

# A Review of Theoretical and Experimental Results on Schemata in Genetic Programming

Riccardo Poli and W. B. Langdon

School of Computer Science  
The University of Birmingham  
Birmingham B15 2TT, UK  
E-mail: {R.Poli,W.B.Langdon}@cs.bham.ac.uk

**Abstract.** Schemata and the schema theorem, although criticised, are often used to explain why genetic algorithms (GAs) work. A considerable research effort has been produced recently to extend the GA schema theory to Genetic Programming (GP). In this paper we review the main results available to date in the theory of schemata for GP and some recent experimental work on schemata.

## 1 Introduction

Genetic Programming (GP) has been applied successfully to a large number of difficult problems [8, 7, 2]. However a relatively small number of theoretical results are available to try and explain why and how it works.

Since John Holland's seminal work in the mid seventies and his well known schema theorem (see [6] and [3]), schemata are often used to explain why GAs work (although their usefulness has been recently criticised, e.g. in [4] and [1]). In particular it is believed that GAs solve problems by hierarchically composing relatively fit, short schemata to form complete solutions (building block hypothesis). So the obvious way of creating a theory for GP is to define a concept of schema for parse trees and to extend Holland's schema theorem.

One of the difficulties in obtaining theoretical results using the idea of schema is that the definition of schema for GP is much less straightforward than for GAs and a few alternative definitions have been proposed in the literature. All of them define schemata as composed of one or multiple trees or fragments of trees. In some definitions [8, 10, 11, 18, 19] schema components are *non-rooted* and, therefore, a schema can be present multiple times within the same program. This, together with the variability of the size and shape of the programs matching the same schema, leads to some complications in the computation of the schema-disruption probabilities necessary to formulate schema theorems for GP. In more recent definitions [14, 17] schemata are represented by *rooted* trees or tree fragments. These definitions make schema theorem calculations easier. In particular, in our work [14] we have proposed a new simpler definition of schema for GP which is much closer to the original concept of schema in GAs. This concept of schema suggested a simpler form of crossover for GP, called one-point crossover,

which allowed us to derive a simple and natural schema theorem for GP with one-point crossover. We will critically review the main results obtained to date in the theory of schemata for GP in Sect. 3 after briefly recalling and reformulating Holland’s schema theory for binary GAs in Sect. 2.

Although theoretical results on schemata are very important, it is well known that schema theorems only model the disruptive effects of crossover and represent only short-term predictions. At the time of writing this paper, only one empirical study on GP schemata had been carried out [12]. This analysed the effects of standard crossover, one-point crossover and selection only on the propagation of schemata in real, although small, populations. We describe the main results of this study in Sect. 4 and we draw some conclusions in Sect. 5.

## 2 Background

As highlighted in [16], a schema is a subspace of the space of possible solutions. Usually schemata are written in some language using a concise notation rather than as ordinary sets, which would require listing all the solutions they contain: an infeasible task even for relatively small search spaces.

In the context of binary representations, a schema (or similarity template) is a string of symbols taken from the alphabet  $\{0,1,\#\}$ . The character  $\#$  is interpreted as a “don’t care” symbol, so that a schema can represent several bit strings. For example the schema  $\#10\#1$  represents four strings: 01001, 01011, 11001 and 11011.<sup>1</sup> The number of non- $\#$  symbols is called the *order*  $\mathcal{O}(H)$  of a schema  $H$ . The distance between the furthest two non- $\#$  symbols is called the *defining length*  $\mathcal{L}(H)$  of the schema. Holland obtained a result (the schema theorem) which predicts how the number of strings in a population matching a schema varies from one generation to the next [6]. The theorem<sup>2</sup> is as follows:

$$E[m(H, t + 1)] \geq m(H, t) \cdot \underbrace{\frac{f(H, t)}{\bar{f}(t)}}_{\text{Selection}} \cdot \underbrace{(1 - p_m)^{\mathcal{O}(H)}}_{\text{Mutation}} \cdot \underbrace{\left[ 1 - p_c \frac{\mathcal{L}(H)}{N - 1} \left( 1 - \frac{m(H, t) f(H, t)}{M \bar{f}(t)} \right) \right]}_{\text{Crossover}} \quad (1)$$

where  $m(H, t)$  is the number of strings matching the schema  $H$  at generation  $t$ ,  $f(H, t)$  is the mean fitness of the strings in the population matching  $H$ ,  $\bar{f}(t)$  is the mean fitness of the strings in the population,  $p_m$  is the probability of mutation per bit,  $p_c$  is the probability of one-point crossover per individual,  $N$

<sup>1</sup> As pointed out by one of the reviewers, both schemata and strings could be seen as logic formulae, and the problem of matching a string  $h$  against a schema  $H$  could be formalised using standard substitution techniques used in Logic Programming.

<sup>2</sup> This is a slightly different version of Holland’s original theorem (see [3, 21]).

is the number of bits in the strings,  $M$  is the number of strings in the population, and  $E[m(H, t + 1)]$  is the expected value of the number of strings matching the schema  $H$  at generation  $t + 1$ . The three horizontal curly brackets beneath the equation indicate which operators are responsible for each term. The bracket above the equation represents the probability of disruption of the schema  $H$  at generation  $t$  due to crossover,  $P_d(H, t)$ . Such a probability depends on the frequency of the schema in the mating pool but also on the intrinsic *fragility* of the schema  $\frac{\mathcal{L}(H)}{N-1}$ .

The representation for schemata based on don't care symbols and binary digits just introduced is not the only one possible. To make it easier for the reader to understand some of the differences between the GP schema definitions introduced in the following sections, in the following we propose (and then modify) a different representation for GA schemata.

A GA schema is fully determined by the defining bits (the 0's and 1's) it contains and by their position. Instead of representing a schema  $H$  as a string of characters, one could equivalently represent it with a list of pairs  $H = \{(c_1, i_1), \dots, (c_n, i_n)\}$ , where the first element  $c_j$  of each pair would represent a group of contiguous defining bits which we call a *component* of the schema, while the second element  $i_j$  would represent the position of  $c_j$ . For example the schema  $\#10\#1$  would have two components and could be represented as  $\{(10, 2), (1, 5)\}$ .

Although this formal list-of-pairs-based representation is not explicitly used by GA users and researchers, the informal idea of schemata as groups of components is used quite often. For example, when we say that a schema has been disrupted by crossover, we mean that one or more of its components have not been transmitted to the offspring. We do not usually mean that the offspring sample a subspace different from the one represented by the schema (although this is, of course, an entirely equivalent interpretation). Likewise, when we explain the building block hypothesis [3, 6] by saying that GAs work by hierarchically composing relatively fit, short schemata to form complete solutions, we mean that crossover mixes and concatenates the components (defining bits) of low order schemata to form higher order ones. We do not usually mean that crossover is allocating more samples to higher-order schemata representing the intersection of good, lower-order ones.

The schema representation just introduced also seems to be often implicitly assumed when interpreting the GA schema theorem. Obviously, there is no distinction between the number of strings (in the population) sampling the subspace represented by a given schema and the number of times the schema components are jointly present in the strings of the population. Nonetheless, the GA schema theorem is often interpreted as describing the variations of the number of instances of schema components. For example, if  $H = \#10\#1$  the theorem could be interpreted as a lower bound for the expected number of components 10 at position 2 and 1 at position 5 within the population at the next generation.

To reduce the gap between the GA schema theory and some of the GP schema theories introduced in the next section, it is important to understand

the effects of altering our component-centred schema representation by omitting the positional information from the pairs. Syntactically this would mean that we represent a schema as a list of groups of bits (components)  $H = \{c_1, \dots, c_n\}$ , like for example  $\{10,1\}$ . Semantically we could interpret a schema as the set containing all the strings which include as substrings the components of the schema (in whatever position). For example, the schema  $\{11,00\}$  would represent all the strings which include both the substring 11 and the substring 00.

An important consequence of removing positional information from the schema definition is that it is possible for a string to include multiple copies of the components of a schema. For example, the components of the schema  $\{10,01\}$  are jointly present four times in the string 10101. This means that with a position-less schema definition there is a distinction between the number of strings sampling the subspace represented by a given schema and the number of times the schema components are jointly present in the strings of the population.

### 3 GP Schema Theories

While the previous argument might look academic for fixed-length GAs, it is actually very relevant for GP. In some GP schema definitions information about the positions of the schema components is omitted. This has led some researchers to concentrate their analysis on the propagation of such components (often seen as potential building blocks for GP) in the population rather than on the way the number of programs sampling a given schema change over time.

#### 3.1 Theories on Position-less Schema Component Propagation

Given the popularity of Holland’s work, Koza [8, pages 116–119] made the first attempt to explain why GP works producing an informal argument showing that Holland’s schema theorem would work for GP as well. The argument was based on the idea of defining a schema as the subspace of all trees which contain a pre-defined set of subtrees. According to Koza’s definition a schema  $H$  is represented as a set of S-expressions. For example the schema  $H = \{(+ 1 x), (* x y)\}$  represents all programs including at least one occurrence of the expression  $(+ 1 x)$  and at least one occurrence of  $(* x y)$ . This definition of schema was probably suggested by the fact that Koza’s GP crossover moves around subtrees. Koza’s definition gives only the defining components of a schema not their position, so the same schema can be instantiated (matched) in different ways, and therefore multiple times, in the same program. For example, the schema  $H = \{x\}$  can be instantiated in two ways in the program  $(+ x x)$ .

Koza’s work on schemata was later formalised and refined by O’Reilly [10, 11] who derived a schema theorem for GP in the presence of fitness-proportionate selection and crossover. The theorem was based on the idea of defining a schema as an unordered collection (a multiset) of subtrees and tree fragments. Tree fragments are trees with at least one leaf that is a “don’t care” symbol ( $\#$ ) which can be matched by any subtree (including subtrees with only one node).

For example the schema  $H = \{(+ \# x), (* x y), (* x y)\}$  represents all the programs including at least one occurrence of the tree fragment  $(+ \# x)$  and at least *two* occurrences of  $(* x y)$ .<sup>3</sup> The tree fragment  $(+ \# x)$  is present in all programs which include a  $+$  the second argument of which is  $x$ . Like Koza's definition O'Reilly's schema definition gives only the defining components of a schema not their position. So, again the same schema can be instantiated in different ways, and therefore multiple times, in the same program.

O'Reilly's definition of schema allowed her to define the concept of order and defining length for GP schemata. In her definition the order of a schema is the number of non- $\#$  nodes in the expressions or fragments contained in the schema. The defining length is the number of links included in the expressions and tree fragments in the schema plus the links which connect them together. Unfortunately, the definition of defining length is complicated by the fact that the components of a schema can be embedded in different ways in different programs. Therefore, the defining length of a schema is not constant but depends on the way a schema is instantiated inside the programs sampling it. As also the total number of links in each tree is variable, this implies that the probability of disruption  $P_d(H, h, t)$  of a schema  $H$  due to crossover depends on the shape, size and composition of the tree  $h$  matching the schema. The schema theorem derived by O'Reilly,

$$E[i(H, t + 1)] \geq i(H, t) \cdot \frac{f(H, t)}{f(t)} \cdot \left( 1 - p_c \cdot \overbrace{\max_{h \in \text{Pop}(t)} P_d(H, h, t)}^{P_d(H, t)} \right), \quad (2)$$

overcame this problem by considering the maximum of such a probability,  $P_d(H, t) = \max_{h \in \text{Pop}(t)} P_d(H, h, t)$  which may lead to severely underestimating the number of occurrences of the given schema in the next generation ( $p_c$  is the probability of crossover). It is important to note  $i(H, t)$  is the number of *instances* of the schema  $H$  at generation  $t$  and  $f(H, t)$  is the mean fitness of the instances of  $H$ . This is computed as the weighted sum of the fitnesses of the programs matching  $H$ , using as weights the ratios between the number of instances of  $H$  each program contains and the total number of instances of  $H$  in the population. The theorem describes the way in which the components of the representation of a schema propagate from one generation to the next, rather than the way the number of programs sampling a given schema change during time. O'Reilly discussed the usefulness of her result and argued that the intrinsic variability of  $P_d(H, t)$  from generation to generation is one of the major reasons why no hypothesis can be made on the real propagation and use of building blocks (short, low-order relatively fit schemata) in GP. O'Reilly's schema theorem did not include the effects of mutation.

In the framework of his GP system based on context free grammars (CFG-GP) Whigham produced a concept of schema for context-free grammars and the related schema theorem [18, 20, 19]. In CFG-GP programs are the result of applying a set of rewrite rules taken from a pre-defined grammar to a start-

<sup>3</sup> We use here the standard notation for multisets, which is slightly different from the one used in O'Reilly's work.

ing symbol  $S$ . The process of creation of a program can be represented with a derivation tree whose internal nodes are rewrite rules and whose terminals are the functions and terminals used in the program. In CFG-GP the individuals in the population are derivation trees and the search proceeds using crossover and mutation operators specialised so as to always produce valid derivation trees.

Whigham defines a schema as a partial derivation tree rooted in some non-terminal node, i.e. as a collection of rewrite rules organised into a single derivation tree. Given that the terminals of a schema can be both terminal and non-terminal symbols of a grammar and that the root of a schema can be a symbol different from the starting symbol  $S$ , a schema represents all the programs that can be obtained by completing the schema (i.e. by adding other rules to its leaves until only terminal symbols are present) and all the programs represented by schemata which contain it as a component. When the root node of a schema is not  $S$ , the schema can occur multiple times in the derivation tree of the same program. This is the result of the absence of positional information in the schema definition. For example, the schema  $H = (A \xrightarrow{\pm} FAA)$  can be instantiated in two ways in the derivation tree,  $(S (A (F +) (A (F -) (A (T 2)) (A (T x))) (A (T x))))$ , of the program  $(+ (- 2 x) x)$ .

Whigham's definition of schema leads to simple equations for the probabilities of disruption of schemata under crossover,  $P_{d_c}(H, h, t)$ , and mutation,  $P_{d_m}(H, h, t)$ . Unfortunately as with O'Reilly's, these probabilities vary with the size of the tree  $h$  matching the schema. In order to produce a schema theorem for CFG-GP Whigham used the average disruption probabilities of the instances of a schema under crossover and mutation,  $\bar{P}_{d_c}(H, t)$  and  $\bar{P}_{d_m}(H, t)$ , and the average fitness  $f(H, t)$  of such instances. The theorem is as follows:

$$E[i(H, t + 1)] \geq i(H, t) \frac{f(H, t)}{f(t)} \cdot \{[1 - p_m \bar{P}_{d_m}(H, t)][1 - p_c \bar{P}_{d_c}(H, t)]\}, \quad (3)$$

where  $p_c$  and  $p_m$  are the probabilities of applying crossover and mutation. By changing the grammar used in CFG-GP this theorem can be shown to be applicable both to GAs with fixed length binary strings and to standard GP, of which CFG-GP is a generalisation (see the GP grammar given in [19, page 130]). Like in O'Reilly's case, this theorem describes the way in which the components of the representation of a schema propagate from one generation to the next, rather than the way the number of programs sampling a given schema change over time. The GP schema theorem obtained by Whigham is different from the one obtained by O'Reilly as the concept of schema used by the two authors is different. Whigham's schemata represent derivation-tree fragments which always represent single subexpressions, while O'Reilly's schemata can represent multiple subexpressions.

### 3.2 Theories on Positioned Schema Propagation

Recently two new schema theories (developed at the same time and independently) have been proposed [17, 14] in which schemata are represented using

rooted trees or tree fragments. The rootedness of these schema representations is very important as they reintroduce in the schema definition the positional information lacking in previous definitions of schema for GP. As a consequence a schema can be instantiated at most once within a program and studying the propagation of the components of the schema in the population is equivalent to analysing the way the number of programs sampling the schema change over time.

Rosca [17] has proposed a definition of schema, called *rooted tree-schema*, in which a schema is a rooted contiguous tree fragment. For example, the rooted tree-schema  $H = (+ \# x)$  represents all the programs whose root node is a + the second argument of which is x. Rosca derived the following schema theorem for GP with standard crossover (when crossover points are selected with a uniform probability):

$$E[m(H, t + 1)] \geq m(H, t) \frac{f(H, t)}{f(t)}. \quad (4)$$

$$\left[ 1 - (p_m + p_c) \underbrace{\sum_{h \in H \cap Pop(t)} \frac{\overbrace{\mathcal{O}(H)}^{P_d(H, h, t)}}{N(h)} \frac{f(h)}{\sum_{h \in H \cap Pop(t)} f(h)}}_{P_d(H, t)} \right],$$

where  $N(h)$  is the size of a program  $h$  matching the schema  $H$ ,  $f(h)$  is its fitness, and the order of a schema  $\mathcal{O}(H)$  is the number of defining symbols it contains.<sup>4</sup> Rosca did not give a definition of defining length for a schema. Rosca's schema theorem involves the evaluation of the weighted sum of the fragilities  $\frac{\mathcal{O}(H)}{N(h)}$  of the instances of a schema within the population, using as weights the ratios between the fitness of the instances of  $H$  and the sum of the fitness of such instances.

In [14] we have proposed a simpler definition of schema for GP which has allowed us to derive a new schema theorem for GP with a new form of crossover, which we call one-point crossover. In the definitions of GP schema mentioned above, in general schemata divide the space of programs into subspaces containing programs of different sizes and shapes. Our definition of schema partitions the program space into subspaces of programs of fixed size and shape.

We define a *schema* as a tree composed of functions from the set  $\mathcal{F} \cup \{=\}$  and terminals from the set  $\mathcal{T} \cup \{=\}$ , where  $\mathcal{F}$  and  $\mathcal{T}$  are the function set and the terminal set used in a GP run. The symbol = is a "don't care" symbol which stands for a *single* terminal or function. In line with the original definition of schema for GAs, a schema  $H$  represents programs having the same shape as  $H$  and the same labels for the non- $=$  nodes. For example, if  $\mathcal{F} = \{+, -\}$  and  $\mathcal{T} = \{x, y\}$  the schema  $H = (+ (- = y) =)$  would represent the four programs  $(+ (- x y) x)$ ,  $(+ (- x y) y)$ ,  $(+ (- y y) x)$  and  $(+ (- y y) y)$ . Our definition of schema is in some sense lower-level than those adopted by others, as a smaller number of trees can be represented by schemata with the same number of "don't care"

<sup>4</sup> We have rewritten the theorem in a form which is slightly different from the original one to highlight some of its features.

symbols and it is possible to represent one of their schemata by using a collection of ours.

The number of non- $=$  symbols is called the *order*  $\mathcal{O}(H)$  of a schema  $H$ , while the total number of nodes in the schema is called the *length*  $N(H)$  of the schema. The number of links in the minimum subtree including all the non- $=$  symbols within a schema  $H$  is called the *defining length*  $\mathcal{L}(H)$  of the schema. For example the schema  $(+ (- = =) x)$  has order 3 and defining length 2. These definitions are independent of the shape and size of the programs in the actual population.

In order to derive a GP schema theorem for our schemata we used very simple forms of mutation and crossover, namely point mutation and one-point crossover. *Point mutation* is the substitution of a function in the tree with another function with the same arity or the substitution of a terminal with another terminal. *One-point crossover* works by selecting a common crossover point in the parent programs and then swapping the corresponding subtrees like standard crossover. In order to account for the possible structural diversity of the two parents, one-point crossover starts analysing the two trees from the root nodes and considering for the selection of the crossover point only the parts of the two trees which have the same topology (i.e. the same arity in the nodes encountered traversing the trees from the root node) [14].

The new schema theorem provides the following lower bound for the expected number of individuals sampling a schema  $H$  at generation  $t + 1$  for GP with one-point crossover and point mutation:

$$E[m(H, t + 1)] \geq m(H, t) \frac{f(H, t)}{\bar{f}(t)} (1 - p_m)^{\mathcal{O}(H)} \cdot \left\{ 1 - p_c \left[ p_{\text{diff}}(t) \left( 1 - \frac{m(G(H), t) f(G(H), t)}{M \bar{f}(t)} \right) + \frac{\mathcal{L}(H)}{(N(H) - 1)} \frac{m(G(H), t) f(G(H), t) - m(H, t) f(H, t)}{M \bar{f}(t)} \right] \right\} \quad (5)$$

where  $p_m$  is the mutation probability (per node),  $G(H)$  is the zero-th order schema with the same structure of  $H$  where all the defining nodes in  $H$  have been replaced with “don’t care” symbols,  $M$  is the number of individuals in the population,  $p_{\text{diff}}(t)$  is the conditional probability that  $H$  is disrupted by crossover when the second parent has a different shape (i.e. does not sample  $G(H)$ ), and the other symbols have the same meaning as in Equation 2 (see [14] for the proof). The zero-order schemata  $G(H)$ ’s represent different groups of programs all with the same shape and size. For this reason we call them *hyperspaces* of programs. We denote non-zero-order schemata with the term *hyperplanes*.

Equation 5 is more complicated than the corresponding version for GAs [3, 6, 21] because in GP the trees undergoing optimisation have variable size and shape. This is accounted for by the presence of the terms  $m(G(H), t)$  and  $f(G(H), t)$ , which summarise the characteristics of the programs belonging to the same hyperspace in which  $H$  is a hyperplane.

In [14] we analysed Equation 5 in detail and discussed the likely interactions between hyperspaces and hyperplanes and the expected variations of the probability  $p_{\text{diff}}(t)$  during a run. In particular we conjectured that, given the

diversity in the initial population, in general  $p_{\text{diff}}$  would be quite close to 1 and the probability of schema disruption would be quite high at the beginning of a run. Schema disruption and creation by crossover would then heavily counteract the effects of selection, except for schemata with above average fitness and short defining-length whose shape  $G(H)$  is also of above average fitness and is shared by an above average number of programs. We also conjectured that, in the absence of mutation, after a while the population would start converging, like a traditional GA, and the diversity of shapes and sizes would decrease. During this second phase, the competition between schemata belonging to the same hyperspace would become more and more important and the GP schema theorem would asymptotically tend to the standard GA schema theorem. Therefore, during this phase *all* schemata with above average fitness and short defining-length would tend to have a low disruption probability. These conjectures were later corroborated by the experimental study summarised in Sect. 4.

In [11] serious doubts have been cast on the correctness of the building block hypothesis for standard GP. However, our analysis of Equation 5 led us to suggest that it cannot be ruled out in GP with one-point crossover. Nonetheless, as suggested in [5] it is possible that, in order to fully understand what are the building blocks which GP uses to create complete solutions, it will be necessary to study also the propagation of *phenotypical schemata*, i.e. of sets of structurally different but functionally equivalent programs. Unfortunately, it seems very difficult at this stage to imagine how such schemata could be represented and detected.

## 4 Experimental Studies on GP Schemata

Some researchers have studied the changes over time of quantities which are relevant to schema theorems but which do not require the direct manipulation of all the schemata in a population. For example, Rosca has studied the average of the ratio  $f(h)/N(h)$  (an important term in his schema theorem) for all programs in the population and for the best program in the population. However, we will not describe studies of this kind here as they do not explicitly represent and study the schemata in a population but only the programs it contains.

Whatever definition of schema is embraced, it is quite easy to see that the number of different schemata or schema instances contained in a single individual or, worse, in a population of individuals, can be extremely large. For example, using our definition of schema, the number of different schemata contained in a single program  $h$  of length  $N(h)$  is  $2^{N(h)}$ , which is large even for short programs. This, together with the complexity of detecting all the instances of a schema with some schema definitions, is probably the reason why only one experimental study on the properties of GP schemata had been done at the time of writing this paper [12]. We describe the main results of this work in the remainder of this section.

In our experiments we have only been able to study (our) schemata in real GP populations by limiting the depth of the programs being evolved to three or

four levels and the arity of the functions used by GP to two arguments. Given these restrictions we decided to use in our experiments the XOR problem, which can be solved with these settings. In the experiments we used the function set  $\mathcal{F}=\{\text{AND, OR, NAND, NOR}\}$  and the terminal set  $\mathcal{T}=\{x_1, x_2\}$ . With these choices and a maximum depth of 2 (the root node is at level 0) we were able to follow the evolution of all the schemata in a population of 50 individuals for 50 generations. (The reader is invited to refer to [12] for a description of other experiments in which we studied subsets of the schemata present in populations of programs with maximum depth 3 and for other details on the experimental setting.)

In a first series of experiments we studied the effects of selection on the propagation of single schemata and on the number of hyperplanes and hyperspace sampled by the programs in the population in ten runs. In each run all schemata in the population were recorded together with: the average fitness of the population in each generation, the average fitness of each schema in each generation, the number of programs sampling the schema in each generation, and the order, length and defining length of the schema. Using these data we were also able to study schema diversity in the population. Diversity was assessed in terms of number of different programs, of schemata and of hyperspaces in each generation.

According to the schema theorem, when selection only is acting, schemata with below average fitness should tend to disappear from the population while above-average schemata should be sampled more frequently. This effect was indeed observed in our runs at the level of both hyperplanes and hyperspaces. However, the results shown in Fig. 1(a) suggest that selection is less effective on hyperspaces than on programs. Indeed, the rate at which hyperspaces disappear is lower than for programs. This can be explained by considering that the average deviation of the fitness of high-order schemata (e.g. programs) from the population fitness is in general bigger than for low-order schemata. Therefore, (positive or negative) selection will be stronger on average for high-order schemata and programs.

Not surprisingly, in none of our runs we observed the exponential schema growth/decay often claimed to happen in GAs [3]. On the contrary, the growth or decay of schemata diverged quite significantly from being monotonic. This was mainly due to the fact that we used small populations of 50 individuals and that genetic drift interfered with the natural growth or decay in the number of schema instances. This effect becomes prominent when the selective pressure decreases, i.e. when nearly all the programs in the population have the same fitness.

In a second set of experiments we considered the effects of normal crossover on the propagation of schemata. As shown in Fig. 1(b), the situation is quite different from the previous case. In an initial phase programs were still subject to a relatively strong selection pressure. As in the selection-only case, hyperspaces were not as strongly affected by it. As soon as the population fitness started saturating the effects of crossover became prevalent. These effects are the constant creation of new individuals and the destruction of old ones. They

prevented the population from converging and maintained a high number of different schemata in the population. Towards the end of the runs, there were several times more schemata than in the selection-only case. Standard crossover prevented the competition between hyperspaces from ending as hyperspaces were constantly repopulated.

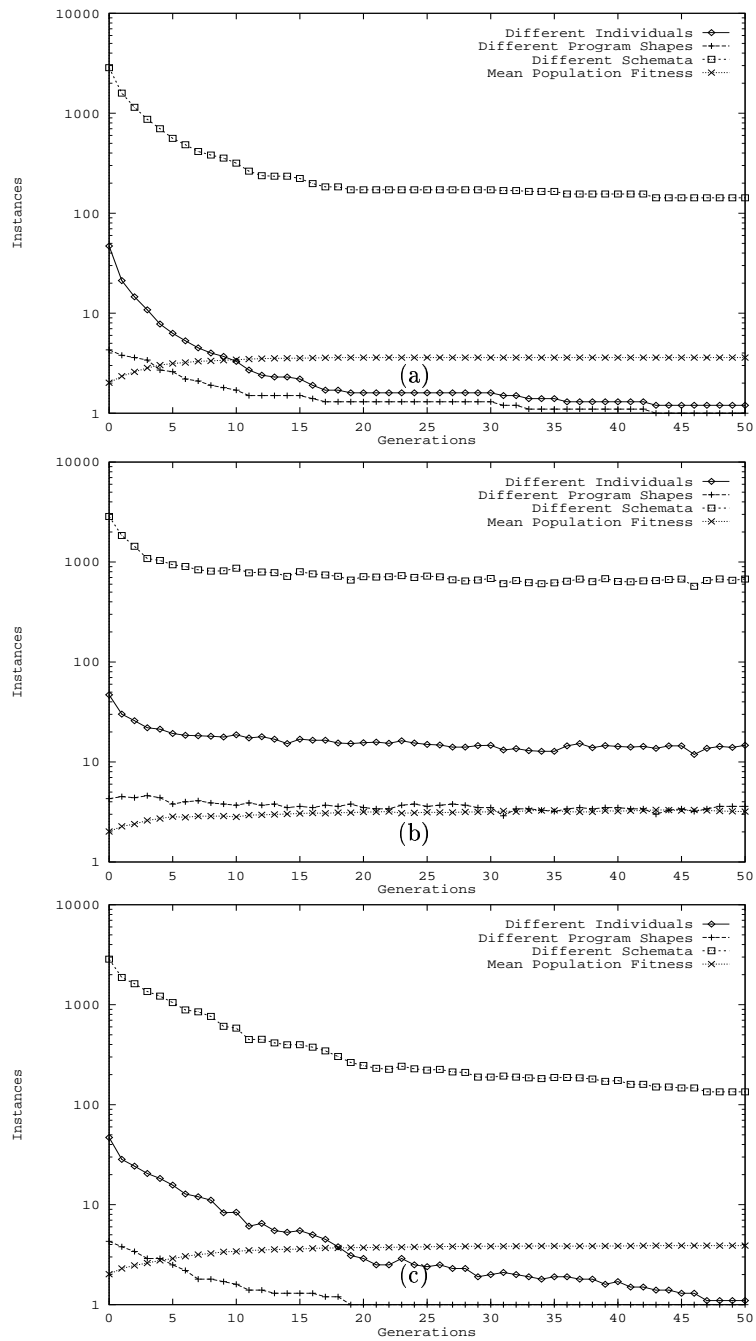
In a final set of experiments we studied the effects of one-point crossover. One-point crossover behaved quite differently from standard crossover. This behaviour is illustrated by the plots of the population fitness and schema diversity averaged over 10 different runs (Fig. 1(c)). These plots show that the average number of different individuals per hyperspace tended to vary very slowly during the runs. The competition between low-order schemata ended relatively more quickly than the competition between higher order ones. When the competition between hyperspaces was over (not later than generation 19), the one between the programs sampling them was still active. In some cases this took quite a few more generations to finish, but it always did it and, like in the case of selection only, the population always converged. A comparison of the plots in Figures 1(c) and 1(a) reveals that, on average, the rate at which the diversity of high-order schemata changes is reduced by one-point crossover. This is due to the continuous creation of new schemata.

Our experimental study revealed that when the size of the populations used is small genetic drift can be a major component in schema propagation and extinction. This certainly happened in our experiments as soon as the selective pressure decreased, both when selection only was present and when one-point crossover was used. Genetic drift could not become a major driving force in runs with standard crossover, because of the relatively large schema disruption/creation probability associated with it.

Although in the first few generations of runs with standard crossover selection was a primary driving force, the disruption and innovation power of standard crossover remained very strong throughout the entire runs and the population never converged. This contributed to maintaining a certain selective pressure even in late generations, as the average population fitness never reached its maximum.

The relatively large random oscillations of the total number of schemata observed in all runs with standard crossover seem to suggest that, however large, the probability of schema disruption  $P_d(H, t)$  in Equation 2 is not constant but may vary significantly from one generation to the next and should be considered a stochastic variable as suggested in [11].

One-point crossover allowed the convergence of the population. So, on average we must assume that its schema disruption probability is smaller than for standard crossover. However, if we compare the total number of schemata in the first few generations of the runs with one-point crossover and the runs with standard crossover, we can see that *initially one-point crossover is as disruptive as standard crossover*. This corroborates our conjecture that schema disruption would be quite high at the beginning of runs with one-point crossover.



**Fig. 1.** Average population fitness and schema diversity in 10 GP runs of the XOR problem with: (a) selection only, (b) normal crossover, (c) one-point crossover.

The experiments also corroborate our conjecture that with one-point crossover, after this first highly-disruptive phase in which different hyperspaces compete, the competition between schemata belonging to the same hyperspace becomes the most important effect. In the experiments after generation 19 only one hyperspace was present in all runs. This means that the different programs in such hyperspace had all the same size and shape and that GP with one-point crossover was actually working like a GA with a non-standard crossover. This corroborates our conjecture that the GP schema theorem asymptotically tends to the GA schema theorem.

This behaviour suggests that, while the building block hypothesis seems really in trouble when standard crossover is used, it is as relevant to GP with one-point crossover as it is for traditional GAs.

The convergence properties of GP with one-point crossover suggested that, like in traditional GAs, mutation could be a very important operator to recover lost genetic material. A recent study [13] in which one-point crossover has been compared with standard crossover and an even more constrained form of crossover (called *strict one-point crossover*) has demonstrated that this hypothesis is correct and that point mutation is vital to solve efficiently Boolean classification problems with one-point crossover. (Interestingly, one-point crossover also improves significantly the performance of standard crossover.)

## 5 Conclusions

In this paper we have reviewed the main results available to date on the theory of schemata for GP, including our own work [14] which is based on a simple definition of the concept of schema for GP which is very close to the original concept of schema in GAs. We have also summarised the first experimental study on GP schemata [12].

At this point the reader might be asking himself or herself: “Is there a *right* schema definition and a *right* schema theory for GP?” In our opinion the answer to this question should be “no”, for the following reasons.

We take the viewpoint that a schema is a subspace of the space of possible solutions that schemata are mathematical tools to describe which areas of the search space are sampled by a population. For schemata to be useful in explaining how GP searches, their definition must make the effects of selection, crossover and mutation comprehensible and relatively easy to calculate. A possible problem with some of the earlier definitions of schema for GP is that they make the effects on schemata of the genetic operators used in GP perhaps too difficult to evaluate. However, we believe that any schema definition which, with a particular set of genetic operators, is amenable to mathematical analysis and can lead to some theoretical insight on the inner operation of GP is equally valid and useful.

So, we believe that good different schema theories should not be seen as competitors. When the same operators are used, different schema definitions and schema theorems are simply different views of the same phenomenon. If

integrated, they can lead to a deeper understanding. Even more, when it is possible to represent a schema of the particular kind using a set of schemata of another kind it is possible to export some results of the one schema theory to the schemata of another one and *vice versa*. For example, if we consider one of Rosca's schemata,  $H_R$ , it is easy to prove that it can be represented using a collection of our schemata, say  $H_{PL_1}, \dots, H_{PL_M}$ . Then it is possible to export the results of the Rosca's schema theory to our schemata and our schema theory to Rosca's schemata considering that  $E[m(H_R, t)] = \sum_i E[m(H_{PL_i}, t)]$ .

More theoretical work will be needed to explore the relations between different theories and the ways they can integrate and cross-fertilise each other. We also believe that more work needs to be done to evaluate the effects of schema creation. This has been one of the main themes of our recent research work, carried out in collaboration with Una-May O'Reilly (MIT AI Lab). This has led to the formulation of new general schema theorems which estimate the mean, the variance and the signal-to-noise ratio of the number of individuals sampling a schema in the presence of schema creation, as well as to the estimation of the short-term probability of extinction of newly created schemata [15].

At the same time, we also think that it will be necessary to produce more empirical studies on the creation and disruption of schemata and on the building block hypothesis. In very recent work we have performed the second study of this kind using our schemata to investigate the possible deceptiveness of the Ant problem [9].

## Acknowledgements

The authors wish to thank the members of the EEBIC (Evolutionary and Emergent Behaviour Intelligence and Computation) group for useful discussions and comments, and the reviewers of this paper for their very constructive suggestions. This research is partially supported by a grant under the British Council-MURST/CRUI agreement and by a contract with the Defence Research Agency in Malvern.

## References

1. L. Altenberg. The Schema Theorem and Price's Theorem. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 23–49, Estes Park, Colorado, USA, 31 July–2 Aug. 1994 1995. Morgan Kaufmann.
2. P. J. Angeline and K. E. Kinneer, Jr., editors. *Advances in Genetic Programming 2*. MIT Press, Cambridge, MA, USA, 1996.
3. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, Massachusetts, 1989.
4. J. J. Grefenstette. Deception considered harmful. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, San Mateo, CA, 1993. Morgan Kaufman.
5. T. Haynes. Phenotypical building blocks for genetic programming. In E. Goodman, editor, *Genetic Algorithms: Proceedings of the Seventh International Conference*, Michigan State University, East Lansing, MI, USA, 19-23 July 1997. Morgan Kaufmann.

6. J. Holland. *Adaptation in Natural and Artificial Systems*. MIT Press, Cambridge, Massachusetts, second edition, 1992.
7. K. E. Kinneer, Jr., editor. *Advances in Genetic Programming*. MIT Press, Cambridge, MA, 1994.
8. J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.
9. W. B. Langdon and R. Poli. Why ants are hard. Technical Report CSRP-98-4, University of Birmingham, School of Computer Science, January 1998.
10. U.-M. O'Reilly. *An Analysis of Genetic Programming*. PhD thesis, Carleton University, Ottawa-Carleton Institute for Computer Science, Ottawa, Ontario, Canada, 22 Sept. 1995.
11. U.-M. O'Reilly and F. Oppacher. The troubling aspects of a building block hypothesis for genetic programming. In L. D. Whitley and M. D. Vose, editors, *Foundations of Genetic Algorithms 3*, pages 73–88, Estes Park, Colorado, USA, 31 July–2 Aug. 1994 1995. Morgan Kaufmann.
12. R. Poli and W. B. Langdon. An experimental analysis of schema creation, propagation and disruption in genetic programming. In E. Goodman, editor, *Genetic Algorithms: Proceedings of the Seventh International Conference*, Michigan State University, East Lansing, MI, USA, 19–23 July 1997. Morgan Kaufmann.
13. R. Poli and W. B. Langdon. Genetic programming with one-point crossover. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing*. Springer-Verlag London, 23–27 June 1997.
14. R. Poli and W. B. Langdon. A new schema theory for genetic programming with one-point crossover and point mutation. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 278–285, Stanford University, CA, USA, 13–16 July 1997. Morgan Kaufmann.
15. R. Poli, W. B. Langdon, and U.-M. O'Reilly. Short term extinction probability of newly created schemata, and schema variance and signal-to-noise-ratio theorems in the presence of schema creation. Technical Report CSRP-98-6, University of Birmingham, School of Computer Science, January 1998.
16. N. J. Radcliffe. Forma analysis and random respectful recombination. In *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, 1991.
17. J. P. Rosca. Analysis of complexity drift in genetic programming. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, and R. L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 286–294, Stanford University, CA, USA, 13–16 July 1997. Morgan Kaufmann.
18. P. A. Whigham. A schema theorem for context-free grammars. In *1995 IEEE Conference on Evolutionary Computation*, volume 1, pages 178–181, Perth, Australia, 29 Nov. - 1 Dec. 1995. IEEE Press.
19. P. A. Whigham. *Grammatical Bias for Evolutionary Learning*. PhD thesis, School of Computer Science, University College, University of New South Wales, Australian Defence Force Academy, October 1996.
20. P. A. Whigham. Search bias, language bias, and genetic programming. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 230–237, Stanford University, CA, USA, 28–31 July 1996. MIT Press.
21. D. Whitley. A genetic algorithm tutorial. Technical Report CS-93-103, Department of Computer Science, Colorado State University, August 1993.