Appeared in the Proceedings of Genetic Programming'98, Madison, Wisconsin

On the Search Properties of Different Crossover Operators in Genetic Programming

Riccardo Poli and W.B. Langdon
School of Computer Science, The University of Birmingham, Birmingham B15 2TT, UK
{R.Poli,W.B.Langdon}@cs.bham.ac.uk
Phone: +44-121-414-3739/4791

ABSTRACT

In this paper we study and compare the search properties of different crossover operators in genetic programming (GP) using probabilistic models and experiments to assess the amount of genetic material exchanged between the parents to generate the offspring. These operators are: standard crossover, one-point crossover and a new operator, uniform crossover. Our analysis suggests that standard crossover is a local and biased search operator not ideal to explore the search space of programs effectively. One-point crossover is better in some cases as it is able to perform a global search at the beginning of a run, but it suffers from the same problems as standard crossover later on. Uniform crossover largely overcomes these limitations as it is global and less biased.

1 Introduction

Genetic programming (GP) is a non-deterministic search technique to explore the space of possible programs. It is very well known from the AI literature that the performance of any search algorithm depends crucially on the representation adopted for the states (tentative solutions) of the problem and the operators used to produce new states [Russell and Norvig, 1995]. In this paper we concentrate on the latter and ask whether different crossover operators have the features necessary to explore the space of possible programs efficiently and effectively.

In the recent past this question has bothered ourselves and a number of other GP researchers when the idea that crossover was central and that mutation was unnecessary in GP strongly stressed in Koza's work [Koza, 1992, Koza, 1994] was questioned. Indeed, recent results highlight a limited performance advantage of crossover compared to mutation based operators [O'Reilly and Oppacher, 1996, Luke and Spector, 1997, Angeline, 1997, Chellapilla, 1997].

In [Poli, 1997] we proposed the hypothesis that this unexpected ineffectiveness of GP crossover might be due to the fact that crossover is a local and biased search operator which, as a result, is unable to search properly the space of programs. It is local in the sense that the offspring it produces inherit most of their code from one parent most of the times. It is biased as, probabilistically, it concentrates on certain types of local adjustments, namely very near the leaves of the tree. This hypothesis is supported by simple calculations (similar to those reported in [Rosca and Ballard, 1995, Rosca, 1996]), by experiments in which the expected size of the offspring of full trees of very different depth was estimated and by other experiments and models of crossover behaviour in the MAX problem described in [Langdon and Poli, 1997].

In contrast, most crossover operators used in Genetic Algorithms (GAs) tend to include in the offspring material coming from both parents without a bias towards one or the other. For example, one-point crossover, uniform crossover, etc. all transfer genetic material from each parent with a 50% probability on average. If the parents are quite different this leads to produce offspring which are a long way away from their parents in the genotype space. In this sense we could say that GA crossover operators are global search operators.

GA crossover operators are also able to generate offspring which are very close to their parents. This can happen in two cases: a) when by chance the offspring inherits most of its genetic material from one or the other parent (for some crossover operators, like uniform crossover, this is a relatively unlikely event), or b) when the parents are quite close in the genotype space. In this cases we may say that crossover behaves like a local search operator.

Both these properties are very important: globality of search leads to the exploration of the search space and also to the ability to escape from local maxima, while locality allows the refinement of inaccurate solutions. Standard GP crossover seems to have only a distorted (biased) version of one of them.[1] The questions we address in

[1] To improve this situation recently some researchers have proposed schemes to modify the node selection probability at different levels in the

this paper are: are the properties of locality and globality present, and to which extent, in different GP crossover operators? The operators we will consider in this study are standard crossover, one-point crossover, an operator inspired by Genetic Algorithms (GAs) introduced in our recent research on GP schemata [Poli and Langdon, 1997c, Poli and Langdon, 1997a, Poli and Langdon, 1998], and uniform crossover, a new GA-inspired operator.

To answer these questions we will proceed along the following lines extending and completing the ideas presented in [Poli, 1997]. Firstly we will propose a simple general theory of the amount of genetic material exchanged between the parents in each crossover operator. Then, we apply the theory to an idealised case to show (1) how the amount of genetic material exchanged between the parents in standard crossover may be, on average, quite small, i.e. the search may be within a small neighbourhood of the parents (local), (2) how one-point crossover's search is more global initially but becomes similar to standard crossover later on, and (3) how uniform crossover has more global search properties throughout a run. These results are presented in Section 3. Finally, we will study the amount of genetic material exchanged by the parents to create the offspring in more realistic GP runs on the even-4 parity problem in the presence of different crossover operators and function sets. These experiments are described in Section 4. We discuss our results and draw some conclusions in Section 5.

2 Standard, One-point and Uniform Crossover for GP

Standard GP crossover works by randomly selecting one crossover point in each parent tree and then by swapping the subtrees attached to such nodes to obtain the offspring. In recent research we introduced a new crossover operator, one-point crossover, which works by swapping subtree, but it uses a single common crossover point as explained below.

The way one-point crossover works was suggested by considering the three steps involved in one-point crossover in binary GAs. These steps are schematised in Figure 1(a). They are: 1) alignment of the parent strings, 2) selection of a common crossover point, 3) swap of the right-end sides of the strings to obtain two offspring. If all the trees in a GP population had exactly the same shape and size then a crossover operator involving exactly the same three steps could easily be implemented. The nodes in the trees would play the role of the digits in the bit strings, the links would play the role of the gaps between digits. Figure 2(a) depicts this situation. In the figure the alignment process corresponds to translating the graphical representation of one of the parents until its root node coincides with the root node of the other.

What would happen if one tried the same procedure with trees having different shapes? To a certain extent it would still

tree so as to make bigger changes more likely [Rosca and Ballard, 1996, Harries and Smith, 1997, Ito et al., 1998].

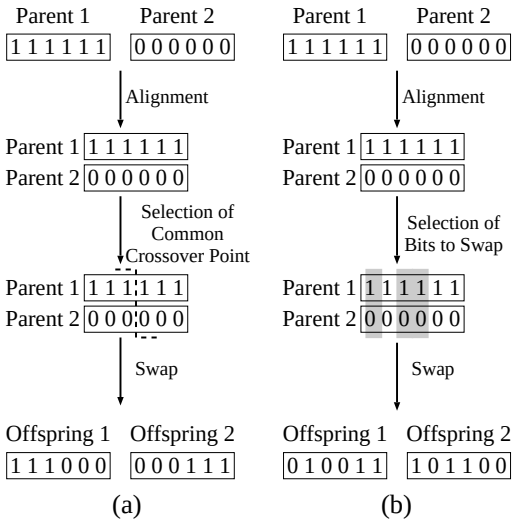

Figure 1: (a) One-point crossover in binary GAs, and (b) uniform crossover in binary GAs.

be possible to align the trees. After the alignment it would be easy to identify the links which overlap, to select a common crossover point and swap the corresponding subtrees. Our one-point crossover operator for GP is based on these simple steps, which are illustrated in Figure 2(b).

From the implementation point of view, the three phases involved in one-point crossover are as follows. Copies of the two parent trees are recursively (jointly) traversed starting from the root nodes to identify the parts with the same shape, i.e. with the same arity in the nodes visited (*alignment*). Recursion is stopped as soon as an arity mismatch between corresponding nodes in the two trees is present. All the nodes and links encountered are stored. They form a tree fragment that we call the *common region*. A random crossover point is selected with a uniform probability among the stored links (*crossover point selection*). The two subtrees below the common crossover point are swapped in exactly the same way as in standard crossover (*swap*).

An interesting variant of one-point crossover, which we call *strict one-point crossover* [Poli and Langdon, 1997b], behaves exactly like one-point crossover except that the crossover point can be located only in the parts of the two trees which are exactly the same (i.e. which have the same functions in the nodes encountered traversing the trees from the root node). The links eligible as crossover points in strict one-point crossover are a subset of those eligible in standard one-point crossover.

By analysing other forms of GA crossover it is possible to define new forms of crossover for GP. In this paper we want to introduce a new one which we call *uniform crossover* as it was suggested by considering the steps involved in uniform crossover in binary GAs. These steps are schematised in Figure 1(b). They are: 1) alignment of the parent strings, 2) selection of the bits to swap, 3) swap of the selected bits to obtain two offspring. If all the trees in a GP population had exactly the same shape and size then exactly the same

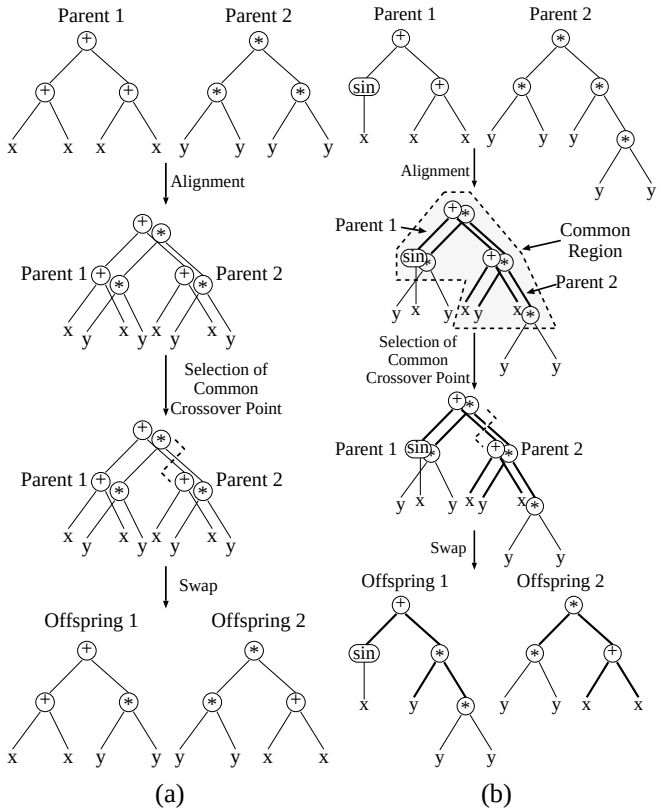

Figure 2: (a) One-point crossover for GP when both parents have the same shape, and (b) general form of one-point crossover for GP. In (b) the links that can be selected as common crossover points are drawn with thick lines.

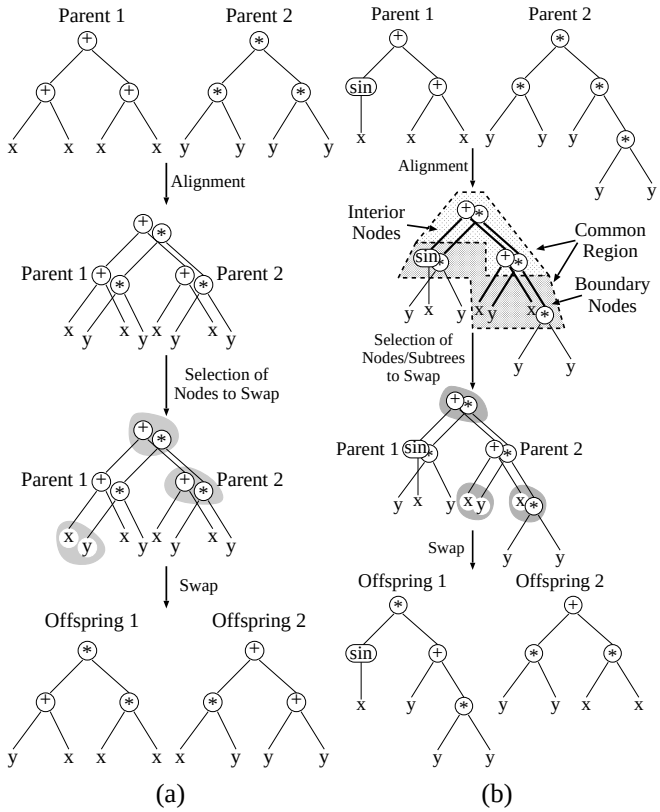

Figure 3: (a) Uniform crossover for GP when both parents have the same shape, and (b) general form of uniform crossover for GP. In (b) the nodes that can be selected for swapping are those included in the common region.

three steps could easily be implemented. Figure 3(a) depicts this situation. When the parent trees have different shapes we can proceed similarly to the one-point case. The differences between the two operators can be easily inferred from Figure 3(b). Again, firstly, copies of the two parent trees are recursively (jointly) traversed starting from the root nodes to identify the common region, i.e. the parts with the same arity in the nodes visited. All the nodes encountered are stored. Then nodes in the common parts are swapped with a uniform probability. If a node belongs to the boundary of the common region (i.e. if it is a leaf of the tree fragment representing the common region) and is a function then also the subtree below it is swapped, otherwise only the node label is swapped.

Like for one-point crossover, it is possible to define a *strict uniform crossover* operator which behaves exactly like uniform crossover except that the nodes/subtrees swapped can be located/rooted only in the parts of the two trees which are exactly the same.

3 Analysis of Crossover Search Properties

In GAs the offspring generated by crossover may be very similar to or very different from their parents depending on the similarity between the parents and the amount of genetic material exchanged to form the offspring. Maximum exploration of the search space is achieved when the offspring include 50% of the material from one parent and the other 50% from the other. The fact that the parents "exchange" a lot of genetic material is not a sufficient condition for the offspring to be significantly different from the parents. The exchanged material must also be different.

In this section we model mathematically the effects of standard crossover, one-point crossover and uniform crossover (both in their normal and strict versions) in terms of the amount of genetic material exchanged between the parents. In this way we can identify the presence or absence of one of the necessary conditions for a good exploration of the search space: the production of offspring sufficiently different from their parents so as to explore the search space globally and efficiently. We also want to check whether different forms of crossover have the other necessary property of a good search method: the ability to perform local search. This is necessary to explore well the hills in the fitness landscape.

3.1 Node Density Functions

Let us consider a particular program tree h in the population and define the *node density function* of h, $p(d|h)$, as the fraction of nodes at depth d in h. If we selected nodes in the tree with a uniform probability, the node density func-

tion would coincide with the (conditional) probability that a node at depth d is selected (given that the program in which the nodes are selected is h). Also, let us call $S(d,h)$ the expected size of the subtrees at depth d in the program h and $S(h) = S(0,h)$ the size of h.

As h has $p(d|h)S(h)$ nodes at depth d, by hypothesising that all subtrees rooted at depth d have the same structure and size, it is possible to estimate the expected number of nodes in such subtrees:

$$S(d,h) = \frac{\overbrace{\sum_{x=d}^{\infty} p(x|h)S(h)}^{\text{Nodes at depth} \geq d}}{\underbrace{p(d|h)S(h)}_{\text{Nodes at depth } d}} = \frac{\sum_{x=d}^{\infty} p(x|h)}{p(d|h)}. \quad (1)$$

The node density function and $S(d,h)$ can also be used to compute the average branching factor at depth d in tree h:

$$B(d|h) = \frac{\overbrace{S(d+1,h)p(d+1|h)}^{\text{Nodes at depth } d+1}}{\underbrace{S(d,h)p(d|h)}_{\text{Nodes at depth } d}}$$

3.2 Genetic Material Exchanged by Crossover

The amount of genetic material exchanged between the parents to generate the offspring during crossover can be quantified using the number of nodes inserted (or removed) to obtain the offspring. In normal crossover and one-point crossover this corresponds to the size of the subtrees swapped. In uniform crossover, this corresponds to the size of the subtrees swapped plus the number of nodes swapped. In the following we will consider these crossover operators separately.

For the sake of simplicity let us consider a GP system in which crossover points are selected with uniform probability among the nodes in the parent trees.

The expected size S_r of the subtree removed from the first parent during crossover can be computed as follows:

$$E[S_r] = \sum_{h_1,h_2} \sum_{d_1,d_2} S(d_1,h_1)p(d_1,d_2|h_1,h_2)p(h_1)p(h_2) \quad (2)$$

where the summation over h_1 and h_2 is carried out over all the individuals in the population, the summation over d_1 and d_2 (the depths of the the crossover point in the first parent and second parent, respectively) is from 0 to ∞, $p(h_1)$ and $p(h_2)$ are the probabilities of individuals h_1 and h_2 being selected,[2] and $p(d_1,d_2|h_1,h_2)$ is the probability that a crossover point at depth d_1 is chosen in the first parent *and* a crossover point at depth d_2 is chosen in the second parent *given that* the first parent is h_1 and the second parent is h_2.

[2] If fitness proportionate selection is used $p(h) = f(h)/(M\bar{f})$, where M is the population size, $\bar{f}$ is the average population fitness and $f(h)$ is the fitness of program h.

The crossover operators described in Section 2 and analysed in the following are symmetric with respect to the two parents. So, $E[S_r]$ also coincides with the amount of genetic material removed from the second parent, i.e. the genetic material *inserted* in the first parent. For this reason in the following we will use the term *amount of genetic material exchanged* when referring to $E[S_r]$.

3.2.1 Standard Crossover Case

If standard crossover is used, then the crossover points are randomly selected independently. This means that

$$p(d_1,d_2|h_1,h_2) = p(d_1|h_1)p(d_2|h_2)$$

where $p(d_1|h_1)$ and $p(d_2|h_2)$ are the node density functions of the programs h_1 and h_2, respectively. By substituting this equation into Equation 2 it follows that the expected value of the amount of genetic material exchanged by crossover $S_{x_{\text{Std}}}$ is

$$\begin{aligned} E[S_{x_{\text{Std}}}] &= \sum_{h_1}\sum_{d_1} S(d_1,h_1)p(d_1|h_1)p(h_1) \\ &\quad \cdot \underbrace{\Big[\sum_{h_2} p(h_2)\Big(\overbrace{\sum_{d_2} p(d_2|h_2)}^{=1}\Big)\Big]}_{=1} \\ &= \sum_{h_1} p(h_1) \sum_{d_1} S(d_1,h_1)p(d_1|h_1) \end{aligned} \quad (3)$$

3.2.2 One-point Crossover Case

If one-point crossover is used, the crossover point in the second parent cannot be selected independently from the one in the first parent. Indeed, in one-point crossover d_1 and d_2 are always the same. Therefore, $p(d_1,d_2|h_1,h_2) = 0$ if $d_1 \neq d_2$. As a consequence, from Equation 2 we obtain

$$\begin{aligned} E[S_{x_{1\text{pt}}}] &= \sum_{h_1} p(h_1) \sum_{d_1} S(d_1,h_1) \sum_{h_2} p(h_2) \\ &\quad \cdot \underbrace{\sum_{d_2} p(d_1,d_2|h_1,h_2)}_{p(d_1,d_1|h_1,h_2)} \\ &= \sum_{h_1} p(h_1) \sum_{d_1} S(d_1,h_1) \sum_{h_2} p(h_2)p(d_1|h_1,h_2) \\ &= \sum_{h_1} p(h_1) \sum_{d_1} S(d_1,h_1)p_{1\text{pt}}(d_1|h_1,t) \end{aligned} \quad (4)$$

where we renamed $p(d_1,d_1|h_1,h_2)$ as $p(d_1|h_1,h_2)$ for brevity and

$$p_{1\text{pt}}(d_1|h_1,t) \stackrel{def}{=} \sum_{h_2} p(h_2) \cdot p(d_1|h_1,h_2)$$

is the probability that a node at depth d_1 is selected for one-point crossover in individual h_1 given the other individuals

present in the population at generation t. As $p(d_1|h_1,h_2)$ can be seen as the node density function of the common region of h_1 with respect to h_2, $p_{1\mathrm{pt}}(d_1|h_1,t)$ can be thought of as the node density function of the expected common region of h_1 with respect to the whole population.

3.2.3 Uniform Crossover Case

In uniform crossover two forms of exchange of genetic material take place: exchange of nodes and exchange of subtrees. Subtrees are exchanged when nodes at the boundaries of the common region are selected for swap. We call these, *boundary nodes*; all the other nodes in the common region are termed *interior nodes* (see Figure 3(b)).

Let $N_b(d_1|h_1,h_2)$ be the number of boundary nodes at depth d_1 in the first parent and $N_i(d_1|h_1,h_2)$ the number of interior nodes at depth d_1. If p_s is the probability of swapping a node/subtree then we can compute the expected amount of genetic material exchanged during uniform crossover as follows:

$$E[S_{x_{\mathrm{Unif}}}] = p_s \sum_{h_1,h_2} p(h_1)p(h_2) \sum_{d_1} \Big[\underbrace{N_i(d_1|h_1,h_2)}_{\text{Swap of nodes}} + \underbrace{N_b(d_1|h_1,h_2)S(d_1,h_1)}_{\text{Swap of subtrees}} \Big]$$

If no boundary nodes were present at level d_1 then

$$N_i(d_1|h_1,h_2) = B(d_1-1|h_1)N_i(d_1-1|h_1,h_2).$$

However, in general some interior nodes at one level will generate boundary nodes at the next. Their number is given by the following recurrence relation:

$$N_b(d_1|h_1,h_2) = B(d_1-1|h_1)N_i(d_1-1|h_1,h_2) - N_i(d_1|h_1,h_2).$$

This relation is valid for $d_1 > 0$ and as long as $B(d_1-1|h_1)$ is defined (i.e. for $p(d_1-1|h_1) > 0$). If we use it in the expression of $E[S_{x_{\mathrm{Unif}}}]$ we obtain

$$\begin{aligned} E[S_{x_{\mathrm{Unif}}}] = p_s \sum_{h_1,h_2} p(h_1)p(h_2) \Big[& N_i(0|h_1,h_2) + \\ & N_b(0|h_1,h_2)S(0,h_1) + \\ & \sum_{d_1=1}^{\infty} \Big(N_i(d_1|h_1,h_2)(1-S(d_1,h_1)) + \\ & N_i(d_1-1|h_1,h_2)B(d_1-1|h_1)S(d_1,h_1) \Big) \Big] \end{aligned}$$

where we used the fact that $N_i(d_1|h_1,h_2) = 0$ for all d_1's such that $p(d_1,h_1) = 0$. As $N_b(0|h_1,h_2) = 1 - N_i(0|h_1,h_2)$ from this equation we obtain:

$$\begin{aligned} E[S_{x_{\mathrm{Unif}}}] &= p_s \sum_{h_1,h_2} p(h_1)p(h_2) \Big[S(0,h_1) + \\ &\quad \sum_{d_1} N_i(d_1|h_1,h_2) \Big(1 - S(d_1,h_1) + \\ &\quad B(d_1|h_1)S(d_1+1,h_1) \Big) \Big] \end{aligned}$$

As the size of any subtree at depth d_1 is one plus the sum of the size of the arguments of the root of the subtree, we have that the expected size of a subtree in h_1 is

$$S(d_1,h_1) = 1 + B(d_1|h_1)S(d_1+1,h_1)$$

which substituted in the previous equation yields:

$$\begin{aligned} E[S_{x_{\mathrm{Unif}}}] &= p_s \sum_{h_1,h_2} p(h_1)p(h_2)S(0,h_1) \\ &= p_s \sum_{h_1} p(h_1)S(h_1) = p_s E[S(h_1)] \end{aligned} \quad (5)$$

i.e. the amount of genetic material exchanged by uniform crossover is a constant fraction of the average size of the programs in the population.

3.3 Comparison between Crossover Operators

3.3.1 Standard vs. One-Point Crossover

The formulae describing the amount of genetic material exchanged in standard (Equation 3) and one-point crossover (Equation 4) are quite similar. The only difference is that with one-point crossover the probability of selecting a crossover point at a certain depth, $p_{1\mathrm{pt}}(d_1|h_1,t)$, varies over time for any given program. At the beginning of a run, if there is enough diversity in the initial population, $p_{1\mathrm{pt}}(d_1|h_1,t) \ll p(d_1|h_1)$ for big values of d_1 and, consequently, $p_{1\mathrm{pt}}(d_1|h_1,t) \gg p(d_1|h_1)$ for small values of d_1, especially for the strict form of one-point crossover. So, at the beginning of a run we should expect that $E[S_{x_{1\mathrm{pt}}}] \gg E[S_{x_{\mathrm{Std}}}]$. Indeed if we hypothesised a maximally diverse population in which

$$p(d_1|h_1,h_2) = \begin{cases} 1 & \text{if } d_1 = 0 \text{ and } h_1 \neq h_2 \\ 0 & \text{if } d_1 \neq 0 \text{ and } h_1 \neq h_2 \\ p(d_1|h_1) & h_1 = h_2 \end{cases}$$

then $E[S_{x_{1\mathrm{pt}}}] \approx \sum_{h_1} p(h_1)S(0,h_1)$, i.e. the expected exchanged size would approach the expected size of the programs in the population $E[S(h)]$. As in any case $E[S(h)] \geq E[S_{x_{1\mathrm{pt}}}]$, we can infer that with a sufficiently diverse population $E[S(h)] \gg E[S_{x_{\mathrm{Std}}}]$, i.e. that standard crossover is a local search operator. The fact that populations with standard crossover do not converge seems to suggest that $E[S_{x_{\mathrm{Std}}}]$ never approaches zero. So, standard crossover may be unable to effectively explore the neighbourhood of optima in the fitness landscape and to refine partial solutions.

Given that one-point crossover makes the population converge [Poli and Langdon, 1997a], at the end of a run all nodes in every tree could be selected freely, as in standard crossover. Hence

$$\lim_{t\to\infty} p_{1\mathrm{pt}}(d_1|h_1,t) = p(d_1|h_1) \Rightarrow \lim_{t\to\infty} E[S_{x_{1\mathrm{pt}}}] = E[S_{x_{\mathrm{Std}}}],$$

i.e. standard and one-point crossover asymptotically exchange the same amount of genetic material. However, this

does not mean that one-point crossover is unable to explore the maxima of the fitness landscape. Indeed, like one-point crossover in GAs, as the population converges the material replaced by GP one-point crossover tends to be more and more similar to the original material. Thus, GP one-point crossover becomes more and more local.

3.3.2 Standard/One-Point vs. Uniform Crossover

Despite the fact that in general multiple subtrees are swapped in a single application of uniform crossover while only one subtree is swapped in standard and one-point crossover, the form of Equation 5 is considerably simpler than Equations 3 and 4. This is because the amount of genetic material exchanged by transferring trees is exactly complemented by the genetic material exchanged by transferring single internal nodes (which cannot happen with standard and one-point crossover as they transfer a single subtree).

At the beginning of a run, if there is enough diversity in the initial population, uniform crossover will tend to swap only a few relatively large subtrees very near the root. This is particularly true for the strict version of this operator. As shown by the experiments in Section 4, uniform crossover, like one-point crossover, makes the population converge. So, over time an increasing number of nodes and progressively smaller trees will be swapped, until at the end of a run all nodes in every tree can be selected freely for swap and $N_b(d_1|h_1,h_2) = 0$ everywhere. At this stage only single nodes will be swapped.

Unlike standard and one-point crossover, uniform crossover (on average) builds offspring by using a fixed proportion p_s of the genetic material of one parent and a fixed proportion $1-p_s$ of the genetic material of the other parent. So, it can be a very local or very global search operator depending on the value of the parameter p_s. Once p_s is fixed, this character does not change during the run, unlike standard and one-point crossover.

If $p_s = 0.5$ uniform crossover is the GP equivalent of uniform crossover for binary strings. In this case the operator performs a maximal global interpolative search. However, as the population will converge, like for linear GAs, over time the operator will restrict the search to smaller and smaller parts of the genome, and transforms into a local operator.

3.3.3 Binary Fully-Balanced Population Model

To get an intuitive idea of how large or small $E[S_{x_{1\text{pt}}}]$, $E[S_{x_{\text{Std}}}]$ and $E[S_{x_{\text{Unif}}}]$ can be, let us consider a simple idealised model. Let us imagine that the initial population is created using the "full" initialisation method with the same initial depth d_{max}. If for simplicity one imagines that only arity-2 functions are used, then (for $d \leq d_{max}$):

$$p(d|h) = \frac{2^d}{2^{d_{max}+1}-1} \quad \text{and} \quad S(d,h) = 2^{d_{max}+1-d} - 1,$$

where h is a generic program in the population. If standard crossover is used, using these equations and Equation 3 we obtain that in the first generation ($t = 0$):

$$\begin{aligned} E[S_{x_{\text{Std}}}] &= \sum_{h_1} p(h_1) \sum_{d_1} \left(\frac{2^{d_{max}+1} - 2^d}{2^{d_{max}+1}-1} \right) \\ &= \sum_{h_1} p(h_1) \left(\frac{2^{d_{max}+1} d_{max} + 1}{2^{d_{max}+1}-1} \right) \\ &= \frac{2^{d_{max}+1} d_{max} + 1}{2^{d_{max}+1}-1} \approx d_{max} \approx \log_2(S(h)). \end{aligned}$$

For $t > 0$ standard crossover will start creating trees with different shapes. As usually some sort of bloating is present, we expect $E[S_{x_{\text{Std}}}]$ to grow. Even if it has been reported that trees tend not to grow balanced [Soule and Foster, 1997], it may still take several generations before trees are statistically significantly unbalanced. So, we should expect the growth of $E[S_{x_{\text{Std}}}]$ to be very slow, probably much slower than the growth of the average program size.

As all programs have the same shape, one-point crossover is able to select freely any node as a crossover point, like standard crossover. So, at generation $t = 0$

$$E[S_{x_{1\text{pt}}}] = E[S_{x_{\text{Std}}}] \approx d_{max}.$$

Since, in this situation, the offspring produced by one-point crossover will all have the same shape as their parents, $p_{1\text{pt}}(d_1|h_1,t) = p(d_1|h_1)$, $\forall h_1, t$, and the equation above will hold for the whole run.

So, with a population of fully balanced trees the amount of genetic material exchanged by standard crossover and one-point crossover at generation 0 grows linearly with the depth of the programs. However, it grows logarithmically with the size of the trees and the ratio between exchanged size and program size vanishes as the depth (size) of the programs grows:

$$\lim_{d_{max}\to\infty} \frac{E[S_r]}{S(h)} \approx \lim_{d_{max}\to\infty} \frac{d_{max}}{2^{d_{max}+1}-1} = 0.$$

In the case of uniform crossover the amount of genetic material exchanged is:

$$E[S_{x_{\text{Unif}}}] = p_s E[S(0,h_1)] = p_s(2^{d_{max}+1} - 1).$$

Since, in this situation, the offspring produced by uniform crossover will all have the same shape as their parents this equation will hold for the whole run.

The equation shows that $E[S_{x_{\text{Unif}}}]$ grows exponentially with the depth of the tree and for $p_s = 1/2$ can be considerably larger than $E[S_{x_{1\text{pt}}}] = E[S_{x_{\text{Std}}}] \approx d_{max}$. As a result,

$$\frac{E[S_{x_{\text{Unif}}}]}{S(0,d)} = p_s$$

is constant for any value of d_{max} and t.

3.3.4 Summary of Comparison

The theoretical results reported in the previous sections suggest that in a converged population of large trees, both standard crossover and one-point crossover would exchange very

small amounts of genetic material. If this is generally true, the *search performed by these operators becomes local and biased* (if only small subtrees are exchanged, they must be quite close to the leaves of the tree). Initially one-point crossover (especially the strict version) will exchange significantly more genetic material. So, initially the search performed by one-point crossover will be *global*. This may not be true for standard crossover.

Uniform crossover starts by swapping large subtrees near the root, like one-point crossover. So, initially the two operators may behave similarly, and *uniform crossover is initially a global search operator*. As the population starts converging, uniform crossover becomes more and more *local* in the sense that, the offspring it produces are progressively more similar to their parents. However, unlike standard and one-point crossover at this stage the search is, in some sense, *largely unbiased* as any node in the parents has the same chance of being inherited by the offspring. The search is not fully unbiased as the shape of the trees cannot be changed at this stage of a run.

As both one-point crossover and uniform crossover can be imagined as "interpolating" (i.e. exploring the space) between pairs of parents, it is very important that the initial population contains a diverse and representative sample of the whole search space of programs (or at least of a sufficiently large part of it) if we intend to perform a global search of it. This suggests that trees in the initial population should be large (with a size of the order of that we normally accept for end-of-run solutions) and diverse in terms of shape and nodes. Indeed in the experiments reported in [Poli and Langdon, 1997b] we observed a marked beneficial effect of using bigger initial tree depths. Also, like for the corresponding GA operators, mutation should be considered an integral part of the algorithm, to avoid premature convergence, whenever these operators are used.

4 Experimental Results

In order to verify the predictions of the theory developed in the previous section, we decided to perform a set of experiments in which the amount of genetic material exchanged by crossover was measured in real runs with different crossover operators. In the experiments we used different function sets so as to test the behaviour of the operators in the presence of functions with the same arity and functions with different arities (different arity function sets may reduce the size of the common region in one-point and uniform crossover and alter the node density function and the branching factor of programs).

The problem we selected is the Even-4 parity problem which consists of finding a combination of Boolean functions in a function set $\mathcal{F}$ (see below) and terminals from the set $\mathcal{T}$={`x1, x2, x3, x4`} which returns `true` if an even number of the 4 inputs `xi` is `true` and `false` otherwise. The fitness function for this problem is simply the number of entries of the truth table of the even-4 parity function correctly represented by each program. Two different function sets were used in different experiments: $\mathcal{F}_1$={`OR, AND, NOR, NAND`} and $\mathcal{F}_2$={`OR, AND, NOT`}. All functions have arity two, except `NOT` which has arity 1.

In our experiments we used a generational GP system with a population size of 1000, an initial tree depth of 8, the "ramped half-and-half" initialisation procedure [Koza, 1992], elitist selection, tournament selection with tournament size 7, a crossover probability of 0.7. All runs lasted for 50 generations (we *did not* stop the runs when the first solution was found). In the experiments we used standard crossover, one-point crossover and uniform crossover with $p_s = 0.5$.[3]

For each parameter setting we performed 20 independent runs. In each run we recorded: the average fitness of the population, the fitness of the best individual in the population, the average size of the individuals in the population, the average of the amount of genetic material exchanged by crossover during one generation, and the ratio between the average size and the average amount of genetic material exchanged. These parameters are plotted in Figures 4, 5 and 6 (a)–(b).

Figure 4 shows that the amount of genetic material exchanged by standard crossover is between 3 and 4% with the 2-arity function set $\mathcal{F}_1$. Initially this is slightly higher (10%) when the mixed arity function set $\mathcal{F}_2$ is used, but it drops to values below 5% within a few generations. This means that at all stages of a run standard crossover tends to produce offspring which inherit most of their code from one parent or the other, most of the times, i.e. it is a local search operator. Experiments with mutation did not show any significant difference with respect to this behaviour (the only difference being that average and max fitness did not converge).

Figure 5 shows that the amount of genetic material exchanged by one-point crossover is initially relatively large (18%) with $\mathcal{F}_1$, but that it drops very quickly below 5%. However, when the mixed arity functions are used, one-point crossover exchanges a lot more genetic material, starting at 59% and dropping below 10% only after 7 generations, never going below 7%. This happens because arity mismatches are much more likely to happen with $\mathcal{F}_2$ and therefore the common region will initially tend to remain quite small. These results suggest that at the beginning one-point crossover may be able to explore the search space globally, but only if mixed arity function sets are used. Experiments with mutation did not show any significant difference with respect to this behaviour (except that average and max fitness did not converge).

Figure 6 shows that uniform crossover has a significantly different behaviour. As predicted by the theory the amount of genetic material exchanged remains constantly 50% independently of the function set used. Figure 6 also reports the ratio between the average size and the average amount of ge-

[3] We also performed experiments, not reported, with the strict versions of one-point and uniform crossover as well as using fitness proportionate selection. Also experiments were performed with point mutation, in which a function in the tree is substituted with another function with the same arity or a terminal is substituted with another terminal with a constant probability (we used a mutation probability per node of 1/128) [Poli and Langdon, 1997c].

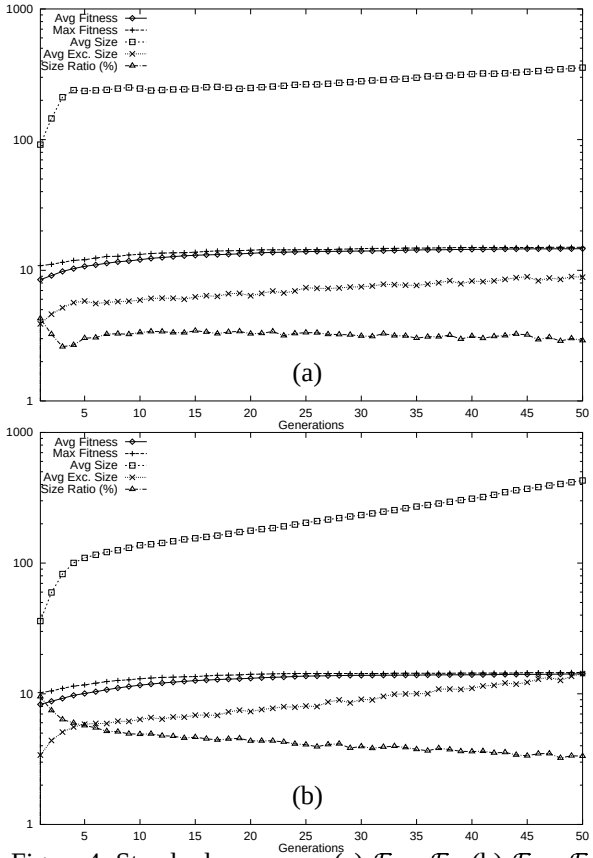

Figure 4: Standard crossover: (a) $\mathcal{F} = \mathcal{F}_1$, (b) $\mathcal{F} = \mathcal{F}_2$.

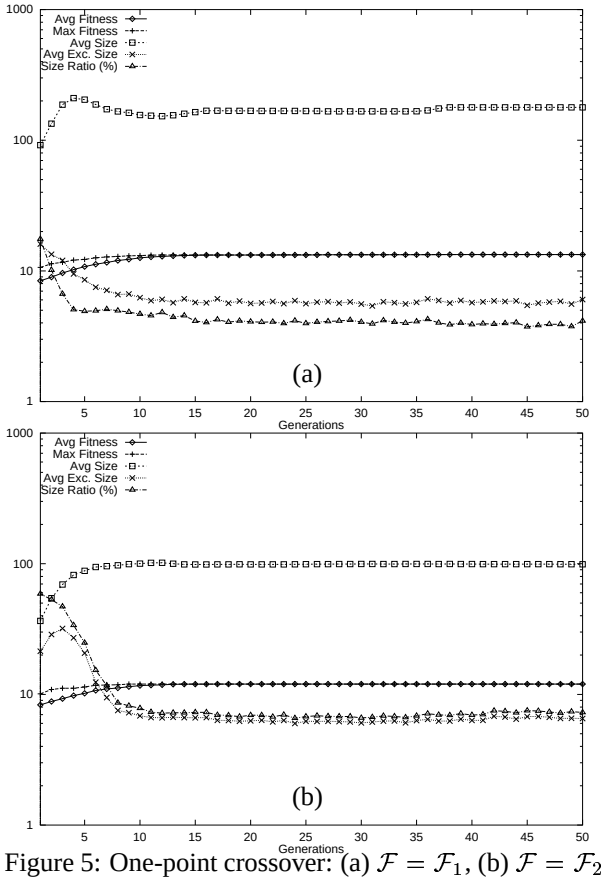

Figure 5: One-point crossover: (a) $\mathcal{F} = \mathcal{F}_1$, (b) $\mathcal{F} = \mathcal{F}_2$.

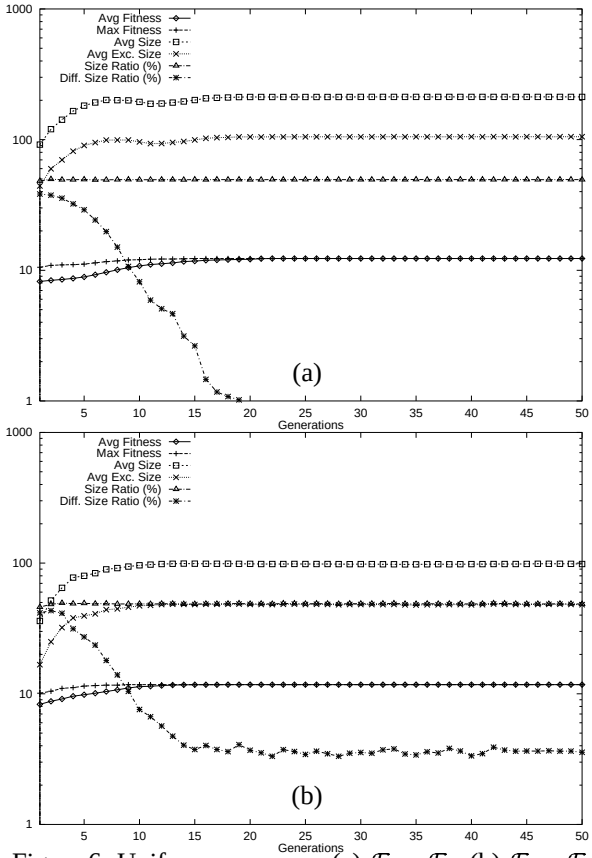

Figure 6: Uniform crossover: (a) $\mathcal{F} = \mathcal{F}_1$, (b) $\mathcal{F} = \mathcal{F}_2$.

netic material exchanged which was actually different in the two parents. As expected this is quite large at the beginning but decreases.[4] This means that uniform crossover is a global search operator which, thanks to the convergence of the population towards a common shape (i.e. when GP starts behaving like a GA [Poli, 1997]) becomes progressively beneficially local like all GA crossover operators.

5 Discussion and Conclusions

In this paper we have studied and compared the search properties of different crossover operators in genetic programming (GP) using probabilistic models and experiments which investigate the amount of genetic material exchanged between the parents to generate the offspring. These operators are: standard crossover, one-point crossover and uniform crossover, a new GA-inspired operator presented here for the first time.

Our analysis suggests that standard crossover is a local search operator which can only produce offspring which inherit most of their code from one parent most of the times. In

[4] The remaining 3–4% of effective genetic material exchanged by uniform crossover after generation 15 when a multi-arity function set is used is due to the fact that in some runs all individuals reached the same fitness level before a common shape had been found. In such runs genetic drift made the population converge to a common shape and the amount of effective genetic material exchanged approach zero long after generation 50.

addition standard GP crossover is biased as it is able to perform only certain types of local adjustments, typically very close to the leaves. These two facts indicate that standard crossover might not an ideal search operator as it cannot explore the search space quickly, it can only reach certain areas of the search space and can get stuck in local maxima. Perhaps this behaviour is the reason why crossover-based GP has seldom been shown to have a edge over mutation-based approaches.

One-point crossover, a form of crossover introduced in our recent research [Poli and Langdon, 1997c], is better as, in certain conditions, it is able to perform a global search of the search space, at the beginning of a run. However, the search becomes local and biased generation after generation.

Uniform crossover seems able to overcome these limitations. Like one-point crossover, uniform crossover starts global and becomes local, but in a different way, like GA operators do. Uniform crossover is much less biased as it allows any node in the parent trees to be transferred to the offspring with the same probability at any stage of the run.

We claim nothing about the performance differences in terms of computational effort to find a solution between the different operators discussed in this paper, although our experience suggests that one-point crossover performs at least as well as standard crossover on the even-parity problems [Poli and Langdon, 1997b]. However, we expect this to depend entirely on the particular fitness landscape at hand. The point we want to make with this paper is that, if GP is meant to search the whole space of programs rather than to behave like a set of parallel stochastic hill-climbers, then operators which allow global search must be used. Although we believe that uniform crossover is one such operator, we also think that the GP community will have to devote more work on finding other good such operators.

Acknowledgements

The authors wish to thank the members of the EEBIC group.

References

[Angeline, 1997] Angeline, P. J. (1997). Subtree crossover: Building block engine or macromutation? In *Proc. of GP'97*, pp. 9–17, Stanford University, CA, USA. Morgan Kaufmann.

[Chellapilla, 1997] Chellapilla, K. (1997). Evolutionary programming with tree mutations: Evolving computer programs without crossover. In *Proc. of GP'97*, pp. 431–438, Stanford University, CA, USA. Morgan Kaufmann.

[Harries and Smith, 1997] Harries, K. and Smith, P. (1997). Exploring alternative operators and search strategies in genetic programming. In *Proc. of GP'97*, pp. 147–155, Stanford University, CA, USA. Morgan Kaufmann.

[Ito et al., 1998] Ito, T., Iba, I., and Sato, S. (1998). Non-destructive depth-dependent crossover for genetic programming. In *Proc. of the First European Workshop on Genetic Programming, EuroGP'98*, Paris. Springer-Verlag. To appear.

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

[Koza, 1994] Koza, J. R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge Massachusetts.

[Langdon and Poli, 1997] Langdon, W. B. and Poli, R. (1997). An analysis of the MAX problem in genetic programming. In *Proc. of GP'97*, pp. 222–230, Stanford University, CA, USA. Morgan Kaufmann.

[Luke and Spector, 1997] Luke, S. and Spector, L. (1997). A comparison of crossover and mutation in genetic programming. In *Proc. of GP'97*, pp. 240–248, Stanford University, CA, USA. Morgan Kaufmann.

[O'Reilly and Oppacher, 1996] O'Reilly, U.-M. and Oppacher, F. (1996). A comparative analysis of GP. In Angeline, P. J. and Kinnear, Jr., K. E., editors, *Advances in Genetic Programming 2*, chapter 2, pp. 23–44. MIT Press, Cambridge, MA, USA.

[Poli, 1997] Poli, R. (1997). Is crossover a local search operator? Position paper at the Workshop on Evolutionary Computation with Variable Size Representation at ICGA-97. Available at http://www.ai.mit.edu/people/unamay/icga-ws.html.

[Poli and Langdon, 1997a] Poli, R. and Langdon, W. B. (1997a). An experimental analysis of schema creation, propagation and disruption in genetic programming. In *Proc. of ICGA'97*, Michigan State University, East Lansing, MI, USA. Morgan Kaufmann.

[Poli and Langdon, 1997b] Poli, R. and Langdon, W. B. (1997b). Genetic programming with one-point crossover. In *Proc. of the Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing.* Springer-Verlag London. Forthcoming.

[Poli and Langdon, 1997c] Poli, R. and Langdon, W. B. (1997c). A new schema theory for genetic programming with one-point crossover and point mutation. In *Proc. of GP'97*, pp. 278–285, Stanford University, CA, USA. Morgan Kaufmann.

[Poli and Langdon, 1998] Poli, R. and Langdon, W. B. (1997d). A review of theoretical and experimental results on schemata in genetic programming. In *Proc. of the First European Workshop on Genetic Programming, EuroGP'98,* Paris. Springer-Verlag. To appear.

[Rosca, 1996] Rosca, J. (1996). Generality versus size in genetic programming. In *Proc. of GP'96*, pp. 381–387, Stanford University, CA, USA. MIT Press.

[Rosca and Ballard, 1995] Rosca, J. and Ballard, D. H. (1995). Causality in genetic programming. In *Proc. of ICGA'95*, pp. 256–263, Pittsburgh, PA, USA. Morgan Kaufmann.

[Rosca and Ballard, 1996] Rosca, J. P. and Ballard, D. H. (1996). Complexity drift in evolutionary computation with tree representations. Technical Report NRL5, University of Rochester, Computer Science Department, Rochester, NY, USA.

[Russell and Norvig, 1995] Russell, S. J. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prendice Hall, Englewood Cliffs, New Jersey.

[Soule and Foster, 1997] Soule, T. and Foster, J. A. (1997). Code size and depth flows in genetic programming. In *Proc. of GP'97*, pp. 313–320, Stanford University, CA, USA. Morgan Kaufmann.