A Genetic Algorithm with Multiple Reading Frames

Terence Soule
Department of Computer Science, University of Idaho
Moscow, ID 83844-1010
email: tsoule@cs.uidaho.edu
phone: 208-885-7789

Amy E. Ball
Moscow, ID 83843
Email: aeball@att.net

Abstract

Multiple reading frames are an important feature of gene expression in biological systems. Multiple reading frames allow several genes to be encoded in the same region of DNA. This produces an inherent form of information compression. In some organisms this compression is so extensive their genes are effectively longer than their DNA. In this paper a modification of a simple genetic algorithm (GA) is introduced that uses multiple reading frames. It is shown that some information compression does occur. Interestingly, while the GA does utilize information compression where necessary there is evidently strong evolutionary pressure to limit this compression.

1 Introduction

Recently there has been considerable interest in applying the basic processes of biological gene expression to artificial forms of evolution, most notably genetic algorithms (GAs). Gene expression, the process by which DNA produces proteins, is a complex process which has a significant impact on how DNA actually 'produces' an organism. Several researchers have explored the impact of artificial gene expression on artificial evolution (see [1] for a summary of this work) leading to potentially beneficial modifications of the basic GA and other artificial evolutionary models.

One important feature of the gene expression process that has received limited attention from the evolutionary computation field is multiple reading frames. Multiple reading frames occur because there are three distinct 'translations' of a given segment of DNA. Each of these translations can encode different proteins. Thus, DNA is inherently capable of a certain amount of information compression.

In this paper we examine a GA that uses an analogy of gene expression that allows multiple reading frames. The primary questions are whether a GA can use reading frames and whether a GA can exploit the information compression capability inherent in multiple reading frames.

2 Background

The primary genetic material of most biological organisms is Deoxyribonucleic acid (DNA). DNA itself is composed of four nucleic acids: adenine (A), cytosine (C), guanine (G), and thymine (T). Every protein component of an organism is made from a long sequence of As Cs Gs and Ts.

Interestingly, most of the DNA contained in a typical chromosome does not contain instructions for making proteins or other functional products. Much of this non-coding DNA serves a structural purpose or serves to regulate how and when, the coding portion of the DNA is utilized or to facilitate the coding process itself. Finally, much of the non-coding DNA serves no known purpose.

The coding portion of DNA is organized into sets of three nucleic acids known as codons. Thus, there are 64 codons in the genetic code of all biological organisms (4^3). Each of the 20 amino acids is specified by at least one codon, which is interpreted in a complex two-step process. In the first step, transcription, the genetic code embedded in DNA is reinterpreted as messenger ribonucleic acids (mRNAs). In the second step, translation, the mRNA is read and used to construct protein. Because the coding portions of DNA are separated on a chromosome, the function of some of the codons is to initiate or to terminate the transcription process. The genetic code specifies 3 termination or "stop" codons and one initiation or "start" codon. The remaining 60 codons specify 19 of the 20 amino acids that make up proteins. The 20th amino acid, methionine, is specified by the start codon itself. (In fact, the initiation of transcription is dependent upon the presence of several specific but non-coding regions of DNA that must be adjacent to the start codon. Only when these DNA regions are present does the "start" codon actually initiate transcription, otherwise it specifies only the inclusion of methionine in the final protein.) There is considerable redundancy in the genetic code, with 61 codons specifying 20 amino acids. Only methionine is specified by a single codon. The remaining 19 amino acids are each specified by as few as two or as many as six codons. There are many useful sources for a further review of the biological genetic code and the processes of transcription and translation (see, for example the summary by Kargupta [2], or a genetics text such as *Genetics* [3]).

The arrangement of the genetic code into functional codons each consisting of three contiguous nucleic acids leads to a phenomenon known as reading frames. Because one can start reading the DNA sequence at either the first, second, or third nucleic acid of a codon there are three different possible sequences each of which can be interpreted separately. For example the DNA sequence:

...AACGTTGACTGCTAGTTTCACATGCGTACT...

can be organized into three distinct sets of codons,

... AAC GTT GAC TGC TAG TTT CAC ATG CGT ACT ...
...A ACG TTG ACT GCT AGT TTC ACA TGC GTA CT...
...AA CGT TGA CTG CTA GTT TCA CAT GCG TAC T...

(Any other reading frame will produce codons that are a subset of one of the primary three reading frames.) The fact that three full and unique coding sequences can be found in one DNA strand allows for a great deal of information compression, as several genes can overlap within the strand of DNA by using different reading frames.

In general, it is unclear to what extent most biological organisms take advantage of multiple reading frames to compress the genetic information contained in their DNA. However, there is precedence for such a process occurring in viruses, bacteria, and eukaryotes. Notably, the bacteriophage ϕX174 includes many regions where two and three genes overlap. The overlapping is so extensive that the total coding length of the bacteriophage is longer than its DNA[4].

The floating building block representation introduced by Wu and Lindsay shares many features with the biological process outlined above [5]. The representation uses a start tag to mark the beginning of genes and an identity tag that determines the 'function' of the gene. The length of the genes are fixed so no end tag (stop codon) is required.

In the floating building block representation a second start tag may appear within an 'active' gene. This leads to overlap similar to that produced by multiple reading frames. Additionally, the use of identity tags allowed some genes to be over or under specified.

Experimental results with the floating representation were positive, particularly for longer chromosomes. It performed significantly better than a standard GA on the two test functions, Royal Road and a version of symbolic regression. Much of the benefits seemed to arise because longer chromosomes allow additional over specification of genes and hence additional exploration of the search space within the same size population. In addition, multiple floating genes gave the GA more flexibility to explore different physical arrangements of the genes. Although the floating representation allows genes to overlap, the experiments did not examine the extent of overlap or how it changed over time.

Messy GAs also bear some resemblance to biological genes (see for example [6]). In messy GAs each 'gene' has an associated tag that determines the gene's location in a transcribed version of the genome. This allows the genes to be rearranged on the evolving chromosome.

3 The Encoding

Multiple reading frames require an encoding scheme that produces codons. We have attempted to write the simplest possible encoding rules that capture the essential properties of biological reading frames. In particular, the encoding is designed to include variable length genes using start and stop codons, potentially overlapping genes, and non-coding regions.

As noted above, nature uses 64 (4^3) codons. Our chromosomes are binary strings, so we choose a codon length of 5 which also produces 64 (2^5) codons. Of course, there is nothing obviously significant about the value 64. It was chosen simply to be similar the biological value. Non-binary encodings and other codon lengths could certainly be used and may be beneficial for some problems.

Codons were transcribed as follows:

$$\begin{array}{c} 11111 = \text{start} \\ 00000 = \text{stop} \\ \text{ijklm} = \text{jklm for all other cases} \end{array}$$

The start and stop sequences (11111 and 00000) act as regulatory codons. The other codons create a four bit sequence by dropping the first bit of the five bit codon. This creates two encodings for each 4 bit sequence (jklm can be encoded as 0jklm or 1jklm) except 0000 and 1111, which have one code apiece (10000 and 01111 respectively). The region between a start codon and a stop codon is equivalent to a biological gene. Although very simple, this encoding captures the essential properties of the genetic code necessary to study multiple reading frames.

Because our codons are five bits long there are five separate reading frames. The genes are evenly distributed among the reading frames. Thus, for N genes reading frame 1 specifies genes 1 through N/5, reading frame 2 specifies genes N/5+1 through 2N/5, etc. For example, our GA is tested on several function optimization problems (see Section 4). Each function depends on 25 variables. Thus, 25 values need to be specified. Each value is specified by a single gene, so there are 5 genes per reading frame.

Transcription of the first gene begins at the first start codon and continues until a stop codon or the end of the chromosome is reached. Transcription of the next gene begins at the next start codon and so forth, until the end of the chromosome is reached or all genes of that reading frame have been specified. If the end of the chromosome is reached before all of the gene are specified, the remaining variables are unspecified. Regions of the chromosome which do not fall between start-stop pairs are non-coding. Start codons in the middle of an open reading frame are ignored and thus are also non-coding.

In our experiment, each of the transcribed strings represents the value of one of the variables written in Gray code. Because the start and stop codons can occur at arbitrary locations the binary strings produced by the transcription process have arbitrary lengths. To avoid overflow errors at most the first 64 bits of the transcribed string are considered. Because the transcribed strings may have different lengths their range of possible values is variable. So, each transcribed string is scaled to a value between zero and one (by dividing by $2^{the\ string's\ length}$) and then is rescaled to the range appropriate to the problem.

Our encoding varies from other gene-like encodings in several significant ways. It combines floating genes with variable length genes. This gives the GA more control over the amount of genetic overlap (and hence the potential amount of information compression) than other encodings. It does not use identity tags, so over specification is not a possibility (although under specification is, if there are too few start codons). Our experiments focus on how the potential problems and advantages of overlapping genes are handled by the evolutionary algorithm.

4 Test Problems

We tested our GA on modifications of four of the functions from the De Jong test suite. We used functions F1, F3, F7 (Schwefel's function), and F8 (Griewangk's function) [7, 8]. The goal is to find the function maximums. The fitness of an individual is the returned value.

Our modified functions were:

$$f_1(x_i) = \sum_{i=1}^{25} x_i^2 \qquad x_i \epsilon [-5.12, 5.12] \tag{1}$$

$$f_3(x_i) = \sum_{i=1}^{25} integer(x_i) \qquad x_i \epsilon [-5.12, 5.12] \tag{2}$$

$$f_7(x_i) = \sum_{i=1}^{25} x_i sin(\sqrt{|x_i|}) \qquad x_i \epsilon [-512, 512] \tag{3}$$

$$f_8(x_i) = \sum_{i=1}^{25} x_i^2/4000 - \prod_{i=1}^{25} cos(x_i/\sqrt{i}) + 1 x_i \epsilon [-512, 512] \tag{4}$$

Clearly reading frames are only meaningful when there are sufficient genes to make use of the reading frames. Thus, each of the functions has been modified from the original version to include 25 variables. The value 25 was chosen strictly for convenience and consistency.

5 The Genetic Algorithms

These experiments compare a standard GA to a GA using codons and multiple reading frames. The parameters common to both GAs are summarized in Table 1. Both versions of the GA are generational.

For test functions F1, F2 and F3 the chromosome length is 250 bits. For the standard GA this meant each gene (variable value) is represented by exactly

Population Size	200
Crossover rate	0.8
Crossover type	one point
Mutation rate	1/Chromosome length
Selection	3 member tournament
Type	Generational
Elitism	1 member
Trials	30

Table 1: Summary of the parameters used with the GA with and without reading frames.

10 bits. For the GA with reading frames there are roughly 1250 bits available, or 50 bits per gene. (5 reading frames with 250 bits apiece. The actual number is slightly smaller because the reading frames lose a few bits at the beginning and end of the chromosome.)

Of course, with reading frames the number of bits encoding each solution is variable and in practice is much smaller than 50. Much of the chromosome may be noncoding. Some of the bits must be start and stop codons and the transcription process causes a 'loss' of 1/5 of the bits. In fact, the results show that the number of bits per variable evolves in a systematic way over the course of a GA run.

For F8 the chromosome length was 400. The standard GA uses 16 bits per variable. The GA with reading frames was subject to the same considerations discussed above.

6 Results

We begin by examining the performance of the GA on our test problems. Although our primary interest is how evolution arranges the genes in different reading frames, it is important that the GA perform reasonably. Additionally, the evolution of fitness does provide some insight into how the GA is adapting the reading frames.

The results of the GA and the GA with reading frames, averaged over 30 trials, are shown in Table 2. The GA using reading frames produces significantly better solutions than the standard GA in two of the four test cases (F1 and F8) and better average solutions in one case (F8). These data show that a GA can successfully use reading frames despite the added complexity of reading frames.

Additionally, there seems to be a specific reason for the poor performance on F3. For F3 the optimal solution is a maximum value (5.12) for all variables. This corresponds to a string of all ones, e.g. 11111111. When reading frames are used the sequence 11111111 is encoded as 01111 01111. However, this only produces an optimal solution in the first reading frame. In the second reading frame the sequence 01111 01111 is read as 0 11110 1111 and in the third frame it is read as 01 11101 111. Thus, for this problem the optimal solution for one reading frame is non-optimal for the other frames.

As a test we used a the following modification of the F3 function:

$$f_3'(x_i) = \sum_{i=1}^{25} |integer(x_i)| \qquad (5)$$

This function has several more potential solutions than the standard F3 function. Thus, it should be easier for the GA to find overlapping sequences within optimal (or near optimal) genes. The results are also shown in Table 2 and the GA with reading frames does outperform the standard GA. These results suggest that for some problems, such as F3, the encoding of the optimal solution(s) may make it extremely difficult to find optimal genes with overlapping regions.

Figure 1 shows the fitness of the best individuals (averaged over 30 trials) evolved with the standard GA and with reading frames on function F8. Initially the GA with reading frames performs much more poorly. Under specification of the variables is a likely cause of this poor initial performance.

Thus, we examined exactly how many variables are being specified by the GA with reading frames. This corresponds to the number of genes (or start-stop pairs) the GA is producing. Clearly, whether the GA can evolve sufficient genes across multiple reading frames is an important question. Figure 2 shows the number of variables specified at each generation for F1.

Initially, the average individual has approximately five of the twenty-five variables specified. This low value occurs simply because in randomly generated binary strings of 250 bits there will be relatively few start-stop pairs. However, over the course of evolution the number of specified variables grows steadily, almost reaching 24.97 for the best individuals and 24.24 for the average individual. Additionally, the best individuals consistently have more variables specified than the average. This is a clear indication that there is strong pressure on the GA to specify more of the variables.

The results for the other three test problems are similar and are summarized in Table 3. For the F8 function the initial number of specified variables is higher (14.3 for the best individual and 7.44 for the average individ-

	Best individual			Average individual		
	Standard GA	Reading frames	p	Standard GA	Reading frames	p
F1	625	642	< 0.01	613	607	< 0.1
F3	112	67.7	< 0.01	108	61.3	< 0.01
F7	19600	16900	< 0.01	19300	16400	< 0.01
F8	1550	1600	< 0.01	1600	1570	< 0.01
F3'	112	121	< 0.01	109	114	< 0.1

Table 2: Results at generation 50 for the best and average individuals. The raw numbers represent the optimal value achieved and are averaged over 30 trials. The p value is generated using Student's two-tailed test.

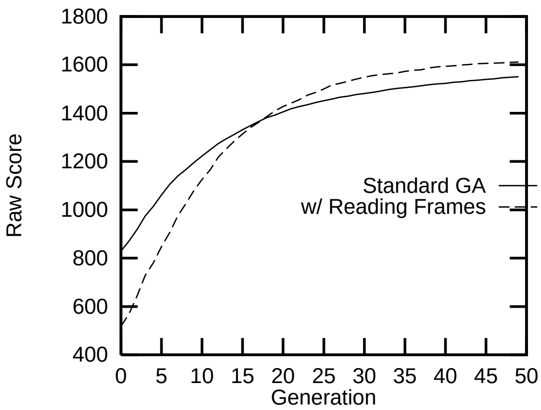

Figure 1: Fitness of the best individuals evolved with and without reading frames. These data are the average of 30 trials. The poor initial performance of the GA with reading frames is attributed to its failure to specific all of the variables.

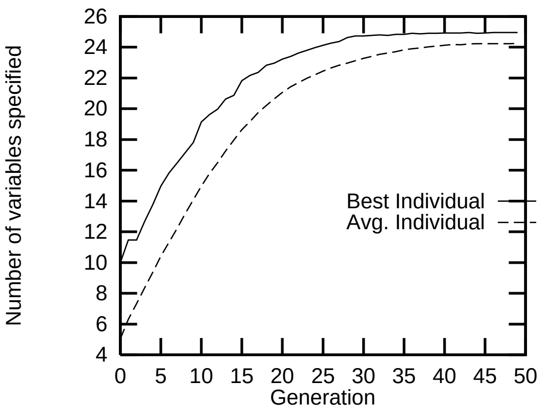

Figure 2: Number of variables specified, or number of genes per chromosome, for problem F1. Initially many of the variables are under specified due to a shortage of start and stop codons, but as evolutions proceeds the GA succeeds in specifying (almost) all 25 variables.

ual) simply because the chromosome is longer, which allows more start-stop pairs in the initial random chromosomes. Thus, a second feature of the GA with reading frames is that it can introduce new genes as necessary, by increasing the number of regulatory codons in the chromosomes. Similar behavior has been seen with messy GAs and floating building blocks, but is more difficult here because of the requirements to produce both start and stop codons and the complications introduced by overlap between the reading frames.

Finally, we consider the issue of information compression. Given that the GA is introducing additional genes, how long are these genes and how much do they overlap? To explore this question we focus on functions F7 and F8. These are the most interesting functions in that they are reasonably complex and they lead to dramatically different performance by the GA with reading frames.

Figure 3 shows the average length of the genes, when all genes are specified, for problems F7 and F8. (This is the length in bits, not including the start and stop codons.) In the early generations none of the solutions are fully specified and no data is shown. As noted above, the chromosomes used with F8 are longer and therefore include more start-stop pairs. Thus, fully specified solutions are found sooner.

For both problems the average gene length decreases over time. However, it is clear that this decrease is much faster and longer lasting for F8. (These runs were extended to 100 generations to illustrate this difference.) By the final generation the length of the genes for F8 has declined below 5, implying that some

Test Problem	Average number of variables specified initially	Average number of variables specified in generation 50
F1	5.04	24.2
F3	5.05	22.9
F3'	5.09	24.3
F7	5.09	24.3
F8	7.44	24.7

Table 3: Average number of specified variables in the initial and final generations. By the final generation the GA has introduced sufficient regulatory codons to specify almost all of the variables.

of the genes consist of only a start-stop pair.

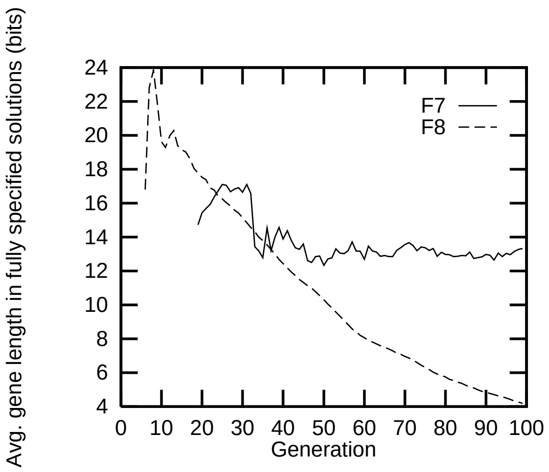

Figure 3: Average length of the genes in individuals with fully specified solutions. In the early generations none of the solutions are fully specified and no data is presented. Once the solutions are fully specified the average length of the genes decreases; steadily for F8 and quite slowly for F7.

There is another critical fact to draw from Figure 3. Consider generation 32 where the average gene length for both problems is roughly 15 bits. Including the start and stop codons each gene takes up, on average, 25 bits. Because the data is only taken from chromosomes in which all 25 variables are specified this means that the total length of the genes in one solution, measured in bits, is 625 (25 genes times 25 bits per gene). However, the lengths of the chromosomes for the F7 and F8 problems are only 250 and 400 bits respectively. The GA has succeeded in compressing a total gene length of 625 bits into 250 and 400 bits respectively.

However, it is also true that performance on F8, where there is less compression, is much better. In fact, by the final generation the average gene length for F8 is roughly 14 (including start and stop codons). This leads to a total gene length of 350, which is less than the chromosome length. Thus, although the GA can perform information compression within the multiple reading frames it seems to perform better when it does not need to.

Because the GA has found a solution (or solutions) to F8 that can be encoded in very short binary sequences, there are fewer bits per variable that need to be determined to find the solution. Whereas the standard GA must optimize all 250 bits in its chromosome, the GA with reading frames must only optimize the bits within genes. By reducing this number the GA reduces the size of the solution space and makes the problem easier. It is also possible that the genes are less susceptible to the effects of crossover and mutation as they represent a smaller percentage of the entire chromosome.

Finally, we are interested in how much overlap actually occurs between genes in different reading frames. A given bit can be read in five different reading frames. Thus, each bit can 'participate' in from 0 to 5 genes. Figure 4 shows the number of bits that participate in 0, 1, 2 or 5 genes (the values for 3 and 4 genes are omitted for clarity) for the F7 function. These results are the average of 30 trials. Each bit that participates in more than one gene represents a point of overlap. Bits where zero genes overlap are non-coding.

Initially, the majority of the bits do not participate in any gene, i.e. most of a chromosome is non-coding. Again this is because in the initial random individuals there are relatively few start-stop codon pairs. However, the number of bits that do not participate decreases extremely rapidly. There are corresponding increases in the number of bits that participate in one or more genes.

There are two important features of this figure. First, the amount of overlap is quite high. Many bits are participating in several (and often all five) genes. This explains how genes whose total length average 625 bits can be squeezed into a chromosome 250 bits long.

The second important feature of the graph is how the amount of overlap changes over time. The number of bits that participate in zero genes decreases rapidly until roughly generation 18. Generation 18 also roughly corresponds to the point when all twenty-five variables are specified. This suggests that the GA attempts to rapidly specify all of the variables. The steady

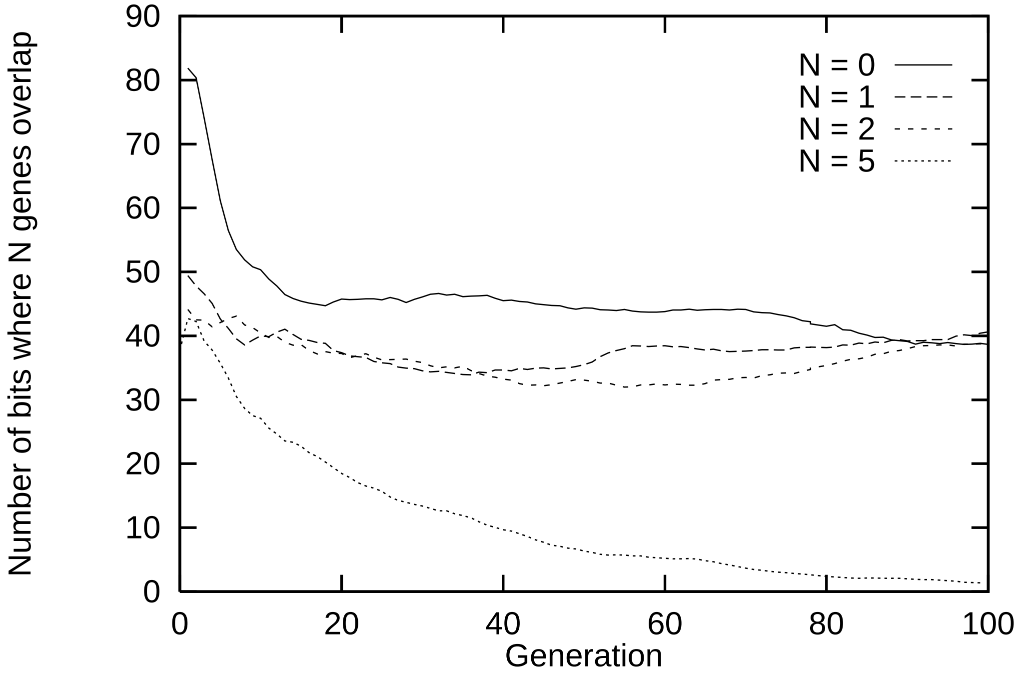

Figure 4: Number of bits with overlap and the amount of overlap (0, 1, 2 or 5 genes) at those bits, for the F7 function. The count of bits where 3 or 4 genes overlap are left out for clarity. The rapid decrease in the number of bits with 0 overlap, i.e. no gene is using them, illustrates that the chromosomes are evolving to utilize more of the available chromosome length. Fluctuations in the other values shows that gene overlap is occurring between the reading frames and that the amount of overlap appears to be subject to evolutionary pressure.

decrease in the number of bits shared by five genes suggests that the GA is also trying to separate the genes to minimize overlap. This seems reasonable, as it should be easier to optimize bits that participate in fewer genes. Thus, the overall behavior of the GA appears to be extremely sophisticated.

Figure 5 shows the number of bits participating in 0, 1, 2, or 5 genes for the F8 function (again the 3 and 4 gene cases have been omitted for clarity). The results are somewhat different from those seen with the F7 function. Except in the very earliest generations the number of non-coding bits increases throughout the trial and the number of bits participating in five genes decreases. Apparently there is much less pressure to produce overlapping genes.

However, the number of bits where two genes overlap does increase for roughly the first fifty generations and then starts to decline. Thus, it still appears that initially there is strong pressure to specify all of the variables. Later there is pressure to reduce the amount of overlap as much as possible. However, because the F8 function can apparently be solved with relatively short genes the amount of overlap never needs to be as high as for the F7 function. Additionally, the increased length of the chromosome for F8 allows less overlap even if the genes were the same length.

7 Discussion

It is clear that a GA can introduce new 'genes' as necessary to solve a given problem, even with the difficulties imposed by using start and stop codons and overlapping genes. More importantly, where necessary the GA can take advantage of multiple reading frames to overlap genes producing a form of information compression. In the most extreme case genes totaling 625 bits in length are represented in a chromosome 250 bits long. However, it is also clear that too much overlap between genes significantly degrades the performance of the GA. Thus, the potential benefits of this compression may be limited. Interestingly, after introducing sufficient genes, which often requires considerable overlap, the GA appears to reduce the amount of overlap (and hence the amount of compression) on its own.

It appears that multiple reading frames can be useful on problems where the length of the optimal solution is unknown, but the maximum possible solution length is quite large. Rather than requiring a chromosome as long as the maximum solution length, reading frames can be used to compress the chromosome. As

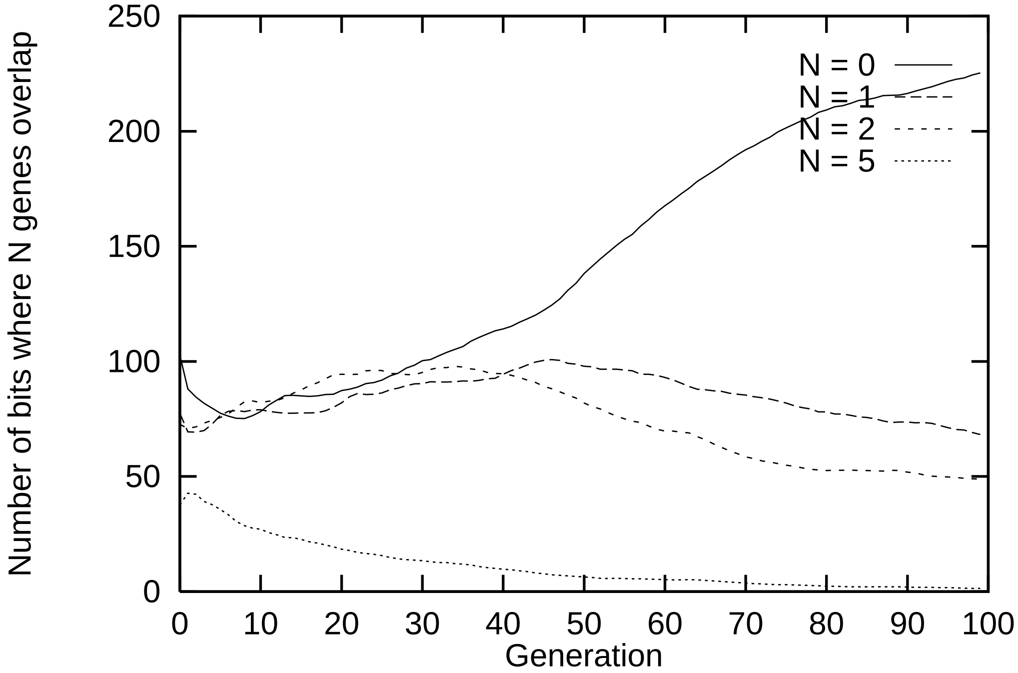

Figure 5: Number of bits with overlap and the amount of overlap (0, 1, 2 or 5 genes) at those bits, for the F8 function. The count of bits where 3 or 4 genes overlap are left out for clarity. The rapid increase in the number of bits with 0 overlap, i.e. no gene is using them, illustrates that the chromosomes are evolving very short genes. Fluctuations in the other values shows that gene overlap is occurring between the reading frames and that the amount of overlap appears to be subject to evolutionary pressure.

observed here the GA can be expected to adjust the actual length of the genes within the chromosome as necessary to solve the problem.

Results with the F3 and F3' problems emphasize a potential difficulty in using multiple reading frames. The choice of codons may make it difficult for a GA to find overlapping genes that encode the optimal solution. Without this overlap the benefits of multiple reading frames are lost. A likely solution to this problem is to increase the amount of redundancy within the encoding. Additional redundancy makes it more likely that there are ways to produce optimal genes with overlapping regions.

References

[1] H. Kargupta, "The Genetic Code and the Genome Representation," GECCO-2000 Workshop on Gene Expression, held in Las Vegas, Nevada, July 2000.

[2] H. Kargupta, "Gene: Expression: The Missing Link in Evolutionary Computation," GECCO-2000 Workshop on Gene Expression, held in Las Vegas, Nevada, July 2000.

[3] P. J. Russell, *Genetics* (Scot Foresman and Company, Glenveiw, Illinois, 1990).

[4] D. Graur and W. Li, *Fundamentals of Molecular Evolution, Second Edition* (Sinauer Associates, Sunderland, MA, 2000).

[5] A. S Wu and R. K. Lindsay, "A Comparison of the Fixed and Floating Building Block Representation in the Genetic Algorithm," *Evolutionary Computation*, 4:2, 1996.

[6] D. E. Goldberg, B. Korb and K. Deb, "Messy Genetic Algorithms: Motivation, Analysis, and First Results," in *Complex Systems*, **3**(5) (1989) 493-530.

[7] K. A. De Jong, *Analysis of Behavior of a Class of Genetic Adaptive Systems*, PhD Thesis, University of Michigan, Ann Arbor, MI, 1975.

[8] M. A. Potter, K. A. De Jong, "A cooperative Coevolutionary Approach to Function Optimization," The Third Parallel Problem Solving from Nature, held in Jerusalem, Israel, 1994.

[9] D. E. Goldberg, *Genetic Algorithms in Search Optimization, and Machine Learning* (Addison-Wesley Publishing, Reading, MA, 1989).