A New Crossover Operator for Rapid Function Optimisation Using a Genetic Algorithm.

Bill Keller and Rudi Lutz
School of Cognitive and Computing Sciences
The University of Sussex
email: {billk,rudil}@cogs.susx.ac.uk

Abstract

This paper describes experiments with a new genetic operator, *randomised and/or crossover* (RAOC), which shows good performance on a range of function optimisation problems. Originally motivated by work on grammatical inference, RAOC was observed to outperform more traditional crossover operators both in terms of its ability to locate global optima in the search space and in terms of its speed (measured as the number of function evaluations taken to reach a solution). This paper describes an extensive empirical study comparing RAOC with a number of more standard operators on a range of function optimisation problems. The results of this study bear out the earlier observations and demonstrate that RAOC is of wider applicability than to the original grammatical inference problem which motivated it.

1 Introduction

Much of the power of genetic algorithms (GAs) derives from the use of appropriate genetic operators for recombining or *crossing* the encodings of candidate solutions. Crossover operators allow for the rapid identification of good regions of the search space by combining successful characteristics of partial solutions in novel ways. It is not surprising that researchers have investigated a variety of crossover operators in an attempt to improve the performance of GAs.

We have investigated the behaviour of a new crossover operator, *randomised and/or crossover* (RAOC), which was introduced in the context of a GA based approach to the problem of stochastic grammatical inference. RAOC was observed consistently to outperform more conventional operators in this domain. In the present paper, RAOC is properly defined and compared to several other operators on a range of function optimisation problems of the kind examined in other studies. The results of this comparison demonstrate that RAOC is of wider applicability than to the grammatical inference problem which originally motivated it. In general, RAOC outperforms other operators both in terms of its ability to find global solutions in the search space and the speed with which these solutions are identified.

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.

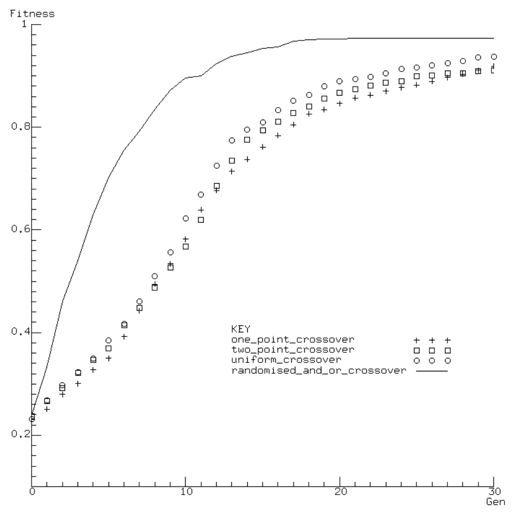

Figure 3: Comparison of different crossover operators on the $a^n b^n$ problem

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 $1 \leq j \leq k \leq \mathit{chromlen}$. Define c_1 and c_2 by:

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

  and

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

- *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:

- *Randomised and/or crossover* (*RAOC*): This is defined in terms of the family of α-and/or operators. Each time two parents are chosen to reproduce, pick a value of α at random (uniform distribution) and then carry out α-and/or crossover. Note that α varies with each mating.

For comparison purposes we also define the following operator:

- *Randomised uniform crossover* (*RUC*): This is defined in terms of the family of α-crossover operators. Each time two parents are chosen to reproduce, pick a value of α at random (uniform distribution) and then carry out α-crossover. Again, note that α varies with each mating.

4 The Test Problems

In order to compare the various crossover operators we used a set of 22 "standard" optimisation problems, which we will refer to as F1-F8, SF1-SF8 and P1-P6. F1-F5 are five problems originally due to De Jong [3]. F6 is Ackley's function [1], F7 is the Schubert Function [9] and F8 is Schaffer's sine envelope sine wave function [6]. Davis [2] has suggested that GAs can take advantage of the position of the optima for these problems relative to Hamming "cliffs", and suggested that "shifted" versions of the problems may provide a better test. Accordingly, SF1-SF8 are shifted versions of F1-F8, where the process of shifting will be described more fully in section 5. Finally, P1-P6 are 6 n-peak problems used in a recent study of multi-point crossover operators by De Jong and Spears [4].

The objective functions for the 22 problems are described next[1]:

F1 This function has a single optimal value of 0.0, and is defined by:

$$-\sum_{i=1}^{3} x_i^2$$

for $-5.12 \leq x_i \leq 5.12$

F2 This function has a single optimal value of 0.0, and is defined by

$$-100(x_1^2 - x_2)^2 - (1 - x_1)^2$$

for $-2.048 \leq x_i \leq 2.048$

F3 This has a single optimal value of 25.0, and is defined by:

$$-\sum_{i=1}^{5} intof(x_i)$$

for $-5.12 \leq x_i \leq 5.12$

[1]Note that we have inverted the normal definitions of these functions as our GA performs function maximisation rather than minimisation.

F4 This is a "noisy" function with random noise added to its values every time it is evaluated. Without the noise it has a single optimal value of 0.0 at $x_i = 0,\ 1 \leq i \leq 30$. It is defined by:

$$-(\sum_{i=1}^{30} i x_i^4 + Gauss(0,1))$$

for $-1.28 \leq x_i \leq 1.28$ where $Gauss(\mu, \sigma)$ is random Gaussian noise with mean μ, and standard deviation σ.

F5 This function has an optimal value of 499.002, but there are numerous suboptimal maxima. It is defined by:

$$500 - \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{1+i+(x_1-a_i)^6+(x_2-b_i)^6}}$$

for $-65.536 \leq x_i \leq 65.536$, where a_i and b_i are defined by:

$$a_i = 16((i \bmod 5) - 2)$$

and

$$b_i = 16(\lfloor i/5 \rfloor - 2)$$

F6 This function has a maximum value of 0.0, and is defined by:

$$20e^{\left(-0.2\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}\right)} + e^{\left(\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)\right)} - 20 - e$$

for $-30.0 \leq x_i \leq 30.0$.

F7 This function has a global maximum of 186.731, and is defined by:

$$-\prod_{i=1}^{2}\sum_{j=1}^{5} j\cos((j+1)x_i + j)$$

for $-10 \leq x_i \leq 10$. Within this range there are 760 local maxima, of which 18 are global maxima.

F8 This function has a single global maximum of 1.0, and is defined by:

$$0.5 - \frac{(\sin(\sqrt{x_1^2 + x_2^2}))^2 - 0.5}{1 + 0.001(x_1^2 + x_2^2)^2}$$

for $-100 \leq x_i \leq 100$. It should be noted however that there are numerous local maxima.

Problems SF1-SF8 are defined according to the functions F1-F8 given above. The difference lies is in the way in which the genomes are decoded, and will be explained later in section 5. Following De Jong and Spears [4], we define P1-P6 as follows:

$$
\begin{aligned}
\mathrm{P1} &= \bigwedge_{1 \le i \le 30} Q_i \\
\mathrm{P2} &= \mathrm{P1} \vee (Q_1 \wedge \bigwedge_{1 \le i \le 30} \overline{Q_i}) \\
\mathrm{P3} &= \mathrm{P2} \vee (Q_1 \wedge \bigwedge_{1 \le i \le 15} \overline{Q_i} \wedge \bigwedge_{16 \le j \le 30} Q_j) \\
\mathrm{P4} &= \mathrm{P3} \vee (\overline{Q_1} \wedge \bigwedge_{1 \le i \le 15} Q_i \wedge \bigwedge_{16 \le j \le 30} \overline{Q_j}) \\
\mathrm{P5} &= \mathrm{P4} \vee (Q_1 \wedge \bigwedge_{1 \le i \text{ odd} \le 30} \overline{Q_i} \wedge \bigwedge_{1 \le j \text{ even} \le 30} Q_j) \\
\mathrm{P6} &= \mathrm{P5} \vee (\overline{Q_1} \wedge \bigwedge_{1 \le i \text{ odd} \le 30} Q_i \wedge \bigwedge_{1 \le j \text{ even} \le 30} \overline{Q_j})
\end{aligned}
$$

These are Boolean satisfiability problems defined over the 30 propositional variables $Q_1 \ldots Q_{30}$. We associate with each problem P_m a function h_m from valuations (assignments of truth values to variables) to [-1, 1] such that $h_m(V) = 1$ if and only if P_m is satisfied under valuation V. The function h_m is defined as follows. Let E_m $(1 \le m \le 5)$ denote that Boolean expression such that $\mathrm{P}_{m+1} = \mathrm{P}_m \vee \mathrm{E}_m$. So, for example, P2 = P1 $\vee$ E1, where

$$\mathrm{E1} = (Q_1 \wedge \bigwedge_{1 \le i \le 30} \overline{Q_i})$$

and so on. Note that each of the E_m is actually unsatisfiable. Now, define:

$$score(X, V) = \frac{T(X, V) - F(X, V)}{30}$$

where $T(X, V)$ is defined as the number of literals (propositional variables or their negation) in expression X that are *true* under valuation V and $F(X, V)$ is defined as the number of literals in X that are *false* under valuation V.

Finally, define:

$$
\begin{aligned}
h_1(V) &= score(\mathrm{P}_1, V) \\
h_{m+1}(V) &= \max(h_m(V), score(\mathrm{E}_m, V))
\end{aligned}
$$

The function h_m has the property that it has a single, global maximum value of 1, corresponding to assigning *true* to all the variables Q_i. However, there are $m - 1$ local maxima corresponding to nearly satisfying the extra, unsatisfiable expressions $\mathrm{E}_1, \ldots, \mathrm{E}_{m-1}$.

5 The Genetic Algorithm

For the study we used a distributed GA having a structured population consisting of a two dimensional toroidal grid with $N \times N$ squares. Each grid square 'contained' a single individual. The algorithm involved repeatedly executing the following mating strategy:

- select an individual at random, taking the fittest of its immediate neighbours as its partner;
- produce two children in the usual way by crossover and mutation and evaluate according to the objective function;
- replace the weakest parent by the fittest child.

Note that because this strategy does not involve converting fitness values to probabilities, we could work with the values of the function to be optimised directly. In general, we did not normalize function values as part of the decoding process.

For problems F1–F8 we used 30 bits to encode each parameter. For example, F4, which has 30 parameters, had a total chromosome length of 900 bits. The bits representing each parameter were decoded as two's complement integers, and then scaled to bring the resulting number into the required range. For the "shifted" problems, SF1-SF8, the 30 bits representing each parameter were first decoded just as for F1-F8. However, 10% of the absolute value of the result was then taken away. If the result fell below the permitted range, then the value was 'wrapped around' to the top of the range by the same amount that it had gone below (see [2] for more details). The six n-peak problems, P1-P6 were all carried out with a chromosome length of 30 bits (i.e. 1 bit per Boolean variable) with the obvious decoding strategy of 1 representing *true*, and 0 representing *false*.

6 The Experiments

We compared the performance of the different crossover operators on each of the test problems described above. Initially, this was done with the following crossover operators:

- 1-point crossover
- 2-point crossover
- uniform crossover
- randomised and/or crossover

A total of 20 runs, each of 100 generations were carried out for each crossover operator on each problem (where a generation was arbitrarily defined as 128 function evaluations). This was repeated for different population sizes (4, 16, 64, 256 and 1024). For all runs, the mutation rate was fixed at $1/chromlen$ At the start of each run, the population was initialized randomly: each bit in each genome was chosen to be either 1 or 0 with equal probability.

Further runs were also carried out to compare randomized and/or crossover with randomized uniform crossover and α-and/or crossover, with α fixed at 0.75. This was done to see whether the advantages of RAOC come from random variations around uniform crossover (since 0.5-and/or crossover is the same as uniform crossover) and to rule out the possibility that we would do as well with 0.75-and/or crossover (since 0.75 is the effective mean value of α in RAOC).

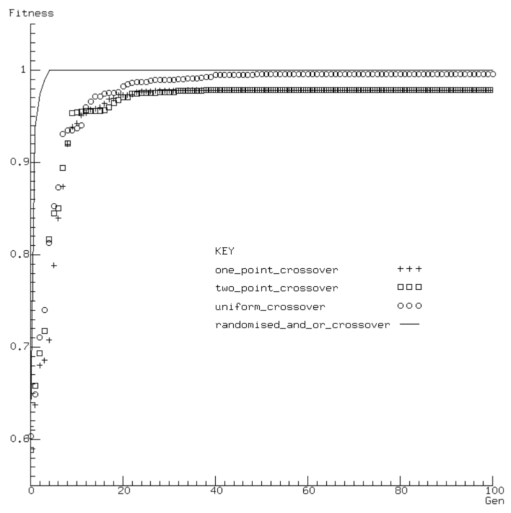

Figure 4: Comparison of different crossover operators on F8 (population size 256)

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	**100**	**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**	3058	**0.0**	**100**	**616**	**0.0**
SF6	**100**	3341	**0.0**	**100**	3080	**0.0**	**100**	3000	**0.0**	**100**	**659**	**0.0**
F7	**100**	3211	**186.731**	**100**	**2915**	**186.731**	95	4187	186.731	90	4195	186.706
SF7	**100**	3348	186.731	**100**	**3074**	**186.732**	95	4178	186.729	**100**	3741	186.731
F8	65	2804	0.985	45	3040	0.976	85	3115	0.993	**100**	**549**	**1.0**
SF8	45	4089	0.976	55	3119	0.98	95	3430	0.998	**100**	**687**	**1.0**

Table 1: Results for population size 256

dramatic, with an increased hit rate and a much lower average number of evaluations. This is illustrated in Figure 4 which shows the graph of fitness (for the best of each generation) for the various operators on problem F8.

On the other hand, it should be noted that there are a few problems on which RAOC actually performs *least* well. In particular this is the case for SF3, SF5, and F7 (it also fares less well on SF7). It is not yet clear why these problems pose such difficulties for RAOC. One possibility that we are currently investigating is that the operator performs rather like a very efficient stochastic hill-climber. It is notable, for example, that the problems on which it does not do very well seem to have plateaus, and we suspect that it does badly in landscapes with many plateaus.

As stated earlier, we also compared RAOC with 0.75-and/or crossover and randomised uniform crossover. The results are given in table 2. It can clearly be seen from this table that although randomised uniform, and 0.75-crossover occasionally outperform randomised and/or crossover, in general randomised and/or crossover performs best. This would seem to indicate that it is not simply the use of a crossover operator "centered" around uniform crossover that gives the improvement. nor is it simply the fact that randomising the parameter averages out at 0.75.

Size	RUC			0.75-AOC			RAOC		
256	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	0.997	**100**	1160	**1.0**	**100**	**727**	**1.0**
P4	75	2742	0.996	**100**	1122	**1.0**	**100**	**779**	**1.0**
P5	75	2830	0.996	**100**	1135	**1.0**	**100**	**798**	**1.0**
P6	65	2816	0.994	**100**	1136	**1.0**	**100**	**838**	**1.0**
F1	**100**	1366	**0.0**	**100**	574	**0.0**	**100**	**436**	**0.0**
F2	95	1684	-0.001	**100**	**1284**	-0.0	**100**	1641	**-0.0**
F3	**100**	3081	**25.0**	**100**	943	**25.0**	**100**	**598**	**25.0**
F4	0	∞	-0.641	**100**	1590	**-0.045**	**100**	**938**	-0.08
F5	95	1295	498.992	95	**1021**	498.982	**100**	1184	**499.002**
SF1	**100**	1635	**0.0**	**100**	585	**0.0**	**100**	**426**	**0.0**
SF2	80	2217	-0.002	**100**	1722	**-0.0**	90	**1206**	-0.001
SF3	**85**	**6812**	**24.85**	60	9900	24.6	20	7499	24.2
SF4	0	∞	-0.799	**100**	1477	-0.121	**100**	**951**	**-0.043**
SF5	**90**	2882	**499.001**	85	**2865**	499.001	75	4225	498.901
F6	**100**	2899	**0.0**	**100**	987	**0.0**	**100**	**616**	**0.0**
SF6	**100**	3027	**0.0**	**100**	945	**0.0**	**100**	**659**	**0.0**
F7	85	**3977**	186.665	**100**	4007	**186.731**	90	4195	186.706
SF7	90	4264	186.716	**100**	4669	186.731	**100**	**3741**	**186.731**
F8	80	2655	0.991	**100**	884	**1.0**	**100**	**549**	**1.0**
SF8	75	3611	0.989	**100**	1067	**1.0**	**100**	**687**	**1.0**

Table 2: Comparison with RU and 0.75-AOC for population size 256

8 Conclusions

Our results show that randomised and/or crossover consistently outperforms 1-point, 2-point, and uniform crossover on nearly all of the test problems. This confirms the results of earlier experiments on the grammatical inference problem. The improvement in performance is often quite dramatic, with both an increase in the success rate, and an increase in the speed at which solutions are found. This suggests that RAOC should be considered as a possible alternative to other crossover operators whenever such a choice is feasible.

9 Acknowledgements

We would like to thank Malcolm McIllhagga and Rob Duncan for use of their workstations. We would also like to thank Phil Husbands, and Inman Harvey for various suggestions relating to this work.

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,* Doctoral Thesis, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor.

[4] De Jong, K.A. and Spears, W.M. (1990) An analysis of the interacting roles of population size and crossover. *Proc. 1st Int'l Conf. on Parallel Problem Solving from Nature,* Dortmund, Germany, October 1990.

[5] Keller,W. and Lutz, R. (1996) Learning stochastic context-free grammars from corpora using a genetic algorithm. Cognitive Science Research Paper 444, School of Cognitive and Computing Sciences, The University of Sussex. To appear in *Proceedings of the 3rd International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA-97).*

[6] Schaffer, J.D., Caruana, R.A., Eshelman, L.J., and Das, R. (1989) A study of control parameters affecting online performance of genetic algorithms for function optimization. *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishing.

[7] Spears, W.M. and De Jong, K.A. (1991) On the virtues of parameterized uniform crossover. *Proceedings of the Fourth Internationa Conference on Genetic Algorithms*, San Diego, VA, pp.237-244.

[8] Syswerda, G. (1989) Uniform crossover in genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishing.

[9] Torn, A. and Zilinskas, A. (1989) *Global Optimization.* Springer-Verlag, Berlin.