wavelet as

$$\psi_{j,i}(x) = 2^{\frac{j}{2}}\ \psi(2^j x - i). \tag{1}$$

Figure 1 shows an example wavelet and a translated and dilated version of that wavelet.

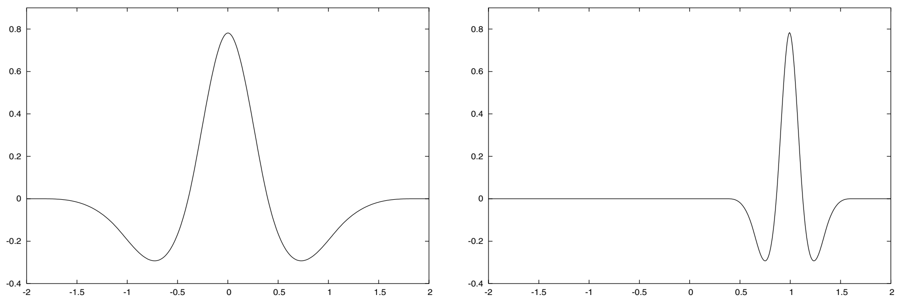

**Fig. 1.** A wavelet $\psi$ and the translated and dilated version $\psi_{1,2}$. The wavelet shown is a cubic spline wavelet.

All the translates of $\psi$ for a specific dilation $j$ span a function space $W_j$:

$$W_j = span\{\ \psi_{j,i}\ |\ i \in \mathbb{Z}\}. \tag{2}$$

The $W_j$ are called *wavelet spaces* or *detail spaces*, because each of them adds a level of detail to the wavelet representation of a function. All of the detail spaces combined form a basis in which any function can be expressed.

The process of decomposing a function into *wavelet coefficients* (a scaling factor for each of the $\psi_{j,i}$) is called *wavelet transform.* If the parameters $i$ and $j$

take on dicrete values, we have a *discrete wavelet transform* or *DWT*, essentially leading to a finite number of coefficients.

In order to compute the DWT of a function $f$, we need to find one wavelet coefficient $\gamma_{j,i}$ for each $\psi_{j,i}$, such that

$$f = \sum_{j,i} \gamma_{j,i} \psi_{j,i}. \tag{3}$$

If a wavelet basis (i.e. the set of all $\psi_{j,i}$) is *orthogonal*, then the $\gamma_{j,i}$ are given by

$$\gamma_{j,i} \;=\; \langle f, \psi_{j,i} \rangle \;=\; \int_{-\infty}^{\infty} f(x)\overline{\psi_{j,i}}(x)dx, \tag{4}$$

where the bar denotes the complex conjugate. Otherwise, a *dual* wavelet $\widetilde{\psi}$ is necessary such that $\psi$ and $\widetilde{\psi}$ together are *biorthogonal*, which basically means that the transform must be invertible. We can then use $\widetilde{\psi}$ for determining the wavelet coefficients (eqn. 4), and the original wavelet for the inverse DWT (eqn. 3). Note that an orthogonal wavelet is just a special case of a biorthogonal one where $\widetilde{\psi} = \psi$.

## 2.2 Filters and the Fast Wavelet Transform

Computing a wavelet transform in the way just described is expensive and cumbersome. However, an algorithm called the *Fast Wavelet Transform* or *FWT* allows computing the wavelet coefficients by recursively applying a pair of digital filters to the data, much like the Fast Fourier Transform reduces a DFT to computing a few finite sums.

A *digital filter* can be defined by giving a sequence of real numbers called *filter coefficients*. It is applied by convolution with an input sequence. A filter is said to have *finite impulse response (FIR)*, if its coefficients are non-zero only on a finite range. A FIR filter can be represented by a finite number of coefficients and the index of the leftmost non-zero coefficient.

It turns out that the filter pair used in the FWT uniquely determines the mother wavelet $\psi$ and also (in the biorthogonal case) the dual wavelet $\widetilde{\psi}$. In order to define a valid wavelet transform, a filter pair must be *complementary*, which is the same as saying that the associated wavelet must be biorthogonal.

Ensuring that a filter pair is complementary and that the individual filters are finite are the basic contraints in wavelet design.

## 2.3 Lifting

The Lifting scheme, introduced by Sweldens [5] in 1996, offers an easy way to construct complementary filter pairs. A finite filter, called a *lifting step*, is used to generate a new filter pair from an existing pair. Multiple lifting steps can be applied consecutively. In [3], Sweldens and Daubechies proved two important properties of lifting:

– Lifting preserves biorthogonality, i.e. if the original filter pair is complementary, then so is the new pair, no matter what lifting step is applied.
– Any wavelet with finite filters can be expressed as a sequence of lifting steps. Starting with the trivial wavelet transform (called the *Lazy Wavelet*), all possible wavelets can be reached by applying a finite number of finite-length lifting steps.

These two properties make lifting a powerful tool for constructing new wavelets.

### 2.4 Wavelets and Signal Compression

In signal compression applications, wavelets are used as the transformation part of a transform coder. Figure 2 shows the general structure of a transform coder.

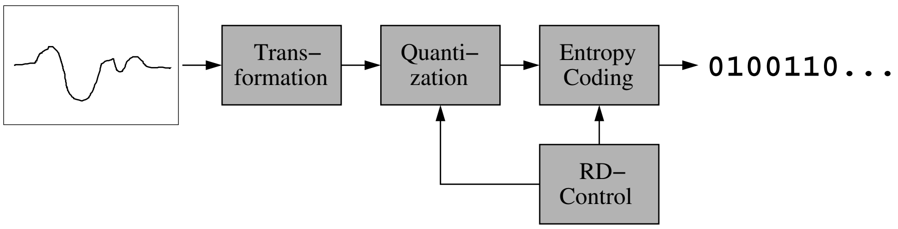

**Fig. 2.** The structure of a transform coder. The signal is first decorrelated using an invertible transform, and then quantized and entropy coded. The rate-distortion (RD) unit controls the quantization to minimize the distortion within the available bit rate.

The first step is to apply an invertible transform to the data in order to decorrelate it. Examples of such transforms are the discrete cosine transform (DCT) and the discrete wavelet transform (DWT). The performance of a transform coder depends largely on how well the transform decorrelates the signal. A well decorrelated signal consists mainly of values close to zero.

The transform coefficients are then quantized, i.e. expressed using symbols from a finite alphabet, and entropy coded, using as little space or bandwidth as possible. The rate-distortion (RD) unit controls the quantization in order to achieve minimal distortion within the available bit rate.

Examples for transform coders are the DCT-based JPEG standard and the wavelet-based JPEG2000 standard.

## 3 Related Work

The idea of adaptive wavelet bases is not new. Traditionally, this has been done by so-called *dictionary* methods, where a basis is selected from an overcomplete set of predefined functions called *atoms*. Examples of such methods are the *best* *basis* algorithm [6] and *wavelet packets* [7]. [8] and [9] use evolutionary algorithms for adaptive dictionary methods. Note that dictionary methods do not come up with new wavelets. Instead, they simply try to select the best combination of atoms to form a basis.

The lifting technique has provided new ways of adapting wavelets. For example, Claypoole et al. [10] use lifting to adapt a wavelet transform to a given signal by optimizing data-based prediction error criteria.

In [11] and [12], genetic algorithms are used for the design of digital filters.

Several stochastic optimization techniques have been applied to the design of wavelets. Monro and Sherlock [13] use simulated annealing to find wavelets with balanced uncertainty in space and frequency. Hill et al. [14] use a genetic algorithm to find windowed trigonometric functions that can be used in a continuous wavelet transform.

To our knowledge, the combination of lifting and genetic algorithms has not been explored previously.

## 4 Evolving Wavelets

In section 2.3, two interesting properties of lifting were mentioned:

– It preserves biorthogonality, and
– any wavelet can be expressed as a sequence of lifting steps.

These two properties make sequences of lifting steps an effective representation for wavelets in a genetic algorithm, because (1) any random sequence of lifting steps will encode a valid (i.e. biorthogonal) wavelet, and (2) any wavelet can be represented using the genetic code. In this section, a coevolutionary genetic algorithm that evolves wavelets encoded as lifting steps will be described.

**Algorithm.** The coevolutionary GA used is closely related to the *Enforced Sub-Populations (ESP)* neuroevolution algorithm introduced by Gomez and Miikkulainen [1]. ESP evolves a number of populations of individual neurons in parallel. In the evaluation phase, ESP repeatedly selects one neuron from each sub-population to form candidate networks. The fitness of a particular neuron is the average fitness of all networks in which it participated.

This concept can be easily applied to wavelet evolution: Several populations of lifting steps are evolved in parallel, and are randomly combined to form wavelets, which are then evaluated. No migration or crossover occurs between sub-populations. Figure 3 describes the algorithm in more detail.

**Representation.** A lifting step is represented as a fixed-length sequence of floating point numbers for the filter coefficients, and a single integer for the leftmost index of the filter. Using a fixed number of fixed-length steps limits the number of wavelets that can be represented. However, it also limits the length of the wavelet filters, which is a desirable effect. Also, most wavelets used in practice can be factored into a small number of short lifting steps [3], so this limitation is unlikely to interfere with finding good solutions.

```
WAVELET-ESP
input: N, the number of sub-populations
       L, the lengths of the lifting filters
       M, the size of each sub-population
       P, the mutation rate

1. INITIALIZE
Create M filters of length L for each of the N sub-populations, and randomize
them.

2. EVALUATE
Select N lifting steps, one from each sub-population, and evaluate the resulting
wavelet. Add the fitness to the cumulative fitness of all participating steps. Repeat
until each step has been evaluated 10 times on average.

3. RECOMBINE
Rank the lifting steps in each sub-population by their average fitness. Each step in
the top quartile is recombined with a higher-ranking step. The offspring is mutated
with probability P and replaces the lowest-ranking half of each sub-population.

4. REPEAT
Repeat the EVALUATE-RECOMBINE cycle for a fixed number of generations.
```

**Fig. 3.** The ESP algorithm applied to wavelets. Several populations of lifting steps are evolved in parallel, and are combined in the evaluation phase to form wavelets.

**Initialization.** Each chromosome is initialized by setting the values of the coefficients to random values from a gaussian distribution with mean 0 and variance 0.5, and setting the leftmost index of each filter to a random integer between -2 and 2. This reflects the values commonly found in lifting steps.

**Crossover.** The crossover operator performs simple one-point crossover on the coefficients. The integers representing the leftmost indices of the parent filters are randomly assigned to the children.

**Mutation.** A chromosome is mutated by adding low-variance gaussian noise to a random filter coefficient and/or adding $\pm 1$ to the integer representing the leftmost index.

**Fitness Evaluation.** In signal compression, the ideal measure of fitness would be the performance in an actual transform coder as described in section 2.4. However, there are two problems with this approach. First, evaluating a wavelet using a transform coder is almost prohibitively expensive. Second, in order to make a fair comparison between two wavelets, either the available number of bits needs to be fixed and the resulting distortion used as a fitness measure, or vice versa. Both options are inexact and expensive for actual transform coders.

Figure 4 shows a definition of the evaluation function. It is an idealized version of a transform coder: Instead of quantizing and entropy-coding the wavelet coefficients, it uses only a certain percentage of the coefficients for reconstruction and sets the rest to zero. This is much less expensive and allows choosing the compression ratio exactly, which means that the resulting distortion can be used directly as a fitness measure. Villasenor et al. [15] have used a similar but even simpler method to evaluate wavelets with good results.

```
EVALUATION FUNCTION
input:  D, the input data
        W, a candidate wavelet
        R, the compression ratio
return: The fitness of W.

1. TRANSFORM
Transform D using the wavelet W.

2. COMPRESS
Sort the resulting wavelet coefficients. Keep only the largest $R \times |D|$. Set the rest to zero.

3. RECONSTRUCT
Perform an inverse transform using W and the altered wavelet coefficients.

4. MEASURE THE ERROR
Measure the resulting distortion ($L_2$ error) E and return 1/E.
```

**Fig. 4.** The evaluation function is an idealized version of a transform coder: Instead of quantizing and entropy-coding the wavelet coefficients, it uses only part of the coefficients for reconstruction and sets the rest to zero.

## 5 Experiments

The algorithm described in the previous section was evaluated on the task of compressing cubic splines, i.e. 1D-sequences of data sampled from cubic spline curves. Input sequences of length 256 were generated from 16 random control points from the interval $[-1, 1]$. Figure 5 shows an example input curve. This type of input data has several advantages:

1. The optimal compression ratio is known, because the number of random control points is known.

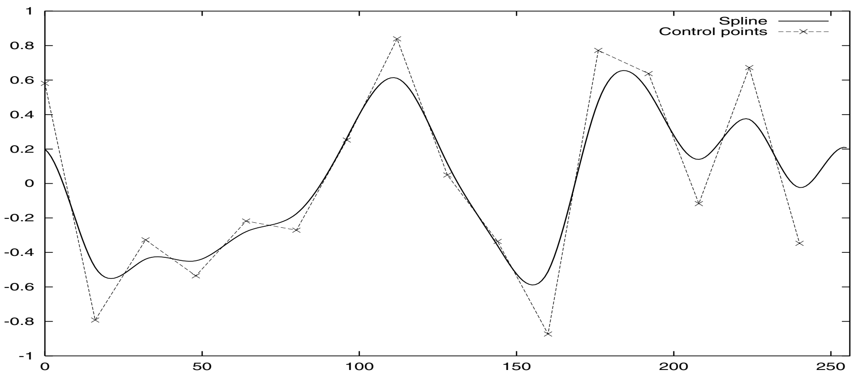

**Fig. 5.** An example for the input data used in the experiment. A spline curve is created from 16 random control points.

2. Optimal wavelets for the task are known (the class of cubic spline wavelets, like the one shown in figure 1).
3. Cubic Splines are known to be an acceptable model for a wide range of input data, including images. Some of the best known wavelets for image compression are spline wavelets [4].

## 5.1 Methodology

The algorithm was run 30 times for 40 generations. After every generation, the best wavelet found so far was evaluated on a test set of 50 spline curves. Fresh test and training data were used for every run.

Preliminary experiments were conducted to determine the best parameter settings. The algorithm turned out to be very robust; similar results were obtained for a wide range of parameters.

The following parameters were used for the reported results: The population size was 400 for each sub-population, which means that 4000 evaluations took place each generation. The algorithm evolved 6 sub-populations in parallel, each of which contained lifting steps of length 2. The mutation rate was 0.4. The evaluation function used 26 of the 256 wavelet coefficients for reconstruction of the signal, i.e. the compression ratio was roughly 10:1.

## 5.2 Results

Figure 6 shows the learning curve (the average performance on the test set) averaged over 30 runs. The bars are 95% confidence intervals for the expected performance.

The horizontal lines show the performance of two well-known orthogonal wavelets [16], and the biorthogonal 9/7 wavelet introduced by Antonini [17], probably the most popular wavelet for image compression. These comparisons

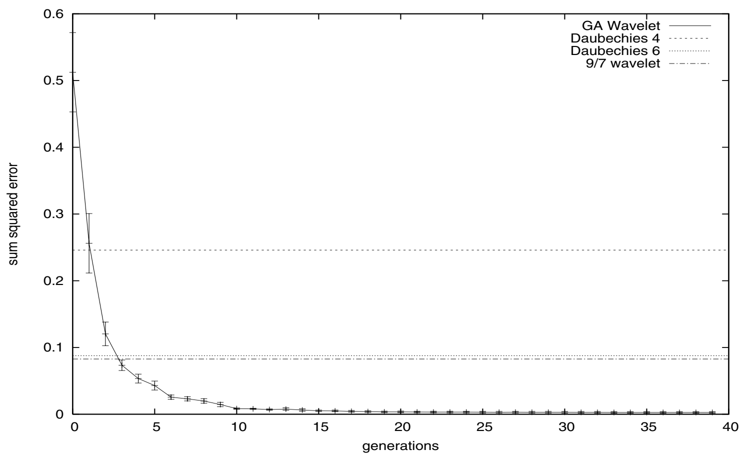

**Fig. 6.** The learning curve (average performance on the test set) averaged over 30 runs. The bars are 95% confidence intervals. The horizontal lines show the performance of some well-known wavelets.

are intended to put the performance of the wavelets into perspective. As mentioned before, the optimal wavelets for this kind of input data are cubic spline wavelets like the one in figure 1. Using such a wavelet would result in zero compression error. Note, however, that for real-world applications, an optimal wavelet is generally not known.

Figure 7 shows some of the best wavelets found by the GA after 40 generations. Note that these wavelets are locally very similar to the wavelet in figure 1, but also that all of them differ in overall shape. These are characteristics shared with actual cubic spline wavelets.

Figure 8 illustrates how the compression performance develops during a typical run. The wavelets used are the winners of generations 1, 5, 10, 20 and 40. The plots on the left compare an example spline curve with the reconstructed version after 10:1 compression. The plots on the right show a rate-distortion curve for the same wavelet, i.e. the sum squared compression error as a function of the percentage of wavelet coefficients used for reconstruction.

The best wavelet of the first generation performs very poorly. Keep in mind, however, that no evolution has taken place yet; we are simply looking at the best wavelet from a random population.

After five generations, the compressed curve is already much closer to the original. The rate-distortion curve shows that the error has been reduced by over an order of magnitude. By generation 10, the compressed curve is even closer to the original curve, and by generation 20, the left plot no longer shows a difference between the two. Between generations 20 and 40, the error decreases by another order of magnitude.

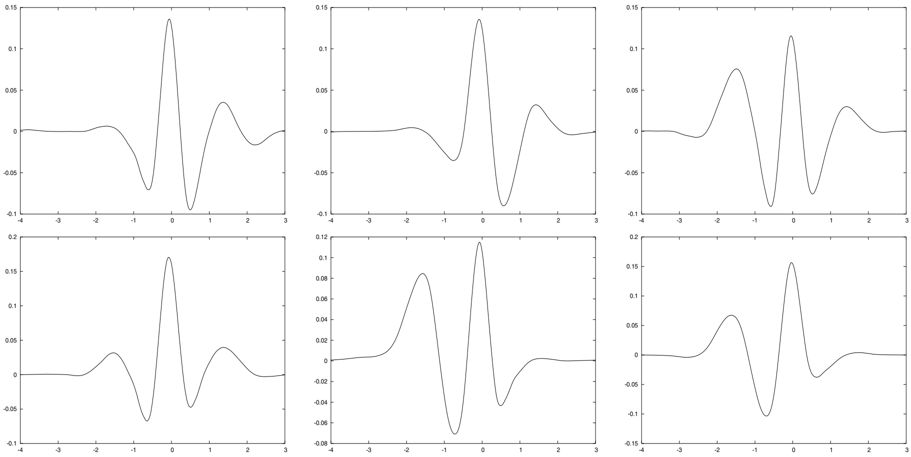

**Fig. 7.** Some of the best wavelets found in different runs of the GA. All 30 runs produced near-optimal wavelets.

The GA has clearly found a near-optimal wavelet for the compression of cubic splines. The winner wavelets from all 30 runs show similar performance.

## 6 Future Work

The next step will be to test the algorithm on real-world data. Current work focuses on the evaluation of the algorithm on a compression application using natural images.

The design of non-separable two-dimensional wavelets has received much attention in the literature (see e.g. [18]). The algorithm presented in this paper could be adapted to this case without major changes.

The most interesting possibility for future research, however, is the use of non-linear lifting predictors. Evolutionary neural networks, for example, could be used instead of the FIR filters used in this paper, which might lead to wavelets with more power to express end exploit regularities in the input data.

## 7 Conclusions

In this paper, we described a coevolutionary genetic algorithm that evolves biorthogonal wavelets encoded as a series of lifting steps. Applied to the task of compressing cubic spline curves, the algorithm consistently found near-optimal wavelets that outperformed some well-known wavelet bases.

The experiments reported in this paper show that the combination of lifting and genetic algorithms provides a powerful framework for adaptive wavelet design. Evolution of sequences of lifting steps seems to be a friendly problem for genetic algorithms, especially for the coevolutionary approach used.

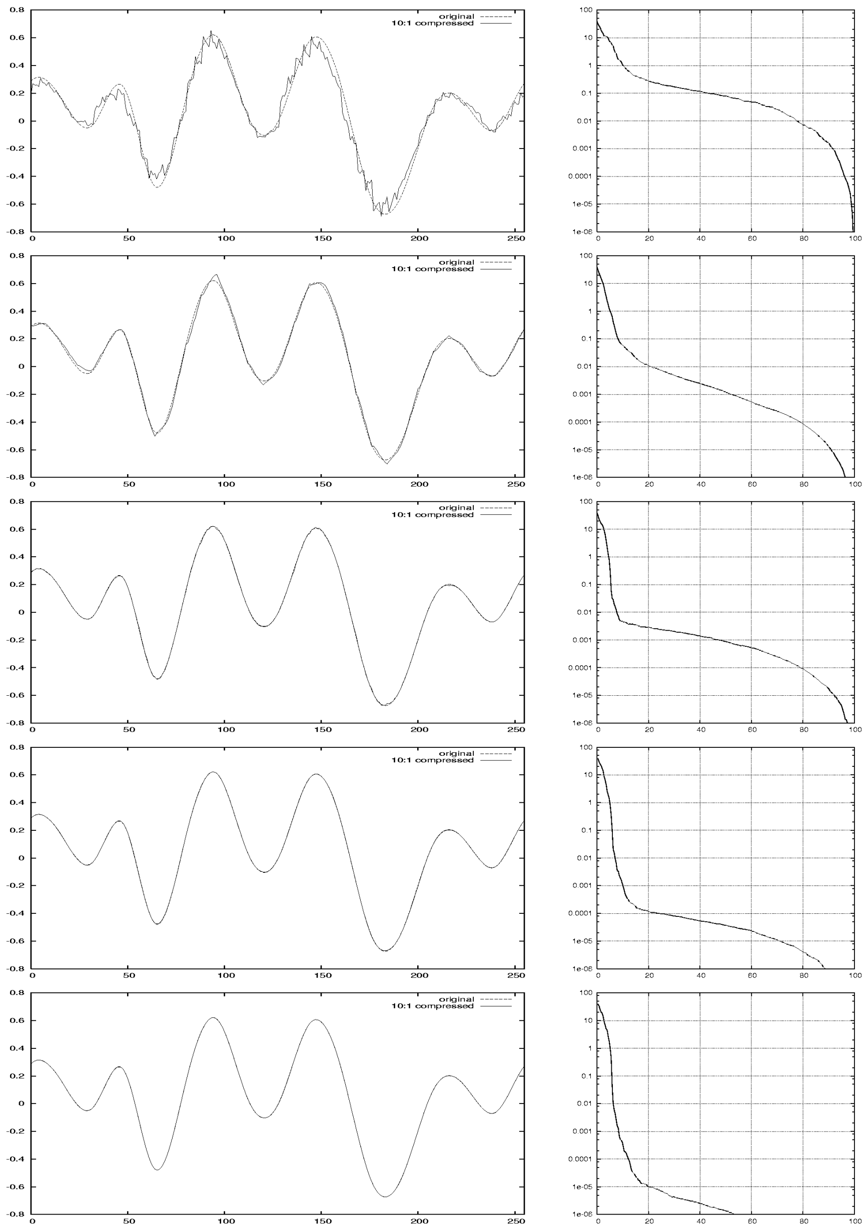

**Fig. 8.** The compression performance of the best wavelets after 1, 5, 10, 20, and 40 generations, from top to bottom. The plots on the left show the original data and the reconstructed data after 10:1 compression. The plots on the right show the distortion as a function of the percentage of coefficients used. By generation 40, the error has dropped by well over 3 orders of magnitude.

## References

1. Gomez, F., Miikkulainen, R.: Solving non-markovian control tasks with neuroevolution. In: Proceedings of the International Joint Conference on Artificial Intelligence, San Francisco, CA (1999) 1356–1361
2. Jawerth, B., Sweldens, W.: An overview of wavelet based multiresolution analyses. SIAM Rev. **36** (1994) 377–412
3. Daubechies, I., Sweldens, W.: Factoring wavelet transforms into lifting steps. Journal of Fourier Analysis and Applications **4** (1998) 245–267
4. Davis, G., Nosratinia, A.: Wavelet-based image coding: An overview. Applied and Computational Control, Signals and Circuits **1** (1998)
5. Sweldens, W.: The lifting scheme: A custom-design construction of biorthogonal wavelets. Journal of Applied and Computational Harmonic Analysis **3** (1996) 186–200
6. Coifman, R., Wickerhauser, V.: Entropy-based algorithms for best basis selection. IEEE Transactions on Information Theory **38** (1992) 713–718
7. Wickerhauser, M.: Adapted Wavelet Analysis from Theory to Software. A. K. Peters, Wellesley, MA (1994)
8. Lankhorst, M.M., van der Laan, M.D.: Wavelet-based signal approximation with genetic algorithms. In: Evolutionary Programming. (1995) 237–255
9. Liu, C., Wechsler, H.: Face recognition using evolutionary pursuit. In: Proceedings of the Fifth European Conference on Computer Vision, Freiburg, Germany (1998)
10. Claypoole, R., Braniuk, R., Nowak, R.: Adaptive wavelet transforms via lifting. Transactions of the International Conference on Acoustics, Speech and Signal Processing (1998) 1513–1516
11. Erba, M., R.Rossi, Liberali, V., Tettamanzi, A.: Digital filter design through simulated evolution. In: Proceedings of the ECCTD 01, Espoo, Finland (2001)
12. Lee, A., Ahmadi, M., Jullien, G., Miller, W., Lashkari, R.: Design of 1-d fir filters with genetic algorithms. In: ISSPA 5th International Symposium. (1999) 955–958
13. Monro, D., Sherlock, B.: Space-frequency balance in biorthogonal wavelets. Transactions of the IEEE Int. Conf. on Image Processing **1** (1997) 624–627
14. Hill, Y., O'Keefe, S., Thiel, D.: An investigation of wavelet design using genetic algorithms. In: Microelectronic Engeneering Research Conference. (2001)
15. Villasenor, J., Belzer, B., Lia, J.: Wavelet filter evaluation for image compression. IEEE Transactions on Image Processing **2** (1995) 1053–1060
16. Daubechies, I.: Orthonormal bases of compactly supported wavelets. Comm. Pure Appl. Math. (1988) 909–996
17. Antonini, M., Barlaud, M., Mathieu, P., Daubechies, I.: Image coding using wavelet transform. IEEE Transactions on Image Processing (1992)
18. Sweldens, W.: The lifting scheme: A construction of second-generation wavelets. SIAM J. Math. Anal. **29** (1997) 511–546