Survival of the Sickest: A Site-Specific Recombination Operator for Accelerated Function Optimization

Stephen Drake Phil Husbands

School of Cognitive and Computing Sciences
University of Sussex
Brighton, UK, BN1 9QH
Email: {stevedr, philh}@cogs.susx.ac.uk

Abstract

This paper describes experiments with a new crossover operator which is based on the mechanism of biological site-specific recombination. By using hill-climbing to gather additional information about the fitness landscape, it increases the constructive power of crossover. The nature of the operator calls for a somewhat unusual selection strategy which, in contrast with traditional methods, automatically selects a relatively unfit member of the population to be a parent. When applied to difficult continuous-variable function optimization problems, the operator is seen to perform better than standard one-point crossover in terms of quality of solutions found and its speed in finding them (as measured by the number of function evaluations carried out).

1 Introduction

A considerable body of literature has amassed which attempts to explain crossover's place among evolutionary algorithms, and research has yielded increasingly sophisticated analyses since the time when conventional wisdom decreed that recombination was the primary engine of optimization while mutation was regarded as being somehow 'less powerful'. However, a complete and accurate analysis is still proving elusive, with much of the research either posing more questions than it answers (questions such as: is crossover redundant? Is crossover simply macromutation?) or contradictory, in the case of Holland's Schema Theorem versus Goldberg's Building Blocks Hypothesis [11]. There appears to be scant work which marries investigation of such theoretical bases as those mentioned above to the efficacy of crossover as observed empirically (other than evaluating its performance under variations of parameter values (for example, [4, 9]) or in recombination strategy itself, such as in [10, 12]). Moreover, it can be illustrated (see section 5) that, very often, standard one-point crossover constructs better solutions for only a relatively short length of the time; it very quickly becomes redundant, and a GA can then only rely on mutation to explore different areas of the search space. Thus the immediate motivation for this paper was to improve the performance of crossover by increasing the length of time for which it constructs better solutions. An 'intelligent' operator is proposed, which incorporates a generic – that is, problem non-specific – mechanism inspired by biological site-specific recombination. Biological site-specific recombination works by mediating proteins that bind specific target sequences in a strand of DNA, and catalyse recombination at those positions [6].

2 An Artificial Site-Specific Recombination Operator

This operator incorporates a hill-climbing technique into its mechanism, and can so be viewed as a hybrid. Hybrid approaches have traditionally involved employing problem-specific search techniques, or relegating the additional technique to a supplementary capacity by bolting it onto the end of the GA: for example, applying some local search to the fittest individual (or, indeed, every individual) at the end of a generation, or some hill-climbing at the end of a run. However, in this site-specific recombination (S-SR) operator, hill-climbing plays a much more important rôle, and can be seen as imbuing the operator with its 'intelligence'.

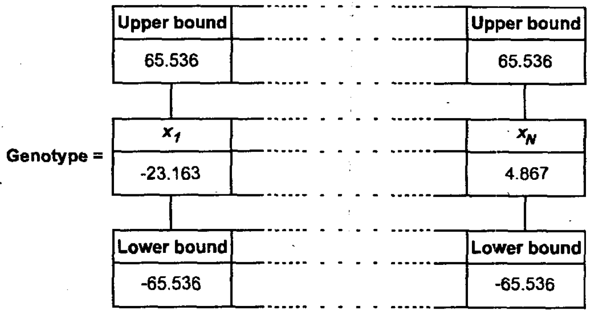


Figure 1: Allele bounds

2.1 Allele Bounds

The structure of an individual in the population has been modified to incorporate a higher and a lower 'bound' for every gene of its genotype, as illustrated in figure 1. (These are initialised respectively to be the higher and lower values of the evaluation function's variable range.) During the random generation of a population, and after every mating cycle, an approximation to random-mutation hill-climbing (RMHC) is applied to a specified percentage of an individual's genes. RMHC mutates random loci in the current best genotype until a maximum number of evaluations have been performed (or until the optimum string has been found), only retaining mutations which lead to an equal or higher fitness [8]. This strategy was adapted to accommodate real-valued genotypes and the concept of allele bounds.

The values of a gene's allele bounds are altered according to the effects of hill-climbing on that gene: if incrementing a gene's value by a constant STEP_SIZE brings about a fitness increase, its lower bound is set to the gene's new value; conversely, if there occurs a decrease in fitness, the gene, together with its upper bound, is reset to its previous value. (The above operates in reverse if hill-climbing decrements a gene's value; incremental hill-climbing is applied first; if this effects a fitness decrease, the gene's value is reset and decremental hill-climbing is applied.) The algorithm is represented by the following pseudo-code:

```
REPEAT N TIMES
 Select random Gene in CurrentGenotype
 Increment Gene by StepSize
 IF CurrentGenotype is fitter OR of equal fitness THEN
  Lower bound of Gene = Gene
 ELSE IF CurrentGenotype is less fit THEN
  Reset Gene to original value
  Upper bound of Gene = Gene
  Decrement Gene by StepSize
  IF CurrentGenotype is fitter OR of equal fitness THEN
   Upper bound of Gene = Gene
  ELSE IF CurrentGenotype is less fit THEN
   Reset Gene to original value
   Lower bound of Gene = Gene
  END IF
 END IF
END REPEAT
```

2.2 Selection and Recombination

Mating was restricted to small neighbourhoods of individuals as detailed in section 3.2. In contrast to traditional methods of selection, the least fit member of a neighbourhood is selected as the first parent. (Preliminary experiments were carried out, of which each set of 50 runs consecutively selected the next fittest neighbour to be the first parent. Results showed that the worse the first parent is, the more effective the S-SR operator proves to be – unsurprisingly, given the nature of the operator, since there is more scope for improvement in an unfit individual.)

A set of allele bounds determines the general direction in the search space which an individual takes by defining the range in which the bounds' gene may find a profitable point. Moreover, if an increase in a gene's value due to hill-climbing effects an increase in the individual's fitness then presumably moving in the immediately opposite direction would prove detrimental to the individual's fitness. The S-SR operator looks for a mate which contains genes whose alleles lie between the corresponding gene's allele bounds in the first parent. The neighbour containing the most such 'desirable' genes is selected as a mate and one offspring is produced which consists of all these desirable genes and the remaining genes of the first parent.

Thus the pseudocode for S-SR selection and crossover is as follows:

```
SELECTION:
 Neighbourhood = array of six vicinal population members
 MostDesirableNo is the greatest number of desirable genes found in any
 neighbour so far = 0
 CurrentFave is the current neighbour with the greatest number of desirable
 genes
 CurrentDesirableNo is the current number of desirable genes found in
 current neighbour
 Sort Neighbourhood in order of increasing fitness
 Parent1 = Neighbourhood[1]
 FOR i from 2 to NeighbourhoodSize DO
  CurrentDesirableNo = 0
  FOR j from 1 to GenotypeLength DO
   IF jth gene of Neighbourhood[i] > jth lower allele bound AND < jth
     upper allele bound of Parent1 THEN
    CurrentDesirableNo = CurrentDesirableNo + 1
     Append value of j to Neighbourhood[i]'s DesirableGenesList
   END IF
  END FOR
  IF CurrentDesirableNo ≥ MostDesirableNo THEN
   MostDesirableNo = CurrentDesirableNo
   CurrentFave = Neighbourhood[i]
  END IF
 END FOR
 Parent2 = CurrentFave

CROSSOVER:
 Child = Parent1
 FOR i from 1 to MostDesirableNo DO
  j = value of ith element in Parent2's DesirableGenesList
  ith gene of Child = jth gene of Parent2
 END FOR
```

3 Experiments

3.1 Functions Used

In order to assess the performance of S-SR in relation to standard one-point crossover, various optimization problems were used, as represented by the following functions.

The first three, to be minimized, are the last in a suite of five functions originally constructed by De Jong [3] and which were intended to represent common difficulties among optimization problems in an isolated manner.

F1 De Jong's F3 has a single optimal value of 0, and is defined by

$$\sum_{1}^{5} \text{integer}(x_i) \quad \text{for } -5.12 \le x_i \le 5.12$$

F2 De Jong's F4 is 'noisy': random Gaussian noise is added to its value every time it is evaluated, and is defined by

$$\sum_{i=1}^{30} ix_i^4 + \text{Gauss}(0,1) \quad \text{for } -1.28 \le x_i \le 1.28$$

F3 De Jong's F5 has a global minimum of 0.002 - although there are many suboptimal minima - and is defined by

$$0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6} \quad \text{for } 65.536 \le x_i \le 65.536$$

The following two functions, to be maximized, are taken from a set due to Baluja [1]. Both have a global maximum of 100,000 at the origin, where the variable ranges are $-2.56 \le x_i \le 2.56$. A small constant $C = 1e\text{-}5$ is added to the denominator of the functions to avoid division by zero.

F4 Trial solutions to this function are affected by high epistasis; that is, there exists a high degree of inter-dependence between loci: the variables in the first portions of the solution string have a large influence on the quality of the rest of the solution – small changes in their values can cause large changes in the evaluation of the solution [1].

$$\frac{1}{C + \left| |y_1| + \sum_{i=2}^{100} |y_i| \right|} \quad \text{where } y_1 = x_1 \text{ and } y_i = x_i + \sin(y_{i-1})$$

F5 In contrast to the previous function, trial solutions to this function are affected by low epistasis; that is, there exists a low degree of inter-dependence between loci.

$$\frac{1}{C + \sum_{i=1}^{100} |0.024(i+1) - x_i|}$$

F6 To simulate a multi-peak problem the following constrained function was defined by Keane [5].

$$\frac{\left| \sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i) \right|}{\sqrt{\sum_{i=1}^{n} ix_i^2}} \quad \text{for } 0 < x_i < 10,\ i = 1,\ldots,n$$

$$\text{subject to } \prod_{i=1}^{n} x_i > 0.75 \text{ and } \sum_{i=1}^{n} x_i < \frac{15n}{2}.$$

Details of the penalty function used with the above function can be found in [5].

F7 This is the most complex in a set of multimodal functions q constructed by Corana et al. [2]. These functions, to be minimized, are obtained by defining a regular, rectangular grid in a space R^n and a set of open, non-overlapping, rectangular subdomains, each centred around the node of the grid. The definition of the function $q_n(\mathbf{x})$ of n variables is a rectangular subdomain of R^n centred at the origin and including several nodes of the grid. The function q_n is a paraboloid with axes to the coordinate directions except inside the open subdomains mentioned above, where it is constant with a value lower than the lowest value of q_n computed on the boundary of each subdomain. These subdomains are like a set of 'holes' representing local minima of q_n and introducing strong discontinuities in the test function. The total number of local minima of the test function $q_n(\mathbf{x})$ is 10^{5n}-1, and for the purposes of this comparison, n = 10. The absolute minimum lies on the origin and has value 0. (A comprehensive definition of the function can be found in [2].)

3.2 The Genetic Algorithm

A geographically distributed GA was used, with the population spread across a two-dimensional toroidal grid of size 15×15, each cell of which contained a single individual. Mating was restricted to small groups of individuals which were generated as follows:

- Select a grid cell at random.
- Build a neighbourhood of six individuals around the current cell by, for each neighbour, generating x- and y-distances from the current cell dependent upon a binomial approximation to a Gaussian distribution where n = 4 and p = 0.85. Thus, for individuals in cells at consecutive distances away from the current cell, the probabilities of being selected are 0.52, 0.37, 0.1 and 0.01. (The direction of the distances

– up, down, left or right – are chosen at random.)

- Rank the neighbourhood members according to their fitness.

The selection of parent individuals differed in strategy depending on which method of crossover was to be applied: in anticipation of site-specific recombination, the least fit member of the neighbourhood was chosen automatically as the first parent, and a mate selected according to the selection strategy described in section 2.2. However, if conditions precluded S-SR (that is, if none of the first parent's neighbours contained desirable genes, or if the number of desirable genes in a neighbour equalled the length of the genotype) standard one-point crossover was used. In this instance, parents were selected according to a linear selection function favouring the fitter individuals. After mating, in both cases, the fittest child replaced a member of the neighbourhood selected according to the inverse of this function.

3.3 Mutation

Mutation was applied after crossover with a probability of 0.01, and the experiments were carried out essentially using two different methods with both one-point crossover and S-SR. Method 1 incorporated a fairly standard method of real-valued mutation: in 90% of mutations, a gene was selected randomly, and mutated according to a uniform distribution centred on the current value and of width 10% of the gene range. In the remaining 10%, the selected gene was simply assigned a random value from within its entire range. Method 2 granted genotypes a greater chance of exploring more remote areas of the search space by assigning a randomly selected gene a random value within its range (while resetting the gene's allele bounds to those parameters respectively for S-SR) 100% of the time. However, resetting the allele bounds could mislead S-SR. Recall that the neighbour with the most desirable genes is chosen to be the second parent; with a gene's allele bounds reset, S-SR would now identify any value for that gene's equivalent in a potential mate as desirable. Therefore, it is possible that a neighbour with several desirable genes – plus one misleadingly desirable gene – would be chosen in favour of a neighbour with one less desirable gene, but which ultimately would have proved more profitable. To help rectify this, a third mutation method, 2.1, for S-SR, acted in the same way as the second, but, in addition, it was ensured that when hill-climbing was subsequently applied, it would be applied to the mutated gene first and foremost, thus updating its allele bounds once again in anticipation of a later S-SR operation.

3.4 Hill-Climbing

A major concern while designing the S-SR operator was the potentially enormous expense incurred due to repeated function evaluations during the hill-climbing stages: for example, using one-point crossover with genotypes of any length for a run of 300 generations, the total number of evaluations would be 67500; however, using S-SR with genotypes of length 100 (and assuming hill-climbing is applied to every gene), the number of evaluations for a single run potentially could range from 6817500 to 13567500. Therefore in order to reduce this expense, hill-climbing was only applied to specified percentages of randomly selected genes; the percentages used were 50%, 30%, 10%, 5% and 1%.

In order that the comparison was fair in terms of the amount of function evaluations, all runs were set to last the number of evaluations equal to the minimum number of evaluations possible in a run of 300 generations using S-SR, with hill-climbing applied to 50% of a genotype, i.e. $((1 + 50) \times 225) \times 300 = 3442500$ for a genotype of length 100; $((1 + 25) \times 225) \times 300 = 1755000$ for a genotype of length 50, and so on.

3.5 Local Search

In addition to comparing S-SR with simple one-point crossover, a subsequent set of runs were carried out in order to compare S-SR performance with a GA using one-point crossover which was also boosted by local search. The method of local search was shown to perform well in studies such as [7] and comprised 200 mutations applied to the current fittest individual after every 225 breeding cycles. (A breeding cycle includes hill-climbing during S-SR runs.)

4 Results

De Jong published his suite of test functions in 1975 [3], and they have been used extensively ever since, becoming a standard test bed. However, it is interesting to note how the increase in computing power and GA efficiency has rendered trivial what once represented common difficulties among optimization problems: while comparing the one-point and site-specific operators, both implementations invariably found the optimum during the first few generations (if not during the initialisation of the population) for all of the three De Jong problems used. Therefore this section will concentrate only on results obtained for the other four functions.

Differences for both one-point (see tables 1 and 2) and S-SR (see tables 3 and 4) between the standard mutation type, method 1, and the less constrained method 2 showed that on average the latter yielded slightly better results. (Table 5 shows that, for S-SR, method 2.1 performed best.) Only on F7, and using S-SR, did method 1 yield substantially better results.

In comparison with one-point crossover, S-SR – with the appropriate amount of hill-climbing and, typically, mutation type 2.1 – performed better on all functions except F5: the best mean achieved by S-SR was 2.06 as opposed to 2.63 by one-point.

In comparison with alternative strategies, S-SR improved on results detailed in [1] with regard to quality of solutions and the speed with which they were found for F4. On F5, S-SR only performed worse than a multiple-restart stochatsic hill-climbing method (MRSH) when incorporating Gray encoding. On F6, S-SR proved competitive with results detailed in [5] which were achieved by a GA which incorporated elitism and niching. On F7, S-SR achieved significantly better results than are given in [2].

Function	Std. Dev.	Mean	Best	Worst
F4	0.002105	0.044284	0.04869	0.038817
F5	0.066223	0.950103	1.110562	0.775135
F6	0.041488	0.540729	0.621828	0.449878
F7	1.91E+06	1.52E+06	4.29E+04	8.18E+06

Table 1: results for runs using one-point crossover, mutation method 1

Function	Std. Dev.	Mean	Best	Worst
F4	0.002008	0.04099	0.045272	0.036733
F5	0.251945	2.631828	3.266276	2.17429
F6	0.023615	0.601511	0.646866	0.551677
F7	4.26E+05	2.54E+05	3.26E+03	2.77E+06

Table 2: results for runs using one-point crossover, mutation method 2

Save for the occasional anomaly, a fortuitous pattern was observed with regard to the amount of hill-climbing applied during S-SR runs: in general, performance – in terms of both the quality of solutions and the speed with which they were found – improved as the amount of hill-climbing applied, and therefore the number of function evaluations, was reduced (although such evidence for runs with F6 is more ambiguous). This is illustrated in figure 3; the percentage of genes to which hill-climbing was applied in a particular run is included where convenient.

The addition of local search improved significantly the performance of runs using one-point crossover, except on F4. However, S-SR by itself still beat these runs for two of the functions, F4 and F7; adding local search to S-SR improved its performance even more and S-SR then still only performed worse than one-point with local search on one function, F5. Figure 4 illustrates runs using local search with both one-point crossover and S-SR (mutation type 2.1) respectively, averaged over fifty runs as before.

Function	Std. Dev.	Mean	Best	Worst
F4 50% HC	0.031607	0.136203	0.193491	0.086794
F4 30% HC	0.033396	0.140692	0.201837	0.072893
F4 10% HC	0.021976	0.13131	0.226073	0.067965
F4 5% HC	0.028029	0.152747	0.216624	0.092069
F4 1% HC	0.004488	0.045702	0.056913	0.036926
F5 50% HC	0.043518	0.724667	0.825794	0.611054
F5 30% HC	0.048233	0.752722	0.851369	0.657559
F5 10% HC	0.04193	0.802317	0.923555	0.724445
F5 5% HC	0.078568	0.763717	0.864478	0.345488
F5 1% HC	0.059918	0.916119	1.117762	0.774005
F6 50% HC	0.070388	0.504706	0.602896	0.312791
F6 30% HC	0.076663	0.489892	0.7	0.353127
F6 10% HC	0.061244	0.464481	0.588138	0.340804
F6 5% HC	0.054148	0.467895	0.57794	0.360941
F6 1% HC	0.054685	0.430142	0.564291	0.318137
F7 50% HC	3.510209	0.501458	0	25.072918
F7 30% HC	25.072918	0.437464	0	21.873184
F7 10% HC	37.858428	5.408347	0	270.41735

Table 3: results for runs using S-SR, mutation type 1

Function	Std. Dev.	Mean	Best	Worst
F4 50% HC	0.024879	0.153171	0.204769	0.087742
F4 30% HC	0.02661	0.16219	0.215401	0.115917
F4 10% HC	0.013164	0.13987	0.159356	0.10867
F4 5% HC	0.023869	0.161398	0.221522	0.113216
F4 1% HC	0.004367	0.052315	0.06248	0.043504
F5 50% HC	0.040895	0.771428	0.878448	0.690911
F5 30% HC	0.045143	0.81307	0.912884	0.737229
F5 10% HC	0.04596	0.865588	0.966334	0.763616
F5 5% HC	0.048425	0.85548	0.947234	0.727778
F5 1% HC	0.077953	1.303436	1.480968	1.154428
F6 50% HC	0.044161	0.627462	0.739852	0.739852
F6 30% HC	0.04405	0.629603	0.730867	0.502184
F6 10% HC	0.040037	0.616966	0.721764	0.535523
F6 5% HC	0.054592	0.615548	0.711064	0.466847
F6 1% HC	0.030055	0.58117	0.671302	0.490647
F7 50% HC	1.28E+05	2.13E+04	3.55E+00	9.07E+05
F7 30% HC	2.79E+04	5.56E+03	5.74E+00	1.92E+05
F7 10% HC	100.077	83.853	5.390	355.805

Table 4: results for runs using S-SR, mutation type 2

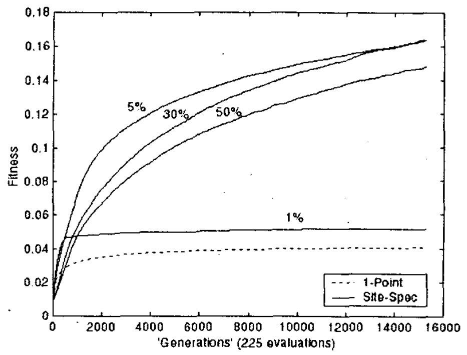
A pattern of behaviour similar to that of the first set of runs was evident, although the inclusion of local search appeared to slow down convergence, particularly with functions F4, F5 and F7. However, with regard to F5, whereas typically solutions improved much faster using S-SR than one-point at the beginning of a run (until subsequently being overtaken) during the second set of runs, one-point invariably performed better than S-SR from the outset when local search was applied. Figure 4 illustrates levels of performance for runs using local search.

Function	Std. Dev.	Mean	Best	Worst
F4 50% HC	0.029256	0.147835	0.184381	0.074057
F4 30% HC	0.030406	0.163875	0.215593	0.077442
F4 10% HC	0.01805	0.144346	0.194791	0.109044
F4 5% HC	0.015724	0.163612	0.197257	0.133717
F4 1% HC	0.004473	0.051899	0.064543	0.038423
F5 50% HC	0.039983	0.769808	0.866052	0.686105
F5 30% HC	0.058376	0.829737	0.976668	0.74918
F5 10% HC	0.054346	0.849967	1.005628	0.762234
F5 5% HC	0.06074	0.856513	1.088315	0.718303
F5 1% HC	0.192987	2.069425	2.5074	1.446473
F6 50% HC	0.042654	0.619801	0.733129	0.482178
F6 30% HC	0.036691	0.622573	0.719462	0.560513
F6 10% HC	0.041164	0.626924	0.744267	0.563771
F6 5% HC	0.039392	0.618657	0.724491	0.521155
F6 1% HC	0.05638	0.576746	0.675144	0.434858
F7 50% HC	9.37E+03	1.99E+03	4.43E+00	6.06E+04
F7 30% HC	79.270	75.484	0.894	280.827
F7 10% HC	87.234	83.041	0.427	354.467

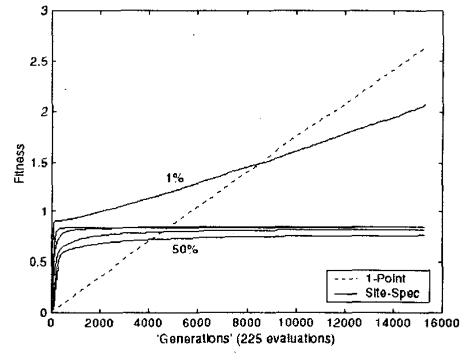
Table 5: results for runs using S-SR, mutation type 2.1

5 Discussion

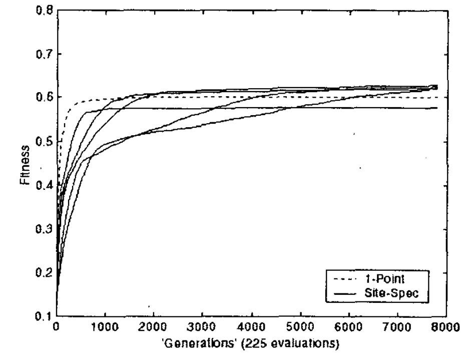
On examining the results, particularly those gathered for functions F4 and F5 given their opposing characteristics, it is encouraging to see that S-SR performs comparatively well on problems with high epistasis – problems which both GAs and hill-climbing traditionally find difficult. The repeated application of hill-climbing undoubtedly will assist a GA to some extent, as traditional hybrid approaches illustrate. However, regardless of any benefits reaped from hill-climbing per se, clearly, the rate of solution improvement using S-SR is very often much faster than with one-point, particularly at the beginning of a search. Such an assertion is reinforced by recalling that the reduction in the hill-climbing percentage invariably increases this speed significantly. Nevertheless, there would appear to be distinct optimum levels of hill-climbing for the different functions: figure 3 illustrates that although reducing the amount of hill-climbing invariably increases the speed of solution improvement in the early stages of the search, sometimes runs with the least amount of hill-climbing ultimately are not the best performers. Similarly, on F4 (the first of the two Baluja functions used) the optimum hill-climbing percentage appears to be around 5%, whereas percentages of 1% and 10% elicited poor results. Conversely, on F5 (the sister Baluja function) a hill-climbing percentage of 1% elicits a performance far better than those using higher percentages. Moreover, either too low a level of hill-climbing or too high a level of hill-climbing results in an increased rate of convergence.



(a)



(b)



(c)

Figure 3: average performance of runs for (a) F6, (b) F5 and (c) F6

Function	Std. Dev.	Mean	Best	Worst
F4	0.002434	0.042893	0.047696	0.037158
F5	4.320856	43.787925	55.267357	34.799171
F6	0.044467	0.703679	0.785085	0.61512
F7	695.948	489.109	17.844	4654.804

Table 6: results for runs using one-point crossover (mutation method 2), with local search

Function	Std. Dev.	Mean	Best	Worst
F4 50% HC	0.014196	0.174359	0.209247	0.143221
F4 30% HC	0.0192	0.17248	0.211882	0.139093
F4 10% HC	0.013208	0.14103	0.173231	0.111771
F4 5% HC	0.020423	0.166046	0.209749	0.118235
F4 1% HC	0.00322	0.047479	0.058447	0.038107
F5 50% HC	0.183986	1.785747	2.305293	1.335613
F5 30% HC	0.256185	2.617804	3.10274	2.085571
F5 10% HC	0.672608	6.609144	8.259431	5.186248
F5 5% HC	1.268163	12.2427	15.192928	9.572442
F5 1% HC	3.224481	31.630345	39.186485	25.348616
F6 50% HC	0.031754	0.765916	0.815092	0.62449
F6 30% HC	0.027081	0.771703	0.820694	0.705847
F6 10% HC	0.032556	0.775546	0.824942	0.656737
F6 5% HC	0.030724	0.768574	0.817979	0.686743
F6 1% HC	0.044304	0.739859	0.808975	0.64303
F7 50% HC	3.654	0.522	0	26.100
F7 30% HC	20.813	2.973	0	148.667
F7 10% HC	3.121	0.446	0	22.293

Table 7: results for runs using S-SR with local search

It would appear that, on the functions where S-SR beats one-point, the improved performance is due to there being a much greater number of constructive crossover operations, per generation, during the early stages of the search. (For the purposes of this paper, a 'constructive crossover' is defined as one that yields at least one offspring which is fitter than either parent; one generation is taken to equal 225 breeding cycles.) Indeed, with the exception of F6 (where the percentage of crossover operations per generation remained high throughout runs of 300 generations), when one-point is used the percentage of constructive operations falls to a negligible – if not non-existent – level in a very short time. In contrast, a relatively high percentage of constructive operations is maintained for longer when S-SR is used; furthermore, constructive operations (albeit a small number) is evident for the duration of a run. This is illustrated using F7 in figure 5. It seems likely that the S-SR selection strategy plays an important part in the success of the operator. GAs have always been based on the concept 'survival of the fittest', which implicitly dictates that the least fit members of a population are left to die out. Conversely, the best S-SR performance is observed when the least fit member of a neighbourhood is selected and resuscitated via the donation of genes from a fitter neighbour.

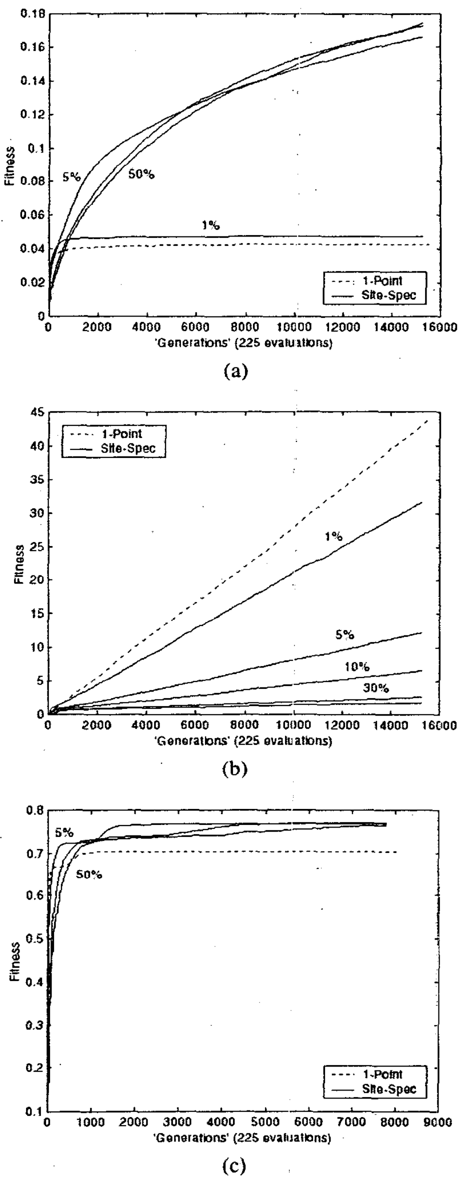


Figure 4: average performance of runs for (a) F6, (b) F5 and (c) F6 with local search

Perhaps it is unsurprising that more constructive crossovers are observed when the S-SR strategy is to take an unfit individual and make it fitter, but one possible explanation may be that fit schemata concealed within unfit individuals, which normally would be overlooked by selection, instead are being exploited. Indeed, this would seem likely if the least fit individual mates with its fittest neighbour to produce offspring fitter than either of its parents (thereby fulfilling the axiom which dictates that 'opposites attract'!).

That constructive crossover operations are still being carried out even at the very end of a run perhaps indicates a slowing of convergence – despite the acceleration of improvement, and that, intuitively, the nature of the site-specific operator suggests an *increased* rate of convergence. Another possibility may be that S-SR makes the most of small differences between a converged population; or that, in a population of converged fitness, one individual benefits from receiving a particular gene from another individual which is of similar fitness but which inhabits a different area of the search space.

6 Conclusion

The immediate objective of designing an operator which attempts to increase the probability of affecting a constructive crossover, thereby increasing the power of a GA, has been fulfilled: it has been shown empirically that using site-specific crossover on some (currently!) difficult functions will accelerate optimization to a considerable degree as well as usually achieving fitter solutions than are reached when one-point is used. However, only continuous function optimization problems have been addressed so far – a set of problems which constitute only a small subset of those to which GAs are routinely applied. Therefore, the operator must be adapted, if necessary, to tackling such combinatorial tasks such as job-shop scheduling and the travelling salesman problem, or even neural network weight optimization. In addition, an analytical investigation of the operator should be carried out in the hope of producing a more complete explanation of the overall rôle of crossover in evolutionary algorithms.

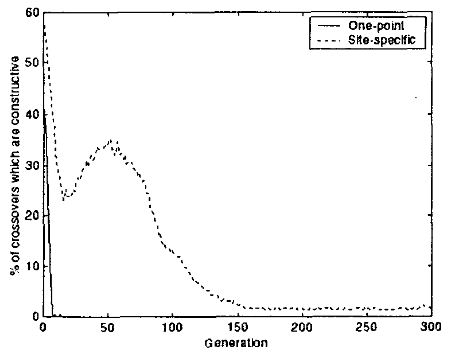


Figure 5: Amount of constructive crossover operations

References

[1] S. Baluja, *An Empirical Comparison of Seven Iterative and Evolutionary Function Optimisation Heuristics*, Internal Paper CMU-CS-95-193, School of Computer Science, Carnegie Mellon University (1995).

[2] A. Corana, M. Marchesi, C. Martini, and S. Ridella, Minimizing Multimodal Functions of Continuous Variables with the 'Simulated Annealing' Algorithm, in *ACM Transactions on Mathematical Software, Vol. 13, No. 3*, September 1987, pages 262-280.

[3] K. DeJong, *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems*, PhD thesis, University of Michigan (1975).

[4] J. J. Grefenstette, Optimization of Control Parameters for Genetic Algorithms, *IEEE Transactions on Systems, Man & Cybernetics 16, No. 1* (1986).

[5] A. Keane, *Experiences with Optimisers in Structural Design*. in Proceedings of the First International Conference on Adaptive Computing in Engineering Design and Control, University of Plymouth (1994).

[6] R. F. Leach, *Genetic Recombination*, Blackwell Science Ltd. (1996).

[7] M. McIlhagga, P. Husbands and R. Ives, A Comparison of Search Techniques on a Wing-Box Optimisation Problem, in H.-M. Voigt, W. Ebeling, I. Rechenberg and H.-P. Schwefel, editors, *PPSN IV*, Springer (1996).

[8] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press (1998).

[9] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, R. Das, A Study of Control Parameters Affecting On-Line Performance of Genetic Algorithms for Function Optimization, in J. D. Schaffer, ed., *Proceedings of the Third ICGA*, Morgan Kaufman (1989).

[10] G. Syswerda, Uniform Crossover in Genetic Algorithms, in J. D. Schaffer, ed., *Proceedings of the Third ICGA*, Morgan Kaufman (1989).

[11] C. Thornton, The Building Block Fallacy, *Complexity Interntational Vol. 4* (1997).

[12] K. Vekari, C. Clack, Selective Crossover in Genetic Algorithms: An Empirical Study, in A. E. Eiben, T. Bäck, M. Schoenauer and H.-P. Schwefel, eds., *PPSN V*, Springer (1998).