E1.3 Recombination parameters

William M. Spears

Naval Research Laboratory

Abstract

One operator that is often used in evolution strategies, genetic algorithms, and genetic programming is recombination, where material from two (or more) parents is used to create new offspring. There are numerous ways to implement recombination. This section will focus mainly on recombination operators that construct potentially useful solutions to a problem from smaller components (often called "building blocks"). Recombination operators that "blend" are discussed in section XXX. This section gives an overview of some of the motivation, issues, theory, and heuristics for "building block" recombination.

E1.3.1 General Background

Although Holland (1975) was not the first to suggest recombination in an evolutionary algorithm (EA) (e.g. see Fraser 1957 or Fogel *et al* 1966), he was the first to place theoretical emphasis on this operator. This emphasis stemmed from his work in adaptive systems, which resulted in the field of genetic algorithms and genetic programming. According to Holland, an adaptive system must persistently test and incorporate structural properties associated with better performance. The object, of course, is to find *new* structures which have a high probability of improving performance significantly.

Holland concentrated on schemata, which provide a basis for associating combinations of attributes with potential for improving current performance. To see this, let us consider the schema AC##, defined over a fixed length chromosome of four genes, where each gene can take on one of three alleles {A, B, C}. If "#" is defined to be a "don't care" (i.e. wildcard) symbol, the schema AC## represents all chromosomes that have an A for their first allele and a C for their second. Since each of the "#" symbols can be filled in with any one of the three alleles, this schema represents $3^2 = 9$ chromosomes.

Suppose every chromosome has a well-defined fitness value (also called "utility," "payoff," etc.). Now suppose there is a population of P individuals, p of which are members of the above schema. The "observed average fitness" of that schema is the average fitness of those p individuals in that schema. It is important to note that these individuals will also be members of other schemata, thus the population of P individuals contains instances of a large number of schemata (all of which have some observed fitness). Holland (1975) stated that a good heuristic is to generate new instances of those schemata whose observed fitness is higher than the average fitness of the whole population, since instances of those schemata are likely to exhibit superior performance.

Suppose the schema AC## does in fact have a high observed fitness. The heuristic states that new samples (instances) of that schema should be generated. Selection (reproduction) does not produce new samples — but recombination can. The key aspect of recombination is that if one recombines two individuals that start with AC, their offspring must also start with AC. Thus one can retain what appears to be a promising building block (AC##), yet continue to test that building block in new contexts.

As stated earlier, recombination can be implemented in many different ways. Some forms of recombination are more appropriate for certain problems than are others. According to Booker

© 1995 IOP Publishing Ltd and Oxford University Press

(1992) it is thus useful to characterize the biases in recombination operators, recognize when these biases are correct or incorrect (for a given problem or problem class), and recover from incorrect biases when possible. The next two subsections summarize much of the work that has gone into characterizing the biases of various recombination operators. Historically, most of the earlier recombination operators were designed to work on low-level "universal" representations, such as the fixed-length low cardinality representation shown above. In fact, most early genetic algorithms used just simple bit-string representations. A whole suite of recombination operators evolved from that level of representation. Subsection E1.3.2 focuses on such "bit-level" recombination operators. Recent work has focused more on problem-class specific representations, with recombination operators designed primarily for those representation. Subsection E1.3.3 focuses on the more problem-class specific recombination operators.

Subsection E1.3.4 summarizes some of the mechanisms for recognizing when biases are correct or incorrect, and recovering from incorrect biases when possible. The conclusion outlines some design principles that are useful when creating new recombination operators. Ideally, one would like firm and precise practical rules for choosing what form and rate of recombination to use on a particular problem, however, such rules have been difficult to formulate. Thus this section concentrates more on heuristics and design principles that have often proved useful.

E1.3.2 Genotypic-level recombination

Theory

Holland (1975) provided one of the earliest analyses of a recombination operator, called 1-point recombination. Suppose there are two parents: ABCD and AABC. Randomly select one point at which to separate ("cut") both parents. For example, suppose they are cut in the middle (AB|CD and AA|BC). The offspring are created by swapping the tail (or head) portions to yield ABBC and AACD. Holland analyzed 1-point recombination by examining the probability that various schemata will be disrupted when undergoing recombination. For example, consider the two schemata AA## and A##A. Each schema can be disrupted only if the "cut-point" falls between its two A's. However, this is much more likely to occur with the latter schema (A##A) than the former (AA##). In fact, the probability of disrupting either schema is proportional to the distance between the A's. Thus, 1-point recombination has the bias that it is much more likely to disrupt "long" schemata than "short" schemata, where the length of a schema is the distance between the first and last "defining position" (a non-wildcard).

De Jong (1975) extended this analysis to include so-called "n-point" recombination. In n-point recombination n cut-points are randomly selected and the genetic material between cut-points is swapped. For example, with 2-point recombination, suppose the two parents ABCD and AABC are cut as follows: A|BC|D and A|AB|C. Then the two offspring are AABD and ABCC. De Jong noted that 2-point (or n-point where n is even) recombination is less likely to disrupt "long" schemata than 1-point (or n-point where n is odd) recombination.

Syswerda (1989) introduced a new form of recombination called "uniform" recombination. Uniform recombination does not use "cut-points" but instead creates offspring by deciding, for each allele of one parent, whether to swap that allele with the corresponding allele in the other parent. That decision is made using a coin-flip (i.e. the swap is made 50% of the time). Syswerda compared the probability of schema disruption for 1-point, 2-point, and uniform recombination. Interestingly, while uniform recombination is somewhat more disruptive of schemata than 1-point and 2-point, it does not have a length bias (i.e. the length of a schema does not affect the probability of disruption). Also, Syswerda showed that the more disruptive nature of uniform recombination can be viewed in another way — it is more likely to construct instances of new schemata than 1-point and 2-point recombination.

De Jong and Spears (1992) verified Syswerda's results and introduced a parameterized version of uniform recombination (where the probability of swapping could be other than 50%). Lowering

 © 1995 IOP Publishing Ltd and Oxford University Press

the swap probability of uniform recombination allows one to lower disruption as much as desired, while maintaining the lack of length bias. Finally, De Jong and Spears characterized recombination in terms of two other measures: productivity and exploration power. The productivity of a recombination operator is the probability that it will generate offspring that are different from their parents. More disruptive recombination operators are more productive (and vice versa). An operator is more explorative if it can reach a larger number of points in the space with one application of the operator. Uniform recombination is the most "explorative" of the recombination operators since, if the Hamming distance between two parents is h (i.e. h loci have different alleles), uniform recombination can reach any of 2^h points in one application of the operator. Moon and Bui (1994) independently performed a similar analysis. Although mathematically equivalent, this analysis emphasized "clusters" of defining positions within schemata, as opposed to lengths.

Eshelman *et al* (1989) considered other characterizations of recombination bias. They introduced two biases, the positional and distributional bias. A recombination operator has positional bias to the extent that the creation of any new schema by recombining existing schemata is dependent upon the location of the alleles in the chromosome. This is similar to the length bias introduced above. A recombination operator has distributional bias to the extent that the amount of material that is expected to be exchanged is distributed around some value or values as opposed to uniformly distributed ranging from 1 to $L-1$ alleles (where the chromosome is composed of L genes). For example, 1-point recombination has high positional and no distributional bias, while 2-point recombination has slightly lower positional bias and still no distributional bias. Uniform recombination has no positional bias but high distributional bias because the amount of material exchanged is binomially distributed. Later, Eshelman and Schaffer (1994) refined their earlier study and introduced "recombinative" bias, which is related to their older distributional bias. They also introduced "schema" bias, which is a generalization of their older positional bias.

Booker (1992) tied the earlier work together by characterizing recombination operators via their recombination distributions, which describe the probability of all possible recombination events. The recombination distributions were used to rederive the disruption analysis of De Jong and Spears (1992) for n-point and parameterized uniform recombination, as well as to calculate precise values for the distributional and positional biases of recombination. This reformulation allowed Booker to detect a symmetry in the positional bias of n-point recombination around $n = L / 2$, which corrected a prediction made by Eshelman *et al* (1989) that positional bias would continue to increase as n gets larger.

Heuristics

The sampling arguments and the characterization of biases that has just been presented have motivated a number of heuristics for how to use recombination and how to choose which recombination to use.

Booker (1982) considered implementations of recombination from the perspective of trying to improve overall performance. The motivation was that allele loss from the population could hurt the sampling of co-adapted sets of alleles (schemata). In the earliest implementations of genetic algorithms one offspring of a recombination event would be thrown away. This was a source of allele loss, since, instead of transmitting all alleles from both parents to the next generation, only a subset was transmitted. The hypothesis was that allele loss rates would be greatly decreased by saving both offspring. That hypothesis was confirmed empirically. There was also some improvement in online (average fitness of all samples) and offline (average fitness of the best samples) performance on the De Jong (1975) test suite, although the offline improvement was negligible.

Booker (1987) also pointed out that, due to allele loss, recombination is less likely to produce children different from their parents as a population evolves. This effectively reduces the sampling of new schemata. To counteract this Booker suggested a more explorative version of recombination, termed "reduced surrogate" recombination, that concentrates on those portions of a chromosome in which the alleles of two parents are not the same. This ensures that a new sample is created. For example, suppose two parents are ABCD and ADBC. Then if one uses 1-point recombination, and

© 1995 IOP Publishing Ltd and Oxford University Press

the cut-point occurs immediately after the A, the two offspring would be identical to the parents. Reduced surrogate recombination would ensure that the cut-point was further to the right.

It has been much more difficult to come up with heuristics for choosing which recombination operator to use in a given situation. Syswerda (1989), however, noted that one nice aspect of uniform recombination is that, due to its lack of length bias, it is not affected by the presence of irrelevant alleles in a representation. Nor is it affected by the position of the relevant alleles on the chromosome. Thus, for those problems where little information is available concerning the relevance of alleles or the length of building blocks, uniform recombination is a useful default.

De Jong and Spears (1990) tempered this view somewhat, by including interactions with population size. Their heuristic was that disruption is most useful when the population size is small or when the population is almost homogeneous. They argued that more disruptive recombination operators (such as 0.5 uniform recombination, or n-point recombination where n is greater than two) should be used when the population size is small relative to the problem size, and less disruptive recombinations operators (such as 2-point, or uniform recombination with a swap probability less than 50%) should be used when the population size is large relative to the problem size. De Jong and Spears demonstrated this with a series of experiments in which the population size was varied.

Schaffer *et al* (1989) made a similar observation. They concentrated on the "recombination rate," which is the percentage of the population to undergo recombination. They observed that high recombination rates are best with small populations, a broad range of recombination rates are tolerated at the middle population sizes, and only low recombination rates are suggested for large population sizes.

Finally, Eshelman and Schaffer (1994) have attempted to match the biases of recombination operators with various problem classes and genetic algorithm behavior. They concluded with two heuristics. The first was that high schema bias can lead to hitchhiking, where the EA exploits spurious correlations between schemata that contribute to performance and other schemata that do not. They recommended using a high recombinative bias and low schema bias recombination to combat premature convergence (i.e., loss of genetic diversity) due to hitchhiking. The second heuristic was that high recombinative bias can be detrimental in trap problems.

E1.3.3 Phenotypic-level recombination

Theory

Thus far the focus has been on fixed-length representations in which each gene can take on one of a discrete set of alleles (values). Schemata were then defined, each of which represent a set of chromosomes (the chromosomes that match alleles on the defining positions of the schema). However, there are problems that do not match well to these representations. In these cases new representations, recombination operators, and theories must be developed.

For example, a common task is the optimization of some real-valued function of real values. Of course, it is possible to code these real values as bit-strings in which the degree of granularity is set by choosing the appropriate number of bits. At this point conventional "schema" theory may be applied. However, there are difficulties that arise using this representation. One is the presence of "Hamming cliffs," in which large changes in the binary encoding are required to make small changes to the real values. The use of Gray codes does not totally remove this difficulty. Standard recombination operators also can have the effect of producing children far removed (in the real-valued sense) from their parents (see Schwefel 1995). An alternative representation is to simply use chromosomes that are real-valued vectors. In this case a more natural recombination operator averages (blends) values within two parent vectors to create a child vector. This has the nice property of creating children that are near the parents. See Davis (1991), Wright (1991), Eshelman and Schaffer (1992), Schwefel (1995), Peck and Dhawan (1995), Beyer (1995) and Arabas *et al* (1995) for other recombination operators that are useful for real-valued vectors. One recombination operator of note is discrete recombination, which is the analogue of uniform recombination on real-valued variables.

Recently, three theoretical studies analyzed the effect of recombination using real-valued representations. Peck and Dhawan (1995) showed how various properties of recombination operators

 © 1995 IOP Publishing Ltd and Oxford University Press

can influence the ability of the EA to converge to the global optima. Beyer (1995) concluded that an important role of recombination in this context is genetic repair, diminishing the influence of harmful mutations. Eshelman and Schaffer (1992) analyzed this particular representation by restricting the parameters to be integer ranges. Their "interval-schemata" represent subranges. For example, if a parameter has range [0, 2], it has the following interval-schemata: [0], [1], [2], [0, 1], [1, 2], and [0, 2]. The chromosomes (1 1 2) and (2 1 2) are instances of the interval-schema ([1, 2] [1] [2]). Long interval-schemata are more general and correspond roughly to traditional schemata that contain a large number of #s. Eshelman and Schaffer used "interval-schemata" to help them predict the failure modes of various real-valued recombination operators.

Another class of important tasks involves permutation or ordering problems, in which the ordering of alleles on the chromosome is of primary importance. A large number of recombination operators have been suggested for these tasks, including partially mapped recombination (Goldberg and Lingle 1985), order recombination (Davis 1985), cycle recombination (Oliver *et al* 1987), edge recombination (Starkweather *et al* 1991) and the group recombination of Falkenauer (1994). Which operators work best depend on the objective function.

A classic permutation problem is the Traveling Salesman Problem (TSP). Consider a TSP of four cities {A, B, C, D}. It is important to note that there are only 4! possible chromosomes, as opposed to 4^4 (e.g. the chromosome AABC is not valid). Also, note that schema ##BC does not have the same meaning as before, since the alleles that can be used to fill in the #'s now depend on the alleles in the defining positions (e.g. a B or a C can't be used in this case). This led Goldberg and Lingle (1985) to define o-schemata, in which the "don't cares" are denoted with !s. An example of an o-schema is !!BC, which defines the subset of all orderings that have BC in the third and fourth positions. For this example there are only two possible orderings, ADBC and DABC. Goldberg considered this to be "absolute" o-schemata, since the absolute position of the alleles is of importance. An alternative would be to stress the relative positions of the alleles. In this case what is important about !!BC is that B and C are adjacent — !!BC, !BC!, and BC!! are now equivalent o-schemata. One nice consequence of the invention of o-schemata is that a theory similar to that of the more standard schema theory can be developed. The interested reader is encouraged to see Oliver *et al* (1987) for a nice example of this, in which various recombination operators are compared via an o-schemata analysis.

There has also been some work on recombination for finite state machines (Fogel *et al* 1966), variable length chromosomes (Smith 1980, De Jong *et al* 1993), chromosomes that are Lisp expressions (Fujiki and Dickinson 1987, Koza 1994, and Rosca 1995), chromosomes that represent strategies (i.e. rule sets) (Grefenstette *et al* 1990), and recombination for multi-dimensional chromosomes (Kahng and Moon 1995). Some theory has recently been developed in these areas. For example, Bui and Moon (1995) developed some theory on multi-dimensional recombination. Also, Radcliffe (1991) generalized the notion of schemata to sets he refers to as "formae."

Heuristics

Due to the prevalence of the traditional bit-string representation in GAs, less work has concentrated on recombination operators for higher-level representations, and there are far fewer heuristics. The most important heuristic is that recombination must identify and combine meaningful building blocks of chromosomal material. Put another way, "recombination must take into account the interaction among the genes when generating new instances" (Eshelman and Schaffer 1992). The conclusion of this section provides some guidance in how to achieve this.

E1.3.4 Control of recombination parameters

As can be seen from the earlier discussion, there is very little theory or guidance on how to choose a priori which recombination operator to use on a new problem. There is also very little guidance on how to choose how often to apply recombination (often referred to as the "recombination rate"). There have been three approaches to this problem, referred to as "static," "predictive," and "adaptive" approaches.

© 1995 IOP Publishing Ltd and Oxford University Press

Static

The simplest approach is to assume that one particular recombination operator should be applied at some static rate for all problems. The static rate is estimated from a set of empirical studies, over a wide variety of problems, population sizes, and mutation rates. Three studies are of note. De Jong (1975) studied the online and offline performance of a GA on the De Jong test suite and recommended a recombination rate of 60% for 1-point recombination (i.e. 60% of the population should undergo recombination). Grefenstette (1986) studied online performance of a GA on the De Jong test suite and recommended that 1-point recombination be used at the higher rate of 95%. In the most recent study, Eshelman *et al* (1989) studied the mean number of evaluations required to find the global optimum on the De Jong test suite and recommended an intermediate rate of 70% for 1-point recombination. Each of these settings for the recombination rate are associated with particular settings for mutation rates and population sizes, so the interested reader is encouraged to consult these references for more complete information.

Predictive techniques

In the static approach it is assumed that some fixed recombination rate (and recombination operator) is reasonable for a large number of problems. However, this will not be true in general. The "predictive" approaches are designed to predict the performance of recombination operators (i.e. to recognize when the recombination bias is correct or incorrect for the problem at hand).

Manderick *et al* (1991) computed fitness correlation coefficients for different recombination operators on various problems. Since they noticed a high correlation between operators with high correlation coefficients and good GA performance their approach was to choose the recombination operator with the highest correlation coefficient. The approach by Grefenstette (1995) was similar in spirit to the Manderick *et al* approach. Grefenstette used a "virtual" GA to compute the past performance of an operator as an estimate of the future performance of an operator. By running the virtual GA with different recombination operators, Grefenstette estimated the performance of those operators in a real GA.

Altenberg (1994) and Radcliffe (1994) have proposed different predictive measures. Altenberg proposed using an alternative statistic referred to as the "transmission function in the fitness domain." Radcliffe proposed using the "fitness variance of formae" (generalized schemata). Thus far all approaches have shown considerable promise.

Adaptive techniques

In both the static and predictive approaches the decision as to which recombination operator and the rate at which it should be applied is fixed prior to actually running the EA. However, since these approaches can make errors (i.e. choose non-optimal recombination operators or rates), a natural solution is to make these choices adaptive. Adaptive approaches are designed to recognize when bias is correct or incorrect, and recover from incorrect biases when possible. For the sake of exposition adaptive approaches will be divided into "tag-based" or "rule-based." As a general rule, tag-based approaches attach extra information to a chromosome, which is both evolved by the EA and used to control recombination. The rule-based approaches generally adapt recombination using control mechanisms and data structures that are external to the EA.

One of the earliest tag-based approaches was by Rosenberg (1967). In this approach integers x_i ranging from zero to seven were attached onto each locus. The recombination site was chosen from the probability distribution defined over these integers, $p_i = x_i / \sum x_i$, where p_i represented the probability of a cross at site i.

Schaffer and Morishima (1987) had a similar approach, by adjusting the points at which recombination was allowed to cut and splice material. They accomplished this by appending an additional L bits to L-bit individuals. These appended bits were used to determine "cut-points" for each locus (a '1' denoted a cut-point while a '0' indicated the lack of a cut-point). If two individuals had n distinct cut-points, this was analogous to using a particular instantiation of n-point recombination.

 © 1995 IOP Publishing Ltd and Oxford University Press

Levenick (1995) also had a similar approach. Recombination was implemented by replicating two parents from one end to the other by iterating the following algorithm:

(1) Copy one bit from parent1 to child1
(2) Copy one bit from parent2 to child2
(3) With some base probability P_b perform a recombination:
swap the roles of the children (so subsequent bits come
from the other parent)

Levenick inserted a meta-bit before each bit of the individual. If the meta-bit was '1' in both parents recombination occurred with probability P_b, else recombination occurred with a reduced probability P_r. The effect was that the probability of recombination could be reduced from a maximum of P_b to a minimum of P_r. Levenick claimed that this method improved performance in those cases where the population did not converge too rapidly.

Arabas *et al* (1995) experimented with adaptive recombination in an evolution strategy. Each chromosome consisted of L real-valued parameters combined with an additional L control parameters. In a standard evolution strategy these extra control parameters are used to adapt mutation. In this particular study the control parameters were also used to adapt recombination, by concentrating offspring around particular parents. Empirical results on four classes of functions were encouraging.

Angeline (1996) evolved Lisp expressions, and associated a recombination probability with each node in the Lisp expressions. These probabilities evolved and controlled the application of recombination. Angeline investigated two different adaptive mechanisms based on this approach and reported that the adaptive mechanisms outperformed standard recombination on three test problems.

Spears (1995) used a simple approach in which one extra tag-bit was appended to every individual. The tag-bits were used to control the use of 2-point and uniform recombination in the following manner:

```
if (parent1[L + 1] = parent2[L + 1] = 1)
then two-point-recombination(parent1, parent2)
else if (parent1[L + 1] = parent2[L + 1] = 0)
    then uniform-recombination(parent1, parent2)
    else if (rand(0,1) < 0.5)
        then two-point-recombination(parent1, parent2)
        else uniform-recombination(parent1, parent2)
```

Spears compared this adaptive approach on a number of different problems and population sizes, and found that the adaptive approach always had a performance intermediate between the best and worst of the two single recombination operators.

Rule-based approaches use auxiliary data structures and statistics to control recombination. The simplest of these approaches use hand-coded rules that associate various statistics with changes in the recombination rate. For example, Wilson (1986) examined the application of GAs to classifier systems and defined an entropy measure over the population. If the change in entropy was sufficiently positive or negative, the probability of recombination was decreased or increased respectively. The idea was to introduce more variation by increasing the recombination rate whenever the previous variation had been "absorbed." Booker (1987) considered the performance of genetic algorithms in function optimization and measured the percentage of the current population that produced offspring. Every percentage change in that measure was countered with an equal and opposite percentage change in the recombination rate.

© 1995 IOP Publishing Ltd and Oxford University Press

Srinivas and Patnaik (1994) considered measures of fitness performance and used those measures to estimate the distribution of the population. They increased the probability of recombination (P_c) and the probability of mutation (P_m) when the population was stuck at local optima and decreased the probabilities when the population was scattered in the solution space. They also considered the need to preserve "good"' solutions of the population and attempted this by having lower values of P_c and P_m for high fitness solutions and higher values of P_c and P_m for low fitness solutions.

Hong *et al* (1995) had a number of different schemes for adapting the use of multiple recombination operators. The first scheme was defined by the following rule: If both parents were generated via the same recombination operator, apply that recombination operator, else randomly select (with a coin flip) which operator to use. This first scheme was very similar to that of Spears (1995) but does not use tag-bits. Their second scheme was the opposite of the first: If both parents were generated via the same recombination operator, apply some other recombination operator, else randomly select (with a coin flip) which operator to use. Their third scheme used a measure called an "occupancy rate," which was the number of individuals in a population that were generated by a particular recombination (divided by the population size). For k recombination operators, their third scheme tried to balance the occupancy rate of each recombination operator around $1/k$. In their experiments the second and third schemes outperformed the first (although this was not true when they tried uniform and 2-point recombination).

Eshelman and Schaffer (1994) provided a switching mechanism to decide between two recombination operators that often perform well, HUX (a variant of uniform crossover where exactly half of the differing bits are swapped at random) and SHX (a version of 1-point recombination in which the positional bias has been removed). Their GA uses restarts — when the population is (nearly) converged the converged population is partially or fully randomized and seeded with one copy of the best individual found so far (the "elite" individual). During any convergence period between restarts (including the period leading up to the first restart), either HUX or SHX is used but not both. HUX is always used during the first two convergences. Subsequently, three rules are used for switching recombination operators:

(1) SHX is used for the next convergence, if during the prior convergence no individual is found that is as good as the elite individual
(2) HUX is used for the next convergence, if during the prior convergence no individual is found that is better than the elite individual, but at least one individual is found that is as good as the elite individual
(3) No change in the operator is made, if during the prior convergence, a new best individual is found (which will replace the old elite individual)

These methods had fairly simple rules and data structures. However, more complicated techniques have been attempted. Davis (1989) provided an elaborate bookkeeping method to reward recombination operators that produced good offspring or set the stage for this production. When a new individual was added to the population, a pointer was established to its parent or parents, and a pointer was established to the operator that created the new individual. If the new individual was better than the current best member of the population, the amount it was better was stored as its "local delta." Local deltas were passed back to parents to produce "inherited deltas." "Derived deltas" were the sum of local and inherited deltas. Finally, the "operator delta" was the sum of the derived deltas of the individuals it produced, divided by the number of individuals produced. These operator deltas were used to update the probability that the operator would be fired.

White and Oppacher (1994) used finite state automatons to identify groups of bits that should be kept together during recombination (an extension of uniform recombination). The basic idea was to learn from previous recombination operations in order to minimize the probability that highly fit schemata will be disrupted in future recombination operations. The basic bit-string representation was augmented at each bit position with an automaton. Each state of the automaton mapped

 © 1995 IOP Publishing Ltd and Oxford University Press

to a probability of recombination for that bit-string location — roughly, given N states, then the probability p_i associated with state i was i/N. Some of the heuristics used for updating the automaton state were:

(1) If the offspring fitness > the fitness of the father(mother)
then reward those bits that came from the father(mother)
(2) If the offspring fitness < the fitness of the father(mother)
then penalize those bits that came from the father(mother)

There were other rules to handle offspring of equal fitness. A reward implied that the automaton moved from state i to state $i+1$, and a penalty implied that the automaton moved from state i to state $i-1$.

Julstrom (1995) used an "operator tree" to fire recombination more often if it produced children of superior fitness. With each individual was an operator tree — a record of the operators that generated the individual and its ancestors. If a new individual had fitness higher than the current population median fitness, the individual's operator tree was scanned to compute the credit due to recombination (and mutation). A queue recorded the credit information for the most recent individuals. This information was used to calculate the probability of recombination (and mutation).

Finally, Lee and Takagi (1993) evolved fuzzy rules for GAs. The fuzzy rules had three input variables based on fitness measures:

(1) x = average fitness / best fitness
(2) y = worst fitness / average fitness
(3) z = change in fitness

The rules had three possible outputs dealing with population size, recombination rate, and mutation rate. All of the variables could take on three values {small, medium, big}, with the semantics of those values determined by membership functions. The rules were evaluated by running the GA on the De Jong test suite and different rules were obtained for online and offline performance. They obtained 51 rules in the fuzzy rulebase. Of those, 18 were associated with recombination. An example was: If (x is small) and (y is small) and (z is small) then the change in recombination rate is small.

In summary, the fixed recombination rate approaches are probably the least successful, but provide reasonable guesses for parameter settings. They also are reasonable settings for the initial stages of the predictive and adaptive approaches. The predictive approaches have had success and appear very promising. The adaptive approaches also have had some success. However, as Spears (1995) indicated, a common difficulty in the evaluation of the adaptive approaches has been the lack of adequate control studies. Thus, although the approaches may show signs of adaptation, it is not clear that adaptation is the cause of performance improvement.

E1.3.5 Discussion

The successful application of recombination (or any other operator) involves a close link with the operator, the representation, and the objective function. This has been outlined by Peck and Dhawan (1995), who emphasized similarity — one needs to exploit similarities between previous high performance samples, and those similar samples must often enough have similar objective function values for the algorithm to be effective. Goldberg (1989) and Falkenauer (1994) made a similar point when they refer to "meaningful building blocks." This has led people to outline various issues that must be considered when designing a representation and appropriate recombination operators.

De Jong (1985) outlined several important issues with respect to representation. First, "near-by-ness" should be preserved, in which small changes in a parameter value should come about from small changes in the representation for that value. Thus, binary encodings of real-valued parameters

© 1995 IOP Publishing Ltd and Oxford University Press

are problematic, since Hamming cliffs separate parameters that are near in the real-valued space and standard recombination operators can produce children far removed (in the real-valued sense) from their parents. Second, it is generally better to have context insensitive representations, in which the legal values for a parameter do not depend on the values of other parameters. Finally, it is generally better to have context insensitive interpretations of the parameters, in which the interpretation of some parameter value does not depend on the values of the other parameters. These last two concerns often arise in permutation or ordering problems, in which the values of the left-most parameters influence both the legal values and the interpretation of those values for the right-most parameters. For example, the encoding of the TSP problem presented earlier is context sensitive, and standard recombination operators can produce invalid offspring when using the representation. An alternative representation could be one in which the first parameter specifies which of the N cities should be visited first. Having deleted that city from the list of cities, the second parameter always takes on a value in the range 1 ... $N-1$, specifying by position on the list which of the remaining cities is to be visited second, and so on. For example, suppose there are four cities {A, B, C, D}. The representation of the tour BCAD is (2 2 1 1) because city B is the second city in the list {A, B, C, D}, C is the second city in the list {A, C, D}, A is the first city in the list {C, D} and D is the first city in the list {D}. This representation is context insensitive and recombination of two tours always yields a valid tour. However, it has a context sensitive interpretation, since gene values to the right of a recombination "cut-point" specify different subtours in the parent and the child.

Radcliffe (1991) outlined three design principles for recombination. First, recombination operators should be "respectful." Respect occurs if crossing two instances of any forma (a generalization of schema) must produce another instance of that forma. For example, if both parents have blue eyes then all their children must have blue eyes. This principle holds for any standard recombination on bit-strings. Second, recombination should "properly assort" formae. This occurs if, given instances of two compatible formae, it must be possible to cross them to produce a child which is an instance of both formae. For example, if one parent has blue eyes and the other has brown hair, it must be possible to recombine them to produce a child with blue eyes and brown hair. This principle is similar to what others called "exploratory power" — e.g. uniform recombination can reach all points in the subspace defined by the differing bits (in one application), while n-point recombination cannot. Thus n-point recombination does not properly assort, while uniform recombination does. Finally, recombination should "strictly transmit." Strict transmission occurs if every allele in the child comes from one parent or another. For example, if one parent has blue eyes and the other has brown eyes, the child must have blue or brown eyes. All standard recombination operators for bit-strings strictly transmit genes.

All of this indicates that the creation and successful application of recombination operators is not "cut-and-dried," nor a trivial pursuit. Considerable effort and thought is required. However, if one uses the guidelines suggested above as a first-cut, success is more likely.

Acknowledgements

I thank Diana Gordon, Chad Peck, Mitch Potter, Ken De Jong, Peter Angeline, David Fogel, and the George Mason University GA Group for helpful comments on organizing this paper.

References

Altenberg L 1994 The schema theorem and price's theorem *Proceedings of the Third Foundations of Genetic Algorithms Workshop* eds. M Vose and D Whitley (San Mateo, CA: Morgan Kaufmann) 23–49

Angeline P 1996 Two self-adaptive crossover operations for genetic programming *Advances in Genetic Programming* **2** 89–110

Arabas J, Mulawka J and Pokrasniewicz J 1995 A new class of the crossover operators for the numerical optimization *Proceedings of the Sixth International Conference on Genetic Algorithms (Pittsburgh, PA)* ed. L Eshelman (San Francisco, CA: Morgan Kaufmann) 42–48

Beyer H-G 1995 Toward a theory of evolution strategies: on the benefits of sex — the $(\mu/\mu,\lambda)$ theory *EC* (Cambridge, MA: MIT Press) **3** **1** 81–112

 © 1995 IOP Publishing Ltd and Oxford University Press

Booker L B 1992 Recombination distributions for genetic algorithms *Proceedings of the Second Foundations of Genetic Algorithms Workshop* ed. D Whitley (San Mateo, CA: Morgan Kaufmann) 29–44

Booker L 1987 Improving search in genetic algorithms *Genetic Algorithms and Simulated Annealing* ed. L Davis (Los Altos, CA: Morgan Kaufmann) 61–73

Booker L B 1982 Intelligent behavior as an adaptation to the task environment *Ph.D dissertation University of Michigan Ann Arbor*

Bui T and Moon B 1995 On multi-dimensional encoding/crossover *Proceedings of the Sixth International Conference on Genetic Algorithms (Pittsburgh, PA)* ed. L Eshelman (San Francisco, CA: Morgan Kaufmann) 49–56

Davis L 1991 Hybridization and numerical representation *Handbook of Genetic Algorithms* ed. L Davis (NY: Van Nostrand Reinhold) 61–71

Davis L 1989 Adapting operator probabilities in genetic algorithms *Proceedings of the Third International Conference on Genetic Algorithms* ed J Schaffer (San Mateo, CA: Morgan Kaufmann) 61–69

Davis L 1985 Applying adaptive algorithms in epistatic domains *Proceedings of the International Joint Conference on Artificial Intelligence*

De Jong K 1975 Analysis of the behavior of a class of genetic adaptive systems *Ph.D Dissertation University of Michigan Ann Arbor*

De Jong K and Spears W 1992 A formal analysis of the role of multi-point crossover in genetic algorithms *Annals of Mathematics and Artificial Intelligence* (Switzerland: J C Baltzer A G Scientific Publishing Company) **5 1** 1–26

De Jong K and Spears W 1990 An analysis of of the interacting roles of population size and crossover in genetic algorithms *Proceedings of the International Conference on Parallel Problem Solving from Nature* eds. H.-P Schwefel and R Männer (Berlin: Springer-Verlag) 38–47

De Jong K, Spears W and Gordon D 1993 Using genetic algorithms for concept learning *Machine Learning* **13** 161–188

Eshelman L and Schaffer D 1994 Productive recombination and propagating and preserving schemata *Proceedings of the Third Foundations of Genetic Algorithms Workshop* eds. M Vose and D Whitley (San Mateo, CA: Morgan Kaufmann) 299–313

Eshelman L and D Schaffer 1992. Real-Coded Genetic Algorithms and Interval-Schemata *Proceedings of the Second Foundations of Genetic Algorithms Workshop* ed. D Whitley (San Mateo, CA: Morgan Kaufmann) 187–202

Eshelman L, Caruana R and Schaffer D 1989 Biases in the crossover landscape *Proceedings of the Third International Conference on Genetic Algorithms* ed J Schaffer (San Mateo, CA: Morgan Kaufmann) 10–19

Falkenauer E 1994 A new representation and operators for genetic algorithms applied to grouping problems *EC* (Cambridge, MA: MIT Press) **2 2** 123–144

Fogel L, Owens A and Walsh M 1966 *Artificial intelligence through simulated evolution* (NY: John Wiley)

Fraser A 1957 *Simulation of Genetic Systems by Automatic Digital Computers I Introduction* Australian Journal of Biological Science **10** 484–491

Fujiki C and Dickinson J 1987 Using the genetic algorithm to generate lisp source code to solve the prisoner's dilemma *Proceedings of the Second International Conference on Genetic Algorithms* ed. J Grefenstette (Hillsdale NJ: Lawrence Erlbaum) 236–240

Goldberg D and Lingle R 1985 Alleles loci and the traveling salesman problem *Proceedings of the First International Conference on Genetic Algorithms and their Applications* ed. J. Grefenstette 154–159

Goldberg D 1989 *Genetic algorithms in search optimization and machine learning* (MA: Addison-Wesley Reading)

Grefenstette J 1995 Virtual genetic algorithms: first results *Navy Center for Applied Research in AI Report AIC-95-013*

Grefenstette J 1986 Optimization of control parameters for genetic algorithms *IEEE Transactions on Systems Man and Cybernetics* **SMC-16 1 1** 122–128

Grefenstette J, Ramsey C and Schultz A 1990 Learning sequential decision rules using simulation models and competition *Machine Learning* **54** 355–381

Holland J 1975 *Adaptation in natural and artificial systems* (Ann Arbor: University of Michigan Press)

Hong I, Kahng A and Moon B 1995 Exploiting synergies of multiple crossovers: initial studies *Proceedings of the IEEE International Conference on Evolutionary Computation*

Kahng A and Moon B 1995 Towards more powerful recombinations *Proceedings of the Sixth International Conference on Genetic Algorithms (Pittsburgh, PA)* ed. L Eshelman (San Francisco, CA: Morgan Kaufmann) 96–103

Koza J 1994 *Genetic programming II: automatic discovery of reusable subprograms* (Cambridge MA: MIT Press)

© 1995 IOP Publishing Ltd and Oxford University Press

Julstrom B 1995 What have you done for me lately? adapting operator probabilities in a steady-state genetic algorithm *Proceedings of the Sixth International Conference on Genetic Algorithms (Pittsburgh, PA)* ed. L Eshelman (San Francisco, CA: Morgan Kaufmann) 81–87

Lee M and Takagi H 1993 Dynamic control of genetic algorithms using fuzzy logic techniques *Proceedings of the Fifth International Conference on Genetic Algorithms* ed. S Forrest (San Mateo CA: Morgan Kaufmann) 77–83

Levenick J 1995 Metabits: generic endogenous crossover control *Proceedings of the Sixth International Conference on Genetic Algorithms (Pittsburgh, PA)* ed. L Eshelman (San Francisco, CA: Morgan Kaufmann) 88–95

Manderick B, de Weger M and Spiessens P 1991 The genetic algorithms and the structure of the fitness landscape *Proceedings of the Fourth International Conference on Genetic Algorithms* eds. R Belew and L Booker (San Mateo CA: Morgan Kaufmann) 143–150

Moon B and Bui T 1994 Analyzing hyperplane synthesis in genetic algorithms using clustered schemata *Parallel Problem Solving from Nature — III Lecture Notes in Computer Science* **806** 108–118

Oliver I, Smith D and Holland J 1987 A study of permutation crossover operators on the traveling salesman problem *Proceedings of the Second International Conference on Genetic Algorithms* ed. J Grefenstette (Hillsdale NJ: Lawrence Erlbaum) 224–230

Peck C and Dhawan A 1995 Genetic algorithms as global random search methods: an alternative perspective *EC* (Cambridge, MA: MIT Press) **3 1** 39–80

Radcliffe N 1994 Fitness variance of formae and performance prediction *Proceedings of the Third Foundations of Genetic Algorithms Workshop* eds. M Vose and D Whitley (San Mateo, CA: Morgan Kaufmann) 51–72

Radcliffe N 1991 Forma analysis and random respectful recombination *Proceedings of the Fourth International Conference on Genetic Algorithms* eds. R Belew and L Booker (San Mateo CA: Morgan Kaufmann) 222–229

Rosca J 1995 Genetic programming exploratory power and the discovery of functions *Proceedings of the Fourth Annual Conference on Evolutionary Programming* 719–736

Rosenberg R 1967 Simulation of genetic populations with biochemical properties *Ph.D Dissertation University of Michigan Ann Arbor*

Schaffer J, Caruana R, Eshelman L 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* ed J Schaffer (San Mateo, CA: Morgan Kaufmann) 51–60

Schaffer J and Eshelman K 1991 On crossover as an evolutionarily viable strategy *Proceedings of the Fourth International Conference on Genetic Algorithms* eds. R Belew and L Booker (San Mateo CA: Morgan Kaufmann) 61–68

Schaffer J and Morishima A 1987 An adaptive crossover distribution mechanism for genetic algorithms *Proceedings of the Second International Conference on Genetic Algorithms* ed. J Grefenstette (Hillsdale NJ: Lawrence Erlbaum) 36–40

Schwefel H.-P 1995 *Evolution and Optimum Seeking* (NY: Wiley)

Smith S 1980 Flexible learning of problem solving heuristics through adaptive search *Proceedings of the 8th International Conference on Artificial Intelligence* 422–425

Spears W and De Jong K 1991 On the virtues of parameterized uniform crossover *Proceedings of the Fourth International Conference on Genetic Algorithms* eds. R Belew and L Booker (San Mateo CA: Morgan Kaufmann) 230–236

Spears W 1992 Crossover or Mutation? *Proceedings of the Second Foundations of Genetic Algorithms Workshop* ed. D Whitley (San Mateo, CA: Morgan Kaufmann) 221–237

Spears W 1995 Adapting crossover in evolutionary algorithms *Proceedings of the Fourth Annual Conference on Evolutionary Programming*

Srinivas M and Patnaik L 1994 Adaptive probabilities of crossover and mutation in genetic algorithms *IEEE Transactions on Systems Man and Cybernetics* **244** 656–667

Starkweather T, McDaniel S, Mathias K, Whitley D and Whitley C 1991 A comparison of genetic sequencing operators *Proceedings of the Fourth International Conference on Genetic Algorithms* eds. R Belew and L Booker (San Mateo CA: Morgan Kaufmann) 69–76

Syswerda G 1992 Simulated crossover in genetic algorithms *Proceedings of the Second Foundations of Genetic Algorithms Workshop* ed. D Whitley (San Mateo, CA: Morgan Kaufmann) 239–255

Syswerda G 1989 Uniform crossover in genetic algorithms *Proceedings of the Third International Conference on Genetic Algorithms* ed J Schaffer (San Mateo, CA: Morgan Kaufmann) 2–9

White T and Oppacher F 1994 Adaptive crossover using automata *Proceedings of the Parallel Problem Solving from Nature Conference* eds. Y Davidor, H.-P Schwefel and R Männer (NY: Springer Verlag)

Wilson S 1986 Classifier system learning of a boolean function *Research Memo RIS-27r Rowland Institute for Science*

 © 1995 IOP Publishing Ltd and Oxford University Press

Wright A 1991 Genetic algorithms for real parameter optimization *Proceedings of the Foundations of Genetic Algorithms Workshop* ed. G Rawlins (San Mateo, CA: Morgan Kaufmann) 205–218

© 1995 IOP Publishing Ltd and Oxford University Press