General Schema Theory for Genetic Programming with Subtree-Swapping Crossover

Riccardo Poli

School of Computer Science, The University of Birmingham
Birmingham, B15 2TT, UK
R.Poli@cs.bham.ac.uk, http://www.cs.bham.ac.uk/~rmp/

Abstract. In this paper a new, general and exact schema theory for genetic programming is presented. The theory includes a microscopic schema theorem applicable to crossover operators which replace a subtree in one parent with a subtree from the other parent to produce the offspring. A more macroscopic schema theorem is also provided which is valid for crossover operators in which the probability of selecting any two crossover points in the parents depends only on their size and shape. The theory is based on the notions of Cartesian node reference systems and variable-arity hyperschemata both introduced here for the first time. In the paper we provide examples which show how the theory can be specialised to specific crossover operators and how it can be used to derive an exact definition of effective fitness and a size-evolution equation for GP.

1 Introduction

Schema theories often provide information about a property of a subset of the population (a schema) at the next generation in terms of macroscopic quantities, like schema fitnesses, population fitness, or number of individuals in a schema, measured at the current generation. Some schema theories express the same macroscopic property using only microscopic quantities, such as the fitnesses of all the individuals in the population, or a mixture of microscopic and macroscopic quantities. We will refer to these as *microscopic schema theories* to differentiate them from the purely macroscopic ones. This distinction is important because the latter are simpler and easier to analyse than the former.

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 [6, 1, 11, 24, 17, 21], only very recently exact schema theories have become available [14, 12] which give exact formulations (rather than lower bounds) for the expected number of instances of a schema at the next generation. These exact theories are applicable to GP with one-point crossover [16, 17, 18]. No exact macroscopic schema theory for standard crossover (or any other GP crossover) has ever been proposed.

This paper fills this theoretical gap and presents a new exact general schema theory for genetic programming which is applicable to standard crossover as well as many other crossover operators. The theory includes two main results describing the propagation of GP schemata: a microscopic schema theorem and a macroscopic one. The microscopic version is applicable to crossover operators which replace a subtree in one parent with a subtree from the other parent to produce the offspring. So, the theorem covers

standard GP crossover [6] with and without uniform selection of the crossover points, one-point crossover [17, 18], size-fair crossover [7], strongly-typed GP crossover [9], context-preserving crossover [3] and many others. The macroscopic version is valid for a large class of crossover operators in which the probability of selecting any two crossover points in the parents depends only on their size and shape. So, for example, it holds for all the above-mentioned crossover operators except strongly typed GP crossover.

The paper is organised as follows. Firstly, we provide a review of earlier relevant work on schemata in Sec. 2. Then, in Sec. 3, we introduce the notion of node reference systems and we use it in Sec. 4 to define the concept of functions over them. Then, in Sec. 5 we show how these ideas can be used to build probabilistic models of different crossover operators. We use these to derive a general microscopic schema theorem for GP with subtree-swapping crossover in Sec. 6. We transform this into a macroscopic schema theorem in Sec. 7. In Sec. 8 we give an example that shows how the theory can be specialised to obtain schema theorems for specific types of crossover operators, and we illustrate how it can be used to obtain other general results, such as a size-evolution equation and an exact definition of effective fitness for GP. Some conclusions are drawn in Sec. 9.

2 Background

Schemata are sets of points of the search space sharing some syntactic feature. 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 we denote with $\alpha(H,t)$ 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 [19]

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

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 [5] and other worst-case-scenario schema theories normally provide a lower bound for $\alpha(H,t)$ or, equivalently, for $E[m(H,t+1)]$.

Obtaining theoretical results for GP using the idea of schema is much less straightforward than for GAs. A few alternative definitions of schema have been proposed in the literature [6, 1, 11, 24, 17, 21], but for brevity here we will describe only the definition introduced in [17, 18]. This is used in the rest of this paper. We will refer to this kind of schemata as fixed-size-and-shape schemata.

Syntactically a GP *fixed-size-and-shape schema* 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. 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))`.

In [17, 18] a worst-case-scenario schema theorem was derived for GP with point mutation and one-point crossover. One-point crossover works by selecting a common crossover point in the parent programs and then swapping the corresponding subtrees, like standard crossover. To account for the possible structural diversity of the two parents, one-point crossover analyses the two trees from the root nodes and considers for the selection of the crossover point only the parts of the two trees, called the *common region*, which have the same topology.[1] As discussed in [15], the theorem in [17, 18] is a generalisation of Holland's schema theorem to variable size structures. This result was improved in [14, 12] where an exact schema theory for GP with one-point crossover was derived which was based on the notion of hyperschema. A *GP hyperschema* is a rooted tree composed of internal nodes from $\mathcal{F} \cup \{=\}$ and leaves from $\mathcal{T} \cup \{=, \#\}$. Again, = is a "don't care" symbols which stands for exactly one node, while # stands for any valid subtree. For example, the hyperschema `(* # (= x =))` represents all the programs with the following characteristics: a) the root node is a product, b) the first argument of the root node is any valid subtree, c) the second argument of the root node is any function of arity two, d) the first argument of this function is the variable `x`, e) the second argument of the function is any valid node in the terminal set. One of the results obtained in [12] is the following macroscopic model

$$\alpha(H,t) = (1-p_{xo})p(H,t) + \qquad (2)$$
$$p_{xo}\sum_k\sum_k \frac{1}{\mathbf{NC}(G_k,G_l)} \sum_{i\in C(G_k,G_l)} p(U(H,i)\cap G_k,t)p(L(H,i)\cap G_l,t)$$

where: p_{xo} is the crossover probability; $p(H,t)$ is the selection probability of the schema H;[2] $G_1, G_2, \cdots$ are all the possible program shapes, i.e. all the possible fixed-size-and-shape schemata containing = signs only; $\mathbf{NC}(G_k,G_l)$ is the number of nodes in the common region between shape G_k and shape G_l; $C(G_k,G_l)$ is the set of indices of the crossover points in such a common region; $L(H,i)$ is the hyperschema obtained by replacing all the nodes on the path between crossover point i and the root node with = nodes, and all the subtrees connected to those nodes with # nodes; $U(H,i)$ is the hyperschema obtained by replacing the subtree below crossover point i with a # node.[3] The steps involved in the construction of $L(H,i)$ and $U(H,i)$ for the schema $H =$ `(* = (+ x =))` are illustrated in Fig. 1. If a crossover point i is in the common region between two programs but it is outside the schema H, then $L(H,i)$ and $U(H,i)$ are empty sets. The hyperschemata $L(H,i)$ and $U(H,i)$ are important because, if one crosses over at point i *any* individual in $L(H,i)$ with *any* individual in $U(H,i)$, the resulting offspring is always an instance of H.

As discussed in [15], it is possible to show that, in the absence of mutation, Eq. 2 generalises and refines not only the GP schema theorem in [17, 18] but also Holland's [5] and more modern GA schema theory [22, 23].

[1] The common region is defined formally in Sec. 4, Eq. 3.

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

[3] Eq. 2 is in a slightly different form than the result in [12]. However, the two results are equivalent since $C(G_k,G_l) = C(G_l,G_k)$ and $\mathbf{NC}(G_k,G_l) = \mathbf{NC}(G_l,G_k)$.

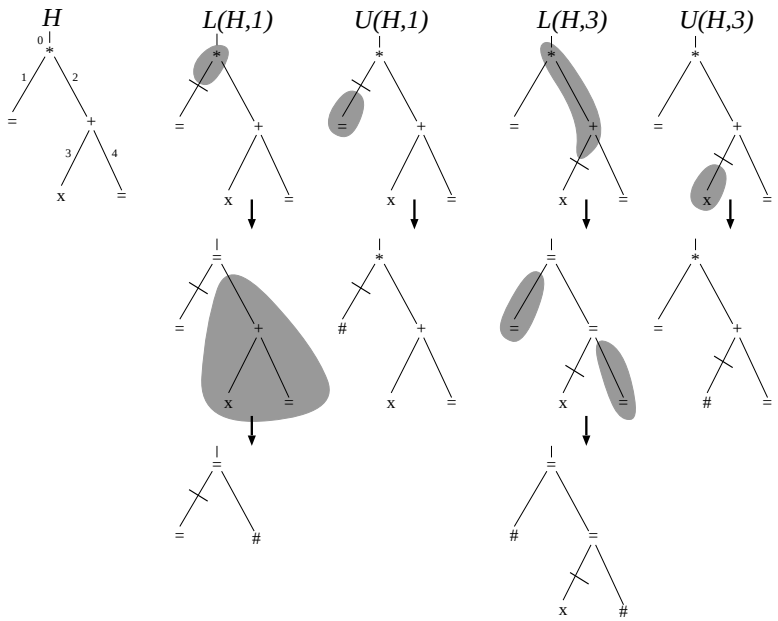

Fig. 1. Example of schema and some of its potential hyperschema building blocks. The crossover points in H are numbered as shown in the top left.

3 Node Reference Systems

Given a syntax tree like, for example, the one in Fig. 2 which represents the S-expression `(A (B C D) (E F (G H)))`, there can be different methods to indicate unambiguously the position of one particular node in the tree. One method is to use the path from the root node [3]. The path can be specified indicating which branch to select to find the target node for every node encountered starting from the root of the tree. This reference system presents the disadvantage of not corresponding to our typical notion of a Cartesian reference system, because the number of coordinates necessary to locate a node grows with the depth of the node in the tree.

A better alternative from this point of view is to organise the nodes in the tree into layers of increasing depth (see Fig. 3), to align them to the left and and then to assign an index to each node in a layer. The layer number d and the index i can then be used to define a Cartesian coordinate system. So, for example, node `G` in Fig. 2 would be at coordinates (2,3). This reference system presents the problem that it is not possible to infer the structure of a tree from the coordinates of its nodes.

A coordinate system similar to this one but without this problem can be defined by assuming that the trees to represent have nodes with arities not bigger than a predefined maximum value $a_{\max}$. Then one could define a node-reference system like the previous one for 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.. Finally one could use the maximal system also to locate the nodes of non-maximal trees. This is possible because a non-maximal tree can always be described using a subset of the nodes and links in the

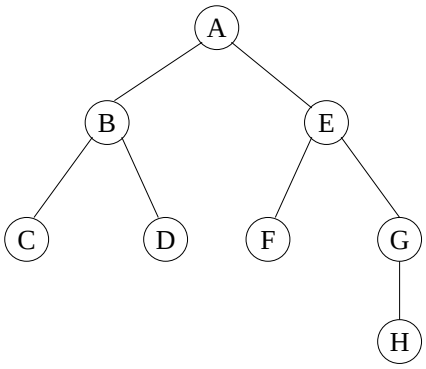

Fig. 2. A sample syntax tree.

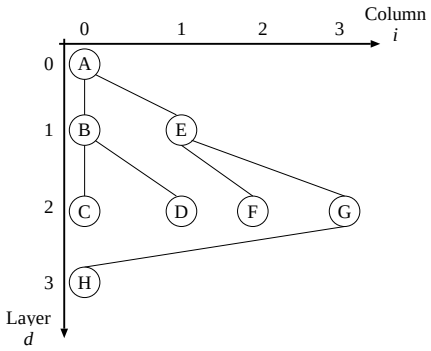

Fig. 3. A Cartesian node reference system.

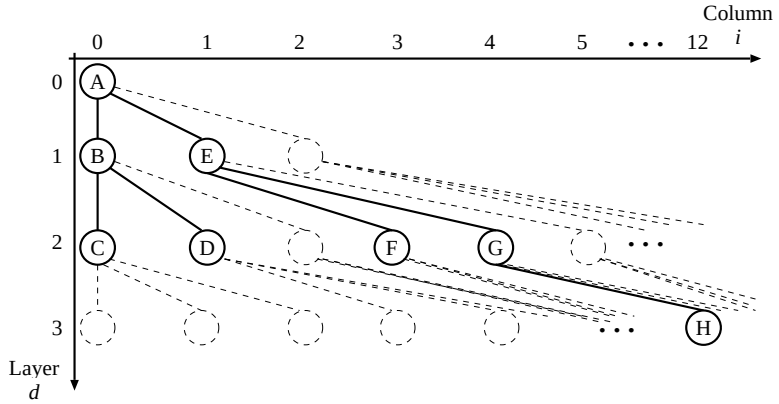

Fig. 4. Tree-independent Cartesian node reference system. The syntax tree for the program `(A (B C D) (E F (G H)))` is drawn with solid lines. Nodes and links of the maximal tree are drawn with dashed lines.

maximal tree. This is illustrated in Fig. 4 assuming $a_{\max} = 3$. So, for example, node `G` in Fig. 2 would have coordinates (2,4) while node `H` would have coordinates (3,12). In this reference system it is always possible to find the route to the root node from any given valid pair of coordinates. Also, if one chooses a_{max} to be the maximum arity of the functions in the function set, it is possible to use this reference system to represent the structure of any program in any population. Because of these properties, we will use this reference system in the rest of this paper.

As clarified in the following section, it is sometimes necessary to be able to express the location of nodes in different trees at the same time. In this case we can extend the node-reference systems just introduced by concatenating the coordinates of the nodes in each reference system into a single multi-dimensional coordinate vector. For example, we can indicate two nodes, (d_1, i_1) in one tree and (d_2, i_2) in another tree, at the same time using the point (d_1, i_1, d_2, i_2) in a four-dimensional coordinate system.

Finally, it should be noted that in the Cartesian reference system in Fig. 4 it is possible to transform pairs of coordinates into integers by counting the nodes in the reference system in breadth-first order. Conversely, it is also possible to map integers

into node coordinates unambiguously. We will use this property to simplify the notation in some of the following sections.

4 Functions over Node Reference Systems

Given a node reference system it is possible to define functions over it. An example of such functions is a function which represents a particular computer program. If one considers the program h one could define it as the function $h(d,i)$ which returns the node in h stored at position (d,i) if (d,i) is a valid pair of coordinates. If (d,i) is not in h then a conventional default value, $\emptyset$, is returned to represent the absence of a node. For example, for the program h=`(A (B C D) (E F (G H)))` represented in Fig. 4, this function would return the following values: $h(0,0) =$ `A`, $h(1,0) =$ `B`, $h(1,1) =$ `E`, $h(1,2) = \emptyset$, $h(2,0) =$ `C`, $h(2,1) =$ `D`, $h(2,2) = \emptyset$, $h(2,3) =$ `F`, $h(2,4) =$ `G`, $h(2,5) = \emptyset$, etc.. So, programs can be seen as functions over the space $\mathbb{N}^2$. Below we will refer to $h(d,j)$ as the *name function* for h and we will denote it with $N(d,i,h)$.

Other examples of node functions include:

- The *size function* $S(d,i,h)$ which represents the number of nodes present in the subtree rooted at coordinates (d,i) in tree h, with the convention that $S(d,i,h) = 0$ if (d,i) indicates an inexistent node. The size function can be defined using the name function (see [13]).
- The *arity function* $A(d,i,h)$ which returns the arity of the node at coordinates (d,i) in h. For example, for the tree in Fig. 4, $A(0,0,h) = 2$, $A(1,0,h) = 2$, $A(2,1,h) = 0$ and $A(2,4,h) = 1$.
- The *type function* $T(d,i,h)$ which returns the *data type* of the node at coordinates (d,i) in h.
- The *function-node function* $F(d,i,h)$ which returns 1 if the node at coordinates (d,i) is in h and it is a function, 0 otherwise. The number of internal nodes in h is therefore given by $\sum_d \sum_i F(d,i,h)$.

Similarly, it is possible to define functions on multi-dimensional node-coordinate systems. Useful functions of this kind include:

- The *position match function*

$$\mathrm{PM}(d_1,i_1,d_2,i_2,h_1,h_2) = \begin{cases} 1 & \text{if } (d_1,i_1) = (d_2,i_2),\ N(d_1,i_1,h_1) \neq \emptyset \text{ and} \\ & N(d_2,i_2,h_2) \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

- The *type match function*

$$\mathrm{TM}(d_1,i_1,d_2,i_2,h_1,h_2) = \begin{cases} 1 & \text{if } T(d_1,i_1,h_1) = T(d_2,i_2,h_2), \\ 0 & \text{otherwise.} \end{cases}$$

- The *common region membership function*:

$$\mathcal{A}(d_1,i_1,d_2,i_2,h_1,h_2) = \\ \begin{cases} 1 & \text{if } (d_1,i_1) = (d_2,i_2) = (0,0), \\ 1 & \text{if } (d_1,i_1) = (d_2,i_2), \\ & A\,(d_1-1, \lfloor i_1/a_{\max} \rfloor, h_1) = A\,(d_2-1, \lfloor i_2/a_{\max} \rfloor, h_2) \text{ and} \\ & \mathcal{A}\,(d_1-1, \lfloor i_1/a_{\max} \rfloor, d_2-1, \lfloor i_2/a_{\max} \rfloor, h_1, h_2) = 1, \\ 0 & \text{otherwise,} \end{cases}$$

where $\lfloor \cdot \rfloor$ is the integer-part function. The notion of common region membership function allows us to formalise the notion of *common region*:

$$C(h_1, h_2) = \{(d, i) \mid \mathcal{A}(d, i, d, i, h_1, h_2) = 1\}. \tag{3}$$

5 Modelling Subtree-swapping Crossovers

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) which leads to the creation of an offspring. 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:

$$p(d, i|h) = \Pr\{\text{A node at depth } d \text{ and column } i \text{ is selected in program } h\}, \tag{4}$$

where we assume that $p(d, i|h)$ is zero for all the coordinates (d, i) which represent inexistent nodes in h.[4] For example, if we consider the tree in Fig. 4 and we select nodes with uniform probability, then $p(d, i|h) = \frac{1}{8}$ if (d, i) is a node, $p(d, i|h) = 0$ otherwise. If instead we select functions with a probability 0.9 and any node with a probability 0.1, like in standard GP crossover [6], then $p(0, 0|h) = p(1, 0|h) = p(1, 1|h) = p(2, 4|h) = 0.2375$, $p(2, 0|h) = p(2, 1|h) = p(2, 3|h) = p(3, 12|h) = 0.0125$, and $p(d, i|h) = 0$ for all other coordinate pairs.

There are many possible uses for $p(d, i|h)$ and other probability distributions over node reference systems (see [13]). However, here we will concentrate on their use in modelling crossover operators.

In general in order to model crossover operators we need to use the following conditional probability distribution function over the space $\mathbb{N}^4$:

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

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 Sec. 4. 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 Eq. 4. We will call *separable* crossover operators for which this relation is true.

Standard crossover is a separable operator. Indeed, assuming uniform selection of the crossover points,

$$p_{\text{StdUnif}}(d_1, i_1, d_2, i_2|h_1, h_2) = \frac{\delta(N(d_1, i_1, h_1) \neq \emptyset)\delta(N(d_2, i_2, h_2) \neq \emptyset)}{N(h_1)N(h_2)}, \tag{5}$$

[4] For this probability distribution we use the notation $p(d, i|h)$ rather than $p(d, i, h)$ to emphasise the fact that $p(d, i|h)$ can be seen as the conditional probability of selecting node (d, i) if (or given that) the program being considered is h. In the rest of the paper, we will do the same for other probabilities distributions.

where $N(h) = S(0,0,h)$ is the number of nodes in h.

For standard crossover with a 90%-function/10%-any-node selection policy, it is easy to show that

$$p_{\mathrm{Std}90/10}(d_1,i_1,d_2,i_2|h_1,h_2) = \left(0.9\frac{F(d_1,i_1,h_1)}{\sum_d\sum_i F(d,i,h_1)} + 0.1\frac{\delta(N(d_1,i_1,h_1)\neq\emptyset)}{N(h_1)}\right)\times \left(0.9\frac{F(d_2,i_2,h_2)}{\sum_d\sum_i F(d,i,h_2)} + 0.1\frac{\delta(N(d_2,i_2,h_2)\neq\emptyset)}{N(h_2)}\right), \tag{6}$$

where $F(d,i,h)$ is the function-node function defined in Sec. 4.

In some crossover operators the selection of the crossover points in the two parents is not performed independently. For example in one-point crossover, the first and second crossover points must have the same coordinates. In this case, if we assume to use a uniform probability of node selection, then

$$p_{1\mathrm{pt}}(d_1,i_1,d_2,i_2|h_1,h_2) = \begin{cases} 1/\mathbf{NC}(h_1,h_2) & \text{if } (d_1,i_1)=(d_2,i_2) \text{ and} \\ & (d_1,i_1)\in C(h_1,h_2), \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $\mathbf{NC}(h_1,h_2)$ is the number of nodes in the common region $C(h_1,h_2)$ between program h_1 and program h_2. This can also be expressed using the common region membership function $\mathcal{A}(d_1,i_1,d_2,i_2,h_1,h_2)$ defined in Sec. 4 (see [13]):

$$p_{1\mathrm{pt}}(d_1,i_1,d_2,i_2|h_1,h_2) = \frac{\mathcal{A}(d_1,i_1,d_2,i_2,h_1,h_2)}{\mathbf{NC}(h_1,h_2)}.$$

Similarly, it is possible to model strongly typed GP crossover [9] and context-preserving crossover [3] using the functions defined in Sec. 4 obtaining:

$$p_{\mathrm{stgp}}(d_1,i_1,d_2,i_2|h_1,h_2) = \frac{\mathrm{TM}(d_1,i_1,d_2,i_2,h_1,h_2)}{\sum_{d_1}\sum_{i_1}\sum_{d_2}\sum_{i_2}\mathrm{TM}(d_1,i_1,d_2,i_2,h_1,h_2)},$$

$$p_{\mathrm{context}}(d_1,i_1,d_2,i_2|h_1,h_2) = \frac{\mathrm{PM}(d_1,i_1,d_2,i_2,h_1,h_2)}{\sum_{d_1}\sum_{i_1}\mathrm{PM}(d_1,i_1,d_1,i_1,h_1,h_2)}.$$

Most other subtree-swapping crossover operators can be modelled using probability distribution functions over node reference systems (see [13] for other examples).

Thanks to these probabilistic models of crossover, it is possible to develop a general schema theory for GP as described in the following sections.

6 Microscopic Exact GP Schema Theorem 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 in Sec. 3, there is a one-to-one mapping between pairs of coordinates and natural numbers.

In order to obtain a schema theory valid for subtree-swapping crossovers, we need to extend the notion of hyperschema summarised in Sec. 2. We will call this new form of hyperschema a *Variable Arity Hyperschema* or *VA hyperschema* for brevity.

Definition 1. *A* Variable Arity hyperschema *is a rooted tree composed of internal nodes from the set $\mathcal{F} \cup \{=, \#\}$ and leaves from $\mathcal{T} \cup \{=, \#\}$, where $\mathcal{F}$ and $\mathcal{T}$ are the function and terminal sets. 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. VA hyperschemata generalise most previous definitions of schema in GP (see [13]).

Thanks to VA hyperschemata and to the notion of probability distributions over node reference systems, it is possible to obtain the following general result:

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*

$$\alpha(H,t) = (1 - p_{xo})p(H,t) + \tag{8}$$
$$p_{xo} \sum_{h_1} \sum_{h_2} p(h_1,t)p(h_2,t) \sum_{i \in H} \sum_{j} p(i,j|h_1,h_2)\delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,j))$$

where: p_{xo} is the crossover probability; $p(H,t)$ is the selection probability of the schema H; the first two summations are over all the 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; $\delta(x)$ is a function which returns 1 if x is true, 0 otherwise; $L(H,i,j)$ is the 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 hyperschema $L(H,i,j)$ represents the set of all programs whose subtree rooted at crossover point j matches the subtree of H rooted in node i. The idea behind its definition is that, if one crosses over at point j *any* individual matching $L(H,i,j)$ and at point i *any* individual matching $U(H,i)$, the resulting offspring is always an instance of H. Before we proceed with the proof of the theorem, let us try to understand with examples how $L(H,i,j)$ is built (refer to Sec. 2 and Fig. 1 to see how $U(H,i)$ is built). In the examples, we will use 2–D coordinates to designate the nodes i and j. Let us consider the schema $H =$`(* = (+ x =))`. As indicated in Fig. 5(a), $L(H,(1,0),(1,1))$ is

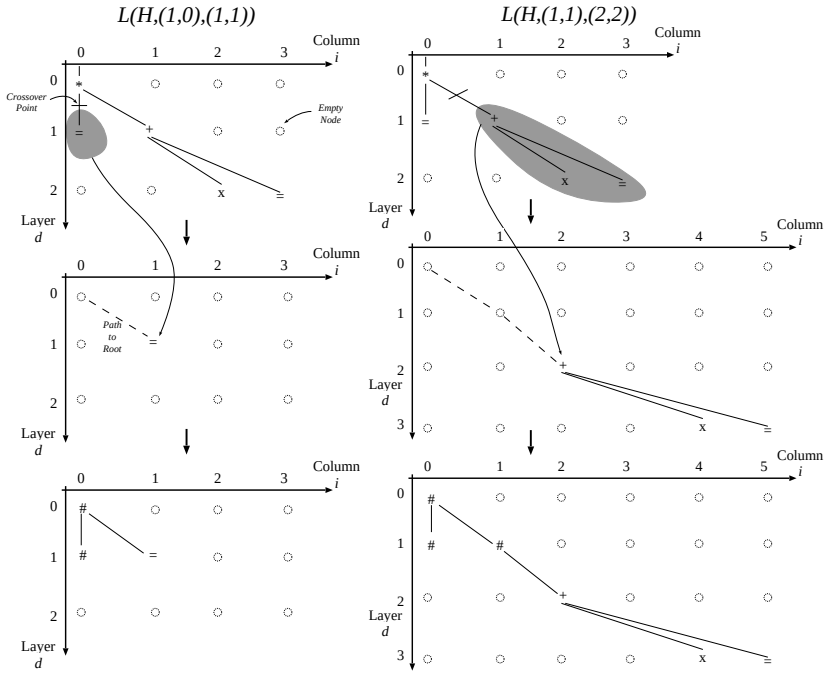

Fig. 5. Phases in the constructions of the VA hyperschema building block $L(H,(d_1,i_1),(d_2,i_2))$ of the schema $H =$`(* = (+ x =))` within a node coordinate system with $a_{max} = 2$ for: (a) $(d_1,i_1) = (1,0)$ and $(d_2,i_2) = (1,1)$ and (b) $(d_1,i_1) = (1,1)$ and $(d_2,i_2) = (2,2)$.

obtained through the following steps: a) we root the subschema below crossover point $(1,0)$, i.e. the symbol `=`, at coordinates $(1,1)$ in an empty reference system, b) we label the node at coordinates $(0,0)$ with a # function node (in this case this is the only node on the path between node $(1,1)$ and the root), and c) we label node (1,0) with a # terminal (this is because the node is to the left of the path between $(1,1)$ and the root, and it is an argument of one of the nodes replaced with #). Another example is provided in Fig. 5(b), where $L(H,(1,1),(2,2))$ is obtained through the following steps. Firstly, we root the subschema below crossover point $(1,1)$, i.e. the tree `(+ x =)`, at coordinates $(2,2)$ in an empty reference system. Note that this is not just a rigid translation: while the `+` is translated to position $(2,2)$, its arguments need to be translated more, i.e. to positions $(3,4)$ and $(3,5)$, because of the nature of the reference system used. Then, we label the nodes at coordinates $(0,0)$ and $(1,1)$ with # functions (these two nodes are on the path between node $(2,2)$ and the root). Finally, we label node (1,0) with a # terminal (this node is the only argument of one of the nodes replaced with # to be to the left of the path between $(2,2)$ and the root node).

Once the concept of $L(H,i,j)$ is available, the theorem can easily be proven.

Proof. Let $p(h_1,h_2,i,j,t)$ be the probability that, at generation t, the selection/crossover process will choose parents h_1 and h_2 and crossover points i and j. 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 [2]) 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:

$$E[\gamma] = \sum_{h_1} \sum_{h_2} \sum_{i} \sum_{j} g(h_1, h_2, i, j, H) p(h_1, h_2, i, j, t). \tag{9}$$

Since γ is a binary stochastic variable, its expected value also represent the proportion of times it takes the value 1. This corresponds to the proportion of times the offspring of h_1 and h_2 are in H.

We can write

$$p(h_1, h_2, i, j, t) = p(i, j|h_1, h_2) p(h_1, t) p(h_2, t), \tag{10}$$

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, while $p(h_1, t)$ and $p(h_2, t)$ are the selection probabilities for the parents. Substituting Eq. 10 into Eq. 9 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

$$E[\gamma] = \sum_{h_1} \sum_{h_2} p(h_1, t) p(h_2, t) \sum_{i \in H} \sum_{j} g(h_1, h_2, i, j, H) p(i, j|h_1, h_2). \tag{11}$$

The contribution to $\alpha(H, t)$ due to selection followed by crossover is $E[\gamma]$. By multiplying this by p_{xo} (the probability of doing selection followed by crossover) and adding the term $(1 - p_{xo})p(H, t)$ due to selection followed by cloning one obtains the r.h.s. of Eq. 8. □

7 Macroscopic Exact GP Schema Theorem

In order to transform Eq. 8 into an exact macroscopic description of schema propagation we will need to make one additional assumption on the nature of the subtree-swapping crossovers used: that the choice of the crossover points in any two parents, h_1 and h_2, depends only on their shapes, $G(h_1)$ and $G(h_2)$, not on the actual labels of their nodes, i.e. that $p(i, j|h_1, h_2) = p(i, j|G(h_1), G(h_2))$. We will term such operators *node-invariant* crossovers.

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*

$$\alpha(H, t) = (1 - p_{xo})p(H, t) + \tag{12}$$
$$p_{xo} \sum_{k,l} \sum_{i \in H} \sum_{j} p(i, j|G_k, G_l) p(U(H, i) \cap G_k, t) p(L(H, i, j) \cap G_l, t),$$

where the schemata G_1, G_2, $\cdots$ are all the possible fixed-size-and-shape schemata of order 0 (program shapes) and the other symbols have the same meaning as in Theorem 1.

Proof. 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 append the l.h.s. of these equations at the end of the triple summation in Eq. 8 and reorder the terms, we obtain:

$$\begin{aligned}
&\sum_{k,l} \sum_{h_1,h_2} p(h_1,t)p(h_2,t) \sum_{i\in H} \sum_{j} p(i,j|h_1,h_2) \times \\
&\quad \delta(h_1 \in U(H,i))\delta(h_1 \in G_k)\delta(h_2 \in L(H,i,j))\delta(h_2 \in G_l) \\
&= \sum_{k,l} \sum_{h_1\in G_k, h_2 \in G_l} p(h_1,t)p(h_2,t) \times \\
&\quad \sum_{i\in H} \sum_{j} p(i,j|h_1,h_2)\delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,j)).
\end{aligned}$$

For node-invariant 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:

$$\begin{aligned}
&\sum_{k,l} \sum_{h_1\in G_k, h_2 \in G_l} p(h_1,t)p(h_2,t) \times \\
&\quad \sum_{i\in H} \sum_{j} p(i,j|G(h_1),G(h_2))\delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,j)) \\
&= \sum_{k,l} \sum_{h_1\in G_k, h_2 \in G_l} p(h_1,t)p(h_2,t) \times \\
&\quad \sum_{i\in H} \sum_{j} p(i,j|G_k,G_l)\delta(h_1 \in U(H,i))\delta(h_2 \in L(H,i,j)) \\
&= \sum_{k,l} \sum_{i\in H} \sum_{j} p(i,j|G_k,G_l) \underbrace{\sum_{h_1\in G_k} p(h_1,t)\delta(h_1 \in U(H,i))}_{p(U(H,i)\cap G_k,t)} \times \\
&\quad \underbrace{\sum_{h_2\in G_l} p(h_2,t)\delta(h_2 \in L(H,i,j))}_{p(L(H,i,j)\cap G_l,t)},
\end{aligned}$$

which completes the proof of the theorem. □

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 $\emptyset$. So, the theorem expresses the total transmission probability of H only using the selection probabilities of a set of lower- (or same-) order schemata.

8 Applications and Specialisations

In this section we give examples that show how the theory can be specialised to obtain schema theorems for specific crossover operators, and we illustrate how it can be used

to obtain other general theoretical results, such as an exact definition of effective fitness and a size-evolution equation for GP.

8.1 Macroscopic Exact Schema Theorem for GP with Standard Crossover

Let us apply Theorem 2 to standard crossover. It is easy to show that standard crossover is node invariant (see [13]). So, we can substitute the expression of $p(i,j|G_k,G_l)$ in Eq. 12 with the expression

$$p(i,j|G_k,G_l) = \frac{\delta(N(i,G_k) \neq \emptyset)\delta(N(j,G_l) \neq \emptyset)}{N(G_k)N(G_l)},$$

obtained from Eq. 5. This yields

$$\alpha(H,t) = (1-p_{xo})p(H,t) + p_{xo} \sum_{k,l} \sum_{i \in H} \sum_{j} \frac{\delta(N(i,G_k) \neq \emptyset)}{N(G_k)} \times \frac{\delta(N(j,G_l) \neq \emptyset)}{N(G_l)} p(U(H,i) \cap G_k, t) p(L(H,i,j) \cap G_l, t).$$

From this we obtain

Theorem 3. *The total transmission probability for a fixed-size-and-shape GP schema H under standard crossover with uniform selection of crossover points is*

$$\alpha(H,t) = (1-p_{xo})p(H,t) + p_{xo} \sum_{k,l} \frac{1}{N(G_k)N(G_l)} \sum_{i \in H \cap G_k} \sum_{j \in G_l} p(U(H,i) \cap G_k, t) p(L(H,i,j) \cap G_l, t). \quad (13)$$

If one further specialises this result to the case of linear structures (see [13]), the schema theorem for linear structures reported in [20] is obtained.

8.2 Size-evolution Equation for GP with Subtree-swapping Crossover

Let us call *symmetric* any crossover operator for which $p(i,j|h_1,h_2) = p(j,i|h_2,h_1)$. Then, by using the microscopic schema theorem in Eq. 8 it is possible to prove (see [13]) the following

Theorem 4. *The expected mean size of the programs at generation $t+1$, $E[\mu(t+1)]$, in a GP system with a symmetric subtree-swapping crossover operator in the absence of mutation can equivalently be expressed in microscopic form as*

$$E[\mu(t+1)] = \sum_{h \in \mathcal{P}} N(h)p(h,t), \quad (14)$$

where $\mathcal{P}$ denotes the population, or in macroscopic form as

$$E[\mu(t+1)] = \sum_{l} N(G_l)p(G_l,t). \quad (15)$$

The theorem is important because it indicates that for symmetric subtree-swapping crossover operators the mean program size evolves as if selection only was acting on the population. This means that if there is a variation in mean size, like for example in the presence of bloat, that can only be attributed to some form of positive or negative selective pressure on some or all the shapes G_l.

From this theorem it follows that on a flat landscape

$$E[\mu(t+1)] = \sum_l N(G_l)p(G_l,t) = \sum_l N(G_l)\frac{m(G_l,t)}{M}$$

(or $E[\mu(t+1)] = \sum_l N(G_l)\alpha(G_l,t-1)$ for infinite populations), whereby

Corollary 1. *On a flat landscape,*

$$E[\mu(t+1)] = \mu(t). \tag{16}$$

For infinite populations, the expectation operator can be removed from the previous theorem and corollary. In [20] we obtained more specific versions of these results for one-point crossover and standard crossover acting on linear structures.

8.3 Effective Fitness for GP with Subtree-Swapping Crossovers

Once an exact schema theorem is available, it is easy to extend to GP with subtree-swapping crossovers the notion of effective fitness provided in [22, 23]:

$$f_{\text{eff}}(H,t) = \frac{\alpha(H,t)}{p(H,t)}f(H,t). \tag{17}$$

By using this definition and the value of $\alpha(H,t)$ in Eq. 12, we obtain the following

Theorem 5. *The effective fitness of a fixed-size-and-shape GP schema H under a node-invariant subtree-swapping crossover operator and no mutation is*

$$f_{\text{eff}}(H,t) = f(H,t)\Big[1 - p_{xo} \times \Big(1 - \sum_{k,l}\sum_{i\in H,j} p(i,j|G_k,G_l)\frac{p(U(H,i)\cap G_k,t)p(L(H,i,j)\cap G_l,t)}{p(H,t)}\Big)\Big]. \tag{18}$$

This result gives the *true effective fitness for a GP schema* under subtree swapping crossover: it is not an approximation or a lower bound.

It is possible to specialise this definition to standard crossover:

Corollary 2. *The effective fitness of a fixed-size-and-shape GP schema H under standard crossover with uniform selection of crossover points is*

$$f_{\text{eff}}(H,t) = f(H,t)\Big[1 - p_{xo} \times \Big(1 - \sum_{k,l}\sum_{i\in H\cap G_k}\sum_{j\in G_l} \frac{p(U(H,i)\cap G_k,t)p(L(H,i,j)\cap G_l,t)}{N(G_k)N(G_l)p(H,t)}\Big)\Big]. \tag{19}$$

In future work we intend to compare this definition with the approximate notions of effective fitness and operator-adjusted fitness introduced in [10] and [4], respectively.

9 Conclusions

In this paper a new general schema theory for genetic programming is presented. The theory includes two main results describing the propagation of GP schemata: a microscopic schema theorem and a macroscopic one. The microscopic version is applicable to crossover operators which replace a subtree in one parent with a subtree from the other parent to produce the offspring. The macroscopic version is valid for subtree-swapping crossover operators in which the probability of selecting any two crossover points in the parents depends only on their size and shape. Therefore, these theorems are very general and can be applied to model most GP systems used in practice.

Like other recent schema theory results [22, 23, 14, 12], our theory gives an exact formulation (rather than a lower bound) for the expected number of instances of a schema at the next generation. One special case of this theory is the exact schema theorem for standard crossover: a result that has been awaited for many years.

As shown by some recent explorations reported in [20, 8], exact schema theories 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. Also, as discussed in [15] for one-point crossover, exact macroscopic theories open the way to future work on GP convergence, population sizing, and deception, only to mention some possibilities. In the future we hope to use the general schema theory reported in this paper to obtain new general results in at least some of these exciting directions.

Acknowledgements

The author would like to thank the members of the EEBIC (Evolutionary and Emergent Behaviour Intelligence and Computation) group at Birmingham, Nic McPhee, Jon Rowe, Julian Miller, Xin Yao, and Bill Langdon for useful discussions and comments.

References

[1] L. Altenberg. Emergent phenomena in genetic programming. In A. V. Sebald and L. J. Fogel, editors, *Evolutionary Programming — Proceedings of the Third Annual Conference*, pages 233–241. World Scientific Publishing, 1994.

[2] 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. 1995. Morgan Kaufmann.

[3] P. D'haeseleer. Context preserving crossover in genetic programming. In *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, volume 1, pages 256–261, Orlando, Florida, USA, 27-29 June 1994. IEEE Press.

[4] D. E. Goldberg. Genetic algorithms and Walsh functions: II. Deception and its analysis. *Complex Systems*, 3(2):153–171, Apr. 1989.

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

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

[7] W. B. Langdon. Size fair and homologous tree genetic programming crossovers. *Genetic Programming And Evolvable Machines*, 1(1/2):95–119, Apr. 2000.

[8] 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.

[9] D. J. Montana. Strongly typed genetic programming. *Evolutionary Computation*, 3(2):199–230, 1995.

[10] P. Nordin and W. Banzhaf. Complexity compression and evolution. In L. Eshelman, editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, pages 310–317, Pittsburgh, PA, USA, 15-19 July 1995. Morgan Kaufmann.

[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. Exact schema theorem and effective fitness for GP with one-point crossover. In D. Whitley, *et al.*, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 469–476, Las Vegas, July 2000. Morgan Kaufmann.

[13] R. Poli. General schema theory for genetic programming with subtree-swapping crossover. Technical Report CSRP-00-16, University of Birmingham, School of Computer Science, November 2000.

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

[15] R. Poli. Exact schema theory for genetic programming and variable-length genetic algorithms with one-point crossover. *Genetic Programming and Evolvable Machines*, 2(2), 2001. Forthcoming.

[16] R. Poli and W. B. Langdon. Genetic programming with one-point crossover. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 180–189. Springer-Verlag London, 1997.

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

[18] 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.

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

[20] 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.

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

[22] C. R. Stephens and H. Waelbroeck. Effective degrees of freedom in genetic algorithms and the block hypothesis. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, pages 34–40, East Lansing, 1997. Morgan Kaufmann.

[23] C. R. Stephens and H. Waelbroeck. Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124, 1999.

[24] 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.