Proceedings of the 2001 IEEE Congress on Evolutionary Computation Seoul, Korea • May 27-30, 2001

Exact GP Schema Theory for Headless Chicken Crossover and Subtree Mutation

Riccardo Poli
School of Computer Science
The University of Birmingham
Birmingham, B15 2TT, UK
R.Poli@cs.bham.ac.uk

Nicholas F. McPhee
Division of Science and Mathematics
University of Minnesota, Morris
Morris, MN, USA
mcphee@mrs.umn.edu

Abstract- Here a new general GP schema theory for headless chicken crossover and subtree mutation is presented. The theory gives an exact formulation for the expected number of instances of a schema at the next generation either in terms of microscopic quantities or in terms of macroscopic ones. The paper gives examples which show how the theory can be specialised to specific operators.

1 Introduction

The theory of schemata in genetic programming has had a difficult childhood. After some excellent early efforts leading to different worst-case-scenario schema theorems [1, 2, 3, 4, 5, 6, 7], exact schema theories have become available only very recently [8, 9, 10, 11]. These new theories give exact formulations (rather than lower bounds) for the expected number of instances of a schema at the next generation, and are applicable to GP with various types of subtree crossover. No exact schema theory for subtree mutation (or any other type of GP mutation) has ever been proposed.

This paper fills this theoretical gap and presents a new general GP schema theory for subtree mutation and headless chicken crossover. *Headless chicken crossover* is a variant of crossover, introduced for GAs in [12] and for GP in [13], in which one of the parents is randomly generated while the other is selected from the population. Our theory gives an exact formulation for the expected number of instances of a schema at the next generation for these operators.

The paper is organised as follows. Firstly, we provide a review of earlier relevant work on schemata in Section 2. Most of the concepts introduced in that section are described extensively, since they are necessary to understand the rest of the paper. Then, we derive general schema theorems for GP with headless chicken crossover and subtree mutation in Sections 3 and 4, respectively. In Section 5 we give examples that show how the theory can be specialised to obtain schema theorems for specific operators and primitive sets. Some conclusions are drawn in Section 6.

2 Background

Schemata are sets of points of the search space sharing some syntactic features. For example, in the context of GAs operating on binary strings, syntactically a schema is a string of symbols from the alphabet {0,1,*}, where the character * is interpreted as a “don't care” symbol. Typically schema theorems are descriptions of how the number of members of the population belonging to a schema vary over time. If $\alpha(H,t)$ is the probability that a newly created individual samples the schema H, which we term the *total transmission probability* of H, an exact schema theorem is simply [14]

$$E[m(H,t+1)] = M\alpha(H,t), \tag{1}$$

where M is the population size, $m(H,t+1)$ is the number of individuals in H at generation $t+1$ and $E[\cdot]$ is the expectation operator. Holland's [15] and other worst-case-scenario schema theories normally provide a lower bound for $\alpha(H,t)$ or, equivalently, for $E[m(H,t+1)]$.

One of the difficulties in obtaining theoretical results on GP using the idea of schema is that its definition is much less straightforward than for GAs. Various definitions have been proposed in the literature [1, 2, 3, 4, 5, 7], but for brevity here we will describe only the definition of fixed-size-and-shape schema introduced in [5, 6] which is what is used in this paper and in other recent work [8, 9, 10, 11, 16].

2.1 GP Schemata

Syntactically a GP *fixed-size-and-shape schema* (or just schema for simplicity) is 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 and terminal sets used in a GP run [5, 6]. The primitive $=$ is a “don't care” symbol which stands for a *single* terminal or function. 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 `(+ x (= y =))` represents the four programs `(+ x (+ y x))`, `(+ x (+ y y))`, `(+ x (* y x))` and `(+ x (* y y))`.

Using this definition, in [5, 6] a worst-case-scenario schema theorem was derived for GP with point mutation and one-point crossover. This result was improved in [8, 9] where an exact schema theory for GP with one-point crossover (but no mutation) was derived.

2.2 Cartesian Node Reference Systems

In [11] a general schema theory for GP with subtree-swapping crossover was presented which was based on the notion of variable arity hyperschema and on the concepts of

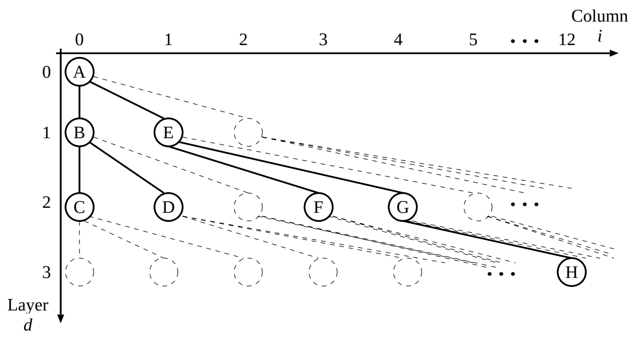

Figure 1: Tree-independent Cartesian node reference system. Nodes and links of the maximal tree are drawn with dashed lines. Only four layers are shown.

Cartesian node reference systems and probability distributions over them. These are also the basis for the new theory presented in this paper. They are described in this and the following sections.

A Cartesian node reference system can be defined by first considering the largest possible tree that can be created with nodes of arity $a_{\max}$. This maximal tree would include 1 node of arity $a_{\max}$ at depth 0, $a_{\max}$ nodes of arity $a_{\max}$ at depth 1, $a_{\max}^2$ nodes of arity $a_{\max}$ at depth 2, etc.. Then one can organise the nodes in the tree into layers of increasing depth and assign an index to each node in a layer. We can then define a coordinate system based on the layer number d and the index i. This reference system can also be used to locate the nodes of non-maximal trees by using a subset of the nodes and links in the maximal tree. So, for example, if $a_{max} = 3$, the nodes in the expression `(A (B C D) (E F (G H)))` would be placed in a node reference system as indicated in Figure 1 where, for example, F is indexed by (2,3). It should be noted that in the this kind of reference system it is possible to transform pairs of coordinates into integers by counting the nodes in breadth-first order (and vice versa). So, nodes A, B, C, D, E, F and G would have indices 0, 1, 4, 5, 2, 7, 8 and 25, respectively. We will use this property to simplify the notation in some of the following sections.

2.3 Functions over Node Reference Systems

Given a node reference system it is possible to define functions over it. An example of such functions is the *name function* $N(d, i, h)$ which returns the node at position (d, i) in a particular tree h; if h does not have a node at position (d, i), a default value of $\emptyset$ is returned. For example, for the tree in Figure 1, $N(0, 0, h) = \mathtt{A}$ and $N(2, 0, h) = \mathtt{C}$, while $N(2, 2, h) = \emptyset$.

Another example of a node function is the *arity function* $A(d, i, h)$ which returns the arity of the node at coordinates (d, i) in h. The function returns -1 if (d, i) is not in h. For example, for the tree in Figure 1, $A(0, 0, h) = 2$, $A(1, 0, h) = 2$, $A(2, 1, h) = 0$ and $A(2, 4, h) = 1$.

Finally, it should be noted that these functions can be applied to schemata too. A useful function in handling schemata is the *defining node function*, $\Delta(d, i, H)$, which returns 1 if the node at coordinates (d, i) is a defining node, 0 if it is a = symbol, -1 if it is not in H.

2.4 Modelling the Selection of Crossover and Mutation Points

Most genetic operators used in GP require the selection of a node where to perform a transformation (e.g. the insertion of a random subtree, or of a subtree taken from another parent). In most cases the selection of the node is performed with a stochastic process of some sort. It is possible to model this process by assuming that a probability distribution is defined over the nodes of each individual. If we use the node-reference system introduced in the previous section, this can be expressed as the function:

$$p(d, i|h) = \Pr\left\{\begin{matrix}\text{A node at coordinates } (d,i) \text{ is}\\ \text{selected in program } h\end{matrix}\right\}, \qquad (2)$$

where we assume that $p(d, i|h)$ is zero for all the undefined coordinates (d, i) in h.[1] For example, if we select nodes with uniform probability from the tree in Figure 1, then $p(d, i|h) = \frac{1}{8}$ if (d, i) exists in h, and $p(d, i|h) = 0$ otherwise.

There are many possible uses for probability distributions over node reference systems. In the following section we will concentrate on their use in modelling crossover operators. Later it will become clear how these can be used to model headless chicken crossover and subtree mutation.

2.5 Modelling Subtree-swapping Crossover

In general in order to model crossover operators we need to use the following conditional probability distribution:

$$p(d_1, i_1, d_2, i_2|h_1, h_2) = \Pr\left\{\begin{matrix}\text{A node at coordinates } (d_1,i_1) \text{ is selected in parent } h_1 \text{ and}\\ \text{a node at coordinates } (d_2,i_2) \text{ is selected in parent } h_2\end{matrix}\right\},$$

with the convention $p(d_1, i_1, d_2, i_2|h_1, h_2) = 0$ if $N(d_1, i_1, h_1) = \emptyset$ or $N(d_2, i_2, h_2) = \emptyset$, where $N(d, i, h)$ is the name function defined in Section 2.3. If the selection of the crossover points is performed independently in the two parents, then

$$p(d_1, i_1, d_2, i_2|h_1, h_2) = p(d_1, i_1|h_1) \cdot p(d_2, i_2|h_2),$$

where $p(d, i|h)$ is defined in Equation 2. We will call crossover operators for which this relation is true *separable*.

Standard crossover with uniform selection of the crossover points is a separable operator with

$$p(d, i|h) = \frac{\delta(N(d, i, h) \neq \emptyset)}{S(h)},$$

where $S(h)$ is the number of nodes in h and $\delta(x)$ is a function which returns 1 if x is true, 0 otherwise.

[1] For this probability distribution we use the notation $p(d, i|h)$ rather than $p(d, i, h)$ since this can be seen as the conditional probability of selecting node (d, i) if (or given that) the program being considered is h.

Also standard crossover with a 90%-function/10%-any-node selection policy is separable. However, it should be noted that some crossover operators, like for example one-point crossover and strongly typed GP crossover, are not separable. Models for these and other crossover operators are described in [11].

Thanks to these probabilistic models of crossover, it is possible to develop a general schema theory for GP as described in the following sections. This theory is the basis for the schema theory for headless chicken crossover and subtree mutation presented later in this paper.

2.6 Exact GP Schema Theorems for Subtree-swapping Crossovers

For simplicity in this and the following sections we will use a single index to identify nodes unless otherwise stated. We can do this because, as indicated previously, there is a one-to-one mapping between pairs of coordinates and natural numbers.

In order to state a schema theorem valid for subtree-swapping crossovers, we need to introduce new form of schema: the *Variable Arity Hyperschema*, or *VA hyperschema* for brevity. A VA hyperschema is a rooted tree composed of internal nodes from the set $\mathcal{F} \cup \{=, \#\}$ and leaves from $\mathcal{T} \cup \{=, \#\}$ [11]. The operator = is a "don't care" symbols which stands for exactly one node, the terminal # stands for any valid subtree, while the function # stands for exactly one function of arity not smaller than the number of subtrees connected to it. For example, the VA hyperschema `(# x (+ = #))` represents all the programs with the following characteristics: a) the root node is any function in the function set with arity 2 or higher, b) the first argument of the root node is the variable x, c) the second argument of the root node is +, d) the first argument of the + is any terminal, e) the second argument of the + is any valid subtree. If the root node is matched by a function of arity greater than 2, the third, fourth, etc. arguments of such a function are left unspecified, i.e. they can be any valid subtree.

We can use VA hyperschemata and the notion of probability distributions over node reference systems to obtain the following general result [11]:

Theorem 1. *The total transmission probability for a fixed-size-and-shape GP schema H under a subtree-swapping crossover operator and no mutation is*

$$\alpha(H,t) = (1-p_{xo})p(H,t) + p_{xo} \sum_{h_1 \in \mathcal{P}} \sum_{h_2 \in \mathcal{P}} p(h_1,t)p(h_2,t) \cdot \sum_{i \in H} \sum_{j} p(i,j|h_1,h_2)\delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,j)) \qquad (3)$$

where: p_{xo} is the crossover probability; $p(H,t)$ is the selection probability of the schema H;[2] *$\mathcal{P}$ is the set of unique individuals in the population; $p(h_1,t)$ and $p(h_2,t)$ are the selection probabilities of parents h_1 and h_2, respectively; the third summation is over all the crossover points (nodes) in the schema H; the fourth summation is over all the crossover points in the node reference system; $p(i,j|h_1,h_2)$ is the probability of selecting crossover point i in parent h_1 and crossover point j in parent h_2; $L(H,i,j)$ is the VA hyperschema obtained by rooting at coordinate j in an empty reference system the subschema of H below crossover point i, then by labelling all the nodes on the path between node j and the root node with # function nodes, and labelling the arguments of those nodes which are to the left of such a path with # terminal nodes; $U(H,i)$ is the hyperschema obtained by replacing the subtree below crossover point i with a # node.*

The functions $L(H,i,j)$ and $U(H,i)$ are designed to return *exactly* the hyperschemata needed to create H using crossover. $U(H,i)$ is the hyperschema representing all the trees that match the *upper* portion of H (i.e., the parts of H not below crossover point i). $L(H,i,j)$ is the hyperschema representing all the trees that match the *lower* portion of H, but where the matching portion is at some arbitrary position j. The combined effect of these definitions is that if one crosses over *any* individual matching $U(H,i)$ at point i with *any* individual matching $L(H,i,j)$ at point j, the resulting offspring is always an instance of H. Further, this is the only way to construct an instance of H.[3]

To better understand how $U(H,i)$ and $L(H,i,j)$ are constructed, let us consider an example; throughout this example we will use the 2–D coordinate system, so positions i and j will in fact be ordered pairs. Let us take our schema to be $H =$`(* = (+ x =))`, and our coordinates to be $i = (1,0)$ and $j = (1,1)$. Then Figure 2 illustrates how we construct $U(H,i)$ (the top two coordinate grids) and $L(H,i,j)$ (the lower three coordinate grids). The top coordinate grid shows the initial schema H, with the crossover point i marked, and the lower part of the schema shaded. The next grid then shows $U(H,i)$, which is obtained by simply replacing the shaded subtree (in this case just the terminal '=') with a '#'. The upper of the three coordinate grids for $L(H,i,j)$ again illustrates the initial schema H with the crossover point i marked. Now, however, the shaded area (the part of H below i) needs to be translated to position j as shown in the second coordinate grid. The third coordinate grid then shows the insertion of '#' symbols (a) along the path from the root to j (in this case just $(0,0)$) and (b) in all argument positions to the left of '#' symbols (in this case just $(1,0)$). This placement of '#' symbols, combined with the fact that we allow '#'s to represent functions of varying arity, ensures that $L(H,i,j) =$`(# # =)` represents *all* the possible trees whose subtrees at position j match the lower part of H (i.e., the part below position i).

Let us denote with $G(H)$ the schema obtained by replac-

[2]In fitness proportionate selection $p(H,t) = \frac{m(H,t)f(H,t)}{M\bar{f}(t)}$, where $m(H,t)$ is the number of programs matching the schema H at generation t, $f(H,t)$ is the mean fitness of such programs, and $\bar{f}(t)$ is the mean fitness of the programs in the population.

[3]$L(H,i,j)$ and $U(H,i)$ are discussed in more detail in [11].

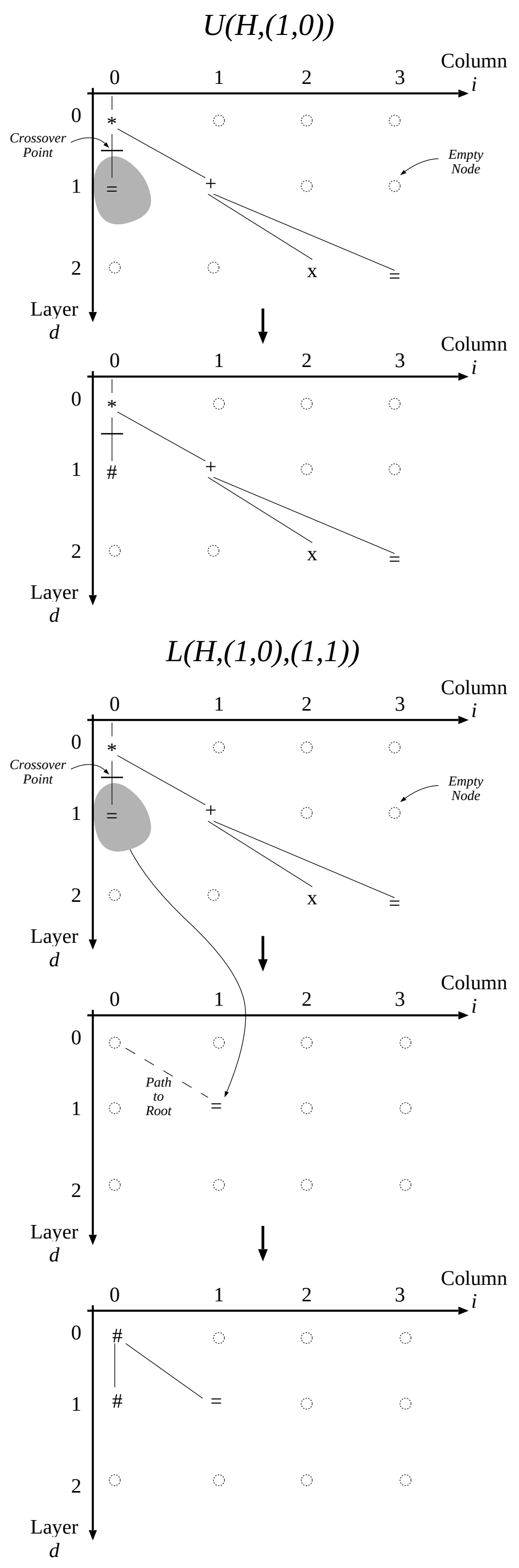

Figure 2: Phases in the constructions of the VA hyperschema building blocks $U(H,(1,0))$ and $L(H,(0,1),(1,1))$ of the schema H =`(* = (+ x =))` within a node coordinate system with $a_{max} = 2$.

ing all the defining nodes in the schema H with = nodes. We will refer to $G(H)$ as the *shape* of H.

If the choice of the crossover points in any two parents, h_1 and h_2, depends only on their shapes, $G(h_1)$ and $G(h_2)$, i.e. if $p(i,j|h_1,h_2) = p(i,j|G(h_1),G(h_2))$, we term the operators *node-invariant*. For node-invariant subtree-swapping crossovers Equation 3 can be transformed into the following exact macroscopic description of schema propagation:

Theorem 2. *The total transmission probability for a fixed-size-and-shape GP schema H under a node-invariant subtree-swapping crossover operator and no mutation is*

$$\begin{aligned}\alpha(H,t) = (1-p_{xo})p(H,t)+ \\ p_{xo}\sum_{k,l}\sum_{i\in H}\sum_{j} p(i,j|G_k,G_l) \\ p(U(H,i)\cap G_k,t)p(L(H,i,j)\cap G_l,t),\end{aligned} \quad (4)$$

where the schemata G_1, G_2, $\cdots$ are all the possible program shapes (i.e. all the fixed-size-and-shape schemata including only = symbols) and the other symbols have the same meaning as in Theorem 1.

The sets $U(H,i)\cap G_k$ and $L(H,i,j)\cap G_l$ either are (or can be represented by) fixed-size-and-shape schemata or are the empty set. So, the theorem indicates which pairs of schemata can contribute to the creation of instances of a schema and with which relative probability. Such schemata can be considered the building blocks for the schema.

2.7 Previous Schema Theories for Mutation

We are aware of only two schema-theory results for mutation applicable to the standard GP representation. We briefly summarise them below.

In [7] Rosca derived a worst-case-scenario schema theorem for rooted-tree schemata, which can be defined as hyperschemata without = symbols and # function nodes. In the case in which only subtree mutation and fitness proportionate selection are present the theorem is equivalent to:

$$E[m(H,t+1)] \geq Mp(H,t)\left[1 - p_m\frac{\sum_{h\in H\cap\mathcal{P}}\frac{m(h,t)f(h)}{S(h)}}{\sum_{h\in H\cap\mathcal{P}}\frac{m(h,t)f(h)}{\mathcal{O}(H)}}\right], \quad (5)$$

where p_m is the mutation probability (per individual), $S(h)$ is the size of a program h matching the schema H, $f(h)$ is its fitness, and $\mathcal{O}(H)$ is the order of a schema defined as the number of defining symbols it contains.

A second result for mutation can be obtained from the worst-case-scenario GP schema theorem for fixed-size-and-shape schemata under point mutation and one-point crossover derived in [5, 6]. In the absence of crossover, this leads to:

$$E[m(H,t+1)] \geq Mp(H,t)(1-p_m)^{\mathcal{O}(H)} \quad (6)$$

where p_m is the mutation probability (per node) and $\mathcal{O}(H)$ (the *order of* H) is the number of non-= symbols in H.

3 Schema Theory for Subtree-swapping Headless Chicken Crossover

Different forms of subtree-swapping headless chicken crossover can be defined depending on whether one returns one or two offspring and whether such offspring inherit their root nodes from the parent which has been randomly generated or the one selected from the population [13]. In this paper we will concentrate on the case in which we generate a single offspring, and the offspring inherits the root from the parent selected from the population.

The schema theory for subtree-swapping headless chicken crossover is a natural extension of the theory for subtree-swapping crossover since the only difference between the two operators is the source of the non-root-donating parent: the population through fitness proportionate selection in the latter case, a stochastic tree generation algorithm in the former case. Therefore, the theorems (and the proofs) provided in this section are also very similar to the corresponding results for subtree-swapping crossover.

Indeed, for the class of operators headless chicken crossover operators defined above we have:

Theorem 3. *The total transmission probability for a fixed-size-and-shape GP schema H under a subtree-swapping headless chicken crossover is*

$$\begin{aligned}\alpha(H,t) = (1-p_{xo})p(H,t) + \\ p_{xo}\sum_{h_1\in\mathcal{P}}\sum_{h_2\in\mathcal{S}} p(h_1,t)\pi(h_2,t)\cdot \\ \sum_{i\in H}\sum_{j} p(i,j|h_1,h_2)\delta(h_1\in U(H,i))\delta(h_2\in L(H,i,j))\end{aligned} \quad (7)$$

where: $\mathcal{S}$ is the space of all possible programs that can be built with the given terminal and function sets, $\pi(h_2,t)$ is the probability that the random tree generation algorithm used will produce program h_2 at generation t, and the other symbols have the same meaning as in Theorem 1.

Proof. Let $p(h_1,h_2,i,j,t)$ be the probability that, at generation t, the selection/crossover/randomisation process will choose parent h_1 taken from the population, parent h_2 randomly generated and crossover points i and j in h_1 and h_2, respectively. Then, let us consider the function

$$g(h_1,h_2,i,j,H) = \delta(h_1\in U(H,i))\delta(h_2\in L(H,i,j)).$$

Given two parent programs, h_1 and h_2, and a schema of interest H, this function returns the value 1 if crossing over h_1 at position i and h_2 at position j yields an offspring in H. It returns 0 otherwise. This function can be considered as a measurement function (see [17]) that we want to apply to the probability distribution of parents and crossover points at time t, $p(h_1,h_2,i,j,t)$.

If h_1, h_2, i, and j are stochastic variables with joint probability distribution $p(h_1,h_2,i,j,t)$, the function $g(h_1,h_2,i,j,H)$ can be used to define a stochastic variable $\gamma = g(h_1,h_2,i,j,H)$. The expected value of γ is:

$$E[\gamma] = \sum_{h_1}\sum_{h_2}\sum_{i}\sum_{j} g(h_1,h_2,i,j,H)p(h_1,h_2,i,j,t). \quad (8)$$

We can write

$$p(h_1,h_2,i,j,t) = p(i,j|h_1,h_2)p(h_1,t)\pi(h_2,t), \quad (9)$$

where $p(i,j|h_1,h_2)$ is the conditional probability that crossover points i and j will be selected when the parents are h_1 and h_2, $p(h_1,t)$ is the selection probability for the root-donating parent and $\pi(h_2,t)$ is the probability that the random tree generation algorithm will produce program h_2 at generation t. Substituting Equation 9 into Equation 8 and noting that if crossover point i is outside the schema H, then $L(H,i,j)$ and $U(H,i)$ are empty sets, lead to

$$E[\gamma] = \quad (10)$$
$$\sum_{\substack{h_1\in\mathcal{P}\\ h_2\in\mathcal{S}}} p(h_1,t)\pi(h_2,t) \sum_{\substack{i\in H\\ j}} g(h_1,h_2,i,j,H)p(i,j|h_1,h_2).$$

Since γ is a binary stochastic variable, its expected value also represents the probability that the offspring produced by headless chicken crossover is in H. So, the contribution to $\alpha(H,t)$ due to selection followed by headless chicken crossover is $E[\gamma]$. By multiplying this by p_{xo} and adding the term $(1-p_{xo})p(H,t)$ due to selection followed by cloning one obtains the r.h.s. of Equation 7. □

This result allows one to calculate the expected proportion of individuals belonging to a schema in the next generation. This is a microscopic model since it requires to consider the properties of each member of the search space, which makes it hard to use it for computational studies. However, this model can be transformed into a macroscopic model for a very general class of headless chicken crossovers.

If we define as *node invariant* a headless chicken crossover in which $p(i,j|h_1,h_2) = p(i,j|G(h_1),G(h_2))$, then we can obtain a macroscopic version of the previous theorem by following a strategy similar to the one used in the proof of Theorem 2, obtaining

Theorem 4. *The total transmission probability for a fixed-size-and-shape GP schema H under a node-invariant subtree-swapping headless chicken crossover is*

$$\begin{aligned}\alpha(H,t) = (1-p_{xo})p(H,t) + \\ p_{xo}\sum_{k,l}\sum_{i\in H}\sum_{j} p(i,j|G_k,G_l) \\ p(U(H,i)\cap G_k,t)\pi(L(H,i,j)\cap G_l,t),\end{aligned} \quad (11)$$

where $\pi(L(H,i,j)\cap G_l,t)$ is the probability of randomly generating programs in $L(H,i,j)\cap G_l$ and the other symbols have the same meaning as in Theorem 2.

Proof. We prove the theorem by transforming Equation 7 into Equation 11. The schemata $G_1, G_2, \cdots$ represent disjoint sets of programs. Their union represents the whole search space.

So, $\sum_k \delta(h_1 \in G_k) = 1$. Likewise, $\sum_l \delta(h_2 \in G_l) = 1$. If we multiply the terms within the quadruple summation in Equation 7 by the l.h.s. of these equations and reorder the terms, we obtain:

$$\begin{aligned}
&\sum_{k,l} \sum_{\substack{h_1 \in \mathcal{P} \\ h_2 \in \mathcal{S}}} p(h_1,t)\pi(h_2,t) \sum_{\substack{i \in H \\ j}} p(i,j|h_1,h_2) \\
&\quad \delta(h_1 \in U(H,i))\delta(h_1 \in G_k)\delta(h_2 \in L(H,i,j))\delta(h_2 \in G_l) \\
&\quad = \sum_{k,l} \sum_{\substack{h_1 \in \mathcal{P} \cap G_k \\ h_2 \in \mathcal{S} \cap G_l}} p(h_1,t)\pi(h_2,t) \sum_{\substack{i \in H \\ j}} p(i,j|h_1,h_2) \\
&\qquad \delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,j)).
\end{aligned}$$

For node-invariant headless chicken crossover operators $p(i,j|h_1,h_2) = p(i,j|G(h_1),G(h_2))$, which substituted into the previous equation gives:

$$\begin{aligned}
&\sum_{k,l} \sum_{\substack{h_1 \in \mathcal{P} \cap G_k \\ h_2 \in \mathcal{S} \cap G_l}} p(h_1,t)\pi(h_2,t) \sum_{\substack{i \in H \\ j}} p(i,j|G(h_1),G(h_2)) \\
&\qquad \delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,j)) \\
&\quad = \sum_{k,l} \sum_{\substack{h_1 \in \mathcal{P} \cap G_k \\ h_2 \in \mathcal{S} \cap G_l}} p(h_1,t)\pi(h_2,t) \sum_{\substack{i \in H \\ j}} p(i,j|G_k,G_l) \\
&\qquad \delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,j)) \\
&\quad = \sum_{k,l}\sum_{i \in H}\sum_{j} p(i,j|G_k,G_l) \\
&\qquad \sum_{h_1 \in \mathcal{P} \cap G_k} p(h_1,t)\delta(h_1 \in U(H,i)) \\
&\qquad \sum_{h_2 \in \mathcal{S} \cap G_l} \pi(h_2,t)\delta(h_2 \in L(H,i,j)).
\end{aligned}$$

Since $\sum_{h_1 \in \mathcal{P} \cap G_k} p(h_1,t)\delta(h_1 \in U(H,i)) = p(U(H,i) \cap G_k,t)$ and $\sum_{h_2 \in \mathcal{S} \cap G_l} \pi(h_2,t)\delta(h_2 \in L(H,i,j)) = \pi(L(H,i,j) \cap G_l,t)$, this completes the proof. □

This and the previous theorems are quite similar to the corresponding theorems for crossover. However, there is one important difference. Once the stochastic tree generation algorithms is known, the quantities $\pi(H,t)$ are numeric constants. So, the schema theorems for headless chicken crossover are linear in the schema selection probabilities, while those for crossover are quadratic.

Theorem 4 indicates which schemata can contribute to the creation of instances of a schema and with which relative probability.

4 Schema Theory for Subtree Mutation

Once the theory for headless chicken crossover is available it is very easy to modify it to become a theory for subtree mutation. It is sufficient to constrain the choice of the crossover point in the random parent to always be the root node. This can be modelled by setting:

$$p(i,j|h_1,h_2) = p(i|h_1)\delta(j=0), \tag{12}$$

where $p(i|h_1)$ is the probability of selecting mutation point i in the root donating parent h_1. As a consequence, the result in Theorem 3 simplifies considerably, leading directly to the following

Corollary 5. *The total transmission probability for a fixed-size-and-shape GP schema H under subtree mutation is*

$$\begin{aligned}
\alpha(H,t) &= (1-p_m)p(H,t)+ \\
&\quad p_m \sum_{h_1 \in \mathcal{P}} \sum_{h_2 \in \mathcal{S}} p(h_1,t)\pi(h_2,t) \cdot \qquad (13) \\
&\quad \sum_{i \in H} p(i|h_1)\delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,0))
\end{aligned}$$

where p_m is the probability of mutation (per individual) and all the other symbols have the same meaning as in Theorem 3.

If the choice of the mutation point in the parent program, h, depends only on its shape, $G(h)$, i.e. $p(i|h) = p(i|G(h))$, we term the mutation operator *node-invariant*. For node-invariant mutation operators it is possible to specialise the results in Theorem 4 obtaining

Corollary 6. *The total transmission probability for a fixed-size-and-shape GP schema H under node-invariant subtree mutation is*

$$\begin{aligned}
\alpha(H,t) &= (1-p_m)p(H,t)+ \qquad (14) \\
&\quad p_m \sum_{k}\sum_{i \in H} p(i|G_k)p(U(H,i) \cap G_k,t)\pi(L(H,i,0),t),
\end{aligned}$$

where all the symbols have the same meaning as in Theorem 4.

Proof. For a node invariant mutation operator, the quantity $p(i,j|G_k,G_l)$ in Equation 11 becomes $p(i|G_k)\delta(j=0)$. So, only terms where $j=0$ remain.

The VA hyperschema $L(H,i,0)$ has no # symbols since it is simply a subtree of H. So, there exists only one shape G_l such that $\pi(L(H,i,0) \cap G_l) \neq 0$. Let us call it $G_{\hat{l}}$. So, only the terms in Equation 11 where $l = \hat{l}$ remain. The proof is completed by noting that $L(H,i,0) \cap G_{\hat{l}} = L(H,i,0)$. □

So, also mutation is a linear operator.

5 Specialisations and Example

In order to use the theory presented in the previous sections it is necessary to define the quantities $\pi(h,t)$ and $\pi(L(H,i,j) \cap G_l,t)$. All other quantities are defined once one chooses a particular crossover-/mutation-point selection algorithm and a particular selection algorithm. It should be noted that $L(H,i,j) \cap G_l$ is always either the empty set or a set which can be represented by fixed-size-and-shape schema, so we will need to be able to express $\pi(H,t)$ for a generic schema H.

In the following subsections we will provide expressions for $\pi(h,t)$ and $\pi(H,t)$ for two very widely used random-tree generation algorithms: the "full" method and the "grow" method [1, 18]. Starting from the root node, both methods use the strategy of creating trees by selecting random nodes recursively along each branch until either a terminal is chosen, or a maximum depth D is reached; only terminals are then chosen at depth D. The two methods differ in that the "full" method only chooses from $\mathcal{F}$ until the depth limit is reached, guaranteeing that each branch is "full" out to depth D, whereas "grow" chooses from $\mathcal{C} = \mathcal{F} \cup \mathcal{T}$, which makes it possible for some branches to have length less than D.

5.1 Probability Distributions for the "Full" Method

Let us start by recursively defining a function $a(d, i, h)$ over a node reference system which returns the probability that the subtree rooted at position (d, i) in h is created when using the "full" method. This is given by:

$$a(d,i,h) = \frac{\delta(A(d,i,h)=0)\delta(d=D)}{|\mathcal{T}|} + \frac{\delta(A(d,i,h)>0)\delta(d<D)}{|\mathcal{F}|} \prod_{n=0}^{A(d,i,h)-1} a(d+1, i\cdot a_{max}+n, h) \tag{15}$$

By modifying appropriately the expression for $a(d, i, h)$ we can generalise it so as to return the probability that the subtree rooted at position (d, i) created when using the "full" method belongs to the subschema of H rooted at the same position, obtaining:

$$\begin{aligned} a(d,i,H) = {} & \delta(A(d,i,H)=0)\delta(d=D)\left(\frac{\Delta(d,i,H)}{|\mathcal{T}|} + 1 - \Delta(d,i,H)\right) + \\ & \delta(A(d,i,H)>0)\delta(d<D)\cdot \\ & \left(\frac{\Delta(d,i,H)}{|\mathcal{F}|} + (1-\Delta(d,i,H))\frac{|\mathcal{C}_{A(d,i,H)}|}{|\mathcal{F}|}\right)\cdot \\ & \prod_{n=0}^{A(d,i,H)-1} a(d+1, i\cdot a_{max}+n, H), \end{aligned} \tag{16}$$

where $\mathcal{C}_k$ is the subset of $\mathcal{C}$ including the functions/terminals of arity k. So, $\cup_{k\geq 0}\mathcal{C}_k = \mathcal{C}$, $\cup_{k\geq 1}\mathcal{C}_k = \mathcal{F}$ and $\mathcal{C}_0 = \mathcal{T}$.

Then, clearly for the "full" method we can define $\pi(H, t) = a(0, 0, H)$ and $\pi(h, t) = a(0, 0, h)$ (which, incidentally, are independent from t).

5.2 Probability Distributions for the "Grow" Method

We proceed in a similar way for the "grow" method. We define a function $b(d, i, h)$ over a node reference system which returns the probability that the subtree of h rooted at position (d, i) be created when using the "grow" method. Then, we generalise the expression for $b(d, i, h)$ so as to return the probability that the subtree rooted at position (d, i) created when using the "grow" method belongs to the subschema of H rooted at the same position, obtaining:

$$\begin{aligned} b(d,i,H) = {} & \delta(A(d,i,H)=0)\delta(d=D)\cdot \\ & \left(\frac{\Delta(d,i,H)}{|\mathcal{T}|} + 1 - \Delta(d,i,H)\right) + \\ & \delta(A(d,i,H)\neq -1)\delta(d<D)\cdot \\ & \left(\frac{\Delta(d,i,H)}{|\mathcal{C}|} + (1-\Delta(d,i,H))\frac{|\mathcal{C}_{A(d,i,H)}|}{|\mathcal{C}|}\right)\cdot \\ & \prod_{n=0}^{A(d,i,H)-1} b(d+1, i\cdot a_{max}+n, H). \end{aligned} \tag{17}$$

with the convention that $\prod_{n=0}^{-1} b(d+1, i\cdot a_{max}+n, H) = 1$.

Then, for the "grow" method we define $\pi(H, t) = b(0, 0, H)$ and $\pi(h, t) = b(0, 0, h)$.

5.3 Example

Let us write a macroscopic, exact schema theorem equation for the schema $(=\ (=\ =))$ assuming that we are using mutation based on the "grow" method with a maximum allowed depth $D = 3$, $p_m = 1$ and uniform selection of the crossover points (i.e. in Equation 14 $p(i|G_k) = 1/S(G_k)$). Let us consider the primitive set $\mathcal{C} = \{\texttt{INC}, \texttt{IGNORE}, \texttt{0}\}$ which can be decomposed into $\mathcal{C}_1 = \{\texttt{INC}, \texttt{IGNORE}\}$ and $\mathcal{C}_0 = \{\texttt{0}\}$. The semantics of these primitives (see [19, 16]) is unimportant for our example. We also assume that at generation t the population does not contain individuals with more than 3 nodes.

In these conditions, by applying Corollary 6 and simplifying we obtain:

$$\begin{aligned} \alpha((=\ (=\ =))) = {} & p(=)\pi((=\ (=\ =))) \\ & +\frac{1}{2}p((=\ =))[\pi((=\ =)) + \pi((=\ (=\ =)))] \\ & +\frac{1}{3}p((=\ (=\ =)))[\pi(=) + \pi((=\ =)) + \pi((=\ (=\ =)))] \end{aligned} \tag{18}$$

By using Equation 17 we then can calculate $\pi(H, t)$ for the schemata $=$, $(=\ =)$, and $(=\ (=\ =))$. For $H = (=\ (=\ =))$ we obtain

$$\begin{aligned} \pi(H,t) = b(0,0,H) & \\ = {} & \delta(A(0,0,H)=0)\delta(0=3) + \\ & \delta(A(0,0,H)\neq -1)\delta(0<3)\frac{2}{3}b(1,0,H) \\ = {} & \frac{2}{3}b(1,0,H) \\ = {} & \frac{2}{3}\delta(A(1,0,H)=0)\delta(1=3) + \\ & \frac{2}{3}\delta(A(1,0,H)\neq -1)\delta(1<3)\frac{2}{3}b(2,0,H) \\ = {} & \frac{4}{9}b(2,0,H) \\ = {} & \frac{4}{9}\delta(A(2,0,H)=0)\delta(2=3) + \\ & \frac{4}{9}\delta(A(2,0,H)\neq -1)\delta(2<3)\frac{1}{3} \\ = {} & \frac{4}{27} \end{aligned}$$

Likewise, we obtain $\pi(=, t) = \frac{1}{3}$ and $\pi((=\ =), t) = \frac{2}{9}$. By substituting these values in Equation 18, we obtain

$$\alpha((=\ (=\ =)), t) = \frac{4}{27}p(=, t) + \frac{5}{27}p((=\ =), t) + \frac{19}{81}p((=\ (=\ =)), t).$$

This equation shows how mutation and selection interact in the creation of instances of `(= (= =))`. It is particularly interesting to study two cases. Firstly, let us consider a needle in a haystack situation in which only programs of length 3 are fit at all, while all other programs have zero fitness. Then, clearly $p((= \ (= \ =)), t) = 1$ and $p(=, t) = p((= \ =), t) = 0$. In this circumstances one would expect the algorithm to keep sampling programs of length 3. However, the expected proportion of programs in `(= (= =))` is only about 24%. This means that the biases of mutation work against the intended biases of selection. In fact, even on a flat landscape the mutation biases impose a dynamics on the population.

6 Conclusions

Here we have presented the first ever exact schema theory for GP with headless chicken crossover and subtree mutation, thus filling an important theoretical gap. The theory is not only an exact formulation for the expected number of instances of a schema at the next generation but it is also very general. So, it is applicable to most subtree-swapping headless chicken crossovers and mutation operators used in practice. In the paper we have also provided examples which show how the theory can be specialised to specific operators.

As shown by some recent explorations reported in [10, 16], exact schema theories have many purposes. They can be used, for example, to study the exact schema evolution in infinite populations over multiple generations, to make comparisons between different operators and identify their biases, to study the evolution of size, and investigate bloat. The exact theory presented here also offers these possibilities as shown in [20], where we have used it to characterise the behaviours and biases of different mutation operators.

Acknowledgements

The authors would like to thank the members of the EEBIC (Evolutionary and Emergent Behaviour Intelligence and Computation) group at Birmingham, in particular Jon Rowe, for useful discussions and comments. The second author would like to extend special thanks to The University of Birmingham School of Computer Science for graciously hosting him during his sabbatical, and various offices and individuals at the University of Minnesota, Morris, for making that sabbatical possible.

Bibliography

[1] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[2] L. Altenberg. Emergent phenomena in genetic programming. In *Evolutionary Programming — Proceedings of the Third Annual Conference*, pages 233–241. World Scientific Publishing, 1994.

[3] U.-M. O'Reilly and F. Oppacher. The troubling aspects of a building block hypothesis for genetic programming. In *Foundations of Genetic Algorithms 3*, pages 73–88, Estes Park, Colorado, USA, 31 July–2 Aug. 1994 1995. Morgan Kaufmann.

[4] 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.

[5] R. Poli and W. B. Langdon. A new schema theory for genetic programming with one-point crossover and point mutation. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 278–285, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[6] R. Poli and W. B. Langdon. Schema theory for genetic programming with one-point crossover and point mutation. *Evolutionary Computation*, 6(3):231–252, 1998.

[7] J. P. Rosca. Analysis of complexity drift in genetic programming. In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 286–294, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[8] R. Poli. Hyperschema theory for GP with one-point crossover, building blocks, and some new results in GA theory. In *Genetic Programming, Proceedings of EuroGP 2000*. Springer-Verlag, 15-16 Apr. 2000.

[9] R. Poli. Exact schema theorem and effective fitness for GP with one-point crossover. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 469–476, Las Vegas, July 2000. Morgan Kaufmann.

[10] R. Poli and N. F. McPhee. Exact schema theorems for GP with one-point and standard crossover operating on linear structures and their application to the study of the evolution of size. In *Genetic Programming, Proceedings of EuroGP 2001*, LNCS, Milan, 18-20 Apr. 2001. Springer-Verlag.

[11] R. Poli. General schema theory for genetic programming with subtree-swapping crossover. In *Genetic Programming, Proceedings of EuroGP 2001*, LNCS, Milan, 18-20 Apr. 2001. Springer-Verlag.

[12] T. Jones. Crossover, macromutation, and population-based search. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 73–80, San Francisco, July 15–19 1995. Morgan kaufmann Publishers.

[13] P. J. Angeline. Subtree crossover: Building block engine or macromutation? In *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 9–17, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[14] R. Poli, W. B. Langdon, and U.-M. O'Reilly. Analysis of schema variance and short term extinction likelihoods. In *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 284–292, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.

[15] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, USA, 1975.

[16] N. F. McPhee and R. Poli. A schema theory analysis of the evolution of size in genetic programming with linear representations. In *Genetic Programming, Proceedings of EuroGP 2001*, LNCS, Milan, 18-20 Apr. 2001. Springer-Verlag.

[17] L. Altenberg. The Schema Theorem and Price's Theorem. In *Foundations of Genetic Algorithms 3*, pages 23–49, Estes Park, Colorado, USA, 31 July–2 Aug. 1994 1995. Morgan Kaufmann.

[18] S. Luke. Two fast tree-creation algorithms for genetic programming. *IEEE Transactions on Evolutionary Computation*, 2000.

[19] N. F. McPhee and J. D. Miller. Accurate replication in genetic programming. In *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 303–309, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[20] N. F. McPhee, R. Poli, and J. E. Rowe. A schema theory analysis of mutation size biases in genetic programming with linear representations. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC 2001 (this volume)*, Seoul, Korea, May 2001.