Structural Emergence with Order Independent Representations

R. Muhammad Atif Azad and Conor Ryan

Department Of Computer Science And Information Systems
University of Limerick
Ireland
{Atif.Azad,Conor.Ryan}@ul.ie

Abstract. This paper compares two grammar based Evolutionary Automatic Programming methods, Grammatical Evolution (GE) and Chorus. Both systems evolve sequences of derivation rules which can be used to produce computer programs, however, Chorus employs a position independent representation, while GE uses polymorphic codons, the meaning of which depends on the context in which they are used.
We consider issues such as the order in which rules appear in individuals, and demonstrate that an order always emerges with Chorus, which is similar to that of GE, but more flexible.
The paper also examines the final step of evolution, that is, how *perfect* individuals are produced, and how they differ from their immediate neighbours.
We demonstrate that, although Chorus appears to be more flexible structure-wise, GE tends to produce individuals with a higher neutrality, suggesting that its representation can, in some cases, make finding the perfect solution easier.

1 Introduction

Chorus [8] is a Genetic Programming (GP) [4] type evolutionary algorithm that evolves programs. It belongs to the same family of algorithms as Grammatical Evolution (GE) [9] and like [2,3,5], uses grammars to evolve the programs.

As with GE, Chorus recognizes a distinction between genotype and phenotype. The genotype is a string of 8 bit integers. It is translated into a high level language program described by a context free grammar given in Backus Naur Form (BNF) using a genotype-phenotype mapping process.

The nature of the mapping process is such that the functionality of the genes is almost never tied with their exact locations on the chromosome. This is the crucial difference between Chorus and GE, as they share many similar traits, as described in Sect. 4. Position independence not only allows the system to evolve the constituents of its individuals, but it also gives the flexibility to evolve their order.

This study is an investigation into any underlying trends that the system may have in forming its individuals and compares it with GE. The study examines the patterns that emerge in the chromosomes over the course of evolution, and considers their consequences. In particular, we wish to investigate if an order can emerge using the position independent representation of Chorus. Emergence of an order in a position independent system can also be interesting if it can be correlated with building block formation. The individuals are also tested for robustness to see the amount of neutrality available in the search space surrounding them. Finally, we also look into the leap the system has to make to reach the ideal fitness. For this purpose we analyse the parents of the ideal solutions and compare them with their perfect offsprings.

The paper first gives a brief introduction to Backus Naur Form. Next, a brief introduction to GE is followed by a description of Chorus. Section 5 then describes the genetic operators used by the systems. The following section describes the experimental setup to carry out this study and comments on the results obtained from the experiments.

2 Backus Naur Form

Backus Naur Form (BNF) is a notation for describing grammars. A grammar is represented by a tuple $\{N, T, P, S\}$, where T is a set of terminals, i.e. items that can appear in legal sentences of the grammar, and N is a set of non-terminals, which are interim items used in the generation of terminals. P is the set of production rules that map the non-terminals to the terminals, and S is a start symbol, from which all legal sentences may be generated.

Below is a sample grammar, which produces individuals similar to those used by Koza [4] in his symbolic regression and integration problems.

```
S = <expr>

<expr>    ::=  <expr> <op> <expr>   (0)
             | ( <expr> <op> <expr>)(1)
             | <pre-op> ( <expr> )  (2)
             | <var>                (3)
<op>      ::= + (4) | - (5) | % (6)
             | * (7)
<pre-op>  ::= Sin (8) | Cos (9)
             | Exp (A)| Log (B)
<var>     ::= 1.0 (C) | X (D)
```

3 Grammatical Evolution

Grammatical Evolution (GE)[9] evolves computer programs in an arbitrary language. Unlike GP, as described in [4], which typically uses abstract syntax trees, GE recognizes a distinction between genotype and phenotype. Genotypes are 8 bit integer (typically referred to as a codon) strings of varying length. Translation to the phenotype is dictated by a context free grammar.

One of the distinguishing features of GE is *intrinsic polymorphism*, that allows it to possess a genotype without non coding regions(introns). Whenever a rule has to be chosen for a non-terminal, a codon is **mod**ed against the number of rules pertaining to that non-terminal. This ensures that every codon produces a rule that is immediately consumed. A complete description of GE can be found in [6].

4 The Chorus System

In a manner similar to GE, Chorus uses 8 bit integer strings to represent the genotypes. However, unlike the intrinsically polymorphic codons of GE, each codon in Chorus corresponds to a particular rule in the grammar. In this way it is similar to GAGS[5], but it does not have the same issue with introns.

Fixed valued codons are different from GE, where the meaning of every codon is dependent upon its predecessors. The 8 bit codon is **mod**ed with the total number of rules in the grammar so as to point to a rule from the grammar. This behaviour differs from GE, where the interpretation of a codon depends upon the state of the mapping process. As described earlier, GE **mod**es a codon with the number of rules relevant to the non-terminal being processed. Depending upon the state of the mapping process, the same codon can be read in for different non-terminals. This suggests that the behaviour of a codon is determined by its situation in the chromosome.

As the total number of rules in the grammar remains constant irrespective of the state of the mapping process, a codon in Chorus behaves in a fixed manner regardless of its location in the genome.

When the mapping begins, the genome is read from left to right, to pick the rules from the grammar so as to arrive at a legal sentence comprising of all terminals. However, fixed valued codons may not necessarily be in the order the mapping process may demand. We keep track of all the rules that we come across in a *concentration table*. The concentration table has an entry for every rule in the grammar and is initialised to all zeros when the mapping begins. Every time a codon is read, concentration of the corresponding rules is incremented in the table.

Consider a sample individual, which has been **mod**ed to give the codons: **8 D D C 2 3**. If we consider the grammar given in Sect. 2, the start symbol is `<expr>` which maps to any of the rules 0 through 3 . While we traverse through the genome to find an *applicable* rule, we can find rules that may be used later. We keep track of all the rules that we come across in the concentration table.

The table is divided into different sections, each pertaining to a particular non-terminal. As the grammar in the Sect. 2 has four non-terminals, the table has four different sections. The first section contains entries for rules 0 through 3, the second consists of rules 4 through 7 and so on. When we have to find a rule corresponding to a non-terminal, we consult the *relevant* section from the table. The rule with the highest concentration at that time is chosen. In case of a tie (e.g at the beginning of the mapping) we read the genome left to right incrementing the concentration of every rule we read in. The reading stops when one of the rules from the relevant section becomes a clear winner. When required, subsequent scanning of the genome will continue from that point.

For the current example, we stop when we encounter rule 2 so that next reading position points to rule 3. The rule with the maximum count in the relevant section (rule 2 in this case) is considered dominant and is chosen, so that the start symbol is expanded into `<pre-op>(<expr>)`. We always stick to the left most non-terminal, a `<pre-op>` in this case. Rule 8 is a clear winner for this case, as it is the only applicable rule that has been read in. We do not need to read from the genome in this case. Thus, we see that a codon may not be consumed the moment it is read in. This delay in the expression of a codon, combined with the requirement to be in majority to make a claim brings the position independence in the system. The important thing is the presence or otherwise of a codon and its location is less so. For more information, consult [8][1].

5 Genetic Operators

The binary string representation of individuals effectively provides a separation of search and solution spaces. This permits us to use all the standard genetic operators at the string level. Crossover is implemented as a simple, one point affair, the only restriction being that it takes place at the codon boundaries.

Bit mutation is employed at a rate of 0.01, whereas crossover occurs with a probability of 0.9. Steady state replacement is used with roulette wheel selection.

As with GE, if an individual fails to map after a complete run through the genome, Chorus uses a wrapping operator to reuse the genetic material. However, the exact implementation of this operator has been kept different from the traditional GE implementation [9]. Repeated reuse of the same genetic material effectively makes the wrapped individual behave like multiple copies of the same genetic material stacked on top of each other in layers. When such an individual is subjected to crossover, the stack is broken into two pieces. When linearized, the result of crossover is different from one or both of its parents at regular intervals. In order to minimize such happenings, Chorus limits wrapping to the initial generation. If an individual undergoes wrapping, it is then *unrolled* by appending all the genetic material at the end of the genome the number of times it is reused. The unrolled individual then replaces the original individual in the population. This altered use of wrapping in combination with position flexibility, promises to maintain the exploitative effects of crossover. Unlike the traditional implementation of GE, those individuals that fail to map in second and subsequent generations are not wrapped, and are simply considered unfeasible.

All the experiments described in this paper employ the altered wrapping both for GE and Chorus, except where mentioned otherwise.

6 Experiments

This paper makes a comparison between the structures of the individuals that are evolved by the two grammar based evolutionary algorithms. To facilitate the cause, we have employed a few different approaches. In the first approach, termed *Segment Rule Profiles*, the individuals are divided into ten segments to see if there are certain kinds of rules dominating certain regions in the genomes. This can help us understand the relationship between the genomes and the derivation trees they encode.

As is the nature of the mapping in both the algorithms, all the genetic material available in the genomes may not be required for the mapping. Instead, a certain percentage of the chromosome may be read in. This study only monitors the *effective* lengths of the genomes that contribute towards the mapping. Considering only effective length is also motivated by the fact that, even though fixed valued codons found in the *tails* or unused portions of Chorus individuals can be sampled, intrinsically polymorphic codons in GE can not be interpreted so as the mapping terminates before reading them in.

We have used two problems to make such a comparison. The first problem is Koza's[4] symbolic regression of the quartic polynomial $(x^4 + x^3 + x^2 + x)$. In order to evolve such a function, rules 0,1,3,4,7 and D from the grammar in Sect. 2 play a significant role. However, if we can use a problem whose solution requires less number of rules, we can have a clearer view of any underlying trends. For this purpose we have used an abbreviated form of the same problem, where the objective function is $x^4 + x^3$.

We then move on to compare the robustness of the individuals produced by each of the algorithms. In this test we compare the perfect individuals produced for the two problems against their nearest neighbourhoods. Bit string representation means that any individual having a hamming distance of 1 exists in the nearest neighbourhood of the individual under observation. Percentage fall off in the fitness is recorded by generating all the nearest neighbours. This provides us with a test of neutrality around the perfect solutions found by the systems.

We also compare the similarity in the rule history of the perfect individuals with their immediate neighbours. It gives us a measure of similarity in structure in the space immediately surrounding those individuals.

We then use the notion of *productive parents* [7], that is, parents that produce perfectly fit offspring, and considers the relationship between these parents and the offspring in terms of the leap in fitness required to get to the ideal fitness.

All the experiments involve 100 independent runs with a population size of 500 spanning 250 generations. Runs terminated upon finding an ideal fitness. All fitness comparisons involve normalized fitness as described in [4]. The experimental setup is primarily the same as used by Koza[4] for the symbolic regression problem.

6.1 Segment Rule Profiles

In GE, every sampled codon decodes to an applicable rule, so the effective length of an individual can be seen as a history of the rules selected.

Position independence in Chorus, however, may not permit such minimal individuals. Effective length in a Chorus individual may possess introns as well as high counts of the competing rules trying to maximise their concentrations to stake a claim. Thus, we have also recorded rule selection histories of the Chorus Individuals to see what kind of derivation trees are being produced.

The figures reported in this section show a snapshot of the 50th generation. It was observed that the rule percentiles reasonably stabilised upon reaching the 50th generation.

Figure 1 shows the rule percentiles in the effective lengths of the Chorus individuals for both the problems. As mentioned earlier, the figures show that the system has figured out the necessary constituents of the ideal solutions, depicted by their higher percentages in different segments. However, some of these significant rules show clearly dominating percentiles in certain segments,

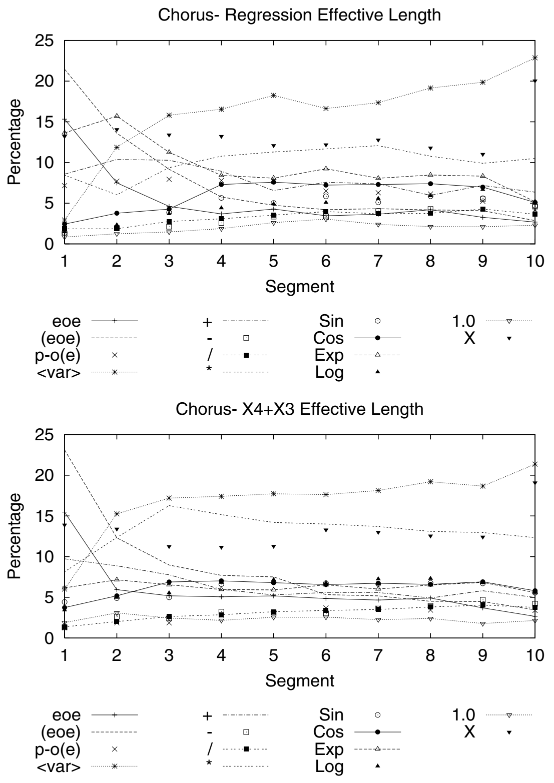

Fig. 1. Rule percentages across the 10 segments for the effective length of Chorus Individuals

and a low profile in the others. Let's consider the regression problem first. If we consider segment 1, the rules `<expr><op><expr>` (rule 0 in the grammar in Sect. 2) and `(<expr><op><expr>)` (rule 1 in the grammar) constitute about 38% of the profile. However, as we move to the right, the percentages of these rules decrease consistently up until segment 5, maintaining a relatively stable proportion thereafter. `<var>`, that links the aforementioned rules to the terminals, has only 2.935% share in segment 1, but as the percentiles of those rules decrease, `<var>` shows a clear tendency to increase attaining its maximum value in the last segment.

The terminal symbol X scores about 13% in segment 1, and does not show as much variation as the rules discussed earlier, except in the last segment.

The two operators $+$ and $*$ start off at the same level. $+$ then takes over briefly for the earlier segments, before $*$ takes the lead in later segments, emphasizing its increased role towards the end of the derivation tree.

Figure 1 shows that the artificial problem shows a trend very similar to the regression problem. The reduced effect of the $+$ operator gives us a much clearer picture of the same phenomenon.

It appears from the findings that the system arranges its genomes in a way that the earlier part primarily consists of the rules that define the structure of the derivation tree and lead to increase in its size. This dominance is further enhanced by the low percentage of the rule `<var>` in the earlier segments, thus ensuring that even a high percentage of the terminal symbol X, will not hinder the structure formation. This is a consequence of the property of the system, which not only allows it to evolve the values of its genes but also the order in which the chromosome may possess them. This can be helpful in forming building blocks by allowing the rules of conflicting roles towards the derivation tree formation to exist in close vicinity.

Figure 2 shows a comparison of the rule selection histories of the two algorithms on the regression problem. The figure for Chorus correlates well with the earlier figure showing a high percentile (about 80%) of the structure imposing rules in earlier segments. `<var>` has a much lower appearance with terminal symbol X being almost non-existent with a share of 0.257%.

GE also shows a clear dominance of the structure forming rules in the first segment. However, there are two noticeable exceptions. The combined percentage of rule 0 (`<expr><op><expr>`) and rule 1 (`(expr><op><expr>)`) is much lower compared to Chorus(about 48% as against 80%). Also, the percentage of `<var>` is much higher, about 18% as against 7.277% in Chorus. The figure is unable to show clearly that the percentage of the terminal X is also higher for GE in segment 1 at 7.312%.

The differences observed in the regression problem become even clearer in the artificial problem (Fig. 3). There is an increase in the difference in combined percentages of rule 0 and rule 1 between GE and Chorus. Also, `<var>` has an increased percentile with GE.

The results of the rule selection history seem to suggest that, while, the individuals produced by Chorus tend to create thinner and deeper derivation trees, GE individuals encode broader trees (see Fig. 4). This fact can be attributed

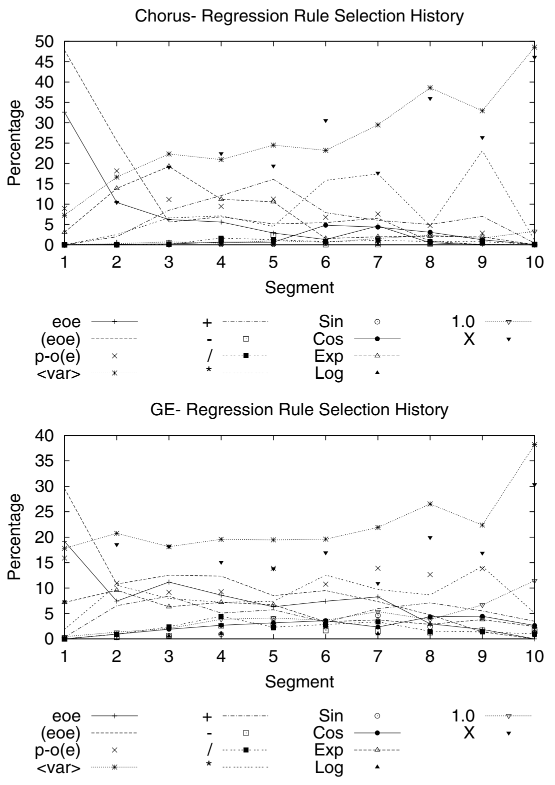

Fig. 2. A comparison of the choices made by GE and Chorus across the 10 segments for the Regression problem

to the depth first mapping employed by both the algorithms. As, earlier on, rules 0 and 1 dominate in Chorus, depth first mapping uses them to increase the depth of the tree. The phenomenon is also visible in GE, but given that it possesses `<var>` with a higher percentage in comparison with Chorus, it may possess broader trees as depicted by Fig. 4. This may have implications towards crossover. We get an impression that Chorus tends to preserve the structure in the earlier part of the genome. Also, the earlier portions possess the terminals that may not have been expressed so far. Thus, while the incoming fragment may contain terminals of its own, it may also be expected to trace a path to the terminals already contained by the receiving individual. Crossover, thus, can potentially behave in a more context preserving fashion.

GE, however, can have more irregularly shaped trees exchanged, possibly leading to a more exploratory behaviour. However, a study focused on the derivation trees can be more conclusive in uncovering such trends.

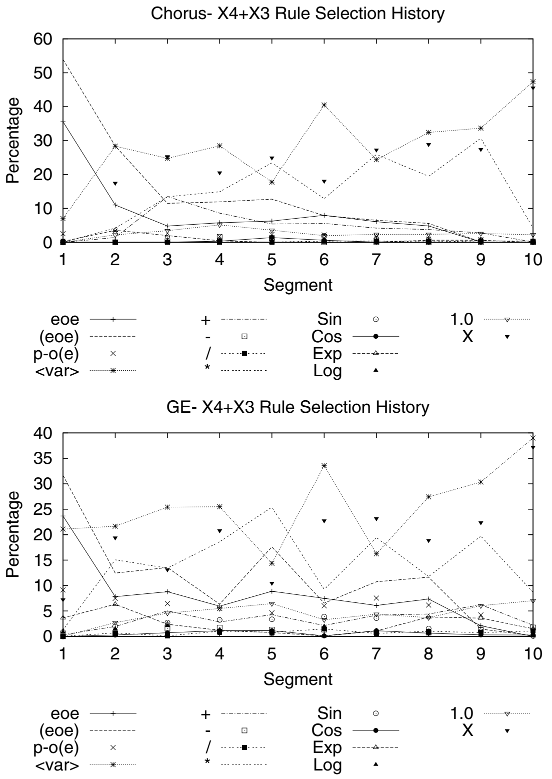

Fig. 3. A comparison of the choices made by GE and Chorus across the 10 segments for the artificial problem

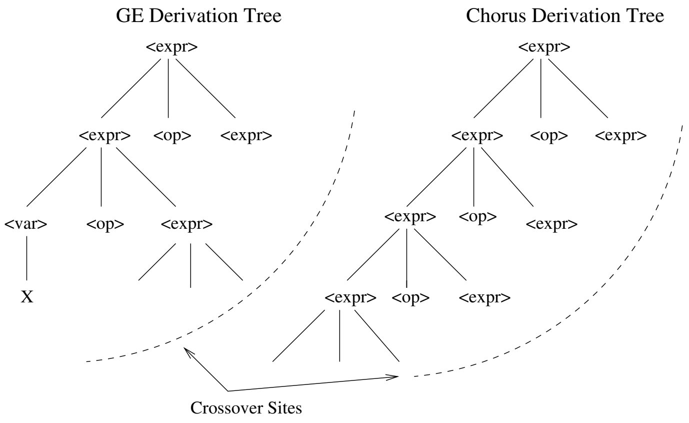

Fig. 4. Partial derivation trees for Chorus and GE

6.2 Fitness Fall Off and Rule History Similarity in the Neighbourhood

In this section we analyse the robustness of the individuals produced by the two algorithms. We consider 32 perfect individuals produced against each problem by GE and Chorus, and generate all of their immediate neighbours. As mentioned earlier, the neighbours are generated only for the effective lengths. This is so because mutations in the tails, will give us exactly the same rule selection histories thus leading to the same phenotypes. This will give a false sense of neutrality in terms of percentage fall off in fitness as well as the similarity in the rule histories. However, the neighbours may require a longer genome length in order to map completely. Thus, we have done two kinds of test. In one case, we allow the neighbours to use the tails if they require them. In the other case we do not allow them to go beyond the effective length of the corresponding perfect individual.

Table 1 explores the neighbourhoods for neutrality in fitness values. The leftmost column shows the fall off in the fitness. The entries in the table show the percentage of immediate neighbourhood lying in a range of fitness fall off. If we consider the regression problem, we can see that Chorus shows 35.017% of the immediate neighbours having perfect fitness even without being allowed to use the tails. With the use of tails, the figure slightly rises to 37.71%. GE shows higher fitness neutrality at about 41.059%. Allowing the use of tails increases this figure very slightly.

The bottom of the table also shows better figures for GE, with Chorus showing high percentage of individuals getting a zero fitness.

The artificial problem also shows that GE seems more robust to handle bit mutation. The allowance to use the tails helps Chorus to decrease the difference in performance. GE, however, remains largely unaffected by it.

Table 2 shows the similarity between the rule histories of the perfect individuals and their neighbours. It shows that most of the neutrality shown in fitness

Table 1. Percentage fall off in Fitness of Immediate Neighbours

%age	Chorus-Regression		GE-Regression		Chorus-$X^4 + X^3$		GE-$X^4 + X^3$	
	No Tails	Tails	No Tails	Tails	No Tails	Tails	No Tails	Tails
0	35.017	37.71	41.059	41.332	33.789	39.129	41.334	41.334
10	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0.018	0.018
40	0	0	0	0	0	0	0.263	0.263
50	0	0	0	0	0	0	0.105	0.105
60	0	0	0	0	0	0	0	0
70	0.0060	0.0060	0.174	0.186	0.018	0.072	0	0
80	6.423	7.188	5.543	5.63	10.391	13.523	9.506	9.664
90	4.031	5.187	4.849	5.146	8.789	10.961	9.979	10.662
100	54.524	49.909	48.375	47.706	47.013	36.314	38.796	37.955

Table 2. Percentage similarity in the Rule Selection History of Immediate Neighbours

%age	Chorus-Regression		GE-Regression		Chorus-$X^4 + X^3$		GE-$X^4 + X^3$	
	No Tails	Tails	No Tails	Tails	No Tails	Tails	No Tails	Tails
0	3.81	3.815	0.409	0.409	4.861	4.843	0.35	0.35
10	2.625	2.613	0.31	0.31	3.829	3.811	0.683	0.683
20	4.303	4.32	0.608	0.608	3.738	3.729	0.735	0.735
30	4.734	4.722	0.57	0.57	3.376	3.222	1.296	1.296
40	4.649	4.546	0.471	0.471	4.001	3.747	0.63	0.63
50	7.347	7.381	0.818	0.818	6.725	6.988	1.576	1.576
60	6.803	6.429	1.128	1.128	7.902	7.476	2.714	2.714
70	5.72	5.811	3.137	3.137	4.435	4.797	7.143	7.143
80	6.281	5.726	11.471	11.471	10.889	7.594	9.191	9.191
90	19.881	18.974	45.3	45.3	18.773	18.049	41.877	41.877
100	33.849	35.663	35.776	35.776	31.472	35.744	33.806	33.806

is basically due to the similar derivation sequences. For instance, in regression problem 33.849% of the neighbours show complete neutrality for Chorus, as against a fitness neutrality of 35.017% (see Table 1), when tails are not allowed. The bulk of fitness neutrality for GE is also derived from neutrality in derivation sequences. However, the difference in 100% fitness neutrality (41.059%) (see Table 1) and 100% rule history similarity(35.776%) is higher compared to Chorus. It suggests that GE has more individuals in the neighbourhood which may be somewhat different in appearance, yet they are able to reach the ideal fitness. A similar trend is observable in the artificial problem.

The results suggest that GE is more tolerant to bit mutation as compared to Chorus. It is helped by intrinsic polymorphism that provides more neutrality in GE codons. A codon is moded only with the number of rules pertaining to the non-terminal which has to be mapped. Every non-terminal in the grammar given in Sect. 2 has 4 rules except `<var>`, which has 2. This means an 8 bit codon can represent the same rule in 64 different ways. The figure rises to 128 for `<var>`.

Chorus, on the other hand, uses fixed valued codons by moding them with the total number of rules in the grammar(14 in this case). This means every codon can represent a rule in about 18 ways, a figure considerably less than that of GE.

6.3 Productive Parents

In this section we compare the fitness of the parents of the ideal solutions to their offsprings. This gives us an idea about the relationship between the parents and their offsprings in terms of fitness.

Table 3 shows that on average the parents are quite distant from their offsprings in the fitness landscape. As is the case in neighbourhood tests, GE does better by evolving parents that are closer to the ideal solutions in terms of fit-

Table 3. Percentage distance between the fitness of productive parents and the ideal fitness. A low score indicates a higher fitness

	Chorus		GE	
	Regression	$X^4 + X^3$	Regression	$X^4 + X^3$
Maximum	91.74	92.95	83.55	82.86
Average	77.11	79.07	66.53	73.8
Minimum	49.31	41.46	21.87	31.7

ness. This suggests that GE has a relatively smoother approach towards the ideal solutions.

Artificial problem reported that 33% of the Chorus parents and 59% of GE parents encoded for the building block $x^3 + x^2$ which is 75% away from the ideal fitness. While it shows the ruggedness of the search space, it is interesting to see how the systems managed to find their way through low fitness area instead of getting trapped into highly fit but possibly misleading areas of search space.

7 Conclusions

Chorus and Grammatical Evolution are from the same family of algorithms, and clearly have much in common as a result. This paper investigates and compares the two algorithms in terms of the individuals they produce. There appears to be a convergence in behaviour, yet some differences have been pointed out. The segment profile tests show that, despite Chorus possessing position independence, an order seems to settle down as the evolution progresses. In particular, the genes that deal with structural information, tend to be at the start of individuals.

This emergent order shows that the system has the potential for building block formation, and produces individuals more conducive for crossover operations. A study focused on derivation trees can shed further light in exploring the effects of crossover.

Intrinsic polymorphism enables GE to have more tolerance towards bit mutation compared to fixed valued codons, while Chorus individuals require significantly larger codons to achieve the same neutrality.

A result of this is that GE also seems to enjoy a smoother passage towards the ideal fitness in terms of fitness distance between the productive parents and the offsprings. Further investigation into the roles of the genetic operators can help us understand it better.

References

1. Azad, R.M.A., Ryan C., Burke, M.E., and Ansari, A.R., A re-examination of The Cart Centering Problem using The Chorus System. In the proceedings of *Genetic and Evolutionary Computation Conference (GECCO), 2002* (pp. 707–715), Morgan Koffman, NYC, NY, 2002.

2. Freeman, J.J., “A Linear Representation for GP using Context Free Grammars” in *Genetic Programming 1998: Proc. 3rd Annu. Conf.*, J.R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D.B. Fogel, M.H. Garzon, D.E. Goldberg, H. Iba, R.L. Riolo, Eds. Madison, Wisconsin: MIT Press, 1998, pp. 72–77.
3. Keller, R. and Banzhaf, W., “GP using mutation, reproduction and genotype-phenotype mapping from linear binary genomes into linear LALR phenotypes” in *Genetic Programming 1996: Proc. 1st Annu. Conf.*, J.R. Koza, D.E. Goldberg, D.B. Fogel, and R.L. Riolo, Eds. Stanford, CA: MIT Press 1996, pp. 116–122.
4. Koza, J.R., Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992.
5. Paterson, N., and Livesey, M., “Evolving caching algorithms in C by GP” in *Genetic Programming 1997: Proc. 2nd Annu. Conf.*, MIT Press, 1997, pp. 262–267. MIT Press.
6. O’Neill M. and Ryan C. Grammatical Evolution. *IEEE Transactions on Evolutionary Computation.* 2001.
7. O’Sullivan, J., and Ryan, C., An Investigation into the Use of Different Search Strategies with Grammatical Evolution. In the proceedings of *European Conference on Genetic Programming (EuroGP2002)* (pp. 268–277), Springer, 2002.
8. Ryan, C., Azad, A., Sheahan, A., and O’Neill, M., No Coercion and No Prohibition, A Position Independent Encoding Scheme for Evolutionary Algorithms – The Chorus System. In the Proceedings of *European Conference on Genetic Programming (EuroGP 2002)* (pp. 131–141), Springer, 2002.
9. Ryan, C., Collins, J.J., and O’Neill, M., Grammatical Evolution: Evolving Programs for an Arbitrary Language, in *EuroGP’98: Proc. of the First European Workshop on Genetic Programming* (Lecture Notes in Computer Science 1391, pp. 83–95), Springer, Paris, France, 1998.