Interactive Interpolating Crossover in Grammatical Evolution

James McDermott, Michael O'Neill, and Anthony Brabazon

Abstract— **Interactive interpolating crossover allows a user to quickly see a large number of individuals formed by interactively-controlled interpolation between two or more parents. We study it here for the first time in the context of grammatical evolution (GE). We define methods of quantifying the behaviour of interpolations and use them to compare two methods of performing interpolation and two encodings for GE, one standard and one new. We conclude that a Cartesian interpolation combined with a novel developmental-style GE encoding gives the most usable results. We make connections between our work and broader issues of genotype-phenotype mappings, landscapes, and operators.**

I. Introduction

A fundamental problem in interactive evolutionary computation (IEC) is the fitness evaluation bottleneck [1]: human evaluation of fitness is very slow, compared to computer evaluation. This reduces the viable population size and generation count, which affects search performance.

A second issue is that sometimes users find typical implementations of the search operators to be somewhat inflexible. Crossover, for example, generally works by random choice of a crossover point, which affects in some sense the relative contribution of the two parents: but it is easy to imagine a scenario in which a user wished to have direct control of this parameter.

Both of these problems provide motivation for the idea of interactive interpolating crossover. Here, the user is given interactive, real-time control of a parameter which affects the relative contributions of two parents to the offspring. In the ideal case, this parameter can be varied to produce an interpolation between the two parents which appears smooth, monotonic, and consistent. The user has control over the relative contributions of the two parents, and can evaluate a large number of distinct individuals quickly, focus on the best areas in the interpolation, and select good individuals for the next generation.

In a real-valued space, and if the map from genotype to phenotype is well-behaved (e.g. has high locality [2], [3], a good fitness-distance correlation [4], etc.), then the natural definition of interpolation—a linear interpolation in the genotype space—will indeed give rise to this ideal case, a smooth interpolation in the phenotype space.

When working in discrete spaces (as in many types of genetic programming (GP)), and in the presence of badly-behaved genotype-phenotype mappings, interpolation will sometimes be less smooth, monotonic, and consistent, and as a result will be less usable. Our first aim in this paper, therefore, is to be able to quantify the behaviour of an interpolation. We propose several methods of measuring interpolation behaviour in terms of phenotypic properties. This is carried out in Section III.

Our previous and ongoing work uses grammatical evolution (GE), a type of GP, for several aesthetic and design problems [5], [6]. The GE search space is discrete, and the genotype-phenotype mapping is in at least some cases badly-behaved. Our second aim, then, is to define interpolation for the first time in the GE setting, and make it behave as well as possible. Therefore we define several alternative implementations applicable to GE, in Section IV and compare their properties using the methods mentioned above, in Section V (we do not report any evolutionary runs). The results are discussed in Section VI. Section VII then gives our conclusions and hints at future work.

To begin, we examine previous work in the areas of GE, interactive EC, and the design of crossover operators.

II. Previous Work

A. The Design of Crossover Operators

The classic GA [7] achieves recombination of genetic material using a one-point crossover operator. The crossover point, a single index into the genome array, is randomly generated, and the "heads" and "tails" (the portions of data before and after this point) are swapped between two parents. Many other possibilities have been proposed for this and other representations, and the properties desirable in a crossover operator have been described in representation-independent terms [8], [9], [10]. GE typically uses a slight modification of the GA one-point crossover to allow for its variable-length representation: one crossover point *per parent* is generated. Variations in GE include multi-point crossover and a crossover sensitive to the number of *used codons* (often less than the number of codons in the genome).

The idea of crossover by interpolation has been used in several non-interactive EC implementations [11], [12], and even outside EC, in artistic, musical, design and graphics software [13], [14]. The colour wheel, familiar in graphics applications, maps a 2-dimensional interpolating controller to a 3-dimensional space. The Metasurface [15] allows multiple synthesizer parameters to be recombined using a 2-dimensional interpolating controller. Bencina [15] and Goudeseune [16] each give accounts of the properties required for successful interpolation. It is seen as important that intermediate points should lie "between" the end-points of the interpolation, in a perceptual sense; and interpolation should be perceived as continuous or smooth. In fact, a model

The authors are with the Natural Computing Research & Applications group, Department of Computer Science and Informatics, UCD CASL, University College Dublin, Dublin 4, Ireland. Email of contact author: jamesmichaelmcdermott@gmail.com. James McDermott is funded by the Irish Research Council for Science, Engineering and Technology under the Empower scheme.

978-1-4244-8126-2/10/$26.00 ©2010 IEEE

for good interpolation is perceptual *linearity*. Psychophysical experimental results suggest that any scalar-valued stimulus (such as sound or colour intensity) gives rise to a non-linear perceptual effect. This non-linearity can be corrected for using mappings derived from the same experiments, so that in the case of interpolation, a linear mapping from the interpolation variable to the perceived effect is achieved. The seminal, though partially disputed, body of experimentation is reported by Stevens [17]. Apart from our initial study [18], we are unaware of any previous research quantifying interpolation properties and the usability of interpolations which deviate from perceptual linearity. The interpolation properties we describe later all measure, in different ways, how much an interpolation deviates from linearity.

B. Interactive EC

IEC is a technique suitable to aesthetic, subjective, and multi-objective domains [19]. The central motivation for using IEC is usually that it is impossible to define a purely computational method of quantifying individuals' fitness values. Human evaluation of fitness is therefore common [20]. Interactive direct selection (obviating the need for explicit fitness) and interactive control of operators have also been used by various authors [21], [22]. Interactive interpolating crossover brings together these techniques, since it allows the user to control the behaviour of the operator and select offspring interactively.

In previous work, we have shown that an interactive interpolating crossover has the potential to serve as an intuitive, usable, and efficient method of interacting with an evolutionary population, helping to alleviate the fitness evaluation bottleneck [23], [24], [25]. The central advantages of the idea are that a large number of individuals can be judged very quickly; that the user can search in a coarse-grained way through poor areas of the interpolation, but focus in a fine-grained way when a good area is encountered; and that wasteful tasks (awarding numerical values to bad individuals, unnecessary mouse-clicks, etc.) are minimised.

We have also developed methods of quantifying the behaviour of interpolations. and used experiments with human subjects to show that they have some predictive power: that is, interpolations with good values according to these methods gave faster, more accurate results in interpolation tasks. Interactive interpolation has not otherwise been applied to a discrete domain. Figures 1 and 2 give an intuitive demonstration of good and bad interpolation behaviour.

C. Introduction to Grammatical Evolution

Grammatical evolution (GE) is a proven technique for representing code in diverse languages with arbitrary constraints on its form [26], [27]. The language syntax and the allowable or desirable syntactic forms are specified by a context-free grammar in Backus-Naur Form. The grammar below generates strings representing abstract coloured trees.

```
<tree> ::= (<node> <tree> <tree>) | <leaf>
<node> ::= black | white
<leaf> ::= red | blue
```

An individual's genome is an integer array, which specifies the grammar productions to be chosen during the derivation process. Consider the genome `[6, 5, 3, 0, 3, 1, ...]`. Starting with the first symbol `<tree>`, we use the genome to choose productions as follows. Taking the first codon (6) mod the number of productions in the `<tree>` rule (2), we get 0, and so choose the 0th production. Thus we replace `<tree>` with `(<node> <tree> <tree>)`. We next replace the left-most non-terminal in this new string, `<node>`. We read the next codon (5), and again there are 2 possible productions, `black` and `white`. 5 mod 2 is 1, so `<node>` is replaced with `white`. Continuing in this way, we get the following derivation.

```
   <tree>
-> (<node> <tree> <tree>) [codon 6 % 2 = 0]
-> (white <tree> <tree>)  [codon 5 % 2 = 1]
-> (white <leaf> <tree>)  [codon 3 % 2 = 1]
-> (white red <tree>)     [codon 0 % 2 = 0]
-> (white red <leaf>)     [codon 3 % 2 = 1]
-> (white red blue)       [codon 1 % 2 = 1]
```

Using the standard lisp-style syntax, this string corresponds to a tree with a white node at the root, with red and blue children (the 6th tree in Figure 2). Note that it is possible that all codons will be used up before all non-terminals have been removed from the grammar. When this occurs, the individual is declared *invalid*, its phenotype and fitness are not defined, and it is not available for selection (exceptions can occur through the use of "wrapping" of the genome, and repair operators, neither of which is used in this paper).

Of particular interest in this context is the behaviour of the genotype-phenotype mapping in GE, and the effects of the mutation and crossover operators. Rothlauf and Oetzel [3] questioned the locality property of the genotype-phenotype mapping in GE, claiming that it was inferior to the locality in standard GP. Previous work has also demonstrated the existence of a "ripple effect" in GE [28]. A single mutation in a chromosome can lead to changes in the *interpretation* of subsequent codons, and therefore large phenotypic changes, even though the subsequent codons themselves remain unchanged. This can be seen as an instance of low locality. It is of interest here because the same effect was also observed in crossover [28]: the tail of a chromosome inherited from one parent may be interpreted differently in the child, due to the ripple effect from the head of the chromosome inherited from the other parent. Any interpolating crossover is likely to encounter the same issue, making the implementation of a well-behaved interpolating crossover very difficult in the context of GE.

III. Measuring Interpolation Behaviour

In the following, we will assume that interpolating crossover is to be carried out between two parents, A and B, producing a single offspring X (because the user must evaluate the offspring in real-time, creating two children is inappropriate). The offspring X depends on the value of a single, real-valued interpolation parameter $t \in [0, 1]$ which is under the control of the user. We require that when $t = 0$, the output $X = A$; when $t = 1$, $X = B$; and for intermediate

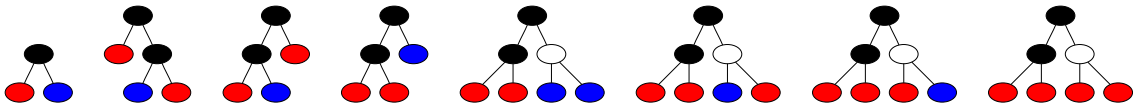


Fig. 1. A relatively well-behaved interpolation. Internal nodes are black and white, and leaves are red and blue. Interpolation is between the left-most and right-most trees, and the input variable (i.e. slider position) increases from left to right. Note that node count is non-decreasing, and that neighbours are very similar to each other.

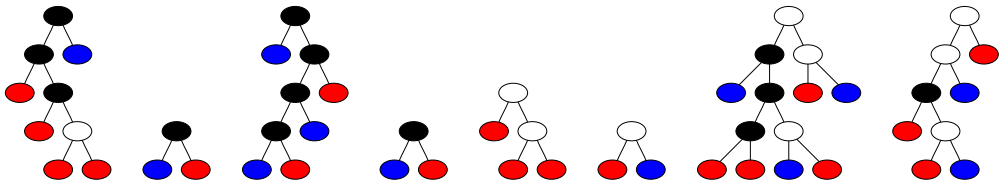


Fig. 2. A badly-behaved interpolation. Again, this is an interpolation between the left-most and right-most trees. Neighbours are not necessarily similar to each other and there is no apparent trend from left to right.

values of t, X should vary "between" A and B in some sense. The user's task during each crossover event is to vary t (using a controller such as an on-screen slider) to choose the best available X.

Our approach to measuring interpolation behaviour is motivated by considering first the one-dimensional linear interpolation between two real numbers, a and b. The output, a real-valued number, is defined as $x = a+t(b-a)$. Such an interpolation has properties such as *continuity*, *monotonicity*, and *consistency*. We believe it is natural to assume that such an interpolation can not be improved upon, from the point of view of the user attempting to find a target item (e.g. a desired sound volume level) which corresponds to a quantity x. The properties mentioned above can be defined in a quantitative way and our previous work has shown that they are, to some extent, predictive of interactive interpolation usability. We now add two further properties. The *correlation* between the input t and the output x is a general-purpose summary of their relationship; and *geometricity*, which is motivated by the work of Moraglio [10], is measured here as the proportion of intermediate values x which are inside the range defined by the end-points a and b. Numerical definitions for these five properties will be given next.

It is easy to see that "changes in direction" will make the user's task more difficult, and so *monotonicity* is important. Since we wish to distinguish not only between monotonic and non-monotonic mappings, but also between mappings with different *degrees of monotonicity*, we define monotonicity to depend on the number of changes of direction in the interpolation:

$$\text{monotonicity} = 1 - \frac{\text{changes}(x)}{\text{length}(x)}$$

where x is the value of the interpolation's output variable, $\text{changes}(x)$ gives the number of times the array x changes direction, and $\text{length}(x)$ is the length of the array.

Locality is the property that a small change in input leads to a small change in output. Succeeding values in the interpolation correspond to small changes in input, so we measure the absolute size of the differences between their outputs:

$$\text{locality} = \frac{1}{1 + \max_i(|x_{i+1} - x_i|)}$$

Consistency is intended to express the idea that all changes to the input have roughly the same effect. We differentiate the output data x to give $x'_i = x_{i+1} - x_i$, and since we expect all values x'_i to be quite similar, we can estimate consistency by calculating the standard deviation:

$$\text{consistency} = \frac{1}{1 + \text{stddev}(x')}$$

Correlation is defined as *the absolute value of* a Pearson's correlation between the input and output variables t and x. A perfect correlation indicates a linear relationship. We take the absolute value since linear-increasing and linear-decreasing relationships are equally good. In cases where the proxy variable is constant the correlation is undefined. However, since this arises for well-behaved interpolations (e.g. both parents and all offspring have the same tree depth) we define our behaviour measure to have the value 1 in this case.

A *geometric* crossover is one in which the offspring is genotypically "between" the parents [10] (p. 36). All of our interpolating crossovers will conform to this definition. We will measure geometricity in the phenotype space as the proportion of values x which are inside the range defined by the endpoints. This is motivated by the idea that values outside this range are unexpected to the user. This approach has been used to measure geometricity for non-interpolating crossovers [30]. This equation assumes $a \leq b$; if not, we swap them.

$$\text{geometricity} = \frac{\#\{i : x_i \in [a, b]\}}{\text{length}(x)}$$

All five properties return higher values for "better" behaviour, as demanded by their names. Note that of the five, both locality and consistency depend on the magnitude of the data. A relatively well-behaved mapping into the range

$[0, 100]$ might well have lower values for locality and consistency than a badly-behaved mapping into the range $[0, 1]$. Therefore it is possible to compare locality and consistency values only between mappings into similar ranges. The others are independent of the magnitude of the data.

The above definitions are phrased in terms of a single output value x, but they can easily be extended to multi-dimensional Cartesian spaces. However we need to show that these definitions are useful in non-Cartesian spaces, such as the space of tree structures common in GP and GE. To achieve this, we take an approach called *proxy variables*. A proxy variable is a numerical property of a phenotype which we use in place of the single output value x in the definitions above. A good example in the case of tree-structured phenotypes is the node-count. This is a numerical quantity, and in a well-behaved interpolation we expect that it should vary smoothly between the two end-points. If two parents have node-counts 5 and 7, then offspring of node-count 22 would indicate a badly-behaved interpolation. Other natural proxy variables include the tree depth, and the count of particular node types. The values at the fitness cases, for symbolic-regression style problems, might be regarded as a set of proxy variables: this approach has been used to characterise the geometricity of crossover [30].

In this paper, we use several proxy variables, applying each measure of interpolation behaviour to the set of values of each variable across an interpolation, and present the results for each. The proxy variables we use are these: node-count; tree depth; and node-count for nodes of type `red` *minus* node-count for nodes of type `blue` (working in the space of abstract trees of coloured nodes).

There is one other important phenotypic property: the *validity* of the phenotype, as described in Section II-C. In an ideal interpolation between valid parents, all of the potential offspring should also be valid. We will report the proportion of invalid individuals achieved through random generation and through interpolations between valid parents.

The final statistic of interest will be the number of times X changes as t is varied from 0 to 1. A good interpolation will not simply contain a single jump from A to B: many small gradual changes are preferable.

IV. Definitions of Interpolation

We will define four possible implementations of interpolation: two interpolation types crossed with two encodings for GE. *Cartesian interpolation* works by regarding the GE search space as being Cartesian, and performing the natural linear Cartesian interpolation. *Single-point interpolation* is similar to a single-point crossover in which the crossover point is under user control.

The first of our two encodings is the standard variable-length integer GE encoding. We also define a new encoding for GE with the intention of giving the search space a more real-valued and Cartesian character: we term this a *developmental* encoding. Either interpolation type can be used with either encoding.

We will use notation such as A_i to indicate the ith codon in A's genome. The terms "head" and "tail" mean initial and final sections of the genomes respectively.

A. Cartesian Interpolation

The GE search space consists of variable-length integer arrays. The integer values are constrained to be in the range $[0, m]$ where typically m is the maximum size of an unsigned integer. We interpolate by setting the tail of the output to be the tail of the longer parent, and the head (up to the length of the shorter) is formed by interpolation of the individual codons of the heads of the parents. That is, for each index $i \in [0, \ell)$ (where ℓ is the length of the shorter parent):

$$X_i = A_i + t(B_i - A_i)$$

For each index $i \geq \ell$, we set $X_i = A_i$ or $X_i = B_i$ according to which parent's genome is longer.

B. Single-Point Interpolation

A standard GE crossover is one-point crossover, where a crossover index less than the length of the shorter parent is chosen randomly, and heads and tails before and after this index are swapped to produce two children. We can make this work as an interpolation operator essentially by allowing the user to control the choice of the crossover index. In practice we make one modification, with the aim of avoiding invalid individuals: the crossover happens as if both parent genomes were the length of the shorter, and the longer parent's extra material is then copied to the end of the offspring. One-point crossover leads to two offspring: in our experiments no bias is introduced by simply discarding the first.

C. Developmental Encoding and Interpolation

It is possible to define methods of mapping integer arrays to strings *other* than the standard GE mapping. All that is required is a method of choosing between possible grammar productions at each step of the derivation. One possibility is for the genome to be an integer array in which each codon corresponds to a single grammar production. Each codon (recall, an integer in $[0, m]$) is divided by m to give a floating-point value in the range $[0, 1]$. Whenever a choice must be made between multiple productions in a single rule, that with the largest corresponding value is chosen. Its value is then decremented by the product of the production values, so that it is not necessarily the largest value at the next iteration of the derivation. These decrements apply only during derivation: the original genetic material is passed on to offspring, if any.

Consider again the simple grammar shown in Section II-C, and suppose m is 100, and we have this genome of 6 values: `[90, 85, 30, 40, 30, 25]`. We now annotate each production with the corresponding codon value (after translation to floating-point):

```
<tree> ::= (<node> <tree> <tree>) [0.9] | <leaf> [0.85]
<node> ::= black [0.3] | white [0.4]
<leaf> ::= red [0.3] | blue [0.25]
```

This time, derivation proceeds as follows. We begin with the start symbol, `<tree>`. Of the two productions we can choose for it, we choose that corresponding to the larger value (0.9), `(<node> <tree> <tree>)`. We then reduce that value by the product of the values for those productions, so the new value associated with `(<node> <tree> <tree>)` is $0.9 - 0.9 * 0.85 = 0.135$. This reduction means that the production with the highest value of its rule at the beginning of derivation is not then chosen at every opportunity.

```
   <tree>
-> (<node> <tree> <tree>) [0.9 > 0.85; 0.9 -> 0.135]
-> (white <tree> <tree>)  [0.3 < 0.4; 0.4 -> 0.28]
-> (white <leaf> <tree>)  [0.135 < 0.85; 0.85 -> 0.736]
-> (white red <tree>)     [0.3 > 0.25; 0.3 -> 0.225]
-> (white red <leaf>)     [0.135 < 0.736; 0.736 -> 0.637]
-> (white red blue)       [0.225 < 0.25; 0.25 -> 0.194]
```

This representation is intended to be loosely "developmental" in the sense of Fontana [31]. The system of real-valued codons with dynamically diminishing values might be seen as an idealised representation of instructions to create stocks of different protein-types, which then diminish as they are used to create cell structures. The original instructions are passed on to the next generation undiminished. The representation has the effect of giving the search space a more real-valued and Cartesian character. Importantly, all individuals now have the same genome length, and by treating codons as floating-point values it becomes natural to perform numerical interpolation. The representation is also comparable to Chorus [29], though the Chorus representation does not have an explicitly Cartesian character.

We can now define interpolation as linear Cartesian interpolation. For each index $i \in [0, n)$ (where n is the number of rules in the grammar and hence the length of each genome):

$$X_i = A_i + t(B_i - A_i)$$

For completeness, we will also test Single-Point Interpolation in the developmental encoding. Its definition is the same as in the standard GE encoding.

V. Experiments and Results

A. Interpolation Behaviour

In this experiment, we perform many interpolations in a space of abstract trees, as defined by the grammar below. It is a non-recursive version, producing trees of depth 5 or less, of the grammar given in Section II-C. With a large enough choice of genotype length, this avoids invalid individuals. The motivation for this choice of grammar is to use an abstract domain, more similar to a design problem than to a typical non-interactive EC problem. This abstract domain also allows us to extract proxy variables unambiguously, and avoids tying our conclusions to a specific domain.

```
<tree0> ::= (<node> <tree1> <tree1>)
<tree1> ::= (<node> <tree2> <tree2>) | <leaf>
<tree2> ::= (<node> <tree3> <tree3>) | <leaf>
<tree3> ::= (<node> <tree4> <tree4>) | <leaf>
<tree4> ::= <leaf>
<node>   ::= black | white
<leaf> ::= red | blue
```

TABLE I
Number of phenotypic changes per 100-step interpolation. Higher is better.

	Mean (Std. dev)
Standard SinglePoint:	7.6 (4.3)
Standard Cartesian:	98.1 (1.9)
Developmental SinglePoint:	4.7 (3.2)
Developmental Cartesian:	29.2 (24.9)

TABLE II
Proportion of invalid individuals occurring in randomly-generated individuals and as interpolation offspring from valid parents.

Representation	Random generation	Interpolation
Standard GE	29%	
SinglePoint		14%
Cartesian		26%
Developmental	26%	
SinglePoint		3%
Cartesian		4%

Interpolation is performed under four conditions, as discussed previously: two types of encoding crossed with two types of interpolating operator. For each condition we randomly generate 100 pairs of parents. The number of times the phenotype changes, as the interpolation parameter t is varied between 0 and 1, is given for each condition in Table I (averaged across 100 interpolations). The proxy variables' behaviour is graphed in Figures 3 and 4 for a randomly-chosen sample of 10 interpolations from the 100 (since these graphs become difficult to read for more than 10). For each interpolation, we summarise the proxy variables' behaviour using the measures of behaviour as discussed in the previous two sections. These measures are then averaged across 100 repetitions. The results are shown in Tables III and IV. Note that in Figures 3 and 4 and in Tables III and IV we omit the results for Depth, to save space. They are very similar in character to those for NodeCount.

B. Invalid Individuals

In this experiment, we briefly investigate the phenomenon of invalid individuals. Recall that in standard GE, an individual is deemed invalid if the codons in its genome are all used up before derivation is complete (see Section II-C). The developmental encoding does not "use up" codons in the same way as standard GE. However, it is sensible to impose a limit in the number of iterations which can be processed during the mapping of a developmental-encoded individual, to avoid potential infinite loops. When this limit is exceeded, the individual is deemed invalid.

To test the impact of invalidity in the two encodings, we use the same, non-recursive grammar as before, and we impose a maximum number of derivation steps (20) which is less than the number required to guarantee that derivation completes (45). We turn wrapping off for the standard encoding (and it does not apply to developmental). We randomly

TABLE III

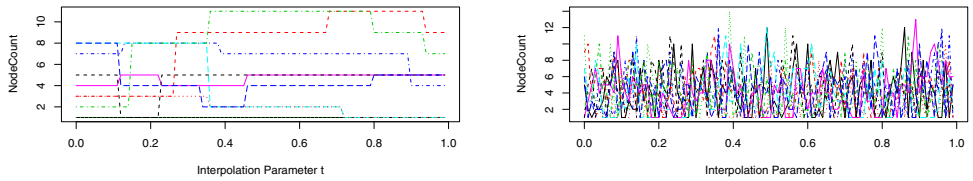
NODECOUNT INTERPOLATION BEHAVIOUR (MEANS AND STANDARD DEVIATIONS). HIGHER IS BETTER. '*' INDICATES A SIGNIFICANT CONTRAST ($p < 0.05$). '+' INDICATES A SIGNIFICANT ADVANTAGE FOR STANDARD-SINGLEPOINT OVER DEVELOPMENTAL-CARTESIAN; '-' THE OPPOSITE.

Representation Interpolation	Standard SinglePoint	Standard Cartesian	Developmental SinglePoint	Developmental Cartesian	Significance
Monotonicity	0.99 (0.01)	0.37 (0.10)	0.99 (0.01)	0.97 (0.04)	* +
Locality	0.32 (0.24)	0.09 (0.01)	0.34 (0.29)	0.33 (0.25)	*
Consistency	0.73 (0.14)	0.21 (0.03)	0.72 (0.16)	0.72 (0.14)	*
Correlation	0.63 (0.29)	0.07 (0.05)	0.67 (0.27)	0.72 (0.26)	* -
Geometricity	0.80 (0.26)	0.45 (0.26)	0.96 (0.06)	0.86 (0.22)	*

TABLE IV

REDBLUE INTERPOLATION BEHAVIOUR.

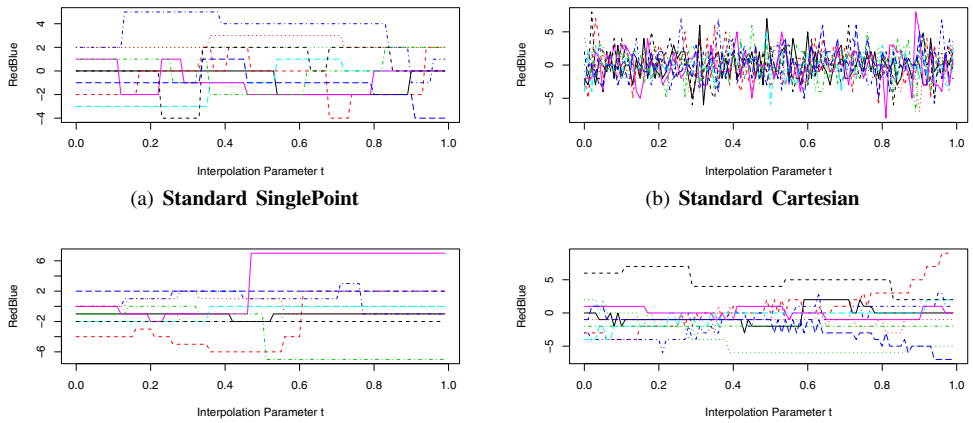
Representation Interpolation	Standard SinglePoint	Standard Cartesian	Developmental SinglePoint	Developmental Cartesian	Significance
Monotonicity	0.97 (0.02)	0.39 (0.05)	0.98 (0.01)	0.92 (0.08)	* +
Locality	0.28 (0.13)	0.10 (0.02)	0.37 (0.33)	0.32 (0.19)	* -
Consistency	0.69 (0.10)	0.24 (0.02)	0.71 (0.20)	0.66 (0.14)	*
Correlation	0.48 (0.28)	0.08 (0.06)	0.72 (0.27)	0.68 (0.27)	* -
Geometricity	0.66 (0.27)	0.48 (0.23)	0.88 (0.17)	0.83 (0.24)	* -



(a) **Standard SinglePoint** (b) **Standard Cartesian**

(c) **Developmental SinglePoint** (d) **Developmental Cartesian**

Fig. 3. Behaviour of **NodeCount** proxy variable in a sample of 10 interpolations.



(a) **Standard SinglePoint** (b) **Standard Cartesian**

(c) **Developmental SinglePoint** (d) **Developmental Cartesian**

Fig. 4. Behaviour of **RedBlue** proxy variable in a sample of 10 interpolations.

generate over 500 individuals for each encoding and report in Table II the proportion which are invalid. For each condition we then perform interpolation between the good individuals, taken in pairs, and report the proportion of invalid individuals obtained, again in Table II. Note that these proportions would all be increased if we used a more recursive grammar, or reduced the number of derivation steps allowed; they would be decreased by using a "smaller" grammar or increasing the number of derivation steps allowed.

VI. Discussion

One surprising result emerged with only a few minor exceptions: all three proxy variables and all five measures of behaviour tended to be quite consistent. A condition which performed well on one tended to perform well on all. This is in contrast to results achieved in previous work for a set of hand-defined interpolations in a simple one-dimensional domain [18]. The differing conclusions drawn using different measures and variables in that experiment confirmed our hypothesis that multiple measures and variables were useful: this new result, in the discrete, grammatical domain, suggests that multiple measures of behaviour and proxy variables may be unnecessary for future work.

Comparing the four experimental conditions, the strongest result is the very poor behaviour of **Cartesian interpolation in the standard representation**. Although the high number of phenotype changes per interpolation (98 of a possible 100: Table I) is good, the figures show that these are not gradual, ordered changes. Rather, the proxy variables change in a very noisy way (Figures 3 and 4). The user of an interpolation like this would experience essentially an unordered selection of individuals, negating any advantage of interpolation. The noisiness is reflected in the values for interpolation behaviour measures (Tables III and IV). This condition is consistently the lowest (worst) for all three proxy variables and all five measures of behaviour. This is the expected result: the GE search space is not Cartesian, in that it is not composed of a fixed number of orthogonal dimensions, so Cartesian navigation in it is the wrong choice.

The very low number of distinct phenotypes available using **single-point interpolation in the developmental representation** (2-8) tends to defeat the purpose of interpolation. The user will not be able to choose from a gradually-changing spectrum of individuals. Although good values are achieved for the measures of behaviour, these are strongly skewed by interpolations of very few distinct individuals, in which it is "easy" to achieve (for example) a perfect monotonicity. They should be treated with caution. Again, this combination of interpolation type and representation is not well-matched, since the individual genes in the developmental genome are essentially continuous in character, and single-point interpolation does not take advantage of this.

The **single-point interpolation in the standard representation** is much better. The graphs appear to show relatively few instances of very bad behaviour. The number of available individuals in the standard single-point interpolations is low (4-12), but significantly better than in single-point developmental. It is similar to the average number of distinguishable individuals available in interpolations in previous work on sound synthesis [25]. Since this work was in the context of a successful real-valued GA, it suggests that this interpolation in the discrete GE domain has the potential to be successful also. Again, the relatively low number of distinct individuals means that we must be cautious in interpreting the good values achieved for interpolation behaviour.

Finally, the **Cartesian interpolation in the developmental representation** appears to perform very well. It achieves a high number of distinct individuals per interpolation, and the graphs seem to show quite a few instances of smooth, trended behaviour. The measures of behaviour confirm this.

The proportion of invalid individuals generated through random generation was quite similar between the two encodings, just over 1 in 4. For individuals produced through interpolation between valid parents, however, there was a very large difference: there were far fewer invalid individuals for the developmental encoding. This is largely explained by the lower number of distinct individuals per interpolation achieved using the developmental encoding.

VII. Conclusions and Future Work

We have developed five quantitative measures of interpolation behaviour, each of which can be applied to any of several "proxy variables" which measure phenotypic properties. These methods allow use to compare, in a principled way, the behaviour of two encodings for GE (the standard encoding and a novel, developmental-style one), and two methods of defining interpolation (a Cartesian one and one derived from single-point crossover). The two encodings and two interpolations were compared in a crossed experimental format. For each of these four conditions, we also analysed numbers of invalid individuals and of distinct phenotypes.

The Cartesian–standard and single-point–developmental conditions can now be disregarded since they perform very poorly, giving very noisy interpolations in one case and too few distinct individuals in the other.

The most interesting comparison to make, therefore, is between Cartesian–developmental and single-point–standard. Each out-performs the other in a few cases, the statistically significant ones being marked in the final columns of Tables III and IV. Experimentally, therefore, the biggest difference between them is the much larger number of individuals produced using the Cartesian-developmental condition. It achieved measures of behaviour which were as good as, if not better than, those for single-point–standard, *despite* this larger number of individuals. Since the larger number of distinct individuals tends to make it more difficult to achieve good measures of behaviour, our conclusion must be that Cartesian–developmental is the best of the four conditions.

There is one caveat. The developmental representation is not compatible with the standard GE representation. That is, an individual can not easily be converted from one representation to another, and a single evolutionary run can

not use both (except in the trivial sense of multiple non-migrating populations). The impact of the new representation on issues such as population dynamics and coverage of the entire phenotype space has not been tested. We emphasise that the only claims made here about this representation's properties relate to interpolation. We have not reported the results of any evolutionary runs, either interactive or non-interactive, using interpolating operators.

Nevertheless, for short, interactive EC runs, our conclusion is that the Cartesian interpolation in the developmental representation is a useful tool and deserves further study. Where compatibility with the standard GE representation is required, the single-point interpolation is a viable alternative. Both possibilities allow the user to view many individuals quickly and to specify the relative contributions of two parents during crossover. They therefore give the user more active control and a more interesting role in IEC, helping to alleviate the fitness evaluation bottleneck and to avoid the fatigue that arises through boring, repetitive tasks.

A. Future Work

An alternative method of interpolation, to be investigated in future work, might ignore genomes altogether to perform interpolation at the level of GE derivation trees. This would use the standard tree edit distance algorithm, which can produce an "edit-script" to add, edit, and delete nodes one by one, changing any tree into any other.

Our results demonstrate major differences in behaviour between the GE encodings. Viewing our behaviour metrics as predictors of problem difficulty (e.g. [2], [30]), it is possible to interpret our results as having implications for performance in non-interactive GE with non-interpolating crossover. This issue will be addressed in future work.

Finally, the central motivation for this work is to improve real-world IEC systems. In future work we will report the impact of different representations and interpolation methods in our GE-based 3D architectural design system [6].

REFERENCES

[1] J. A. Biles, "GenJam: A genetic algorithm for generating jazz solos," in *Proceedings of the 1994 International Computer Music Conference*. Danish Institute of Electroacoustic Music, Denmark: International Computer Music Association, 1994, pp. 131–137.

[2] F. Rothlauf, *Representations for Genetic and Evolutionary Algorithms*, 2nd ed. Physica-Verlag, 2006.

[3] F. Rothlauf and M. Oetzel, "On the locality of grammatical evolution," in *EuroGP*, ser. LNCS, P. Collet, M. Tomassini, M. Ebner, S. Gustafson, and A. Ekárt, Eds., vol. 3905. Springer, 2006, pp. 320–330.

[4] T. Jones and S. Forrest, "Fitness distance correlation as a measure of problem difficulty for genetic algorithms," in *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 184–192.

[5] J. Reddin, J. McDermott, M. O'Neill, and A. Brabazon, "Elevated pitch: Automated grammatical evolution of short compositions," in *Applications of Evolutionary Computing: EvoWorkshops 2004*, ser. LNCS, M. Giacobini *et al.*, Eds., no. 5484. Springer-Verlag, 2009, pp. 579–584.

[6] M. O'Neill, J. McDermott, J. M. Swafford, J. Byrne, E. Hemberg, E. Shotton, C. McNally, A. Brabazon, and M. Hemberg, "Evolutionary design using grammatical evolution and shape grammars: Designing a shelter," *International Journal of Design Engineering*, vol. 3, no. 1, 2010.

[7] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[8] P. D. Surry and N. J. Radcliffe, "Real representations," in *Foundations of Genetic Algorithms 4*, R. K. Belew and M. D. Vose, Eds. San Francisco, CA: Morgan Kaufman, 1997, pp. 343–363.

[9] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artificial Intelligence Review*, vol. 12, no. 4, pp. 265–319, August 1998.

[10] A. Moraglio, "Towards a geometric unification of evolutionary algorithms," Ph.D. dissertation, University of Essex, November 2007. [Online]. Available: http://eden.dei.uc.pt/ moraglio/

[11] A. H. Wright, "Genetic algorithms for real parameter optimization," in *Foundations of Genetic Algorithms 1*, G. J. E. Rawlins, Ed. Los Altos, CA: Morgan Kaufmann, 1991, pp. 205–218. [Online]. Available: http://citeseer.ist.psu.edu/196281.html

[12] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341 – 359, 1997.

[13] D. Van Nort, M. M. Wanderley, and P. Depalle, "On the choice of mappings based on geometric properties," in *Proceedings of the 2004 Conference on New Interfaces for Musical Expression*. National University of Singapore, 2004, pp. 87–91.

[14] A. Hunt, M. M. Waverley, and M. Paradis, "The importance of parameter mapping in electronic instrument design," *Journal of New Music Research*, vol. 32, no. 4, pp. 429–440, 2003.

[15] R. Bencina, "The Metasurface: applying natural neighbour interpolation to two-to-many mapping," in *Proceedings of New Interfaces for Musical Expression 2005*, 2005, pp. 101–104.

[16] C. Goudeseune, "Interpolated mappings for musical instruments," *Organised Sound*, vol. 7, no. 2, pp. 85–96, 2003.

[17] S. S. Stevens, "On the psychophysical law," *Psychological Review*, vol. 64, no. 3, pp. 153–181, 1957.

[18] J. McDermott, "Tree representations and the usability of interpolating controllers," in *Proceedings of i-HCI 2009, the Irish Conference on Human-Computer Interaction*, Trinity College Dublin, September 2009.

[19] H. Takagi, "Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation," *Proc. of the IEEE*, vol. 89, no. 9, pp. 1275–1296, 2001.

[20] C. G. Johnson, "Exploring sound-space with interactive genetic algorithms," *Leonardo*, vol. 36, no. 1, pp. 51–54, 2003.

[21] R. Dawkins, *The Blind Watchmaker*. Harlow, England: Longman Scientific and Technical, 1986.

[22] D. A. Hart, "Toward greater artistic control for interactive evolution of images and animation," in *Applications of Evolutionary Computing*, ser. LNCS, M. Giacobini, Ed., vol. 4448. Springer, 2007, pp. 527–536.

[23] J. McDermott, M. O'Neill, and N. J. L. Griffith, "EC control of sound synthesis," *Evolutionary Computation Journal*, vol. 18, no. 2, 2010.

[24] J. McDermott, N. J. L. Griffith, and M. O'Neill, "Evolutionary GUIs for sound synthesis," in *Applications of Evolutionary Computing*, ser. LNCS, M. Giacobini, Ed., vol. 4448. Springer, 2007, pp. 547–556.

[25] ——, "New-generation methods in an interpolating EC synthesizer interface," in *Applications of Evolutionary Computing*, ser. LNCS, vol. 4974. Napoli, Italy: Springer-Verlag, March 2008.

[26] M. O'Neill and C. Ryan, *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers, 2003.

[27] I. Dempsey, M. O'Neill, and A. Brabazon, *Foundations in Grammatical Evolution for Dynamic Environments*. Springer Verlag, 2009.

[28] M. O'Neill, C. Ryan, M. Keijzer, and M. Cattolico, "Crossover in grammatical evolution," *Genetic Programming and Evolvable Machines*, vol. 4, no. 1, pp. 67–93, 2003.

[29] C. Ryan, A. Azad, A. Sheahan, and M. O'Neill, "No coercion and no prohibition, a position independent encoding scheme for evolutionary algorithms–the chorus system," in *EuroGP*. Springer, 2002, pp. 53–63.

[30] K. Krawiec and P. Lichocki, "Approximating geometric crossover in semantic space," in *GECCO '09: Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2009, pp. 987–994.

[31] A. Fontana, "Cell Tracking: Genesis and Epigenesis in an Artificial Organism," in *Advances in Artificial Life, ECAL 2007*, ser. LNCS, vol. 4648. Springer, 2007, p. 163.