6

Signal Decomposition

WAVELETS AND THE WAVELET TRANSFORM are a recently developed method for signal analysis. In the realm of signal processing, genetic algorithms have mainly been applied to problems in filter design and compression. This chapter discusses the application of genetic algorithms to signal decomposition. The purpose is to extract a set of features characterizing the signal of interest. Often this is realized by decomposing the signal on a set of elementary functions. An example of such a decomposition is the Fourier transform, which decomposes on a basis of harmonic functions. However, in the case of non-stationary signals, i.e., signals whose characteristics change with time, the Fourier transform does not yield a useful characterization of the signal. Such signals can be adequately decomposed on a set of locally supported elementary functions, giving rise to a so-called *time-frequency decomposition*. A genetic algorithm is used to find all parameters of such a decomposition.

The chapter starts with an overview of methods for time-frequency decomposition (Section 6.1), in particular the Short Time Fourier transform and the wavelet transform, and describes the elementary functions used in our decompositions. Section 6.2 describes related work, and Section 6.3 focusses on the adaptation of a genetic algorithm to fit our specific aims and needs. To evaluate these adaptations and to test whether genetic algorithms can be a useful tool in signal analysis, we conduct a series of experiments, described in Section 6.4. Finally, Section 6.5 contains the conclusions drawn from these results.

This chapter was published as [105, 106].

6.1 Time-Frequency Decompositions

It was mentioned in the introduction that the Fourier transform is not to be used in case of non-stationary signals. This is mainly caused by the fact that the harmonic basis functions have global support. For example, a discontinuity occurring somewhere at the time-axis, affects the weights of all basis functions and will therefore be spread out across the whole frequency domain.

A straightforward solution is to multiply the signal with a window function to delimit the signal in time. If this result is Fourier transformed, the information can be interpreted as the spectrum at the time indicated by the position of the window function. The total transform is known as the short-time Fourier transform (STFT), or, in case of a Gaussian window, the Gabor transform [55].

The position of the window function introduces an extra parameter in the STFT. The result can be interpreted as the energy-density vs. the time *and* the frequency; therefore, the STFT

is a *time-frequency decomposition*, giving the energy distribution on the *time-frequency plane*. A common representation is a twodimensional image, where the time is on the *x*-axis, the frequency on the *y*-axis and the STFT result coded as a color or grey-value.

In a general time-frequency decomposition, the signal is decomposed in a set of elementary functions, characterized by a certain position and dimensions in the time-frequency plane. We will call such functions *time-frequency atoms* (tf-atoms). In this section we discuss several methods for time-frequency decomposition and take a look at the characteristics of the corresponding tf-atoms.

6.1.1 Short-Time Fourier Transform

The STFT is the Fourier transform of the product of the signal and a window function $w(t)$.

$$(S_w f)(b,\omega) = \int_{-\infty}^{\infty} f(t)\, e^{-i\omega t}\, \overline{w(t-b)}\, \mathrm{d}t \tag{6.1}$$

The window function is a square integrable function with the special property that $t \cdot w(t)$ is square integrable too. Such a function is well localized in time.

The STFT gives a local spectrum around the time b. It is obvious that for a good time-localization a small window should be used. On the other hand, the window must be large enough to measure a frequency with a certain accuracy. This resembles the uncertainty principle in quantum physics: there is a trade-off between time accuracy and frequency accuracy.

In order to study the localization in more detail, we need to define the position μ_w and the radius σ_w of the window function $w(t)$:

$$\mu_w = \frac{1}{\|w\|_2^2} \int_{-\infty}^{\infty} t\, |w(t)|^2\, \mathrm{d}t\ , \tag{6.2}$$

$$\sigma_w = \frac{1}{\|w\|_2^2} \left\{ \int_{-\infty}^{\infty} (t-\mu_w)^2\, |w(t)|^2\, \mathrm{d}t \right\}^{\frac{1}{2}}\ . \tag{6.3}$$

Let $W(\omega)$ be the Fourier transform of $w(t)$:

$$W(\omega) = \int_{-\infty}^{\infty} w(t) e^{i\omega t}\, \mathrm{d}t\ . \tag{6.4}$$

Similarly, μ_W and σ_W can be defined as the position and the radius in frequency. The function is said to be localized in the *time-frequency window*:

$$[\mu_w - \sigma_w, \mu_w + \sigma_w] \times [\mu_W - \sigma_W, \mu_W + \sigma_W]\ . \tag{6.5}$$

This is a rectangle in the time-frequency plane, whose area has a lower bound due to the uncertainty principle [27]:

$$\sigma_w\, \sigma_W \geq \frac{1}{2}\ . \tag{6.6}$$

Equality can only be obtained using a Gaussian window. This implies that the Gabor transform is the best compromise between time and frequency localization.

For the STFT (6.1), the dimensions of the time-frequency windows become:

$$[b + \mu_w - \sigma_w, b + \mu_w + \sigma_w] \times [\omega + \mu_W - \sigma_W, \omega + \mu_W + \sigma_W]\ . \tag{6.7}$$

6.1.2 Wavelet Transform

Recently, the wavelet transform [27, 86] was introduced as an alternative technique for time-frequency decomposition. The main difference with the windowed Fourier transform is that all basis functions are extracted from a single function (the *mother wavelet*) by shifting in time and scaling.

A *wavelet* $\psi(t)$ is a square integrable function such that the following *admissibility condition* holds:

$$\int_{-\infty}^{\infty} \frac{|\Psi(\omega)|^2}{|\omega|} \, \mathrm{d}\omega < \infty \, , \tag{6.8}$$

where, again, $\Psi(\omega)$ is the Fourier transform of $\psi(t)$. If we assume continuity at $\omega = 0$ the condition implies that $\Psi(0) = 0$. This, in turn, implies that the function must change sign.

From this wavelet we can extract a whole family of equally shaped functions by shifts in time (translation) and scaling (dilation):

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \, \psi\left(\frac{t-b}{a}\right) \quad a \neq 0, b \in \mathbb{R} \, . \tag{6.9}$$

A signal can be decomposed on the members of this family by applying the *wavelet transform*:

$$(W_\psi f)(a,b) = \int_{-\infty}^{\infty} f(t) \, \overline{\psi_{a,b}(t)} \, \mathrm{d}t \, . \tag{6.10}$$

The function can be recovered from the wavelet transform by

$$f(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (W_\psi f)(a,b) \, \psi_{a,b}(t) \, \frac{\mathrm{d}a \, \mathrm{d}b}{a^2} \, . \tag{6.11}$$

The wavelet transform introduces redundancy, because the one-dimensional signal is transformed into a two-dimensional 'coefficient space.' The question arises whether it is possible to recover the original signal from a subset of the wavelet coefficients. If we restrict the class of wavelets this is indeed possible. A sufficient condition is that the family of scaled and shifted copies forms an *orthonormal basis* in the space of square integrable functions. Daubechies [35] has found a countable set of these wavelets $\psi_{a,b}$, where

$$\begin{aligned} a &= 2^{-j} \\ b &= 2^{-j} k \end{aligned} \quad (j,k \in \mathbb{Z}) \, . \tag{6.12}$$

In this case the decomposition reduces to orthogonal projection, and the recovery of the signal from its wavelet-coefficients becomes straightforward:

$$f(t) = \sum_{j,k} c_{jk} \, \psi_{j,k}(t) \, , \tag{6.13}$$

with:

$$c_{jk} = \int_{-\infty}^{\infty} f(t) \, \psi_{j,k}(t) \, \mathrm{d}t \, . \tag{6.14}$$

Let us revisit the localization issue. Assuming that ψ is a window function, we can calculate its time-frequency window. It can be calculated that the wavelet $\psi_{a,b}$ is localized in

$$[b + a\mu_\psi - a\sigma_\psi, b + a\mu_\psi + a\sigma_\psi] \times [\frac{1}{a}\mu_\Psi - \frac{1}{a}\sigma_\Psi, \frac{1}{a}\mu_\Psi + \frac{1}{a}\sigma_\Psi] \, . \tag{6.15}$$

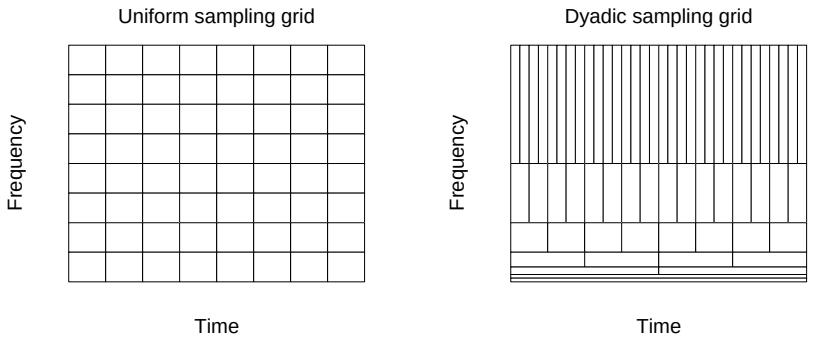

Figure 6.1: Sampling grids for windowed Fourier transform and wavelet transform

Contrary to the windowed Fourier transform (6.1), where the tf-atoms had equal dimensions (6.7), the dimensions of the tf-atoms of the wavelet transform change along the frequency axis. For low frequencies, the windows are wide in time and narrow in frequency, giving good frequency resolution. For high frequency, the windows are wide in frequency and narrow in time, allowing good time resolution. Such a covering of the time-frequency plane is sometimes referred to as *Constant-Q analysis*[1] and is very effective for signal analysis [125]. The specific choice of a and b in (6.12) is related to the dimensions of the time-frequency windows. Figure 6.1 gives the covering of the windowed Fourier transform and the covering according to (6.12): the so-called dyadic sampling grid.

6.1.3 Wave Packets

Despite of the convenient properties of orthogonal wavelets, there was a need for a less restrictive selection of the tf-atoms. Coifman and Meyer [30] have introduced a new class of orthonormal bases called wave packets. Wave packet bases consist of tf-atoms whose time-frequency windows cover the whole plane, but, contrary to wavelet bases, the dimensions are not directly related to the position in the time-frequency plane, see Figure 6.2. In this way it is possible to find a basis which is adapted to the signal. An algorithm implementing this search strategy is the best basis algorithm, introduced by Wickerhauser [156].

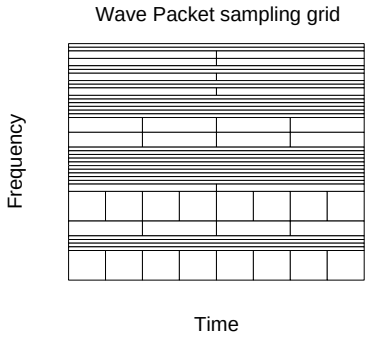

Figure 6.2: A possible sampling grid for wave packets

6.1.4 Matching Pursuit Algorithm

The Matching Pursuit algorithm (MP-algorithm), introduced by Mallat and Zhang [113], is a highly adaptive method for constructing time-frequency decompositions. The general issue is to decompose a function on a set of waveforms, selected appropriately among a large and redundant dictionary.

[1] The **Q** stands for the quality factor of a digital filter, i.e., the ratio between bandwidth and central frequency.

The dictionary $\mathcal{D}$ consists of a finite number of time-frequency atoms g. The algorithm starts with searching the dictionary for the best matching element, i.e., the element g_i which gives a maximum inner product with the signal f. The signal f can be decomposed as

$$f = \langle f, g_i \rangle g_i + Rf \ , \tag{6.16}$$

where Rf is the residual vector after the orthogonal projection on g_i. Note that this residue is orthogonal to g_i. In the next step, the dictionary is searched again for an element g_j best matching the residual, and so on. The decomposition up to n elements is given by:

$$f = \langle f, g_i \rangle g_i + \langle Rf, g_j \rangle g_j + \ldots + R^{n+1} f \ . \tag{6.17}$$

Mallat and Zhang used a dictionary of Gabor functions, by scaling, translating and modulating a Gaussian window function:

$$g_{a,b,\omega}(t) = \frac{1}{\sqrt{|a|}} \, g\left(\frac{t-b}{a}\right) \, e^{i\omega t} \ , \tag{6.18}$$

where

$$g(t) = 2^{\frac{1}{4}} e^{-\pi t^2} \ . \tag{6.19}$$

In (6.18) scaling is controlled by a, translation by b, and modulation by ω.

The Matching Pursuit algorithm is able to decompose a signal into a fixed, pre-defined number of atoms with arbitrary time-frequency windows. This allows a higher degree of adaptation than in the case of wave packets. The tf-plane shows a fixed number of rectangles with constant area but arbitrary dimensions. Moreover, the rectangles are allowed to overlap. In general the plane is not covered completely; the rectangles only need to cover the high energy parts of the signal.

Though the algorithm itself is not linear, the resulting decomposition is a linear combination of tf-atoms. Moreover, there is a major difference with the methods discussed before: The n^{th}-order decomposition has no exact inverse, since the residual is not taken into account. It defines an approximation of the signal with approximation error $R^{n+1} f$. In many cases, the decomposition is continued until the magnitude of the inner-products becomes smaller than a threshold. It is assumed that all significant features of the signal are coded in the n coefficients and that the corresponding residual is only noise.

6.2 Related Work

In the past, the combination of wavelets and evolutionary algorithms has been used in signal processing applications, mainly in the field of filter design [80] and compression. Waagen et al. [151], e.g., use evolutionary programming to select optimal wavelet coefficients for efficient image compression. The approach is based on the optimization of wavelet scale thresholds, and tries to optimize both compression ratio and image quality at the same time.

Another related method is that of Manela et al. [114], which uses a genetic algorithm to fit spline functions to noisy data. A more indirect approach is taken by Rogers [126], which uses a hybrid of the Multivariate Adaptive Regression Splines MARS algorithm of Friedman [53] in which the incremental search for a (linear) spline model of the input data is replaced

by genetic search. An important difference between both methods and our approach is that we do not seek to fit a single function to the input data, but rather a combination of simple, locaky supported basis functions.

6.3 A Genetic Approach to Signal Decomposition

As explained in Section 6.1.4, constructing a time-frequency decomposition of a given signal using a fixed, pre-defined number of atoms with arbitrary time-frequency windows has a number of advantages. To obtain such decompositions, we have employed a genetic algorithm.

In contrast to the MP-algorithm, the optimization process is carried out simultaneously for the whole decomposition. Another difference is that the genetic algorithm operates in the representation domain, i.e., the algorithm manipulates chromosomes without knowing what they represent. As a consequence, the algorithm cannot use domain-specific knowledge, such as the fact that a vector product can be used to find L^2-optimal decompositions.

The resulting decomposition is a linear combination of tf-atoms:

$$\widehat{f} = \sum_{i=1}^{N} c_i g_i \ . \tag{6.20}$$

The tf-atoms g_i in (6.20) are *not* necessarily mutually independent. Again, $\widehat{f}$ can be regarded as an approximation of f.

6.3.1 Choice of the Elementary Function

Any type of elementary function can be used. Since our goal is to obtain a time-frequency decomposition, we would like to use tf-atoms with small tf-windows. Another major consideration is the algebraic complexity of the functions. The following tf-atoms are therefore concentrated on: *Gabor-functions* and *B-splines*.

Gabor functions are used to have maximum flexibility in the parameters of the tf-atoms, retaining the minimal area corresponding to the uncertainty principle (6.6). Instead of the complex function (6.18), the *real* function is used:

$$g_{a,b,\omega,\theta}(t) = g\left(\frac{t-b}{a}\right) \cos(\omega t - \theta) \ , \tag{6.21}$$

where θ is an additional phase parameter. Some examples of (6.21) are given in Figure 6.3. This set of functions covers the so-called Gabor wavelets in which scaling and modulation are coupled to meet (6.9). Note that the factor $1/\sqrt{|a|}$ which occurred in (6.9) is omitted here. Instead, this factor will be coded in the coefficients c_i in (6.20).

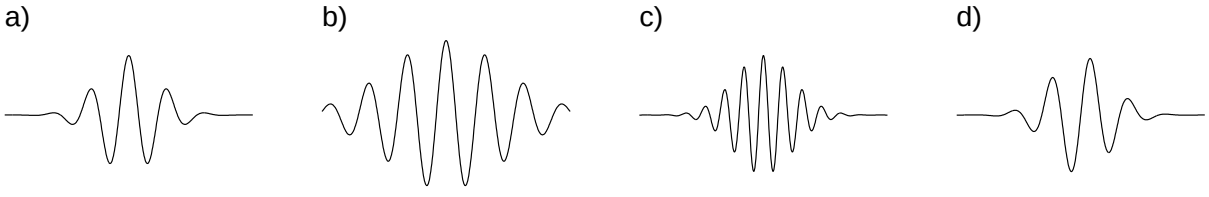

Figure 6.3: The Gabor function (6.21) used in the experiments, for varying parameters:
a) $a = 1.2,\ b = 0,\ \omega = 4,$ and $\theta = 0,$ **b**) same but $a = 2.4,$ **c**) $\omega = 8,$ **d**) $\theta = \frac{\pi}{2}$.

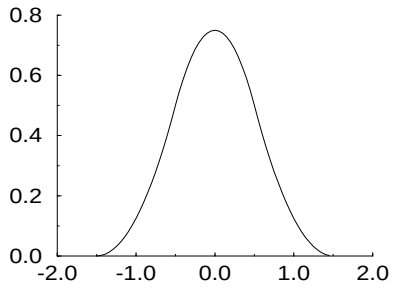

Figure 6.4: The second order *B*-spline.

Polynomial splines are piecewise polynomials which have proven their use in approximation theory. Due to their simple, explicit analytical form they are easy to evaluate and manipulate. A polynomial spline of order n is $n-1$ times continuously differentiable, which guarantees smoothness. Every spline can be represented as a linear combination of *B*-splines, which are smooth functions with compact support. The *B*-spline of order 2 is plotted in Figure 6.4.

Several polynomial spline wavelets are described in the literature. Examples are the *B*-spline wavelets of compact support [27, 146]. However, these wavelets are constructed by a finite linear combination of *B*-splines. Therefore, the *B*-splines themselves are used as tf-atoms in the experiments.

B-splines can be defined in a recursive way:

$$B_m(t) = (B_{m-1} * B_0)(t) \ , \tag{6.22}$$

where the $*$ denotes convolution and $B_0(t)$ is defined as:

$$B_0(t) = \begin{cases} 1 & \text{if } t \in [-\frac{1}{2}, \frac{1}{2}) \ , \\ 0 & \text{otherwise.} \end{cases} \tag{6.23}$$

The set of tf-atoms is now defined by translation and dilation:

$$g_{a,b}(t) = B_2\left(\frac{t-b}{a}\right) \ . \tag{6.24}$$

To employ a genetic algorithm to find suitable parameters for a set of elementary functions that approximate a given signal, the standard genetic algorithm as mentioned in Chapter 2 must be adapted to the specific features of this problem:

Representation: The parameters of the elementary functions have to be represented as chromosomes. A simple way to encode the parameters in the form of bitstrings is given in Section 6.3.2. This encoding has a fixed precision; an adaptive extension of this encoding is also given.

Genetic operators: In defining a set of operators on a bitstring representation, there are two choices. On the one hand, standard binary operators can be used, and on the other hand, special-purpose real-number operators can be employed.

Fitness evaluation: A fitness function to evaluate the quality of the individual population members must be defined. As described in Section 6.3.4, the fitness functions used are based on different norms of the difference between the given signal and its approximation.

Experiments to evaluate these choices are given in Section 6.4.

6.3.2 Representation

If we want to optimize approximations of signals, it is necessary to represent the real-valued coefficients of the tf-atoms as genes; each chromosome should represent a set of functions that approximates the given signal. In any case, the real numbers will be encoded as bitstrings at the lowest level. We can, however, select our genetic operators to work at the bitstring level, or use specialized real-number operators.

The bitstring representation of the real-valued coefficients is straightforward. Each gene has three parameters that determine its mapping into a real number:

- upper limit p;
- lower limit q;
- number of bits ℓ.

Using these parameters, every coefficient w is encoded as:

$$e(w) = \lfloor (2^\ell - 1) \cdot (w - p)/(q - p) \rfloor \ , \tag{6.25}$$

and decoded as:

$$d(e(w)) = p + (q - p) \cdot e(w)/(2^\ell - 1) \ . \tag{6.26}$$

In our experiments, we employed two types of elementary functions: B_2-splines and Gabor functions. B_2-splines have three parameters: dilation a, translation b, and amplitude c. Gabor functions have two more parameters: phase shift θ and modulation ω. To encode a parameter using the linear scheme of (6.25), we need to give a range $[p, q]$.

Assume without loss of generality that the signal f is defined on the time interval $[0, T]$, and sampled at distance Δt, and define $f_{\min} = \min\{ f(t) \mid t \in [0, T] \}$, $f_{\max} = \max\{ f(t) \mid t \in [0, T] \}$, and $r = f_{\max} - f_{\min}$.

The dilation a is encoded exponentially to obtain a uniform distribution of frequencies across the octaves. This resembles the dyadic grid of (6.12) and Figure 6.1. A lower bound of a is such that the shape of the elementary function is still preserved if the function is represented on the sampled interval: its time window must exceed Δt. An upper bound is chosen—rather arbitrarily—such that the function covers the complete time interval.

The translation b determines the position of the elementary function on the time interval. Since the functions extend on both sides of the center with an amount proportional to the dilation a, the bounds of b must depend on a. To guarantee a significant contribution of an elementary function to the overall approximation, it is assumed that the overlap between the time interval of the signal and the time window of the elementary function must be at least 25%, which implies the bounds given below.

For the amplitude $c/\sqrt{a}$ of the elementary functions, one would expect the extremes of the amplitude to be equal to the extremes of the signal. However, different elementary functions might partly cancel out each other. Therefore bounds are chosen such that the range of the amplitude is twice the range of the signal. Again this choice is somewhat arbitrary.

In the case of Gabor functions, the maximum value of ω is determined from the Nyquist criterion, which relates the sampling interval to the maximum frequency present in the sampled signal. The minimum value of ω is 0 to allow unmodulated Gaussians. The bounds of the phase parameter θ are straightforward.

The considerations above lead to the following ranges of the three B_2-spline parameters:

$$\begin{aligned} \log_2 a &\in [-1, 8] \ , \\ b &\in [-0.75a, T + 0.75a] \ , \\ c/\sqrt{a} &\in [f_{\min} - r/2, f_{\max} + r/2] \ . \end{aligned} \tag{6.27}$$

The ranges of the five parameters of the Gabor functions are:

$$\begin{aligned} \log_2 a &\in [-1, 8] \ , \\ b &\in [-a, T + a] \ , \\ c/\sqrt{a} &\in [f_{\min} - r/2, f_{\max} + r/2] \ , \\ \theta &\in [-\pi/2, \pi/2] \ , \\ \omega &\in [0, \pi/\Delta t] \ . \end{aligned} \tag{6.28}$$

For both function types, values in the intervals listed on the right-hand sides in (6.28) and (6.29) are encoded as shown in (6.25). A chromosome encoding a set of m tf-atoms thus consists of $3m$ genes in the case of B_2-splines, and $5m$ genes for Gabor functions.

The truncation in the encoding (6.25) limits the resolution that can be achieved. However, this resolution can be increased by using Dynamic Parameter Encoding (DPE) [131]. DPE, described in Section 2.4.3, implements a strategy of iterative precision refinement by gathering convergence statistics of the top two bits of each (Gray coded) parameter.

6.3.3 Genetic Operators

Selection

The selection is carried out by a stochastic universal sampling algorithm [9], using windowed fitness scaling as described in [69] and Section 2.4.1. In the experiments described in Section 6.4, a window of size of 5 is used. Furthermore, an elitist strategy is employed in which the best individual of the population always survives to the next generation.

Mutation

Three types of real-number mutation are used:

1. Random mutation: a real number $r \in [p, q]$ is mapped to another real number $r' \in [p, q]$ using a uniform random distribution;
2. Gaussian big creep: a real number $r \in [p, q]$ is mapped to $r' = r + N(0, \sigma_{\text{big}})$ where $\sigma_{\text{big}} = 0.2 \cdot (q - p)$;
3. Gaussian little creep: a real number $r \in [p, q]$ is mapped to $r' = r + N(0, \sigma_{\text{little}})$ where $\sigma_{\text{little}} = 0.01 \cdot (q - p)$.

The latter two are variants of Davis' big creep and little creep operators [39] (see also Section 2.4.3). Each of these types has its own merits. Random mutation is useful to maintain a certain level of genetic diversity in the population to prevent premature convergence, big creep makes jumps in a fairly large neighborhood of a point in the search space, and little creep serves to optimize already good values by making small changes.

In the experiments comparing bit to real-number operators (Section 6.4.3), standard binary mutation as explained in Chapter 2 is also used.

Recombination

Standard one-point, two-point, and uniform crossover can of course be used in case of real-valued genes as well. However, we can also construct operators that make use of the special features of the real numbers. For example, if we have two reasonably good values for some gene, a linear combination of these values might do even better. This is the motivation behind arithmetical crossover.

Arithmetical crossover as defined in [115] takes two (real-valued) parent genes s and t and computes their offspring genes s' and t' as a linear combination of the parents' values:

$$\begin{aligned} s' &= a \cdot s + (1-a) \cdot t \ , \\ t' &= (1-a) \cdot s + a \cdot t \ , \end{aligned} \tag{6.29}$$

using a parameter $a \in [0,1]$.

In our case, for each individual gene participating in the crossover, the parameter a is a uniformly random choice from the interval $[0,1]$. We use a combination of two-point and arithmetical crossover: two crossover points are chosen randomly, and the arithmetical crossover takes place between these two points.

6.3.4 Fitness Evaluation

To evaluate the quality of the individuals in the population, we need a fitness function. In the case of signal approximation, the natural choice is to use some norm of the difference between the signal f and the approximation $\hat{f}$ as encoded on the chromosome. In the experiments, the L^1, L^2, L^4, and L^∞ norms are used:

$$\begin{aligned} L^1 &: \|f-\hat{f}\|_1 &&= \sum_i |f(i)-\hat{f}(i)| \ , \\ L^2 &: \|f-\hat{f}\|_2 &&= \left\{ \sum_i |f(i)-\hat{f}(i)|^2 \right\}^{\frac{1}{2}} \ , \\ L^4 &: \|f-\hat{f}\|_4 &&= \left\{ \sum_i |f(i)-\hat{f}(i)|^4 \right\}^{\frac{1}{4}} \ , \\ L^\infty &: \|f-\hat{f}\|_\infty &&= \sup_i |f(i)-\hat{f}(i)| \ . \end{aligned} \tag{6.30}$$

The norm of the error is divided by the signal norm $\|f\|$ to produce a relative error measure. This value is to be minimized by the algorithm, and the corresponding fitness function is computed as explained in Section 6.3.3.

6.4 Experiments

6.4.1 Motivation

In Section 6.3 we discussed the different choices with respect to representation, genetic operators, and fitness functions of the genetic algorithm. To evaluate the assets and drawbacks of these possibilities, a set of experiments was conducted on a series of test signals.

Section 6.4.3 describes the experiments regarding the different representations and operators, and the experiments of Section 6.4.4 are used to evaluate the different fitness functions. To get an indication of the quality ofthe genetic algorithm, the Matching Pursuit algorithm given in Section 6.1.4 was run on one of the test signals (Section 6.4.5).

The overall goal of these experiments is to conclude whether genetic algorithms might be a useful instrument in signal approximation. We do not pretend to construct an algorithm that can be used directly in real-life signal approximation tasks, but we rather want to get a general idea of the advantages and disadvantages of the application of GAs in this domain.

6.4.2 Test Data

The algorithm was tested on three signals. Signal 1, depicted in Figure 6.5, is constructed of B_2-splines with arbitrary translations and dilations, sampled on 256 points. It is continuously differentiable which guarantees a certain smoothness, and it has no oscillating character. Specific difficulties include the non-zero left endpoint, the complex tilings of splines near $t = 150$ and $t = 170$, and the small positive bump near the right endpoint. It is obvious that we approximate this signal using B_2-splines.

The second signal, shown in Figure 6.6, is built from modulated Gaussians, with arbitrary a, b, c, ω, and θ, also sampled at 256 points. It possesses an oscillating character, mainly at medium frequencies. The complex tiling of Gaussians gives rise to high-frequency transients around $t = 5$ and $t = 240$. This signal is approximated with Gabor functions.

The third signal, a part of a real-life gas-chromatogram, is shown in Figure 6.7. The main difficulty of this signal is the asymmetric shape of its three isolated peaks, which cannot be modeled by a single, symmetrical elementary function per peak. Since the signal has no oscillations, we use B_2 splines in the approximation.

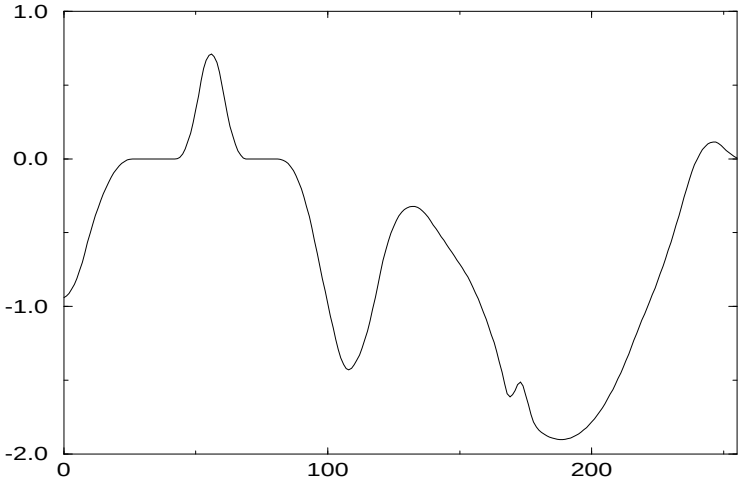

Figure 6.5: Signal 1. Artificial, constructed from B_2-splines. 256 samples.

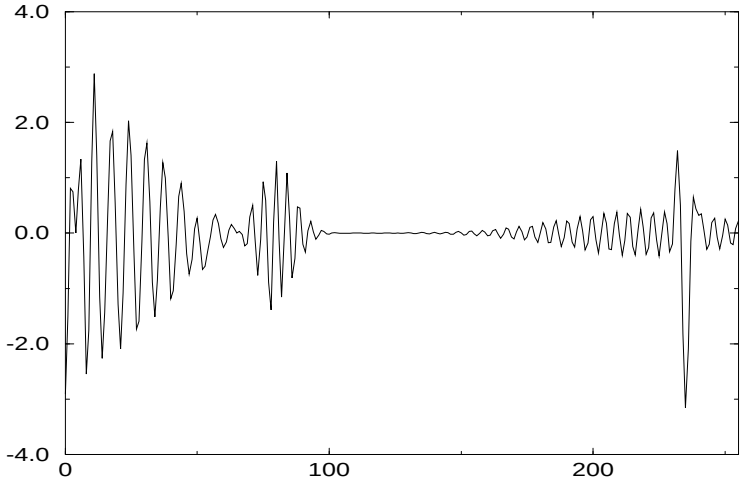

Figure 6.6: Signal 2. Artificial, constructed from modulated Gaussians. 256 samples.

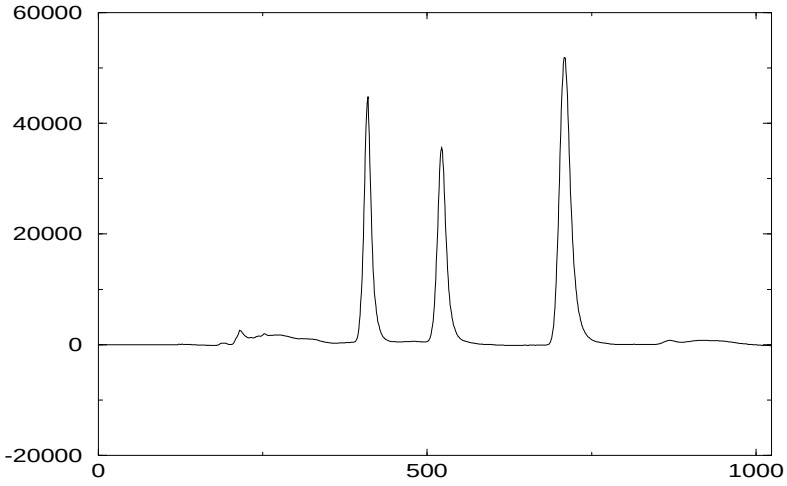

Figure 6.7: Signal 3. Real-life gas-chromatographic data. 1024 samples.

6.4.3 Results Regarding Representation

Binary versus Real Operators

We compared the performance of the GA using binary and real-number operators, for an L^2 spline approximation of signal 1 with 10 B_2-splines and a 16-bit encoding of the genes, and a population size of 50. Standard two-point crossover and bit mutation (crossover rate 0.9, mutation rate 0.002, following the $1/\ell$ heuristic of [116]) are used in the first case, and real-number mutation and crossover as described in Sections 6.3.3 and 6.3.3 in the second, applying mutation types 1, 2, and 3 in a ratio of 1 : 3 : 6, respectively, and an overall mutation rate of 0.03. A graph of the behavior of the GA using bit versus real-number operators, averaged over five runs, is shown in Figure 6.8, and the results are summarized in Table 6.1. In this and subsequent tables and figures, the error bound is determined from the 95% confidence

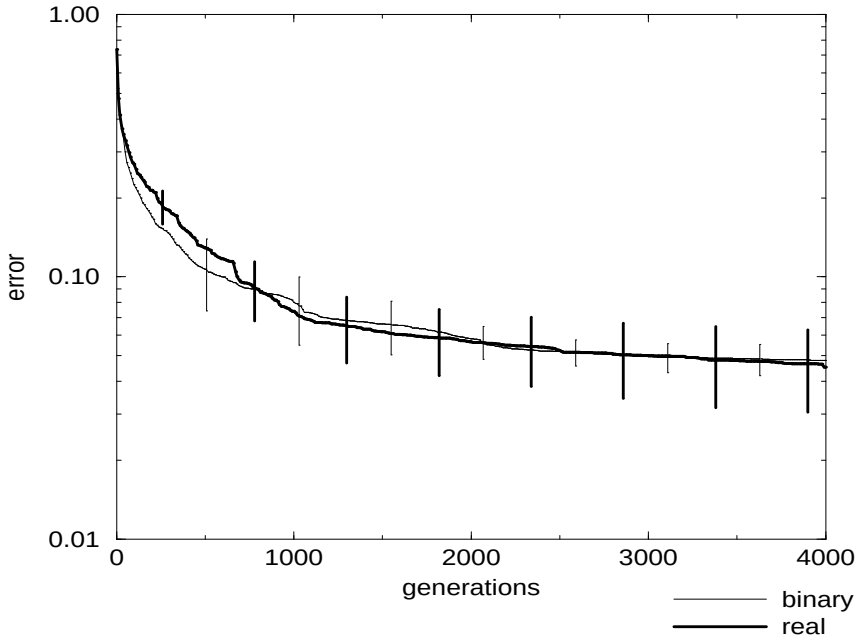

Figure 6.8: L^2 approximation of signal 1. Relative error of best population member vs. generations, averaged over 5 runs, using bit and real-number operators. Vertical lines denote the 95% confidence interval.

	generations	average
bit	4000	$4.80 \cdot 10^{-2} \pm 7.7 \cdot 10^{-3}$
real	4000	$4.53 \cdot 10^{-2} \pm 1.6 \cdot 10^{-2}$

Table 6.1: Relative L^2 norms of the error signal for a B_2-spline approximation of signal 1. Results over 5 runs, using bit and real-number operators.

interval given by a standard t-test.

The outcome of this experiment does not show conclusive evidence in favor of either of the two approaches. The binary approach seems to cause faster convergence, but the end results are slightly worse. The statistical significance is low, however, but we decided to use real-number operators because of their greater flexibility. The next section will show how the different real-number mutation types can be exploited in a favorable manner.

Fixed versus Adaptive Mutation Ratios

Real-number mutation allows us to vary the ratio of the frequencies with which different mutation types are applied. In the first generations, the distribution of values in the population is fairly random, so random mutation is appropriate. As the population starts to converge, most

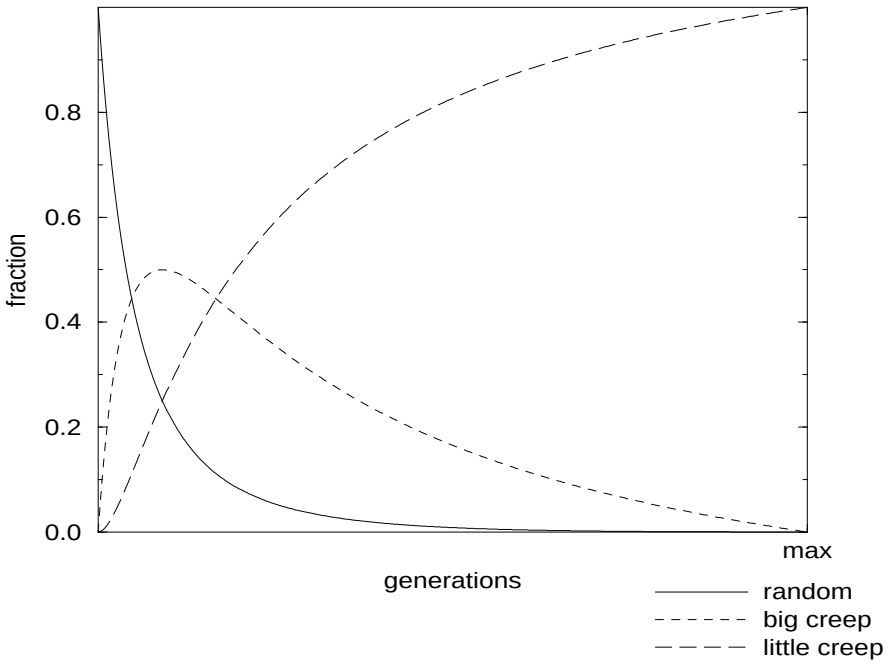

Figure 6.9: Proportion of the different mutation types

genes tend to get better values, and big creep mutation becomes preferable. Finally, when nearly all genes are close to an optimal value, little creep mutation is desirable. Considering this, we designed the adaptive scheme depicted in Figure 6.9. This figure shows the distribution of the relative mutation probabilities used over different generations, for the three mutation types of Section 6.3.3. The total mutation probability remains constant, but the relative application frequencies of the three types change over time.

A similar idea has been employed by Michalewicz, who used a non-uniform mutation operator with mutation values from an interval that decreased exponentially over time in solving a linear-quadratic dynamic control problem [115].

Another approach is Davis' adaptive operator fitness [38]. During a run of his genetic algorithm, performance statistics of all operators (including crossover) are collected, and the relative frequency of each operator is adapted based on these statistics. This scheme improved the performance of his algorithm, but incurred the extra computational burden of computing the operator statistics.

To judge the performance of our adaptive scheme, we compared the performance of the genetic algorithm in both cases, for an L^2 spline approximation of signal 1 with 10 B_2-splines and a 16-bit encoding of the genes. In the fixed case, mutation types 1, 2, and 3 were applied in a ratio of $1 : 3 : 6$, respectively, and the adaptive case employed the ratios of Figure 6.9.

A mutation probability of 0.03 was used in both cases, chosen after some preliminary testing. A graph of the convergence of the algorithm for both cases, averaged over five runs, is shown in Figure 6.10. This shows a small advantage of the adaptive scheme. It seems to converge a little faster and result in slightly better solutions.

However, the statistical significance of these results is low. As shown in Table 6.2, the average of the best values obtained in the five runs is $4.53 \cdot 10^{-2}$ in the fixed case, and $4.01 \cdot 10^{-2}$

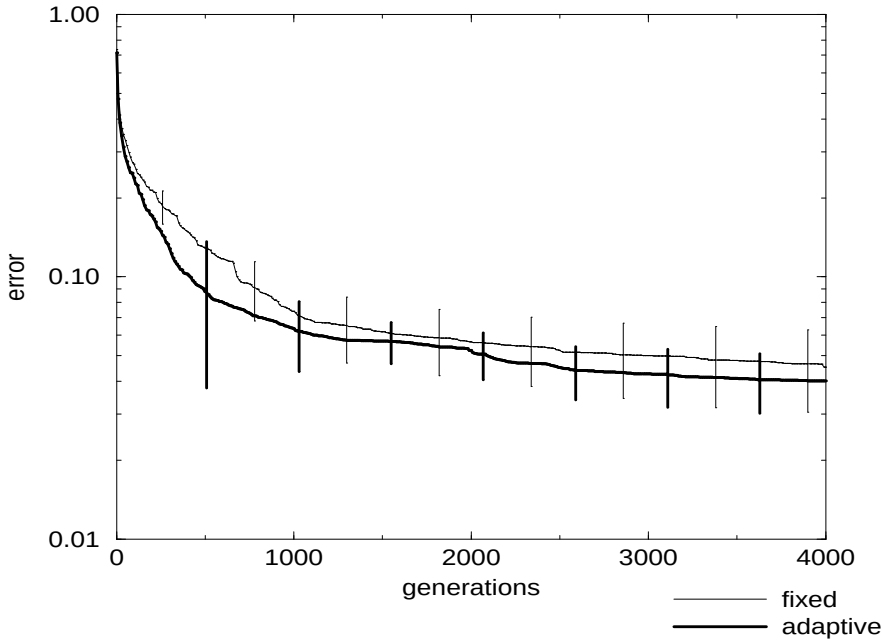

Figure 6.10: L^2 approximation of signal 1. Relative error of best population member vs. generations, averaged over 5 runs, using fixed and adaptive mutation ratios. Vertical lines denote the 95% confidence interval.

	generations	average
fixed	4000	$4.53 \cdot 10^{-2} \pm 1.6 \cdot 10^{-2}$
adaptive	4000	$4.01 \cdot 10^{-2} \pm 1.0 \cdot 10^{-2}$

Table 6.2: Relative L^2 norms of the error signal for a B_2-spline approximation of signal 1. Results over 5 runs, using fixed and adaptive mutation ratios.

in the adaptive case, but the standard deviations are $1.30 \cdot 10^{-2}$ and $8.40 \cdot 10^{-3}$, respectively, giving the 95% confidence intervals as shown. Computational limitations made it impossible to improve these results by doing more simulations.

Dynamic Parameter Encoding

To assess the value of the Dynamic Parameter Encoding described in Section 6.3.2 and Section 2.4.3, an experiment was conducted for an L^2 spline approximation of signal 1 with 10 B_2-splines, using a 16-bit encoding with and without DPE. Figure 6.11 shows a graph of the convergence of the algorithm for these encodings, averaged over five runs. The behavior of both cases is summarized in Table 6.3.

The only significant difference between both cases is the much higher standard deviation of

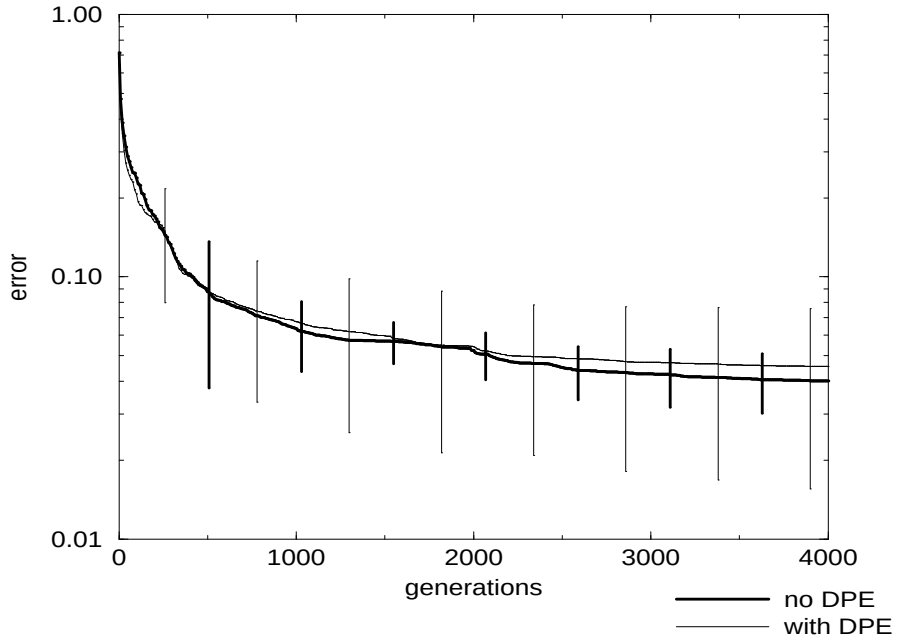

Figure 6.11: L^2 approximation of signal 1. Relative error of best population member vs. generations, averaged over 5 runs, using a 16-bit encoding with and without DPE. Vertical lines denote the 95% confidence interval.

	generations	average
without DPE	4000	$4.01 \cdot 10^{-2} \pm 1.6 \cdot 10^{-2}$
with DPE	4000	$4.55 \cdot 10^{-2} \pm 3.0 \cdot 10^{-2}$

Table 6.3: Relative L^2 norms of the error signal for a B_2-spline approximation of signal 1. Results over 5 runs.

the runs when using DPE. This is caused by the fact the DPE is more vulnerable to premature convergence. Convergence to a local optimum is more likely to occur using DPE, since the search space is reduced by the zoom operator, and the global optimum might occur outside the remaining subspace. This is the cause of the high standard deviation using DPE. Since using DPE does not result in more accurate solutions, we may conclude that the 16-bit encoding without DPE suffices. Therefore, DPE was not used in the subsequent experiments.

6.4.4 Approximation Results

To assess the quality of the approximations obtained by the genetic algorithm, it was tested on the three test signals of Section 6.4.2, using real operators (see Section 6.3.3) and the adaptive mutation scheme described in Section 6.4.3. The different parameters used in the experiments

are listed in Table 6.4. Throughout all experiments, a crossover rate of 0.9 was used. The mutation rates, as listed in Table 6.4, were determined after some preliminary testing.

Wavelet	atoms	no. of params	mut. rate
B_2-spline	10	30	0.03
B_2-spline	15	45	0.02
Gabor-function	10	50	0.03
Gabor-function	15	75	0.02

Table 6.4: Mutation rate

The results of the algorithm with these parameters are shown in Table 6.5. Signals 1 and 3 were approximated using 10 and 15 B_2-splines, and signal 2 was approximated with 10 and 15 Gabor functions. In all cases, the population contained 50 individuals, determined after some preliminary tests.

Norm	signal	atoms	generations	best	average
	1	10	4000	$2.85 \cdot 10^{-2}$	$5.45 \cdot 10^{-2} \pm 3.9 \cdot 10^{-2}$
	1	15	6000	$2.71 \cdot 10^{-2}$	$5.10 \cdot 10^{-2} \pm 3.9 \cdot 10^{-2}$
L^1	2	10	4000	$3.77 \cdot 10^{-1}$	$4.89 \cdot 10^{-1} \pm 3.3 \cdot 10^{-1}$
	2	15	6000	$3.11 \cdot 10^{-1}$	$5.32 \cdot 10^{-1} \pm 1.8 \cdot 10^{-1}$
	3	10	4000	$1.08 \cdot 10^{-1}$	$1.31 \cdot 10^{-1} \pm 4.0 \cdot 10^{-2}$
	3	15	6000	$8.25 \cdot 10^{-2}$	$9.93 \cdot 10^{-2} \pm 3.2 \cdot 10^{-2}$
	1	10	4000	$2.93 \cdot 10^{-2}$	$4.01 \cdot 10^{-2} \pm 1.7 \cdot 10^{-2}$
	1	15	6000	$2.94 \cdot 10^{-2}$	$3.06 \cdot 10^{-2} \pm 1.8 \cdot 10^{-2}$
L^2	2	10	4000	$2.28 \cdot 10^{-1}$	$3.35 \cdot 10^{-1} \pm 2.5 \cdot 10^{-1}$
	2	15	6000	$2.38 \cdot 10^{-1}$	$3.09 \cdot 10^{-1} \pm 4.0 \cdot 10^{-2}$
	3	10	4000	$4.87 \cdot 10^{-2}$	$6.19 \cdot 10^{-2} \pm 2.3 \cdot 10^{-2}$
	3	15	6000	$3.26 \cdot 10^{-2}$	$4.95 \cdot 10^{-2} \pm 2.6 \cdot 10^{-2}$
	1	10	4000	$3.99 \cdot 10^{-2}$	$4.53 \cdot 10^{-2} \pm 4.0 \cdot 10^{-3}$
	1	15	6000	$3.69 \cdot 10^{-2}$	$4.42 \cdot 10^{-2} \pm 1.1 \cdot 10^{-2}$
L^4	2	10	4000	$2.11 \cdot 10^{-1}$	$2.59 \cdot 10^{-1} \pm 7.4 \cdot 10^{-2}$
	2	15	6000	$1.93 \cdot 10^{-1}$	$2.37 \cdot 10^{-1} \pm 1.4 \cdot 10^{-2}$
	3	10	4000	$3.27 \cdot 10^{-2}$	$4.16 \cdot 10^{-2} \pm 2.4 \cdot 10^{-2}$
	3	15	6000	$2.53 \cdot 10^{-2}$	$3.63 \cdot 10^{-2} \pm 8.6 \cdot 10^{-3}$
	1	10	4000	$5.09 \cdot 10^{-2}$	$7.52 \cdot 10^{-2} \pm 4.2 \cdot 10^{-2}$
	1	15	6000	$4.80 \cdot 10^{-2}$	$6.83 \cdot 10^{-2} \pm 2.2 \cdot 10^{-2}$
L^∞	2	10	4000	$2.77 \cdot 10^{-1}$	$4.94 \cdot 10^{-1} \pm 3.6 \cdot 10^{-1}$
	2	15	6000	$3.77 \cdot 10^{-1}$	$4.94 \cdot 10^{-1} \pm 4.0 \cdot 10^{-1}$
	3	10	4000	$3.76 \cdot 10^{-2}$	$4.82 \cdot 10^{-2} \pm 2.3 \cdot 10^{-2}$
	3	15	6000	$3.16 \cdot 10^{-2}$	$3.79 \cdot 10^{-2} \pm 8.8 \cdot 10^{-3}$

Table 6.5: Relative norms of the error signal. Results over 5 runs.

To assess the quality of the different fitness functions (i.e., norms), we did a cross check by calculating all different norms for approximations optimized with respect to a particular norm. The results, for signals 2 and 3, are listed in Tables 6.6 and 6.7, respectively.

	tested			
optimized	L^1	L^2	L^4	L^∞
L^1	$1.07 \cdot 10^{-1}$	$5.22 \cdot 10^{-2}$	$3.69 \cdot 10^{-2}$	$3.49 \cdot 10^{-2}$
L^2	$\underline{1.01 \cdot 10^{-1}}$	$\underline{4.87 \cdot 10^{-2}}$	$3.50 \cdot 10^{-2}$	$3.74 \cdot 10^{-2}$
L^4	$1.26 \cdot 10^{-1}$	$5.42 \cdot 10^{-2}$	$\underline{3.27 \cdot 10^{-2}}$	$\underline{3.10 \cdot 10^{-2}}$
L^∞	$1.60 \cdot 10^{-1}$	$7.43 \cdot 10^{-2}$	$4.88 \cdot 10^{-2}$	$3.76 \cdot 10^{-2}$

Table 6.6: Evaluation of relative norms on signal 2, using 10 Gabor functions

	tested			
optimized	L^1	L^2	L^4	L^∞
L^1	$3.77 \cdot 10^{-1}$	$5.64 \cdot 10^{-1}$	$7.82 \cdot 10^{-1}$	$9.98 \cdot 10^{-1}$
L^2	$\underline{2.19 \cdot 10^{-1}}$	$\underline{2.28 \cdot 10^{-1}}$	$2.68 \cdot 10^{-1}$	$3.45 \cdot 10^{-1}$
L^4	$3.64 \cdot 10^{-1}$	$2.59 \cdot 10^{-1}$	$\underline{2.11 \cdot 10^{-1}}$	$\underline{2.12 \cdot 10^{-1}}$
L^∞	$6.19 \cdot 10^{-1}$	$4.63 \cdot 10^{-1}$	$3.81 \cdot 10^{-1}$	$2.77 \cdot 10^{-1}$

Table 6.7: Evaluation of relative norms on signal 3, using 10 B_2-splines

In both tables, the best scores are underlined, and we would expect these to be on the main diagonal. However, in both tables it can be seen that the L^2 and L^4 norms do better than the L^1 and L^∞ norms. As expected, both L^2 and L^4 are the best norms if one wants to optimize the L^2 or L^4 norms of the approximations. However, the L^2 results score better than L^1 on L^1-optimization, and likewise for L^4 versus L^∞. Possibly, L^∞ pays too much attention to local errors and neglects the global behavior of the approximation, whereas L^1 has a too global character. This may lead us to the conclusion that L^2 and L^4 norms are best suited for our goals.

A potential problem of the genetic algorithm is the fact that a superposition of two or more tf-atoms may be necessary to approximate a (part of) a signal. After a few generations, different chromosomes are likely to contain similar tf-atoms that loosely approximate the same feature of the signal, but these atoms do not necessarily occupy the same position on the chromosomes. Crossover of two such chromosomes and subsequent optimization through mutations can easily result in a chromosome that contains several tf-atoms that, superposed, approximate a single signal feature, but each atom on its own forms a very poor approximation. Crossover on such a chromosome may separate these atoms, which results in a low fitness of the offspring. This dependency between genes is known in biology as *epistasis*: the fitness of a chromosome cannot be expressed as a linear combination of the fitnesses of the individual genes. Epistasis is also a feature of so-called deceptive problems (see Section 2.3.4), and can adversely affect the behavior of the genetic algorithm.

A second problem of the algorithm is the large amount of time consumed. Well over 80% of the total number of instructions is spent on fitness evaluation. Evaluating a single approxima-

tion of signal 1 (256 samples) with 10 B_2-splines takes the equivalent of approx. $1.5 \cdot 10^5$ floating point instructions (3.3 ms per evaluation on a HP735/99 rated at 45.4 Linpack MFlops), and the Gabor functions take even more time. For a population of 50, each generation takes 50 times as much, and several thousands of generations cost in the order of 10^{10} to 10^{11} instructions, i.e., around 15 min. net run time. On a heavily loaded multi-user machine, each run may take as much as an hour. Furthermore, the execution time scales linearly with the number of samples; hence, longer input signals require proportionally more time.

6.4.5 Results of the Matching Pursuit Algorithm

To judge the quality of our results, we applied the Matching Pursuit algorithm described in Section 6.1.4 to some of our test problems, using Zhang's Matching Pursuit Package[2] (MPP). Since this algorithm is based on the L^2 norm and because the implementation does not support B_2-spline approximation, we only compared the results regarding signal 2, using Gabor functions and the L^2 norm.

Table 6.8 lists the results of this algorithm. It is clear that its approximation results are somewhat better than ours. The MP-algorithm uses the fact that the inner product of the signal and one of the tf-atoms can be used directly to assess the contribution of that atom to an L^2-approximation. The GA does not make use of such knowledge; however, it can be used for any type of approximation.

Nr.	L^1	L^2	L^4	L^∞
10	$1.7843 \cdot 10^{-1}$	$1.5795 \cdot 10^{-1}$	$1.4998 \cdot 10^{-1}$	$1.5475 \cdot 10^{-1}$
15	$1.3490 \cdot 10^{-1}$	$1.0417 \cdot 10^{-1}$	$8.9135 \cdot 10^{-2}$	$8.3323 \cdot 10^{-2}$

Table 6.8: The Matching Pursuit algorithm. Relative error norms on signal 2.

Unfortunately, we cannot compare both algorithms on B_2-spline approximation. From Table 6.5 it can be seen that the results of our algorithm are an order of magnitude better for the B_2-spline case than for the Gabor case. Whether this difference also holds for the MP-algorithm remains uncertain.

Another significant difference with the GA approach is that the MP-algorithm takes a net run time in the order of seconds on the problems in which the GA needs several minutes, a difference of nearly two orders of magnitude.

6.5 Discussion

In the previous sections, we have described a genetic algorithm that is able to infer wavelet approximations of sampled input. The algorithm gives reasonably accurate solutions for the three test signals—a non-oscillating artificial spline signal, an oscillating artificial Gaussian signal, and a real-life chromatogram—presented in Section 6.4.2. The best approximation results were obtained by an algorithm that used a real-number encoding in combination with an adaptive mutation scheme that uses three different mutation operators.

[2] copyright New York University, 1993

From the results we may conclude that both the L^1 and the L^∞ norms are less suited for use in the fitness function than the L^2 and L^4 norms. L^∞ seems to put too much stress on local behavior, whereas L^1 does the opposite.

Although the approximation results are reasonable, the genetic algorithm takes much time, which makes it rather unsuitable as a general-purpose function approximator. Moreover, the algorithm does not make use of knowledge about the problem domain structure. In the case of L^2-approximation, it is possible to find the optimal approximation of a signal by orthogonal projection onto a set of basis functions. The Matching Pursuit algorithm makes use of this knowledge which partly explains its better performance. Some knowledge of this type might be included by hybridizing the GA with a local search method such as gradient descent (possibly using a Lamarckian, write-back approach). Such a local search might also alleviate the problems associated with epistatis as mentioned in Section 6.4.4.

However, the algorithm offers a great deal of flexibility. Any type of elementary function can be used, as well as any type of evaluation function. As an extension to this work, we could use an encoding that combines different function types on a single chromosome. Such an approach might be employed to find elementary functions especially suited to approximate certain types of signal data.

The statistical significance of the experimental results is not very high, which can be seen from the relatively large confidence intervals. Due to computational limitations, only five runs of each experiment could be performed; more runs would have increased the reliability.

The effort that was put into adapting the operators to the real-number representation hardly paid off. It did increase the performance of the algorithm a little, but in the light of the statistical reliability discussed above, this improvement is barely significant. There might exist settings of the algorithm's parameters that lead to better approximation results. However, given the computational requirements of the algorithm, a full search of its parameter space is infeasible.

The positive aspect of using real number operators is that they are closer to the domain of the problem than bitstring operators, and therefore easier to understand and adapt to one's needs. Furthermore, using real numbers facilitates hybridization of the genetic algorithm with local optimization techniques such as hillclimbing.