Salzburg Interest Group on Integrated Systems

siGis-21-2001 July 30, 2001

Biologically Inspired Data Compression Induced by Reading Frames on Artificial ptGA Chromosomes

Genetic and Evolutionary Computation Conference (GECCO–2001)
Late–Breaking Papers
July 7–11, 2001, San Francisco, California

Helmut A. Mayer

helmut@cosy.sbg.ac.at

Department of Computer Science
University of Salzburg

Correspondence to:

Helmut Mayer
Universität Salzburg
Institut für Computerwissenschaften
Jakob–Haringer–Straße 2
A–5020 Salzburg
AUSTRIA

Telephone: +43-662-8044-6315
FAX: +43-662-8044-611

Biologically Inspired Data Compression Induced by Reading Frames on Artificial ptGA Chromosomes

Helmut A. Mayer
Department of Computer Science
University of Salzburg
A–5020 Salzburg, Austria

Abstract

In this work we present first experiments with a lossy data compression method which is based on the genetic mechanism of frame shifting occuring during the translation of genetic code into proteins. The basic principle of the presented method is the differential usage of common genetic code which is induced by promoters and terminators marking the start and end of coding regions (exons). Hence, identical sequences of genetic code may be used for the encoding (compression) of a number of different sequences by simply shifting the start position of decoding (decompression). A number of experiments evolving framed chromosomes to compress data generated by a stochastic source of different entropy rates are presented.

1 INTRODUCTION

Research towards the evolution of the representation (or some parts of it) of the genetic code of an *Evolutionary Algorithm* (EA) has not been a major branch in *Evolutionary Computation* (EC). However, when looking at nature it can be seen that the problem of representation is of greatest importance to any evolutionary process. Most of todays EA applications impose more or less constraints on possible representations. In fact, these constraints represent problem knowledge often leading the way to an appropriate genetic representation of the problem. However, some problems could benefit from an EA also evolving its representation (or parts of it).

E.g., when thinking of the evolution of *Artificial Neural Networks* (ANNs) the optimal number of parameters (number of neurons and connections) is not known in advance. Hence, an EA not only evolving the parameter values, but also the number of parameters could be beneficial for specific problems. As a consequence, researchers have transposed increasingly complex genetic mechanisms from nature to machines like the use of *Non–coding* regions on artificial chromosomes. The discrimination between non–coding regions (code not used for transcription of genes) and coding regions is achieved by *Promoter/Terminator* (p/t)–sequences signalling the start and end of a coding region, respectively. Applying genetic operators (mutation, crossover) to these chromosomes changes the location and number of p/t–sequences, thus varying the number of parameters (coding regions).

Combining above ideas we have proposed *ptGA* evolving number, size, and location of parameters, and parts of its representation (Mayer, 1997; Mayer, 1998). A specific mechanism of ptGA borrowed from nature, namely, *Frame Shifting* is investigated in this paper. A biological gene is transcribed into mRNA (messenger RNA) which is translated into a protein at a *Ribosome*. The mRNA base sequence is read in triplets of bases, where each triplet encodes a specific *Amino Acid*. Specific triplets signal start and end (called *Reading Frame*) of the translation process. It has been observed (Lodish et al., 1995) that it may occur that a single reading frame is used for translation of two different proteins by *Frame Shifting*. E.g., the base sequence $AAU - GUU$ could be read as $\ldots A - AUG - UU \ldots$ by shifting one base to the right yielding a completely different protein using the same sequence of bases. In technical terms data is compressed. Hence, we use that biological mechanism as a starting point for similar mechanisms in ptGA chromosomes (Section 2).

1.1 RELATED WORK

A variety of ideas dealing with parameter location on chromosomes, adapting gene size, or changing the number of parameters during the course of algorithmic evolution have been proposed.

The *ARGOT Strategy* (Shaefer, 1987) (Adaptive Representation Genetic Optimizer Technique) employs a form of Lamarckian evolution, as specific knowledge about the population is exploited to change the representation of parameters. Specifically, the resolution of parameter values changes accordingly to the values' standard deviation within the population.

In (Bui and Moon, 1994) the notion of *C–Schemata* is introduced. Here building blocks are clustered by heuristics prior to GA runs dividing the chromosome into sequences with higher and lower density of building blocks. Though, there are no non–coding regions on the chromosome it has been observed that the destructiveness of k–point crossover is reduced by clustering the building blocks.

Messy GAs (Goldberg et al., 1993) allow variable–length chromosomes and relax the fixed–locus assumption of conventional GAs. An examplary messy chromosome of length $l = 3$ has the structure $((10)(21)(11))$, where the first number of each pair of values indicates the specific gene and the second number its value. This GA variant allows for the evolution of number of parameters and their location, whereas the length of parameters on the chromosome is fixed.

Levenick (Levenick, 1991) reported on experiments with binary strings containing five genes with a length of six bases ($e_l = 6$) each being separated by non–coding sequences of length seven ($i_l = 7$). Comparing GA runs with and without insertion of non–coding sequences showed a nearly tenfold increase in successful evolution of optimal individuals. Levenick explains that behavior with the increase of crossover points (he used 1–point–crossover) which do not break a gene when the crossover site is located in the intron region.

In her dissertation (Wu, 1996) Wu centers on experiments with Royal Road functions with chromosomes containing non–coding regions. She reports that the use of non–coding sequences at fixed locations resulted in only slight (or no) increase of GA performance on the Royal Road functions. She also investigated the use of start sequences (promoter sequences) in order to freely position the basic building blocks of the Royal Road function on the chromosome. This increased GA performance, and her thesis is to our knowledge the first work proposing what we call promoter sequences.

2 ptGA INGREDIENTS

We now present the main genetic mechanisms modeled by ptGA, and clear some issues concerning terminology. The biologically motivated extensions to conventional GAs are discussed briefly and explained by means of illustrations.

The main difference to standard GAs is the representation of parameters (genes) on the chromosome. A sequence of bases p_s of a specific length p_l is defined to be the p–sequence. Similarly, the t–sequence is specified by t_s, and t_l. The p/t–sequence bases are given in binary string notations (e.g. $p_s = 010$). The bases downstream a promoter down to the terminator represent an exon. The bases upstream a promoter up to the next terminator are called introns. Each exon is associated with one parameter. It should be noted that biological introns are intra–genetic regions, thus, a number of exons (and introns) build a gene. In a strict biological sense introns are a specific instance of non–coding regions laso existing between complete genes.

In Figure 1 an example chromosome with exons and introns induced by p/t–sequences is depicted.

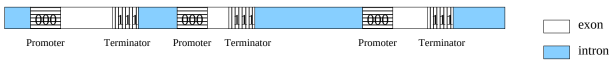

Figure 1: A ptGA chromosome with $p_s = 000$ and $t_s = 111$.

The only limit to the number of exons being possibly evolved by the use of p/t–sequences is the chromosome length l. The following definitions allow a more formal description of ptGA chromosomes. The *Marker Length* m_l and the *Effective Chromosome Length* l_{eff} are given by

$$m_l = p_l + t_l\,, \qquad l_{eff} = l \tag{1}$$

with the p–sequence length p_l, and the t–sequence length t_l. For an unframed chromosome the effective length is simply the length of the chromosome l. The *Normalized Chromosome Length*

$$\bar{l} = \frac{l_{eff}}{\overline{m_l} + \overline{e_l}} \tag{2}$$

should serve as an estimator for the maximum possible number of exons for a given chromosome length l, an average marker length $\overline{m_l}$ with an average exon length $\overline{e_l}$. As in this work we are not concerned with varying marker lengths on the same chromosome we set $\overline{m_l} =$

m_l. Moreover, if we are only interested in the number of exons without any demands on the exon length, we may set $\overline{e_l} = 0$ resulting in

$$\bar{l} = \frac{l_{eff}}{m_l}. \tag{3}$$

Equation 3 implies that a p/t–sequence without any exon bases in between is also counted as an exon.

Another idea is borrowed from the differential usage of p/t–sequences, and the reading frames during translation. If an artificial chromosome is scanned more than once for p/t–sequences, and the start of the scan is shifted by one or more bases for each frame, bases at identical locations can take on different roles in different frames. A single base could possibly be a p–sequence base in one frame, and an exon base in another frame, but could also be a t– or i–sequence in the next frame. This allows a compact representation of information. Two variants of *Framing* will be investigated, namely *Exhaustive* or *Naive Framing*, and *Promoter Framing* (Figure 2).

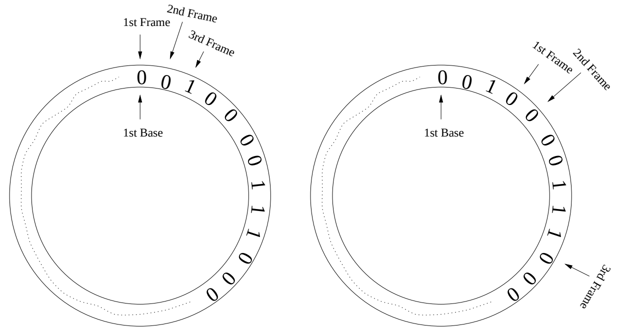

Figure 2: Naive (left) and promoter framing (right) with $p_s = 000$.

To explain the mechanisms of framing the chromosome should be visualized as a closed loop, where the last base is connected to the first base of the genotype. Naive framing starts the first frame on the first base and scans the chromosome for p/t–sequences until the first base is reached again. The second frame starts and ends at the second base. This is done for all l bases of the chromosome. As every possible frame is utilized, this framing variant is also termed exhaustive.

Promoter framing starts the first frame at the first promoter found on the chromosome and scans the genotype until the start point (the first promoter) is encountered, again. The second frame starts and ends at the second promoter relative to the first base of the chromosome. This is done for all promoters (seen from the first base) on the chromosome. Hence, the number of frames depends on the number of promoters. With both framing variants the length of the frame is the chromosome length $l^{(F)}$. An idea similar to naive framing has also been suggested in (Wu, 1996).

The effective length $l_{eff}^{(F)}$ of a framed chromosome is given by

$$l_{eff}^{(F)} = f(l^{(F)})l^{(F)}, \qquad \bar{l}^{(F)} = \frac{f(l^{(F)})l^{(F)}}{{m_l}^{(F)}} \tag{4}$$

where $f(l^{(F)})$ is the number of frames depending on the framing method and on the length of the chromosome to be framed. The length of the framed chromosome (the number of bases) is $l^{(F)}$. Setting the normalized chromosome length $\bar{l}$ for an unframed chromosome equal to $\bar{l}^{(F)}$ gives the *Equivalent Length* $l_E^{(F)}$ for a framed chromosome. It can be seen easily seen that for

$${m_l}^{(F)} = m_l, \qquad \bar{l}^{(F)} = \bar{l} \Rightarrow l_E^{(F)} = \frac{l}{f(l^{(F)})}, \tag{5}$$

hence, the normalized chromosome lengths are equal and can be compared when the chromosome length used with framing $l^{(F)} = l_E^{(F)}$. In the case of naive framing $f(l^{(nF)}) = l^{(nF)}$, as all bases of the chromosome serve as starting points for a frame, regardless of the number and distribution of promoters.

$$l_{eff}^{(nF)} = l^{(nF)2}, \qquad l_E^{(nF)} = \sqrt{l} \tag{6}$$

simply states the fact that for naive framing the effective length of the chromosome is a concatenation of $l^{(nF)}$ frames of length $l^{(nF)}$. Promoter framing induces p frames which gives a $f(l^{(pF)}) = p$ with

$$l_{eff}^{(pF)} = pl^{(pF)}, \qquad l_E^{(pF)} = \frac{l}{p}, \tag{7}$$

thus, the effective length of the promoter framed chromosome is dependent on the number of promoters. Though, we are operating on fixed length chromosomes, this implicitly varies the effective length of promoter framed chromosomes. Moreover, p/t–sequence interactions make things even more complicated. E.g., the bases of the p–sequence $p_s = 000$ may each be part of another p–sequence in another frame. Similarly, this is true for an exemplary t–sequence $t_s = 111$. However, in this case none of the terminator bases can be used for a promoter and vice versa.

3 COMPRESSION EXPERIMENT

From the previous section we conclude that theoretically a framed chromosome may encode the information stored in an unframed chromosome of length l with a framed chromosome length $l^{(F)} \sim \sqrt{l}$ (Naive Framing, Equation 6). Thus, the question arises, if we could evolve a framed ptGA chromosome which represents a given data source as accurately as possible. Obviously, this basic question motivates a variety of experiments and different approaches. However, we start with some simple experiments so as to investigate the basic potential of framed ptGA chromosomes to compress data.

We randomly generate binary bit strings of length $n_m = 100$ with three different probabilities $p_0 = 0.25/0.5/0.75$ for the occurrence of 0s resulting in $p_1 = 0.75/0.5/0.25$ for the occurrence of 1s (*Stochastic* source), respectively. From information theory it is well known that the stochastic binary source with $p = 0.5$ has the maximal entropy of 1 bit and cannot be compressed without loss (Madisetti and Williams, 1998; Storer, 1992).

For each of these stochastic data sources ptGA is started in order to evolve a framed ptGA chromosome effectively compressing the data. We start with a fixed length $l^{(F)} = 20$ giving a compression rate of 5. Also, different p/t–sequences for both framing variants are investigated. Each experiment is repeated 50 times, where new data are generated for each of the 50 runs in order to eliminate "lucky hits".

As the single bits of a stochastic source are statistically independent which is not true for most real world data, we also run experiments with random bit strings of length 10 being duplicated 10 times yielding a binary string of length 100 evidently possessing statistical dependencies (non–stochastic data). Furthermore, we investigate the dependency of reconstruction error on compression rate by varying the fixed length of the framed chromosomes.

The fitness function f is based on the reconstruction error $e_r = \frac{n_e}{n_m}$ with n_e being the number of incorrect bits, and is given by

$$f = \begin{cases} 0 & \text{if } n_d < n_m \\ 1 - e_r & \text{otherwise} \end{cases} \tag{8}$$

n_d is the number of decompressed (deframed) bits which is forced to be at least equal to the number of bits in the data model. Hence, from the fitness value we can easily infer situations where the framed chromosome cannot represent all the data (and we also force ptGA to possibly compress all data).

The following ptGA parameters have been used with all the experiments in this paper:

ptGA Parameters: Population Size = 50, Generations = 100, Crossover Probability $p_c = 0.6$, Mutation Probability $p_m = \frac{1}{l^{(F)}}$, Crossover = 2–Point, Selection Method = Binary Tournament.

4 EXPERIMENTAL RESULTS

Tables 1, 2, and 3 show fitness statistics for evolved framed ptGA chromosomes with various p/t–sequences of length $p_l = t_l = 2$ using stochastic data with varying entropy rate.

Table 1: Fitness of evolved ptGA chromosomes ($l^{(F)} = 20$, $p_s = 00$, $t_s = 00$, averaged on 50 runs).

Stochastic Data		
Source	Naive	Promoter
$p_0 = 0.25$		
Best	0.79	0
Mean	0.7630	0
StdDev	0.0221	0
$p_0 = 0.5$		
Best	0.68	0
Mean	0.6380	0
StdDev	0.0210	0
$p_0 = 0.75$		
Best	0.58	0
Mean	0.5530	0
StdDev	0.0189	0

As can be seen in Table 1 promoter framing is not able to encode all the data (with the given length of the framed chromosome), while naive framing compresses the data with $p_0 = 0.25$ with the smallest loss. This is not surprising, as these data consists of a majority of 1s, so the 0s can be used effectively for the p/t–sequences. Moreover, the entropy of this data source is lower, theoretically allowing improved compression. However, this is also true for the data with $p = 0.75$, but here the strong interactions between the specific p/t–sequence and the data are exemplified. These data contain a majority of 0s, which are also needed for the p/t–sequences. Note also that a t–sequence can never be part of an exon (decompressed data). The consequence in this specific case is that two adjacent 0s can never be encoded.

In Table 2 promoter framing again cannot represent all the data, while naive framing now is able to com-

Table 2: Fitness of evolved ptGA chromosomes ($l^{(F)} = 20$, $p_s = 01$, $t_s = 10$, averaged on 50 runs).

Stochastic Data		
Source	Naive	Promoter
$p_0 = 0.25$		
Best	0.84	0
Mean	0.7980	0
StdDev	0.0187	0
$p_0 = 0.5$		
Best	0.69	0
Mean	0.6540	0
StdDev	0.0165	0
$p_0 = 0.75$		
Best	0.84	0
Mean	0.7790	0
StdDev	0.0390	0

press both biased sources (having the same entropy) with comparable loss. The specific p/t–sequences even enable the representation of the t–sequence $t_s = 10$ in the decompressed (deframed) data, as the last base of an exon can be 1 and the first base of the next exon could be 0.

Table 3: Fitness of evolved ptGA chromosomes ($l^{(F)} = 20$, $p_s = 00$, $t_s = 11$, averaged on 50 runs).

Stochastic Data		
Source	Naive	Promoter
$p_0 = 0.25$		
Best	0.60	0.49
Mean	0.5720	0.4310
StdDev	0.0225	0.0407
$p_0 = 0.5$		
Best	0.70	0.63
Mean	0.6650	0.5580
StdDev	0.0232	0.0385
$p_0 = 0.75$		
Best	0.83	0.80
Mean	0.7870	0.7520
StdDev	0.0287	0.0346

Using the p/t–sequences given in Table 3 promoter framing generates enough data, but also with naive framing the t–sequence interacts heavily with the data generated by $p_0 = 0.25$. Due to these potential problems with the specific choice of t–sequences, we simply omit the t–sequence in the following experiments and define an exon by its promoter (start) and a fixed length e_l. Tables 4, 5, and 6 give the results for the p–sequence $p_s = 00$ and varying exon lengths e_l.

Table 4: Fitness of evolved ptGA chromosomes ($l^{(F)} = 20$, $p_s = 00$, $e_l = 2$, averaged on 50 runs).

Stochastic Data		
Source	Naive	Promoter
$p_0 = 0.25$		
Best	0.82	0.64
Mean	0.7750	0.5780
StdDev	0.0288	0.0312
$p_0 = 0.5$		
Best	0.73	0.67
Mean	0.6850	0.6370
StdDev	0.0242	0.0226
$p_0 = 0.75$		
Best	0.83	0.83
Mean	0.7920	0.7950
StdDev	0.0385	0.0284

With an exon length of $e_l = 2$ promoter framing exhibits problems with the data generated by $p_0 = 0.25$ which can be explained with the specific p–sequence needing two 0s in a majority of 1s. Even if a promoter evolves, it yields only two bases for the decompressed code. Of course, naive framing faces the same problem, but it can "reuse" the same promoter in many different frames. On the other hand, the data generated by $p = 0.75$ interacts symbiotically with the promoter, hence both framing variants produce similar results. The longer exon with $e_l = 4$ does not change this overall picture (Table 5).

Table 5: Fitness of evolved ptGA chromosomes ($l^{(F)} = 20$, $p_s = 00$, $e_l = 4$, averaged on 50 runs).

Stochastic Data		
Source	Naive	Promoter
$p_0 = 0.25$		
Best	0.82	0.64
Mean	0.7760	0.6110
StdDev	0.0299	0.0152
$p_0 = 0.5$		
Best	0.71	0.70
Mean	0.6950	0.6740
StdDev	0.0196	0.0190
$p_0 = 0.75$		
Best	0.82	0.83
Mean	0.7850	0.7910
StdDev	0.0306	0.0264

From Table 6 it can be seen that increasing exon length

Table 6: Fitness of evolved ptGA chromosomes ($l^{(F)} = 20$, $p_s = 00$, $e_l = 8$, averaged on 50 runs).

Stochastic Data		
Source	Naive	Promoter
$p_0 = 0.25$		
Best	0.84	0.70
Mean	0.7990	0.6540
StdDev	0.0281	0.0201
$p_0 = 0.5$		
Best	0.71	0.72
Mean	0.6930	0.6740
StdDev	0.0149	0.0317
$p_0 = 0.75$		
Best	0.84	0.84
Mean	0.7920	0.7820
StdDev	0.0270	0.0377

slightly improves the fitness (reconstruction error) and decreases the problems of p–sequence interactions with promoter framing. In the following experiment (Table 7) we use the p/t–sequences having shown the most promising results and apply the genetic compression method to non–stochastic data (Section 3).

Table 7: Fitness of evolved ptGA chromosomes ($l^{(F)} = 20$) compressing non–stochastic data (averaged on 50 runs).

Non–Stochastic Data ($p_0 = 0.5$)		
p/t	Naive	Promoter
$p_s = 00$, $t_s = 11$		
Best	0.88	0.67
Mean	0.7670	0.5820
StdDev	0.0776	0.0984
$p_s = 00$, $e_l = 8$		
Best	0.85	0.78
Mean	0.7410	0.6960
StdDev	0.0534	0.0442
$p_s = 00$, $e_l = 16$		
Best	0.90	0.87
Mean	0.7560	0.7600
StdDev	0.0615	0.0611

Best and mean fitness values increase slightly, as could be expected, but equally the fitness standard deviation is larger. This implies that the specific p/t–sequences in some cases are "adapted" to the data, but in other cases might impede a better compression. Naturally, this leads to the idea to additionally evolve the p/t–sequences (possibly also their length). First steps in this direction have been presented in (Mayer, 1997).

In Table 8 the results for framed ptGA chromosomes of different length (yielding different compression rates) compressing stochastic data are shown.

Table 8: Fitness of evolved ptGA chromosomes ($p_s = 00$, $e_l = 16$) of different length (averaged on 50 runs).

Stochastic Data ($p_0 = 0.5$)		
Length	Naive	Promoter
$l^{(F)} = 30$		
Best	0.78	0.75
Mean	0.7270	0.7350
StdDev	0.0323	0.0127
$l^{(F)} = 40$		
Best	0.79	0.80
Mean	0.7520	0.7800
StdDev	0.0199	0.0156
$l^{(F)} = 50$		
Best	0.81	0.82
Mean	0.7700	0.7710
StdDev	0.0200	0.0341

The fitness (reconstruction error) increases only slightly (if at all) for decreasing compression rate (Table 8) which could be attributed to the stochastic binary source of maximal entropy which can only be compressed without loss at a compression rate of 1.

5 SUMMARY AND OUTLOOK

We have presented an idea to achieve lossy code compression inspired by a genetical process called shift framing which may occur during translation of mRNA. The main principle of framing is the differential usage of common code. E.g., we could encode two bytes in one byte 01001101 reading the first byte starting with the first bit, and the second byte starting with the second bit (and rotate back to the first bit). The genetic model is further exploited by introducing non–coding sequences which are delimited by promoters and terminators (p/t–sequences) on an artificial chromosome (ptGA). Given binary data, we may evolve a ptGA chromosome such that the genetic code represents a lossy compression of the original data. Decompression is achieved by the two framing variants naive and promoter framing. The fitness of the genotype can be arbitrarily composed by compression rate and reconstruction error.

In this work we have mainly analyzed the basic potential of this approach. Two main conclusions can be drawn from the presented experiments. The reconstruction error decreased with the entropy rates of the stochastic source. This is a prerequisite for any compression algorithm which the presented genetic compression has been demonstrated to have. The experiments also showed that it might be a promising direction to not only evolve the compressed code, but also the p/t–sequences itself. The experiments with externally given p/t–sequences revealed that subtle interactions of (non)coding regions, promoters, and terminators may have a great impact on the reconstruction error. In other words, specific data could demand specific p/t–sequences.

Of course, an important next step is to apply the presented technique to real world data (e.g., images) in order to finally judge the validity of this approach. As a framed chromosome of fixed length generates decompressed code of different lengths different segments of the image data could be compressed with varying compression rates and reconstruction errors (adjustable by the fitness function). The evolution of chromosome length could be another vital component of such an adaptive genetic compression scheme.

References

Bui, T. N. and Moon, B. R. (1994). Analyzing Hyperplane Synthesis in Genetic Algorithms Using Clustered Schemata. Technical Report CSE-94-026, Penn State University, University Park, PA.

Goldberg, D. E., Deb, K., Kargupta, H., and Harik, G. (1993). Rapid, Accurate Optimization of Difficult Problems Using Fast Messy Genetic Algorithms. In Stephanie Forrest, U. o. N. M., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 56–64. University of Illinois at Urbana-Champaign, Morgan Kaufmann.

Levenick, J. R. (1991). Inserting Introns Improves Genetic Algorithm Sucess Rate: Taking a Cue from Biology. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 123–127. University of California, San Diego, Morgan Kaufmann.

Lodish, H., Baltimore, D., Berk, A., Zipursky, S. L., Matsudaira, P., and Darnell, J. (1995). *Molecular Cell Biology*. Scientific American Books, 3rd edition. ISBN 07167-2380-8.

Madisetti, V. and Williams, D., editors (1998). *The Digital Signal Processing Handbook*. CRC/IEEE Press.

Mayer, H. A. (1997). *ptGAs - Genetic Algorithms Using Promoter/Terminator Sequences - Evolution of Number, Size, and Location of Parameters and Parts of the Representation*. PhD thesis, University of Salzburg.

Mayer, H. A. (1998). ptGAs–Genetic Algorithms Evolving Noncoding Segments by Means of Promoter/Terminator Sequences. *Evolutionary Computation*, 6(4):361–386.

Shaefer, C. G. (1987). The ARGOT Strategy: Adaptive Representation Genetic Optimizer Strategy. In Grefenstette, J. J., editor, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 50–55. Naval Research Laboratory, Lawrence Erlbaum Associates.

Storer, J. (1992). *Image and Text Compression*. The Kluwer international series in engineering and computer science. Boston.

Wu, A. S. (1996). *Non–Coding DNA and Floating Building Blocks for the Genetic Algorithm*. PhD thesis, University of Michigan.