An Overview of Genetic Algorithms: Part 2, Research Topics

David Beasley*
Department of Computing Mathematics,
University of Wales College of Cardiff, Cardiff, CF2 4YN, UK

David R. Bull†
Department of Electrical and Electronic Engineering,
University of Bristol, Bristol, BS8 1TR, UK

Ralph R. Martin‡
Department of Computing Mathematics,
University of Wales College of Cardiff, Cardiff, CF2 4YN, UK

University Computing, 1993, **15**(4) 170–181.

1 Introduction

Genetic algorithms, and other closely related areas such as *genetic programming*, *evolution strategies* and *evolution programs*, are the subject of an increasing amount of research interest. This two-part article is intended provide an insight into this field.

In the first part of this article [BBM93a] we described the fundamental aspects of genetic algorithms (GAs). We explained their basic principles, such as task representation, fitness functions and reproduction operators. We explained how they work, and compared them with other search techniques. We described several practical aspects of GAs, and mentioned a number of applications.

In this part of the article we shall explore various more advanced aspects of GAs, many of which are the subject of current research.

2 Crossover techniques

The "traditional" GA, as described in Part 1 of this article, uses 1-point crossover, where the two mating chromosomes are each cut once at corresponding points, and the sections after the cuts exchanged. However, many different crossover algorithms have been devised, often involving more than one cut point. DeJong [DeJ75] investigated the effectiveness of multiple-point crossover, and concluded (as reported in [Gol89a, p119]) that 2-point crossover gives an improvement, but that adding further crossover points reduces the performance of the GA. The problem with adding additional crossover points is that building blocks are more likely to be disrupted. However, an advantage of having more crossover points is that the problem space may be searched more thoroughly.

2.1 2-point crossover

In 2-point crossover, (and multi-point crossover in general), rather than linear strings, chromosomes are regarded as *loops* formed by joining the ends together. To exchange a segment from one loop with that from another

*email: David.Beasley@cm.cf.ac.uk
†email: Dave.Bull@bristol.ac.uk
‡email: Ralph.Martin@cm.cf.ac.uk

loop requires the selection of *two* cut points, as shown in Figure 1. In this view, 1-point crossover can be seen

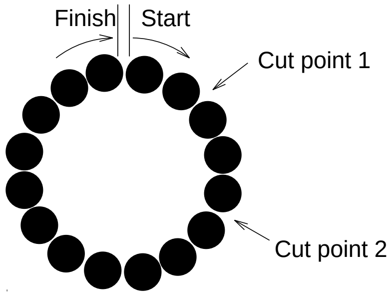

Figure 1: Chromosome Viewed as a Loop

as 2-point crossover with one of the cut points fixed at the start of the string. Hence 2-point crossover performs the same task as 1-point crossover (i.e. exchanging a single segment), but is more general. A chromosome considered as a loop can contain more building blocks—since they are able to "wrap around" at the end of the string. Researchers now agree that 2-point crossover is generally better than 1-point crossover.

2.2 Uniform crossover

Uniform crossover is radically different to 1-point crossover. Each gene in the offspring is created by copying the corresponding gene from one or the other parent, chosen according to a randomly generated *crossover mask*. Where there is a 1 in the crossover mask, the gene is copied from the first parent, and where there is a 0 in the mask, the gene is copied from the second parent, as shown in Figure 2. The process is repeated with the parents exchanged to produce the second offspring. A new crossover mask is randomly generated for each pair of parents.

Offspring therefore contain a mixture of genes from each parent. The number of effective crossing points is not fixed, but will average $L/2$ (where L is the chromosome length).

Crossover Mask	1 0 0 1 0 1 1 1 0 0
Parent 1	1 0 1 0 0 0 1 1 1 0
Offspring 1	1 1 0 0 0 0 1 1 1 1
Parent 2	0 1 0 1 0 1 0 0 1 1

Figure 2: Uniform Crossover

2.3 Which technique is best?

Arguments over which is the best crossover method to use still rage on. Syswerda [Sys89] argues in favour of uniform crossover. Under uniform crossover, schemata of a particular order[1] are equally likely to be disrupted, *irrespective of their defining length*.[2] With 2-point crossover, it is the defining length of the schemata which determines its likelihood of disruption, *not* its order. This means that under uniform crossover, although short defining length schemata are *more* likely to be disrupted, longer defining length schemata are comparatively *less* likely to be disrupted. Syswerda argues that the *total* amount of schemata disruption is therefore lower. Uniform crossover has the advantage that the ordering of genes is entirely irrelevant. This means that re-ordering operators such as inversion (see next section) are unnecessary, and we do not have to worry about positioning genes so as to promote building blocks. GA performance using 2-point crossover drops dramatically if the recommendations of the building block hypothesis [BBM93a] are not adhered to. Uniform crossover, on

[1]The *order* of a schema is the number of bit values it specifies.
[2]The *defining length* is the number of bit positions between the first and last specified bit.

the other hand, still performs well—almost as well as 2-point crossover used on a correctly ordered chromosome. Uniform crossover therefore appears to be more robust.

Eshelman *et al* [ECS89] did an extensive comparison of different crossover operators, including 1-point, 2-point, multi-point and uniform crossover. These were analysed theoretically in terms of positional and distributional bias, and empirically, on several problems. No overall winner emerged, and in fact there was not more than about 20% difference in speed among the techniques (so perhaps we should not worry too much about which is the best method). They found that 8-point crossover was good on the problems they tried.

Spears & DeJong [SD91] are very critical of multi-point and uniform crossover. They stick by the theoretical analyses which show 1- and 2-point crossover are optimal. They say that 2-point crossover will perform poorly when the population has largely converged, due to reduced *crossover productivity*. This is the ability of a crossover operator to produce new chromosomes which sample different points in the search space. Where two chromosomes are similar, the segments exchanged by 2-point crossover are likely to be identical—leading to offspring which are identical to their parents. This is less likely to happen with uniform crossover. They describe a new 2-point crossover operator such that if identical offspring are produced, two new cross points are chosen. (Booker [Boo87] introduced *reduced surrogate crossover* to achieve the same effect.) This operator was then found to perform better than uniform crossover on a test problem (but only slightly better).

In a slightly later paper, DeJong & Spears [DS90] conclude that modified 2-point crossover is best for large populations, but the increased disruption of uniform crossover is beneficial if the population size is small (in comparison to the problem complexity), and so gives a more robust performance.

2.4 Other crossover techniques

Many other techniques have been suggested. The idea that crossover should be more probable at some string positions than others has some basis in nature, and several such methods have been described [SM87, Hol87, Dav91a, Lev91, LR91]. The general principle is that the GA adaptively learns which sites should be favoured for crossover. This information is recorded in a *punctuation string*, which is itself part of the chromosome, and so is crossed over and passed on to descendants. In this way, punctuation strings which lead to good offspring will themselves be propagated through the population.

Goldberg [Gol85, Gol89a] describes a rather different crossover operator, *partially matched crossover* (PMX), for use in order-based problems. (In an order-based problem, such as the travelling salesperson problem, gene values are fixed, and the fitness depends on the *order* in which they appear.) In PMX it is not the *values* of the genes which are crossed, but the *order* in which they appear. Offspring have genes which inherit ordering information from each parent. This avoids the generation of offspring which violate problem constraints. Syswerda [Sys91] and Davis [Dav91d, p72] describe other order-based operators.

The use of problem specific knowledge to design crossover operators for a particular type of task is discussed in Section 13.

3 Inversion and Reordering

It was stated in Part 1 of this article that the *order* of genes on a chromosome is critical for the *building block hypothesis* to work effectively. Techniques for *reordering* the positions of genes in the chromosome during a run have been suggested. One such technique, *inversion* [Hol75], works by reversing the order of genes between two randomly chosen positions within the chromosome. (When these techniques are used, genes must carry with them some kind of "label", so that they may be correctly identified irrespective of their position within the chromosome.)

The purpose of reordering is to attempt to find gene orderings which have better evolutionary potential [Gol89a, p166]. Many researchers have used inversion in their work, although it seems few have attempted to justify it, or quantify its contribution. Goldberg & Bridges [GB90] analyse a reordering operator on a very small task, and show that it can bring advantages—although they conclude that their methods would not bring the same advantages on larger tasks.

Reordering does nothing to lower epistasis (see below), so cannot help with the other requirement of the building block hypothesis. Nor does it help if the relationships among the genes do not allow a simple linear ordering. If *uniform crossover* is used, gene order is irrelevant, so reordering is unnecessary. So, Syswerda [Sys89] argues, why bother with inversion?

Reordering also greatly expands the search space. Not only is the GA trying to find good sets of gene values, it is simultaneously trying to discover good gene orderings too. This is a far more difficult problem to solve.

Time spent trying to find better gene orderings may mean time taken away from finding good gene values.

In nature, there are many mechanisms by which the arrangement of the chromosome(s) may evolve (known as *karyotypic evolution*) [MS89]; inversion is only one of them. In the short term, organisms will be favoured if they evolve to become well adapted to their environment. But in the long term, species are only likely to survive if their karyotypic evolution is such that they can easily adapt to *new* conditions as the environment changes. Evaluation of the genotype takes place rapidly, in each generation. But evaluation of the karyotype takes place very slowly, perhaps over thousands of generations.

For the vast majority of GA applications the environment, as embodied in the fitness function, is static. Taking a hint from nature, it would seem that karyotypic evolution is therefore of little importance in these cases. However, in applications where the fitness function varies over time, and the GA must provide a solution which can adapt to the changing environment, karyotypic evolution may be worth employing.

In a static environment, if we really want to determine the best gene ordering (perhaps because we have a large number of problems, all with similar characteristics), we might try using a meta-GA, in the same way that Grefenstette [Gre86] used a meta-GA to determine a good set of GA parameters. A meta-GA has a population where each member is itself a GA. Each individual GA is configured to solve the same task, but using different parameters (in this case, different gene orderings). The fitness of each individual is determined by running the GA, and seeing how quickly it converges. Meta-GAs are obviously very computationally expensive to run, and are only likely to be worthwhile if the results they provide can be reused many times.

4 Epistasis

The term *epistasis* has been defined by geneticists as meaning that the influence of a gene on the fitness of an individual depends on what gene values are present elsewhere. [MS89] More specifically, geneticists use the term *epistasis* in the sense of a "masking" or "switching" effect. "A gene is said to be epistatic when its presence suppresses the effect of a gene at another locus.[3] Epistatic genes are sometimes called inhibiting genes because of their effect on other genes which are described as hypostatic." [GST90].

Generally, though, there will be far more subtle and complex interactions among large overlapping groups of genes. In particular, there are chains of influence—one gene codes for the production of a protein, which is then involved with a protein produced by another gene to produce a third product, which then reacts with other enzymes produced elsewhere ... and so on. Many genes simply produce intermediate proteins which are used by other processes initiated by other genes. So there is a considerable amount of "interaction" among genes in the course of producing the phenotype, although geneticists might not refer to this as *epistasis*.

When GA researchers use the term *epistasis*, they are generally referring to any kind of strong interaction among genes, not just masking effects, although they avoid giving a precise definition. While awaiting a definitive definition of epistasis in a GA context, we offer our own:

> Epistasis is the interaction between different genes in a chromosome. It is the extent to which the "expression" (i.e. contribution to fitness) of one gene depends on the values of other genes. The degree of interaction will, in general, be different for each gene in a chromosome. If a small change is made to one gene we expect a resultant change in chromosome fitness. This resultant change may vary according to the values of other genes. As a broad classification, we distinguish three levels of gene interaction. These depend on the extent to which the change in chromosome fitness resulting from a small change in one gene varies according to the values of other genes.
>
> - **Level 0—No interaction.** A particular change in a gene always produces the same change in fitness.
> - **Level 1—Mild interaction.** A particular change in a gene always produces a change in fitness of the same sign, or zero.
> - **Level 2—Epistasis.** A particular change in a gene produces a change in fitness which varies in sign and magnitude, depending on the values of other genes.

An example of a level 0 task is the trivial "counting ones task," where fitness is proportional to the number of 1s in the binary string. An example of a level 1 task is the "plateau function," where typically 5 bits are decoded such that the fitness is 1 if all bits are 1, and zero otherwise.

[3] The *locus* is the position within the chromosome.

When GA researchers use the term *epistasis*, they would generally be talking only about level 2. This is how we shall use the term, unless otherwise stated.

Tasks in which all genes are of type 0 or 1 can be solved efficiently by various simple techniques, such as hillclimbing, and do not require a GA [Dav91c]. GAs can, however, outperform simple techniques on more complex level 2 tasks exhibiting many interactions among the parameters—that is, with significant epistasis. Unfortunately, as has already been noted in Part 1 of this article, according to the *building block hypothesis*, one of the basic requirements for a GA to be successful is that there is *low* epistasis. This suggests that GAs will not be effective on precisely those type of problems in which they are most needed. Clearly, understanding epistasis is a key issue for GA research. We need to know whether we can either avoid it, or develop a GA which will work even with high epistasis. This is explored further below, but first we shall describe a related phenomenon.

5 Deception

One of the fundamental principles of GAs is that chromosomes which include schemata which are contained in the global optimum will increase in frequency (this is especially true of short, low-order schemata, known as *building blocks*). Eventually, via the process of crossover, these optimal schemata will come together, and the globally optimum chromosome will be constructed. But if schemata which are *not* contained in the global optimum increase in frequency *more* rapidly than those which *are*, the GA will be mislead, away from the global optimum, instead of towards it. This is known as *deception*.

Deception is a special case of epistasis, and it has been studied in depth by Goldberg [Gol87] [Gol89a, p46][DG91] and others. Deception is directly related to the detrimental effects of epistasis in a GA. Level 2 epistasis is necessary (but not sufficient) for deception.

Statistically, a schema will increase in frequency in the population if its fitness[4] is higher than the average fitness of all schemata in the population. A problem is referred to as *deceptive* if the average fitness of schemata which are *not* contained in the global optimum is greater than the average fitness of those which are. Furthermore, a problem is referred to as *fully* deceptive if "all low-order schemata containing a suboptimal solution are better than other competing schemata" [DG91].

Deceptive problems are difficult to solve. However, Grefenstette [Gre93] cleverly demonstrates that this is not *always* the case. After the first generation, a GA does not get an *unbiassed* sample of points in the search space. Therefore it cannot estimate the global, unbiassed average fitness of a schema. It can only get a *biassed* estimate of schema fitness. Sometimes this bias helps the GA to converge (even though a problem might otherwise be highly deceptive), and other times the bias might prevent the GA converging (even though the problem is not formally deceptive). Grefenstette gives examples of both situations.

6 Tackling epistasis

The problems of epistasis (described above) may be tackled in two ways: as a coding problem, or as a GA theory problem. If treated as a coding problem, the solution is to find a different coding (representation) and decoding method which does not exhibit epistasis. This will then allow a conventional GA to be used. If this cannot be done, the second approach may have to be used.

Vose & Liepins [VL91] show that in principle any problem can be coded in such a way as to make it as simple as the "counting ones task". Similarly, any coding can be made simple for a GA by using appropriately designed crossover and mutation operators. So it is always possible to represent any problem with little or no epistasis. However, for "difficult" problems, the effort involved in devising such a coding will be considerable, and will effectively constitute "solving" the initial problem.

Traditional GA theory, based on the schema theorem, relies on low epistasis. If genes in a chromosome have high epistasis, a new theory may have to be developed, and new algorithms developed to cope with this. The inspiration may once again come from natural genetics, where epistasis (in the GA sense) is very common.

Davis [Dav85a] considers both these approaches. He converts a bin-packing problem, where the optimum *positions* for packing rectangles into a space must be found, into an *order* problem, where the *order* of packing the rectangles had to be found instead. A key part of this is an intelligent decoding algorithm, which uses domain knowledge to find "sensible" positions for each rectangle, in the order specified by the chromosome. This reduces the epistasis in the chromosome. Once the problem has been converted to an order-based one,

[4] The fitness of a *schema* is the average, or expected fitness of chromosomes which contain that schema.

a modified GA theory is required. Goldberg [Gol85] describes how GA theory can be adapted to encompass order-based problems. He introduces the idea of order-schemata (o-schemata), and the PMX crossover method which processes o-schemata in an analogous way to conventional crossover and normal schemata.

Davis & Coombs [DC87] point out that GAs have been made to work even in domains of high epistasis. So, although Holland's convergence proof for a GA assumed low epistasis, there may be another, perhaps weaker, convergence proof for domains of high epistasis. Even rigourous definitions of "low epistasis" and "high epistasis" have yet to be formulated.

Davidor [Dav90] has attempted to develop a technique which allows the degree of epistasis in a problem to be measured. Unfortunately an accurate assessment of epistasis can only be made with a time complexity comparable to an exhaustive search of the problem space. This can be reduced by sampling, but then the results are considerably less accurate—especially for problems with high epistasis.

Davidor also points out that present-day GAs are only suitable for problems of medium epistasis. If the epistasis is very high, the GA will not be effective. If it is very low, the GA will be outperformed by simpler techniques, such as hillclimbing. Until such a time as we have GAs which are effective on problems of high epistasis, we must devise representation schemes (or crossover/mutation operators) which reduce epistasis to an acceptable level.

A technique for achieving this, *expansive coding,* is presented by Beasley, Bull & Martin [BBM93b]. Expansive coding is a technique for designing reduced-epistasis representations for combinatorial problems. Rather than having a representation consisting of a small number of widely interacting genes, a representation is created with a much larger number of more weakly interacting genes. This produces a search space which is larger, yet simpler and more easily solved. They demonstrate that this technique can design reduced complexity algorithms for signal processing.

7 Mutation and Naïve Evolution

Mutation is traditionally seen as a "background" operator [Boo87, p63][DeJ85], responsible for re-introducing inadvertently "lost" gene values (alleles), preventing genetic drift, and providing a small element of random search in the vicinity of the population when it has largely converged. It is generally held that crossover is the main force leading to a thorough search of the problem space.

However, examples in nature show that asexual reproduction can evolve sophisticated creatures *without* crossover—for example bdelloid rotifers [MS89, p239]. Indeed, biologists see mutation as the *main* source of raw material for evolutionary change [Har88, p137]. Schaffer *et al* [SCLD89] did a large experiment to determine optimum parameters for GAs. They found that crossover had much less effect on performance than previously believed. They suggest that "naïve evolution" (just selection and mutation) performs a hillclimb-like search which can be powerful without crossover. They investigate this hypothesis further [SE91], and find that crossover gives much faster evolution than a mutation-only population. However, in the end, mutation generally finds better solutions than a crossover-only regime.

This is in agreement with Davis [Dav91d], who points out that mutation becomes more productive, and crossover less productive, as the population converges.

Despite its generally low probability of use, mutation is a very important operator. Its optimum probability is much more critical than that for crossover [SCLD89]. Even if it is a "background operator," it should not be ignored.

Spears [Spe93] closely compares crossover and mutation, and argues that there are some important characteristics of each operator that are not captured by the other. He further suggests that a suitably modified mutation operator can do everything that crossover can. He concludes that "standard mutation and crossover are simply two forms of a more general exploration operator, that can perturb alleles based on any available information."

Other good performances of naïve evolution have been reported [EOR91, ES91, Esh91]. According to Eshelman [Esh91], "The key to naïve evolution's success (assuming a bit-string representation) is the use of Gray coded parameters, making search much less susceptible to Hamming cliffs. ... I do believe that naïve evolution is a much more powerful algorithm than many people in the GA community have been willing to admit."

8 Non-binary representations

A chromosome is a sequence of symbols, and, traditionally, these symbols have been binary digits, so that each symbol has a cardinality of 2. Higher cardinality alphabets have been used in some research, and some believe them to have advantages. Goldberg [Gol89a, p80][Gol89b] argues that theoretically, a binary representation gives the largest number of schemata, and so provides the highest degree of *implicit parallelism*. But Antonisse [Ant89], interprets schemata differently, and concludes that, on the contrary, high-cardinality alphabets contain more schemata than binary ones. (This has been the subject of more recent discussion [Ang92, Ant92].) Goldberg has now developed a theory which explains why high-cardinality representations can perform well [Gol90]. His theory of *virtual alphabets* says that each symbol converges within the first few generations, leaving only a small number of possible values. In this way, each symbol effectively has only a low cardinality.

Empirical studies of high-cardinality alphabets have typically used chromosomes where each symbol represents an integer [Bra91], or a floating-point number [JM91, MJ91]. As Davis [Dav91d, p65] points out, problem parameters are often numeric, so representing them directly as numbers, rather than bit-strings, seems obvious, and may have advantages. One advantage is that we can more easily define meaningful, problem-specific "crossover" and "mutation" operators. A variety of real-number operators can easily be envisaged, for example:

- Combination operators
  - *Average*—take the arithmetic average of the two parent genes.
  - *Geometric mean*—take the square-root of the product of the two values.
  - *Extension*—take the difference between the two values, and add it to the higher, or subtract it from the lower.
- Mutation operators
  - *Random replacement*—replace the value with a random one
  - *Creep*—add or subtract a small, randomly generated amount.
  - *Geometric creep*—multiply by a random amount close to one.

  For both creep operators, the randomly generated number may have a variety of distributions; uniform within a given range, exponential, Gaussian, binomial, etc.

Janikow & Michalewicz [JM91] made a direct comparison between binary and floating-point representations, and found that the floating-point version gave faster, more consistent, and more accurate results.

However, where problem parameters are *not* numeric, (for example in combinatorial optimisation problems), the advantages of high-cardinality alphabets may be harder to realise.

In GA-digest[5] volume 6 number 32 (September 1992), the editor, Alan C. Schultz, lists various research using non-binary representations. These include Grefenstette's work which uses a rule-based representation to learn reactive strategies (or behaviours) for autonomous agents [SG90, Gre91]. Koza is using a process known as *genetic programming* to learn Lisp programs [Koz92]. Floating point representations have been widely explored [Whi89, JM91, MJ91, ES93], and Michalewicz has looked at a matrix as the data structure [Mic92].

9 Dynamic Operator Probabilities

During the course of a run, the optimal value for each operator probability may vary. Davis [Dav85b] tried linear variations in crossover and mutation probability, with crossover decreasing during the run, and mutation increasing (see above). Syswerda [Sys91] also found this advantageous. However, it imposes a fixed schedule. Booker [Boo87] utilises a dynamically variable crossover rate, depending on the spread of fitnesses. When the population converges, the crossover rate is reduced to give more opportunity for mutation to find new variations. This has a similar effect to Davis's linear technique, but has the advantage of being adaptive.

Davis [Dav89, Dav91d] describes another adaptive technique which is based directly on the success of an operator at producing good offspring. Credit is given to each operator when it produces a chromosome better than any other in the population. A weighting figure is allocated to each operator, based on its performance over the past 50 matings. For each reproductive event, a single operator is selected probabilistically, according

[5] GA-digest is distributed free by electronic mail. Contact GA-List-Request@AIC.NRL.NAVY.MIL to subscribe. Back issues are available by anonymous ftp from: ftp.aic.nrl.navy.mil (in /pub/galist).

to the current set of operator weightings. During the course of a run, therefore, operator probabilities vary in an adaptive, problem dependent way. A big advantage of this technique is that it allows new operators to be compared directly with existing ones. If a new operator consistently loses weight, it is probably less effective than an existing operator.

This is a very interesting technique. It appears to solve a great many problems about choosing operator probabilities at a stroke. It also allows new representations and new techniques to be tried without worrying that much effort must be expended on determining new optimum parameter values. However, a potential drawback of this technique which must be avoided is that it may reward operators which simply locate local optima, rather than helping to find the global optimum.

Going in the opposite direction, several researchers vary the mutation probability by *decreasing* it exponentially during a run [Ack87, Bra91, Fog89, MJ91]. Unfortunately, no clear analysis or reasoning is given as to why this should lead to an improvement (although Fogarty [Fog89] provides experimental evidence). The motivation seems to be that mutation probability is analogous to temperature in simulated annealing, and so mutation rate should be reduced to a low value to aid convergence. However, in Ackley's case [Ack87], probability is varied from 0.25 to 0.02, and most would say that 0.02 is still a rather high value for mutation probability. Ackley does not appear to have thought this through. Fogarty does not say whether he thinks that the improvements he found would apply in other problem areas.

Arguments over whether the trajectory of the mutation probability should increase, decrease, be linear or exponential, become academic if Davis's adaptive algorithm is used.

10 Niche and Speciation

In natural ecosystems, there are many different ways in which animals may survive (grazing, hunting, on the ground, in trees, etc.), and different species evolve to fill each ecological niche. *Speciation* is the process whereby a single species differentiates into two (or more) different species occupying different niches.

In a GA, niches are analogous to maxima in the fitness function. Sometimes we have a fitness function which is known to be multimodal, and we may want to locate all the peaks. Unfortunately a traditional GA will not do this; the whole population will eventually converge on a single peak. Of course, we would expect the population of a GA to converge on a peak of high fitness, but even where there are several peaks of equal fitness, the GA will still end up on a single one. This is due to *genetic drift* [GR87]. Several modifications to the traditional GA have been proposed to solve this problem, all with some basis in natural ecosystems [GR87]. The two basic techniques are to maintain diversity, or to share the payoff associated with a niche.

Cavicchio [GR87] introduced a mechanism he called *preselection*, where offspring replace the parent only if the offspring's fitness exceeds that of the inferior parent. There is fierce competition between parents and children, so the payoff is not so much shared as fought over, and the winner takes all. This method helps to maintain diversity (since strings tend to replace others which are similar to themselves), and this helps prevent convergence on a single maximum.

DeJong [DeJ75] generalised preselection in his *crowding* scheme. In this, offspring are compared with a few (typically 2 or 3) randomly chosen individuals from the population. The offspring replaces the most similar one found, using Hamming distance as the similarity measure. This again aids diversity, and indirectly encourages speciation. Stadnyk [Sta87] found better results using a variation on this. The sampling of individuals was biassed according to inverse fitness, so that new offspring replace others which are in the same niche *and* have low fitness.

Booker [Boo85] uses *restricted mating* to encourage speciation. In this scheme, individuals are only allowed to mate if they are similar. The total reward available in any niche is fixed, and is distributed using a bucket-brigade mechanism. Booker's application is a classifier system, where it is easy to identify which niche an individual belongs to. In other applications, this is generally not a simple matter.

Perry [GR87] solves the species membership problem using a similarity template called an *external schema*. However, this scheme requires advance knowledge of where the niches are, so is of limited use.

Grosso [GR87] simulates partial geographical isolation in nature by using multiple subpopulations and intermediate migration rates. This shows advantages over isolated subpopulations (no migration—equivalent to simply iterating the GA), and completely mixed (panmictic) populations. This is an ideal method for use on a parallel processor system. (Fourman [Fou85] proposed a similar scheme.) However, there is no mechanism for explicitly preventing two or more subpopulations converging on the same niche.

Davidor [Dav91b] used a similar approach, but instead of multiple subpopulations, the population was considered as spread evenly over a two-dimensional grid. A *local mating* scheme was used, achieving a similar effect to multiple subpopulations, but without any explicit boundaries. Davidor found that for a while a wider diversity was maintained (compared with a panmictic population), but eventually the whole population converged to a single solution. Although Davidor describes this as "A naturally occurring niche and species phenomenon", we would argue that he has misused the term "niche". In nature, species only come into direct competition with each other if they are in the same niche. Since Davidor's GA eventually converges to a single species, there can only be one niche.

Goldberg & Richardson [GR87] describe the advantages of *sharing*. Several individuals which occupy the same niche are made to *share* the fitness payoff among them. Once a niche has reached its "carrying capacity," it no longer appears rewarding in comparison with other, unfilled niches. The difficulty with sharing payoff within a niche is that the boundaries of the niche are not easily identified. Goldberg uses a *sharing function* to define how the sharing is to be done. Essentially, the payoff given to an individual is reduced according to a function (a power law) of the "distance" of each neighbour. The distance may be measured in different ways, for example in terms of genotype Hamming distance, or parameter differences in the phenotype. In a 1-dimensional task, this method was shown to be able to distribute individuals to peaks in the fitness function in proportion to the height of the peak.

In a later continuation of this work, Deb & Goldberg [DG89] show that sharing is superior to crowding. Genotypic sharing (based on some distance measure between chromosome strings) and phenotypic sharing (based on the distance between the decoded parameters) are analysed. Phenotypic sharing is shown to have advantages. A sharing function based on Euclidian distance between neighbours implements niches which are hyperspherical in shape. The correct operation of the sharing scheme depends on using the appropriate radius for the hyperspheres. (The radius is the maximum distance between two chromosomes for them still to be considered in the same niche.) The paper gives formulae for computing this, assuming that the number of niches is known, and that they are evenly distributed throughout the solution space.

A mating restriction scheme was also implemented to reduce the production of *lethals* (see Section 11). This only allowed an individual to mate with another from the same (phenotypic) niche (or at random only if there was no other individual in the niche). This showed a significant improvement.

A difficulty arises with niche methods if there are many local maxima with fitnesses close to the global maximum [GDH92]. A technique which distributes population members to peaks in proportion to the fitness of the peak, as the methods described above do, will not be likely to find the global maximum if there are more peaks than population members. Crompton & Stephens [CS91] found that on a real problem, the introduction of niche formation by crowding gave no improvement.

Deb's assumption that the function maxima are evenly distributed gives the *upper bound* on the niche radius, and better results might be obtained using a smaller value. If all the function maxima were clumped together, we would expect the performance to be little better than a GA without sharing. One solution might be to iterate the GA, trying different values for niche radius. An optimum scheme for this could be worth investigating.

A different approach to sharing is described by Beasley, Bull & Martin [BBM93c]. Their *sequential niche* method involves multiple runs of a GA, each locating one peak. After a peak has been located, the fitness function is modified so that the peak is effectively "cancelled out" from the fitness function. This ensures that, on subsequent runs, the same peak will not be re-discovered. The GA is then restarted with a new population. In this way, a new peak is located on each run. This technique has many similarities with fitness sharing. However, instead of the fitness of an individual being reduced (i.e. shared) because of its proximity to other members of the population, individuals have their fitness reduced because of their proximity to peaks located in previous runs. This method has a lower time complexity than that of fitness sharing, but suffers similar problems with regard to choice of niche radius, etc.

11 Restricted Mating

The purpose of restricted mating is to encourage speciation, and reduce the production of *lethals*. A lethal is a child of parents from two different niches. Although each parent may be highly fit, the combination of their chromosomes may be highly unfit if it falls in the valley between the two maxima. Nature avoids the formation of lethals by preventing mating between different species, using a variety of techniques. (In fact, this is the primary biological definition of a "species"—a set of individuals which may breed together to produce viable offspring.)

The general philosophy of restricted mating makes the assumption that if two similar parents (i.e. from the same niche) are mated, then the offspring will be similar. However, this will very much depend on the coding scheme—in particular the existence of building blocks, and low epistasis. Under conventional crossover and mutation operators, two parents with similar genotypes will always produce offspring with similar genotypes. But in a highly epistatic chromosome, there is no guarantee that these offspring will not be of low fitness, i.e. "lethals". Similarity of genotype does not guarantee similarity of phenotype. These effects limit the use of restricted mating.

Restricted mating schemes of Booker [Boo85] and Deb & Goldberg [DG89] have been described above. These restrict mating on the basis of similarities between the genotypes or phenotypes. Other schemes which restrict mating using additional mating template codes (for example [Hol87, p88]) are summarised by Goldberg [Gol89a, p192].

12 Diploidy and Dominance

In the higher lifeforms, chromosomes contain *two* sets of genes, rather than just one. This is known as *diploidy*. (A *haploid* chromosome contains only one set of genes.) Most genetics textbooks tend to concentrate on diploid chromosomes, while virtually all work on GAs concentrates on haploid chromosomes. This is primarily for simplicity, although use of diploid chromosomes might have benefits.

Diploid chromosomes lend advantages to individuals where the environment may change over a period of time. Having two genes allows two different "solutions" to be remembered, and passed on to offspring. One of these will be *dominant* (that is, it will be expressed in the phenotype), while the other will be recessive. If environmental conditions change, the dominance can shift, so that the other gene is dominant. This shift can take place much more quickly than would be possible if evolutionary mechanisms had to alter the gene. This mechanism is ideal if the environment regularly switches between two states (e.g. ice-age, non ice-age).

The primary advantage of diploidy is that it allows a wider diversity of alleles to be kept in the population, compared with haploidy. Currently harmful, but potentially useful alleles can still be maintained, but in a recessive position. Other genetic mechanisms could achieve the same effect. For example, a chromosome might contain several variants of a gene. Epistasis (in the sense of masking) could be used to ensure that only one of the variants were expressed in any particular individual. A situation like this occurs with haemoglobin production [MS89]. Different genes code for its production during different stages of development. During the foetal stage, one gene is switched on to produce haemoglobin, whilst later on a different gene is activated. There are a variety of biological metaphors we can use to inspire our development of GAs.

In a GA, diploidy might be useful in an on-line application where the system could switch between different states. Diploidy involves a significant overhead in a GA. As well as carrying twice as much genetic information, the chromosome must also carry dominance information. There are probably other mechanisms we can use to achieve similar results (for example, keep a catalogue of the best individuals, and try reintroducing them into the population if performance falls). Little work seems to have been done in this area—Goldberg [Gol89a, p148] provides a summary.

13 Knowledge-based Techniques

While most research has gone into GAs using the traditional crossover and mutation operators, some have advocated designing new operators for each task, using domain knowledge [Dav91d]. This makes each GA more task specific (less robust), but may improve performance significantly. Where a GA is being designed to tackle a real-world problem, and has to compete with other search and optimisation techniques, the incorporation of domain knowledge often makes sense.

Suh & Van Gucht [SVG87] and Grefenstette [Gre87] argue that problem-specific knowledge can usefully be incorporated into the crossover operation. Domain knowledge may be used to prevent obviously unfit chromosomes, or those which would violate problem constraints, from being produced in the first place. This avoids wasting time evaluating such individuals, and avoids introducing poor performers into the population.

For example, Davidor [Dav91a] designed "analogous crossover" for his task in robotic trajectory generation. This used local information in the chromosome (i.e. the values of just a few genes) to decide which crossover sites would be certain to yield unfit offspring.

Domain knowledge can also be used to design *local improvement operators*, which allow more efficient exploration of the search space around good points [SVG87]. It can also be used to perform *heuristic initialisation* of the population, so that search begins with some reasonably good points, rather than a random set [Gre87, SG90].

Goldberg [Gol89a, p201–6] describes techniques for adding knowledge-directed crossover and mutation. He also discusses the hybridisation of GAs with other search techniques (as does Davis [Dav91d]).

14 Redundant Value Mapping

A problem occurs when a gene may only have a finite number of discrete valid values. If a binary representation is used, and the number of values is not a power of 2, then some of the binary codes are redundant—they will not correspond to any valid gene value. For example, if a gene represents an object to be selected from a group of 10 objects, then 4 bits will be needed to encode the gene. If codes 0000 to 1001 are used to represent the 10 objects, what do the codes 1010 to 1111 represent?

During crossover and mutation, we cannot guarantee that such redundant codes will not arise. The problem is, what to do about them? This problem has not been greatly studied in the literature (perhaps because most research concentrates on continuous-valued functions, where the problem does not arise). A number of solutions are briefly mentioned by DeJong [DeJ85]:

1. Discard the chromosome as illegal.

2. Assign the chromosome low fitness.

3. Map the invalid code to a valid one.

Solutions 1) and 2) would be expected to give poor performance, since we may be throwing away good gene values elsewhere in the chromosome. There are several ways of achieving 3), including fixed remapping, and random remapping.

In fixed remapping, a particular redundant value is remapped to a specific valid value. (In this case, remapped means that either the actual gene bit pattern is altered, or the decoding process treats the two bit patterns as synonymous.) This is very simple, but has the disadvantage that some values are represented by two bit patterns, while the others are represented by only one. (In the example above, the codes for 10 to 15 may be mapped back to the values 0 to 5, so these values are doubly represented in the code set, while the values 6 to 9 are singly represented).

In random remapping, a redundant value is remapped to a valid value at random. This avoids the representational bias problem, but also causes less information to be passed on from parents to offspring.

Probabilistic remapping is a hybrid between these two techniques. Every gene value (not just the "excess" ones) is remapped to one of two valid values in a probabilistic way, such that each valid value is equally likely to be represented.

Schaffer [Sch85] encountered the simplest version of this problem—three valid states represented by 2 bits. He used fixed remapping—allowing one state to have two binary representations. He also tried using ternary coding to avoid the problem, but performance was inferior.

Belew [Bel89] also used fixed remapping to solve the three-state problem. He points out that not only does one state have *two* representations (while the other two states have only one each), but also that the effective mutation rate for this state is halved (since mutations to one of the bits don't change the state). "There may be opportunities for a GA to exploit this representational redundancy," says Belew.

15 Summary

The two parts of this article have introduced the fundamental principles of GAs, and explored some of the current research topics in more detail. In the past, much research has been empirical, but gradually, theoretical insights are being gained. In many cases it is still too early to say which techniques are robust and general-purpose, and which are special-purpose. Where special-purpose techniques have been identified, work is still required to determine whether these can be extended to make them more general, or further specialised to make them more powerful. Theoretical research can greatly help progress in this area.

Davis [Dav91d] describes a variety of promising ideas. Steady state replacement, fitness ranking, and 2-point crossover (modified so that offspring must differ from their parents) are often good methods to use, although with suitable parent selection techniques, generational replacement may be equally as good [GD91], and uniform crossover can have advantages.

Knowledge-based operators and dynamic operator probabilities are probably going to help solve real world problems. Niche formation still seems like a big problem to be solved—how can all the 'best' maxima be located, while avoiding the not-so-good maxima, which may have only a slightly lower fitness? Ultimately, if the fitness function has very many local maxima, no search technique is ever going to perform well on it. Better methods for designing fitness functions are needed, which can avoid such pitfalls. Similarly, the difficulties of high epistasis must be addressed. Either we must find ways to represent problems which minimise their epistasis, or we must develop enhanced techniques which can cope even where there is high epistasis. There is no doubt that research into GAs will be a very active area for some time to come.

References

[Ack87] D.H. Ackley. An empirical study of bit vector function optimization. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 13, pages 170–204. Pitman, 1987.

[Ang92] Peter J. Angeline. Antonisse's extension to schema notation. *GA-Digest*, 6(35):–, October 1992.

[Ant89] J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 86–91. Morgan Kaufmann, 1989.

[Ant92] Jim Antonisse. Re: Antonisse's extension to schema notation. *GA-Digest*, 6(37):–, November 1992.

[BBM93a] D. Beasley, D.R. Bull, and R.R. Martin. An overview of genetic algorithms: Part 1, fundamentals. *University Computing*, 15(2):58–69, 1993.

[BBM93b] D. Beasley, D.R. Bull, and R.R. Martin. Reducing epistasis in combinatorial problems by expansive coding. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 400–407. Morgan Kaufmann, 1993.

[BBM93c] D. Beasley, D.R. Bull, and R.R. Martin. A sequential niche technique for multimodal function optimization. *Evolutionary Computation*, 1(2):101–125, 1993.

[Bel89] R.K. Belew. When both individuals and popluations search: adding simple learning to the genetic algorithm. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 34–41. Morgan Kaufmann, 1989.

[Boo85] L. Booker. Improving the performance of genetic algorithms in classifier systems. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 80–92. Lawrence Erlbaum Associates, 1985.

[Boo87] L. Booker. Improving search in genetic algorithms. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 5, pages 61–73. Pitman, 1987.

[Bra91] M.F. Bramlette. Initialisation, mutation and selection methods in genetic algorithms for function optimization. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 100–107. Morgan Kaufmann, 1991.

[CS91] W. Crompton and N.M. Stephens. Using genetic algorithms to search for binary sequences with large merit factor. In *Proc. Third IMA Conf on Cryptography and Coding*, pages –, 1991. Not yet published.

[Dav85a] L. Davis. Applying adaptive algorithms to epistatic domains. In *9th Int. Joint Conf. on AI*, pages 162–164, 1985.

[Dav85b] L. Davis. Job shop scheduling with genetic algorithms. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 136–140. Lawrence Erlbaum Associates, 1985.

[Dav89] L. Davis. Adapting operator probabilities in genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69. Morgan Kaufmann, 1989.

[Dav90] Y. Davidor. Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383, 1990.

[Dav91a] Y. Davidor. A genetic algorithm applied to robot trajectory generation. In L. Davis, editor, *Handbook of Genetic Algorithms*, chapter 12, pages 144–165. Van Nostrand Reinhold, 1991.

[Dav91b] Y. Davidor. A naturally occuring niche and species phenomenon: the model and first results. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 257–263. Morgan Kaufmann, 1991.

[Dav91c] L. Davis. Bit climbing, representational bias and test suite design. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 18–23. Morgan Kaufmann, 1991.

[Dav91d] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[DC87] L. Davis and S. Coombs. Genetic algorithms and communication link speed design: theoretical considerations. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 252–256. Lawrence Erlbaum Associates, 1987.

[DeJ75] K. DeJong. *The Analysis and behaviour of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.

[DeJ85] K. DeJong. Genetic algorithms: A 10 year perspective. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 169–177. Lawrence Erlbaum Associates, 1985.

[DG89] K. Deb and D.E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50. Morgan Kaufmann, 1989.

[DG91] K. Deb and D.E. Goldberg. Analyzing deception in trap functions. Technical Report IlliGal 91009, Illigal, December 1991.

[DS90] K. DeJong and W.M. Spears. An analysis of the interacting roles of population size and crossover in genetic algorithms. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, pages 38–47. Springer-Verlag, 1990.

[ECS89] L.J. Eshelman, R. Caruna, and J.D. Schaffer. Biases in the crossover landscape. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 10–19. Morgan Kaufmann, 1989.

[EOR91] Christer Ericson and Ivan Ordonez-Reinoso. Dialogue on uniform crossover. *GA-Digest*, 5(33):–, October 1991.

[ES91] Larry J. Eshelman and J. David Schaffer. GAs and very fast simulated re-annealing. *GA-Digest*, 5(37):–, December 1991.

[ES93] Larry J. Eshelman and J. David Schaffer. Real-coded genetic algorithms and interval schemata. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms, 2*, pages 187–202. Morgan Kaufmann, 1993.

[Esh91] Larry J. Eshelman. Bit-climbers and naive evolution. *GA-Digest*, 5(39):–, December 1991.

[Fog89] T.C. Fogarty. Varying the probability of mutation in the genetic algorithm. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 104–109. Morgan Kaufmann, 1989.

[Fou85] M.P. Fourman. Compaction of symbolic layout using genetic algorithms. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 141–153. Lawrence Erlbaum Associates, 1985.

[GB90] D.E. Goldberg and C.L. Bridges. An analysis of a reordering operator on a GA-hard problem. *Biological Cybernetics*, 62:397–405, 1990.

[GD91] D.E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.

[GDH92] D.E. Goldberg, K. Deb, and J. Horn. Massive multimodality, deception, and genetic algorithms. In R. Männer and B. Manderick, editors, *Parallel Problem Solving from Nature, 2*, pages 37–46. North-Holland, 1992.

[Gol85] D.E. Goldberg. Alleles, loci, and the TSP. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 154–159. Lawrence Erlbaum Associates, 1985.

[Gol87] D.E. Goldberg. Simple genetic algorithms and the minimal, deceptive problem. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 6, pages 74–88. Pitman, 1987.

[Gol89a] D.E. Goldberg. *Genetic Algorithms in search, optimization and machine learning.* Addison-Wesley, 1989.

[Gol89b] D.E. Goldberg. Zen and the art of genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 80–85. Morgan Kaufmann, 1989.

[Gol90] D.E. Goldberg. The theory of virtual alphabets. In H.-P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature*, pages 13–22. Springer-Verlag, 1990.

[GR87] D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Lawrence Erlbaum Associates, 1987.

[Gre86] J.J. Grefenstette. Optimization of control parameters for genetic algorithms. *IEEE Trans SMC*, 16:122–128, 1986.

[Gre87] J.J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, chapter 4, pages 42–60. Pitman, 1987.

[Gre91] J.J. Grefenstette. Strategy acquisition with genetic algorithms. In L. Davis, editor, *Handbook of Genetic Algorithms*, chapter 14, pages 186–201. Van Nostrand Reinhold, 1991.

[Gre93] John J. Grefenstette. Deception considered harmful. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms, 2*, pages 75–91. Morgan Kaufmann, 1993.

[GST90] N.P.O. Green, G.W. Stout, and D.J. Taylor. *Biological Science 1 & 2.* Cambridge University Press, 1990.

[Har88] D.L. Hartl. *A primer of population genetics.* Sinauer Associates Inc., 1988.

[Hol75] J.H. Holland. *Adaptation in Natural and Artificial Systems.* MIT Press, 1975.

[Hol87] J.H. Holland. Genetic algorithms and classifier systems: foundations and future directions. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 82–89. Lawrence Erlbaum Associates, 1987.

[JM91] C.Z. Janikow and Z. Michalewicz. An experimental comparison of binary and floating point representations in genetic algorithms. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 31–36. Morgan Kaufmann, 1991.

[Koz92] John R. Koza. *Genetic Programming: On The Programming Of Computers By Means Of Natural Selection.* MIT Press, 1992.

[Lev91] J. Levenick. Inserting introns improves genetic algorithm success rate: taking a cue from biology. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 123–127. Morgan Kaufmann, 1991.

[LR91] S.J. Louis and G.J.E. Rawlins. Designer genetic algorithms: Genetic algorithms in structure design. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 53–60. Morgan Kaufmann, 1991.

[Mic92] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1992.

[MJ91] Z. Michalewicz and C.Z. Janikow. Handling constraints in genetic algorithms. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 151–157. Morgan Kaufmann, 1991.

[MS89] J. Maynard Smith. *Evolutionary Genetics*. Oxford University Press, 1989.

[Sch85] J.D. Schaffer. Learning multiclass pattern discrimination. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 74–79. Lawrence Erlbaum Associates, 1985.

[SCLD89] J.D. Schaffer, R.A. Caruna, Eshelman L.J., and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 51–60. Morgan Kaufmann, 1989.

[SD91] W.M. Spears and K. DeJong. An analysis of multi-point crossover. In G.J.E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 301–315. Morgan Kaufmann, 1991.

[SE91] J.D. Schaffer and L.J. Eshelman. On crossover as an evolutionarily viable strategy. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann, 1991.

[SG90] A.C. Schultz and J.J. Grefenstette. Improving tactical plans with genetic algorithms. In *Proc. IEEE Conf. Tools for AI*, pages 328–344. IEEE Society Press, 1990.

[SM87] J.D. Schaffer and A. Morishma. An adaptive crossover distribution mechanism for genetic algorithms. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 36–40. Lawrence Erlbaum Associates, 1987.

[Spe93] William M. Spears. Crossover or mutation? In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms, 2*, pages 221–237. Morgan Kaufmann, 1993.

[Sta87] I. Stadnyk. Schema recombination in a pattern recognition problem. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 27–35. Lawrence Erlbaum Associates, 1987.

[SVG87] J.Y. Suh and D. Van Gucht. Incorporating heuristic information into genetic search. In J.J. Grefenstette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 100–107. Lawrence Erlbaum Associates, 1987.

[Sys89] G. Syswerda. Uniform crossover in genetic algorithms. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 2–9. Morgan Kaufmann, 1989.

[Sys91] G. Syswerda. Schedule optimization using genetic algorithms. In L. Davis, editor, *Handbook of Genetic Algorithms*, chapter 21, pages 332–349. Van Nostrand Reinhold, 1991.

[VL91] M. Vose and G. Liepins. Schema disruption. In R.K. Belew and L.B. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 237–242. Morgan Kaufmann, 1991.

[Whi89] D. Whitley. The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In J.D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 116–121. Morgan Kaufmann, 1989.