A

2 The Grammatical Inference Problem

The RAOC operator was originally developed as part of a novel approach to the problem of unsupervised learning of stochastic context-free grammars from corpora [5]. A *stochastic context-free grammar* (SCFG) is a variant of ordinary context-free grammar in which each grammar rule is associated with a probability, a real number in the range [0,1]. The set of production probabilities are called the *parameters* of the SCFG. An example of a simple SCFG is shown in figure 1, with the probability associated with each production given in parentheses. The SCFG generates the language $\{a^n b^n | n \geq 1\}$, where the probability of generating the string ab is 0.6, the probability of generating $aabb$ is 0.24, and so on.

S → A B	(1.0)
A → a	(0.6)
A → C S	(0.4)
B → b	(1.0)
C → a	(1.0)

Figure 1: SCFG for the language $a^n b^n$ $(n \geq 1)$

A *corpus* is a finite set of strings, where each string is associated with an integer representing its frequency of occurrence. An example of a corpus is shown in figure 2. Given a corpus as training data, the problem is to identify a SCFG that models the corpus data as accurately as possible, while generalizing appropriately to the wider language from which the sample strings are drawn.

ab	595
$aabb$	238
$aaabbb$	97
$aaaabbbb$	49
$aaaaabbbbb$	14
$aaaaaabbbbbb$	5

Figure 2: A corpus for the language $a^n b^n$

Our approach employs a genetic algorithm to search for the most likely grammar for a given corpus. Each genome encodes a complete set of parameters for a covering grammar consisting of all possible Chomsky normal form rules over a fixed set of terminal and nonterminal symbols. Since some of the parameters may be zero, a genome effectively picks out a subset of the rules: just those rules with non-zero probability. The fitness of the SCFG represented by a given genome is calculated by summing a measure of the likelihood of the corpus given the grammar and a measure of grammar size favouring smaller or simpler grammars over larger, more complex ones (see [5] for further details).

Experiments were conducted using a number of different crossover operators (definitions are given in section 3). The results of these experiments were unequivocal: RAOC consistently outperformed the other operators. Figure 3 shows a plot of maximum grammar fitness against number of generations for each of the crossover operators tested on the $a^n b^n$ problem. Not only does RAOC find the best solution overall, it also seems to home in on this solution very rapidly. Very similar outcomes were observed for a number of other grammar induction problems and this motivated the more general study described in the following sections.

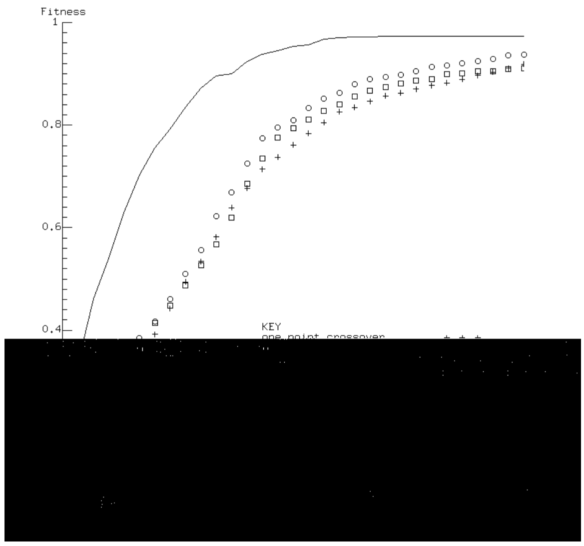

Figure 3: Comparison of different crossover operators on the $a^n b^n$ problem

3 Crossover Operators

A crossover operator C takes two genomes p_1 and p_2 and produces two offspring c_1 and c_2. Let p_{ij} denote the j_{th} bit of genome p_i, and assume the length of a genome (in bits) is *chromlen*. There are a variety of crossover operators that have been developed for different problems, the commonest of which are:

- *One-point crossover (1PC)*: Choose a random k such that $1 \leq k \leq$ *chromlen*. Define c_1 and c_2 by:

$$c_{1i} = \begin{cases} p_{1i} & 1 \leq i < k \\ p_{2i} & \text{otherwise} \end{cases}$$

  and

$$c_{2i} = \begin{cases} p_{2i} & 1 \leq i < k \\ p_{1i} & \text{otherwise} \end{cases}$$

- *Two-point crossover* (*2PC*): Choose random j, k such that $1 \leq j \leq k \leq chromlen$. Define c_1 and c_2 by:

$$c_{1i} = \begin{cases} p_{1i} \; 1 \leq i < j \\ p_{1i} \; k < i \leq chromlen \\ p_{2i} \text{ otherwise} \end{cases}$$

and

$$c_{2i} = \begin{cases} p_{2i} \; 1 \leq i < j \\ p_{2i} \; k < i \leq chromlen \\ p_{1i} \text{ otherwise} \end{cases}$$

- *α-crossover* [8]: This is often referred to as *parameterised uniform crossover* [7] and is really a family of operators, one for each $\alpha \in [0, 1]$. Let X_i be 1 with probability α, and 0 with probability $(1 - \alpha)$. α-crossover can then be defined by:

$$c_{1i} = \begin{cases} p_{1i} \text{ if } X_i = 1 \\ p_{2i} \text{ otherwise} \end{cases}$$

and

$$c_{2i} = \begin{cases} p_{2i} \text{ if } X_i = 1 \\ p_{1i} \text{ otherwise} \end{cases}$$

Note that α-crossover is symmetrical in the sense that, for $0 \leq \alpha <= 0.5$, α-crossover behaves in exactly the same way as $(1 - \alpha)$-crossover with c_1 and c_2 interchanged. 0.5-crossover is usually called *uniform crossover* (*UC*) [8], and is the most commonly used of this family of operators.

In order to define randomised and/or crossover, it is convenient to first define:

- *α-and/or crossover*: Like α-crossover, this is a family of operators, one for each $\alpha \in [0, 1]$. Let X_i be 1 with probability α, and 0 with probability $(1 - \alpha)$, then α-and/or crossover can then be defined by:

$$c_{1i} = \begin{cases} p_{1i} \wedge p_{2i} \text{ if } X_i = 1 \\ p_{2i} \vee p_{2i} \text{ otherwise} \end{cases}$$

and

$$c_{2i} = \begin{cases} p_{1i} \vee p_{2i} \text{ if } X_i = 1 \\ p_{1i} \wedge p_{2i} \text{ otherwise} \end{cases}$$

where $\wedge$ and $\vee$ are the Boolean operators *and* and *or*, respectively.

The effect of this operator is that with probability α, at each bit position the first child gets the logical *and* of the corresponding bits of the parents, while the second gets the logical *or*. Conversely, with probability $(1 - \alpha)$ the first child gets the logical *or* and the second gets the logical *and*. Like α-crossover, this operator is also symmetrical with respect to values of α above and below 0.5. We are now in a position to define the randomized operator:

F4 This is a “noisy” function with random noise added

$$
\begin{aligned}
\mathrm{P1} &= \bigwedge_{1 \leq i \leq 30} Q_i \\
\mathrm{P2} &= \mathrm{P1} \vee (Q_1 \wedge \bigwedge_{1 \leq i \leq 30} \overline{Q_i}) \\
\mathrm{P3} &= \mathrm{P2} \vee (Q_1 \wedge \bigwedge_{1 \leq i \leq 15} \overline{Q_i} \wedge \bigwedge_{16(15)1224615}
\end{aligned}
$$

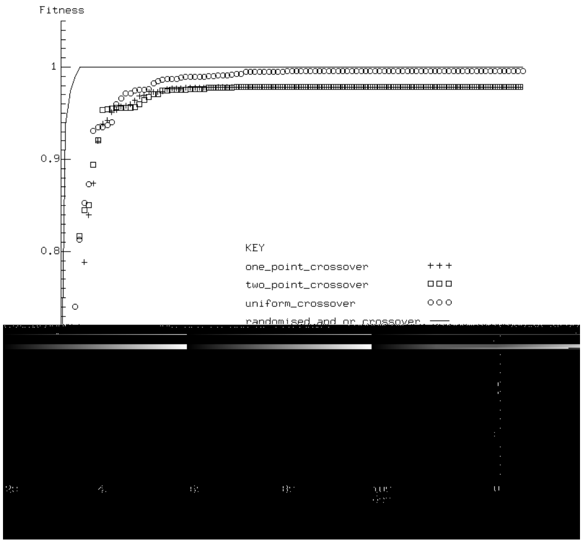

Figure 4: Comparison of different crossover operators on F8 (population size 256)

The performance of the different operators was measured with respect to the following criteria:

- *Hit Rate* (HR). This is the percentage of the 20 runs for a given combination of problem, population size, and crossover operator, in which at least one individual has a fitness of within 0.005 of the optimum value for that problem. In a sense this measures how often a solution to a problem is found.

- *Average Evaluations* (AE). This is the average number of evaluations needed to first obtain an individual satisfying the hit rate criterion. The average is only taken over runs in which such an individual is found. This measures how fast a solution is found, provided that one is found.

- *Average of Best Values* (AV). This is the average over the 20 runs of the fitness of the best individual at the end of each run. This measures how well the system does on the average.

7 Results

The results of the experiments described above bore out earlier observations. In general, RAOC performed best (as measured by HR, AE and AV) on more of the test problems than any of the other operators. This showed up more strongly as the population size increased. Table 1 shows the results for a population size of 256. (In the table, **bold** face type is used to indicate best performance.) At this population size, RAOC is the clear winner on 17 out of the 22 problems. The increase in performance is often quite

Size	1PC			2PC			UC			RAOC		
256	HR	AE	AV	HR	AE	AV	HR	AE	AV	HR	AE	AV
P1	**100**	2278	**1.0**	**100**	2198	**1.0**	**100**	1929	**1.0**	**100**	**650**	**1.0**
P2	**100**	2730	**1.0**	**100**	2714	**1.0**	**100**	2558	**1.0**	**100**	**865**	**1.0**
P3	80	2433	0.997	75	2805	0.996	95	2525	0.999	**100**	**727**	**1.0**
P4	85	3106	0.997	75	3339	0.996	95	2431	0.999	**100**	**779**	**1.0**
P5	85	3036	0.997	75	2788	0.996	80	2616	0.997	**100**	**798**	**1.0**
P6	85	2505	0.997	90	2977	0.998	85	2710	0.997	**100**	**838**	**1.0**
F1	**100**	1832	**0.0**	**100**	1686	**0.0**	**100**	1411	**0.0**	**100**	**436**	**0.0**
F2	75	3096	-0.005	90	3038	-0.002	95	**1524**	-0.001	**100**	1641	**-0.0**
F3	**100**	3423	**25.0**	**100**	3028	**25.0**	**100**	3014	**25.0**	**100**	**598**	**25.0**
F4	0	∞	-1.76	0	∞	-1.395	0	∞	-0.58	[redacted]	**938**	**-0.08**
F5	**100**	1388	**499.002**	**100**	1339	**499.002**	**100**	1545	**499.002**	**100**	**1184**	**499.002**
SF1	**100**	1784	**0.0**	**100**	1799	**0.0**	**100**	1468	**0.0**	**100**	**426**	**0.0**
SF2	90	3171	-0.003	75	1934	-0.003	**95**	1608	-0.001	90	**1206**	**-0.001**
SF3	95	**4193**	24.95	**100**	4991	**25.0**	95	6101	24.95	20	7499	24.2
SF4	0	∞	-2.072	0	∞	-1.519	0	∞	-0.555	**100**	**951**	**-0.043**
SF5	**100**	**1493**	**499.002**	**100**	1693	**499.002**	95	2892	499.002	75	4225	498.901
F6	**100**	3317	**0.0**	**100**	3137	**0.0**	**100**					

Size 256	RUC			0.75-AOC			RAOC		
	HR	AE	AV	HR	AE	AV	HR	AE	AV
P1	**100**	2053	**1.0**	**100**	879	**1.0**	**100**	**650**	**1.0**
P2	**100**	2436	**1.0**	**100**	1023	**1.0**	**100**	**865**	**1.0**
P3	80	2461							

References

[1] Ackley, D.H. and Littman, M.L. (1990) Learning from natural selection in an artificial environment. *Proceedings of the International Joint Conference on Neural Networks,* Vol. 1, pp. 189-193.

[2] Davis, L. (1991) Bit-climbing, representational bias, and test suite design. *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, VA, pp. 18-23.

[3] De Jong, K.A. (1975) *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems,*