C6.1 Niching methods

Samir W Mahfoud

Abstract

Niching methods extend genetic algorithms from optimization to domains such as classification, multimodal function optimization, simulation of complex and adaptive systems, and multiobjective function optimization. Sharing and crowding are two prominent categories of niching methods. Both categories contain algorithms that can successfully locate and maintain multiple solutions within the population of a genetic algorithm.

C6.1.1 Introduction

Niching methods (Mahfoud 1995a) extend *genetic algorithms* (GAs) to domains that require the location and maintenance of multiple solutions. While traditional GAs primarily perform optimization, GAs that incorporate niching methods are more adept at problems in classification and machine learning, multimodal function optimization, *multiobjective function optimization*, and *simulation* of complex and adaptive systems. B1.2 C4.5, F1.9, F1.8

Niching methods can be divided into families or categories, based upon structure and behavior. To date, two of the most successful categories of niching methods are *fitness sharing* (also called *sharing*) and *crowding*. Both categories contain methods that are capable of locating and maintaining multiple solutions within a population, whether those solutions have identical or differing fitnesses.

C6.1.2 Fitness sharing

Fitness sharing, as introduced by Goldberg and Richardson (1987), is a fitness scaling mechanism that alters only the fitness assignment stage of a GA. Sharing can be used in combination with other scaling mechanisms, but should be the last one applied, just prior to selection.

From a multimodal function maximization perspective, the idea behind sharing is as follows. If similar individuals are required to share payoff or fitness, then the number of individuals that can reside in any one portion of the *fitness landscape* is limited by the fitness of that portion of the landscape. Sharing results in individuals being allocated to optimal regions of the fitness landscape. The number of individuals residing near any peak will theoretically be proportional to the height of that peak. B2.7

Sharing works by derating each population element's fitness by an amount related to the number of similar individuals in the population. Specifically, an element's *shared fitness*, F', is equal to its prior fitness F divided by its *niche count*. An individual's niche count is the sum of *sharing function* (sh) values between itself and each individual in the population (including itself). The shared fitness of a population element i is given by the following equation:

$$F'(i) = \frac{F(i)}{\sum_{j=1}^{\mu} \mathrm{sh}(d(i,j))}. \qquad \text{(C6.1.1)}$$

The sharing function is a function of the distance d between two population elements; it returns a '1' if the elements are identical, a '0' if they cross some threshold of dissimilarity, and an intermediate value for intermediate levels of dissimilarity. The threshold of dissimilarity is specified by a constant, σ_{share}; if the distance between two population elements is greater than or equal to σ_{share}, they do not affect each other's shared fitness. A common sharing function is

$$\text{sh}(d) = \begin{cases} 1 - (d/\sigma_{\text{share}})^{\alpha} & \text{if } d < \sigma_{\text{share}} \\ 0 & \text{otherwise} \end{cases} \tag{C6.1.2}$$

where α is a constant that regulates the shape of the sharing function.

While nature distinguishes its niches based upon phenotype, niching GAs can employ either *genotypic* or *phenotypic* distance measures. The appropriate choice depends upon the problem being solved.

C6.1.2.1 Genotypic sharing

In genotypic sharing, the distance function d is simply the Hamming distance between two strings. (The Hamming distance is the number of bits that do *not* match when comparing two strings.) Genotypic sharing is generally employed by default, as a last resort, when no phenotype is available to the user.

C6.1.2.2 Phenotypic sharing

In phenotypic sharing, the distance function d is defined using problem-specific knowledge of the phenotype. Given a function optimization problem containing k variables, the most common choice for a phenotypic distance function is Euclidean distance. Given a classification problem, the phenotypic distance between two classification rules can be defined based upon the examples to which they both apply.

C6.1.2.3 Parameters and extensions

Typically, α is set to unity, and σ_{share} is set to a value small enough to allow discrimination between desired peaks. For instance, given a one-dimensional function containing two peaks that are two units apart, a σ_{share} of 1 is ideal: since each peak extends its reach for $\sigma_{\text{share}} = 1$ unit in each direction, the reaches of the peaks will touch but not overlap. Deb (1989) gives more details for setting σ_{share}.

Population size can be set roughly as a multiple of the number of peaks the user wishes to locate E1.1 (Mahfoud 1995a, b). Sharing is best run for few generations, perhaps some multiple of $\log \mu$. This rough heuristic comes from shortening the expected convergence time for a GA that uses fitness-proportionate selection (Goldberg and Deb 1991). A GA under sharing will not converge population elements atop the peaks it locates. One way of obtaining such convergence is to run a hillclimbing algorithm after the GA.

Sharing can be implemented using any selection method, but the choice of method may either increase or decrease the stability of the algorithm. *Fitness-proportionate* selection with stochastic universal sampling C2.2 (Baker 1987) is one of the more stable options. *Tournament* selection is another possibility, but special C2.3 provisions must be made to promote stability. Oei *et al* (1991) propose a technique for combining sharing with binary tournament selection. This technique, *tournament selection with continuously updated sharing*, calculates shared fitnesses with respect to the new population as it is being filled.

The main drawback to using sharing is the additional time required to cycle through the population to compute shared fitnesses. Several authors have suggested calculating shared fitnesses from fixed-sized samples of the population (Goldberg and Richardson 1987, Oei *et al* 1991). Clustering is another potential remedy. Yin and Germay (1993) propose that a clustering algorithm be implemented prior to sharing, in order to divide the population into niches. Each individual subsequently shares only with the individuals in its niche. As far as GA time complexity is concerned, in real-world problems, a function evaluation requires much more time than a comparison; most GAs perform only $O(\mu)$ function evaluations each generation.

C6.1.3 Crowding

Crowding techniques (De Jong 1975) insert new elements into the population by replacing similar elements. To determine similarity, crowding methods, like sharing methods, utilize a distance measure, either genotypic or phenotypic. Crowding methods tend to spread individuals among the most prominent peaks of the search space. Unlike sharing methods, crowding methods do not allocate elements proportional to peak fitness. Instead, the number of individuals congregating about a peak is largely determined by the size of that peak's basin of attraction under crossover.

By replacing similar elements, crowding methods strive to maintain the preexisting diversity of a population. However, replacement errors may prevent some crowding methods from maintaining individuals in the vicinity of desired peaks. The *deterministic crowding* algorithm (Mahfoud 1992, 1995a) is designed to minimize the number of replacement errors, and thus allow effective niching.

Deterministic crowding works as follows. First it groups all population elements into $\mu/2$ pairs. Then it crosses all pairs and mutates the offspring. Each offspring competes against one of the parents that produced it. For each pair of offspring, two sets of parent–child tournaments are possible. Deterministic crowding holds the set of tournaments that forces the most similar elements to compete.

The pseudocode for deterministic crowding is as follows:

```
**Input:** $g$—number of generations to run, $\mu$—population size
**Output:** $P(g)$—the final population

$P(0) \leftarrow$ initialize()
for $t \leftarrow 1$ to $g$ do
    $P(t) \leftarrow$ shuffle($P(t-1)$)
    for $i \leftarrow 0$ to $\mu/2 - 1$ do
        $p_1 \leftarrow \boldsymbol{a}_{2i+1}(t)$
        $p_2 \leftarrow \boldsymbol{a}_{2i+2}(t)$
        $\{c_1, c_2\} \leftarrow$ recombine($p_1, p_2$)
        $c_1' \leftarrow$ mutate($c_1$)
        $c_2' \leftarrow$ mutate($c_2$)
        if $[d(p_1, c_1') + d(p_2, c_2')] \leq [d(p_1, c_2') + d(p_2, c_1')]$ then
            if $F(c_1') > F(p_1)$ then $\boldsymbol{a}_{2i+1}(t) \leftarrow c_1'$ fi
            if $F(c_2') > F(p_2)$ then $\boldsymbol{a}_{2i+2}(t) \leftarrow c_2'$ fi
        else
            if $F(c_2') > F(p_1)$ then $\boldsymbol{a}_{2i+1}(t) \leftarrow c_2'$ fi
            if $F(c_1') > F(p_2)$ then $\boldsymbol{a}_{2i+2}(t) \leftarrow c_1'$ fi
        fi
    od
od
```

Deterministic crowding requires the user only to select a population size μ and a stopping criterion. As a general rule of thumb, the more final solutions a user desires, the higher μ should be. The user can stop a run after either a fixed number of generations g (of the same order as μ) or when the rate of improvement of the population approaches zero. Full crossover should be employed (with probability 1.0) since deterministic crowding only discards solutions after better ones become available, thus alleviating the problem of crossover disruption.

Two crowding methods similar in operation and behavior to deterministic crowding have been proposed (Cedeño *et al* 1994, Harik 1995). Cedeño *et al* suggest utilizing phenotypic crossover and mutation operators (i.e. *specialized* operators), in addition to phenotypic sharing; this results in further reduction of replacement error.

C6.1.4 Theory

Much of the theory underlying sharing, crowding, and other niching methods is currently under development. However, a number of theoretical results exist, and a few areas of theoretical research have already been defined by previous authors. The characterization of hard problems is one area of theory. For niching methods, the number of optima the user wishes to locate, in conjunction with the number of optima present, largely determines the difficulty of a problem. A secondary factor is the degree to which extraneous optima lead away from desired optima.

Analyzing the distribution of solutions among optima for particular algorithms forms another area of theory. Other important areas of theory are calculating expected drift or disappearance times for desired solutions; population sizing; setting parameters such as operator probabilities and σ_{share} (for sharing); and improving the designs of niching genetic algorithms. For an extensive discussion of niching methods and their underlying theory, consult the article by Mahfoud (1995a).

References

Baker J E 1987 Reducing bias and inefficiency in the selection algorithm *Proc. Int. Conf. on Genetic Algorithms (Cambridge, MA)* ed J J Grefenstette (Hillsdale, NJ: Lawrence Erlbaum Associates) pp 14–21

Cedeño W, Vemuri V R and Slezak T 1994 Multiniche crowding in genetic algorithms and its application to the assembly of DNA restriction-fragments *Evolutionary Comput.* **2** 321–45

Deb K 1989 *Genetic Algorithms in Multimodal Function Optimization* Masters Thesis; TCGA Report 89002, University of Alabama, The Clearinghouse for Genetic Algorithms

De Jong K A 1975 *An Analysis of the Behavior of a Class of Genetic Adaptive Systems* Doctoral Dissertation, University of Michigan *Dissertation Abstracts Int.* **36** 5140B (University Microfilms 76-9381)

Goldberg D E and Deb K 1991 A comparative analysis of selection schemes used in genetic algorithms *Foundations of Genetic Algorithms* ed G J E Rawlins (San Mateo, CA: Morgan Kaufmann) pp 69–93

Goldberg D E and Richardson J 1987 Genetic algorithms with sharing for multimodal function optimization *Proc. 2nd Int. Conf. on Genetic Algorithms (Cambridge, MA)* ed J J Grefenstette (Hillsdale, NJ: Lawrence Erlbaum Associates) pp 41–9

Harik G 1995 Finding multimodal solutions using restricted tournament selection *Proc. 6th Int. Conf. on Genetic Algorithms (Pittsburgh, July 1995)* ed L J Eshelman (San Mateo, CA: Morgan Kaufmann) pp 24–31

Mahfoud S W 1992 Crowding and preselection revisited *Parallel Problem Solving From Nature* vol 2, ed R Männer and B Manderick (Amsterdam: Elsevier) pp 27–36

——1995a *Niching Methods for Genetic Algorithms* Doctoral Dissertation and IlliGAL Report 95001, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory) *Dissertation Abstracts Int.* (University Microfilms 9543663)

——1995b Population size and genetic drift in fitness sharing *Foundations of Genetic Algorithms* vol 3, ed L D Whitley and M D Vose (San Francisco, CA: Morgan Kaufmann) pp 185–223

Oei C K, Goldberg D E and Chang S J 1991 *Tournament Selection, Niching, and the Preservation of Diversity* (IlliGAL Report 91011) University of Illinois, Illinois Genetic Algorithms Laboratory

Yin X and Germay N 1993 A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization *Artificial Neural Nets and Genetic Algorithms: Proc. Int. Conf. (Innsbruck)* ed R F Albrecht, C R Reeves and N C Steele (Berlin: Springer) pp 450–7

C6.2 Speciation methods

Kalyanmoy Deb (C6.2.1–C6.2.4) *and William M Spears* (C6.2.5, C6.2.6)

Abstract

In nature, a species is defined as a collection of phenotypically similar individuals. Many biologists believe that individuals in a sexually reproductive species can be created and maintained by allowing restrictive mating only among individuals from the same species. The connection between the formation of multiple species in nature and in search and optimization problems lies in solving multimodal problems, where the objective is not only to find one optimal solution, but to find a number of optimal solutions. In those problems, each optimal solution may be assumed to constitute a species. Since evolutionary algorithms work with a population of solutions, the concept of natural speciation techniques can be implemented to allow formation of multiple subpopulations, each focusing its search for one optimal solution. This way, multiple optimal solutions can be discovered simultaneously. In this section, a number of speciation techniques are discussed.

C6.2.1 Introduction

Kalyanmoy Deb

Despite some controversy, most biologists agree that a species is a collection of individuals which resemble each other more closely than they resemble individuals of another species (Eldredge 1989). It is also clear that the reproductive process of the sexually reproducing organisms causes individuals to resemble their parents, thereby maintaining a phenotypic similarity among individuals of the community or the *species*. Thus, there is a strong correlation among the reproductively coherent individuals and a phenotypically similar cluster of individuals. Since in evolutionary algorithms a population of solutions is used, artificial species of phenotypically similar solutions can be formed and maintained in the population by restricting their mating to that with similar individuals. Before we outline how to form and maintain multiple species in a population, let us discuss why it could be necessary to form species in the applications of evolutionary algorithms.

In Section C6.1, we saw that multiple optimal solutions in a multimodal optimization problem can be found simultaneously by forming artificial niches (subpopulations) in the population. Each niche can be considered to represent a peak (in the spirit of maximization problems). To capture a number of peaks simultaneously and maintain them for many generations, a niching method is used. Niching helps to emphasize and maintain solutions around multiple optima. However, in niching, the main emphasis is devoted to distributing the population members across different peaks. Thus, the niching technique cannot quite focus its search on each peak and find the exact optimal solutions efficiently. This is because some of the search effort is wasted in the recombination of interpeak solutions, which, in turn, may produce some *lethal* solutions representing none of the peaks. A speciation method used in evolutionary computation (EC) studies, on the other hand, restricts mating to that among like solutions (likeness can be defined phenotypically or genotypically) and discourages mating among solutions of different peaks. If the likeness is defined properly, two parent solutions chosen for mating are likely to represent the same peak. Thus, when like individuals mate with each other, the created children solutions are also similar to the C6.1

parent solutions and are likely to be members of the same peak. This way, the restriction of mating to that among like solutions may reduce the creation of lethal solutions (which represent none of the peaks). This may allow the search to concentrate on each peak and help find the best or near-best optimum solution efficiently. However, in order to apply the speciation technique properly, solutions representing each peak must first be found. Thus, the speciation technique cannot be used independently. In the presence of both niching and speciation, niching finds and maintains subpopulation of solutions around multiple optima and the speciation technique allows us to make an inherent parallel search in each optimum to find multiple optimal solutions simultaneously.

Among the evolutionary algorithms, a number of speciation methods have been suggested and implemented in *genetic algorithms* (GAs). Of the earlier works related to mating restriction in GAs, Hollstien's (1971) inbreeding scheme where mating was allowed between similar individuals in his simulation of animal husbandry problems, Booker's (1982) taxon–exemplar scheme for restrictive mating in his simulation of learning pattern classes, Holland's suggestion of a tag–template scheme (Goldberg 1989), Sannier and Goodman's (1987) restrictive mating in forming separate coherent groups in a population, Deb's (1989) phenotypic and genotypic mating restriction schemes, and Spears's (1994) and Perry's (1984) speciation using tag bits are a few studies. In the following, we discuss some of the above speciation methods in more detail. B1.2

C6.2.2 Booker's taxon–exemplar scheme

Kalyanmoy Deb

Booker (1982) used taxons and exemplars in his *learning algorithm* to reduce the formation of lethal individuals. He defined a taxon as a string (constructed over the three-letter alphabet {0, 1, #}, with a # matching a 0 or a 1). The population is initialized with taxon strings. In his *restricted mating policy*, he wanted to restrict mating among similar taxon strings, which were identified by calculating a match score of the taxon strings with a given exemplar binary string. He allowed partial match scores depending on the matching of the taxon and the exemplar. For the following two taxon strings and the exemplar string, the first taxon matches the exemplar completely. The second taxon matches the exemplar partially (in first, third, and fourth positions): B1.5.2

```
      Taxon        Exemplar
(1  #  0  0  #)  (1  0  0  0  0).
(#  1  #  0  1)
```

If the taxon completely matches the exemplar, a score is assigned as the sum of the string length and the number of #s in the taxon. The partial credit is also assigned based on the number of correct matches and the number of #s in the taxon. In order to implement the restrictive mating policy, he chose parent taxon strings from a sample subpopulation determined based on the available matching taxon strings in the population. If a specified number of matching taxon strings are available in the population, parent strings are chosen uniformly at random from all the matching taxon strings. Otherwise, parent strings are chosen according to a probability distribution calculated based on the match score of the taxon strings. In a number of pattern discovery problems an improved performance is observed with the restricted mating policy.

After the patterns were discovered, Booker extended his above scheme to classify the discovered patterns using a modified string as follows:

```
     Taxon                 Tag
(1  #  0  0  #)   :   (1  0  0  0  0).
```

In addition to the taxon string, a tag string is introduced to classify the discovered taxon strings (or patterns). The taxon strings matching a particular tag string were considered to be in the same class. A similar match score was used, except that this time the matching was performed with the taxon and tag strings. As discussed elsewhere (Goldberg 1989), there is one difficulty with the above tag–taxon scheme. The tag string must be of the same length as the taxon string. This increases the complexity of the classification problem, whereas the same concept can be implemented with shorter tag and template strings, as suggested by Holland; a brief description of this is given by Goldberg (1989).

C6.2.3 The tag–template method

Kalyanmoy Deb

In addition to the functional string (the taxon string in Booker's pattern classification problem), a template and a tag string are introduced. The template string is constructed from the three-letter alphabet (1, 0, and #) as before, but the tag string is a binary string of the same length as the template string. A typical string with the tag and template strings would look like the following:

Template		Tag		Functional string
(#01)	:	(100)	:	(1011001101).

The size of tag and template strings depends on the number of desired solutions. A simple calculation shows that if q different optimal solutions (peaks) are to be found, the minimum string length for the tag and template is $\lceil \log_2 q \rceil$ (Deb 1989). The tag and template strings are created at random in the initial population along with the functional string. These two strings do not affect the fitness of the functional string. However, they are affected by the crossover and the mutation operators, as well. For the template string, the mutation operator must be modified to operate on a three-allele string. The purpose of these strings is to restrict mating. Before crossing a pair of individual strings, their tag and template strings are *matched*. If the match score exceeds a threshold value, the crossover is performed between the two strings as usual; otherwise some other string pair is tested for a possible mating. In this process, the tag and template strings corresponding to the good individuals in early populations are emphasized and an artificial tag is set for solutions in each peak. Later on, since crossing over is only performed between the matched strings, only similar strings (or strings from the same peak) tend to participate in crossover.

Although neither Holland nor Goldberg simulated this speciation method, Deb (1989) (with assistance from David Goldberg) implemented this scheme and applied this technique in solving multimodal test problems. In both cases, GAs with the tag–template scheme performed better than GAs without it.

C6.2.4 Phenotypic and genotypic mating restriction

Kalyanmoy Deb

Deb (1989) has developed two mating restriction schemes based on the phenotypic and genotypic distance between mating individuals. The mating restriction schemes are straightforward. In order to choose a mate for an individual, their distance (in phenotypic mating restriction the Euclidean distance and in genotypic mating restriction the Hamming distance) is computed. If the distance is closer than a parameter σ_{mating}, they participate in the crossover operation; otherwise another individual is chosen at random and their distance is computed. This process is continued until a suitable mate is found or all population members are exhausted, in which case a random individual is chosen as a mate. Deb has implemented both the above mating restriction schemes with a single-point crossover and applied them to solve a number of multimodal test problems. Although, in all his simulations, the parameter σ_{mating} was kept the same as the parameter σ_{share} used in the niching methods, other values of σ_{mating} may also be chosen. It is worthwhile to mention that niching with the σ_{share} parameter is implemented in the selection operator and the mating restriction with the σ_{mating} parameter is implemented in the crossover operator. GAs with niching and mating restriction were found to better distribute the population across the peaks than GAs with sharing alone. Here, we present simulation results for the phenotypic mating restriction scheme adopted in that study. In solving the single-variable, five-peaked function in the interval $0 \leq x \leq 1$

$$\text{maximize} \qquad 2^{-2((x-0.1)/0.8)^2} \sin^6(5\pi x)$$

with $\sigma_{\text{share}} = \sigma_{\text{mating}} = 0.1$, 100 population members after 200 generations without and with phenotypic mating restriction are shown in figure C6.2.1. Stochastic remainder roulette wheel selection and single-point crossover operators are used. The crossover and mutation probabilities are kept as 0.9 and 0.0, respectively. The figures show that, with the mating restriction scheme (the right-hand panel), the number of lethal (nonpeak) individuals has been significantly decreased. This study also implemented a genotypic mating restriction scheme and similar results were obtained. Some guidelines in choosing the sharing and mating restriction parameters are outlined elsewhere (Deb 1989, Deb and Goldberg 1989).

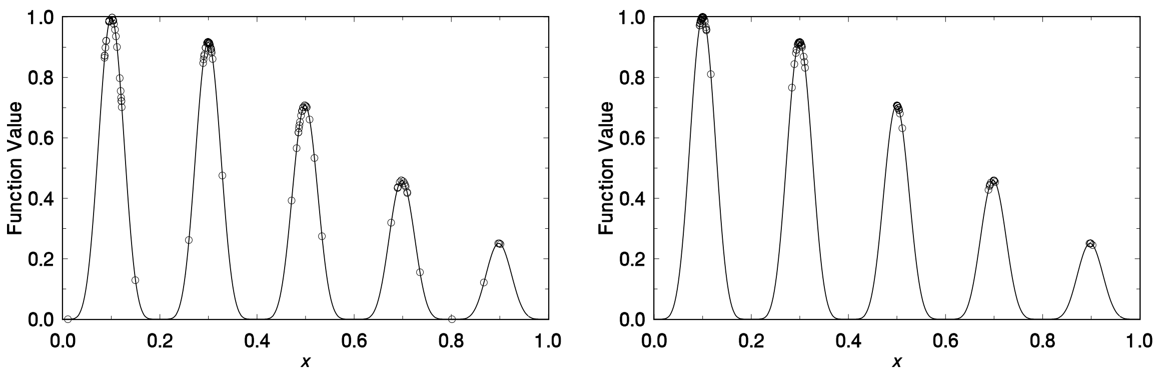

Figure C6.2.1. The distribution of 100 solutions without (left) and with (right) a mating restriction scheme.

C6.2.5 Speciation using tag bits

William M Spears

Another method for identifying species is via the use of tag bits, which are appended to every individual. Each species corresponds to a particular setting of these bits. Suppose there are k different sets of tag bit values at a particular generation of the evolutionary algorithm (EA). Denote these sets as $\{S_0, \ldots, S_{k-1}\}$. The sets are numbered arbitrarily. Each individual belongs to one S_i and all individuals in a particular S_i have the same tag bit values. For example, suppose there is only one tag bit and that some individuals exist with a tag bit value zero and that the remainder exist with tag bit value one. Then (arbitrarily) assign the former set of individuals to S_0 and the latter set to S_1. Let $\| \ \|$ denote the cardinality of the sets.

Spears (1994) uses the tag bits to restrict mating and to perform fitness sharing. With sharing, the perceived fitness, F_i, is a normalization of the objective fitness f_i:

$$F_i = \frac{f_i}{\|S_j\|} \qquad i \in S_j$$

where $\|S_j\|$ is the size of the species that individual i is in.

The average fitness of the population, $\bar{F}$, becomes

$$\bar{F} = \frac{\sum_{i \in S_0} (f_i/\|S_0\|) + \ldots + \sum_{i \in S_{k-1}} (f_i/\|S_{k-1}\|)}{\|S_0\| + \ldots + \|S_{k-1}\|}$$

which is just

$$\bar{F} = \frac{\sum_{i \in S_0} (f_i/\|S_0\|) + \ldots + \sum_{i \in S_{k-1}} (f_i/\|S_{k-1}\|)}{N}$$

since the species sizes have to total N (recall that no individual can lie in more than one species). The expected number of offspring for an individual is now $F_i/\bar{F}$.

Restricted mating is performed by only allowing recombination to occur between individuals with the same tag bit values. Mutation can flip all bits, including the tag bits, thus allowing individuals to change labels. Experimental results, as well as some modifications to the above mechanism can be found in the article by Spears (1994). Code for the algorithm can be found at http://www.aic.nrl.navy.mil/~spears.

Perry's thesis work (Perry 1984) with speciation is extremely similar to the above technique. Perry includes both species and environmental regions in an EA. Species are identified via tag bits and an environmental region is similar to an EA population. Recombination within an environment can occur only on individuals with the same tag bit values. Mutation is allowed to change tag bits, in order to introduce new species. The additional use of a 'migration' operator, which moves individuals from one environment to another, does not have an analog in the work of Spears (1994).

Perry gives an example of two species in an environment—fitness proportional selection is performed, and the average fitness of an environment is

$$\bar{f} = \frac{\sum_{i \in S_0} f_i + \sum_{i \in S_1} f_i}{\|S_0\| + \|S_1\|}$$

or

$$\bar{f} = \frac{\sum_{i \in S_0} f_i + \sum_{i \in S_1} f_i}{N}$$

where N is the population size of the environmental niche. The expected number of offspring is $f_i \, / \, \bar{f}$. One can see that the main difference between the two methods is the use of sharing in the computation of fitness in the work of Spears (1994). Thus it is not surprising that in many of Perry's experimental runs one particular species would eventually dominate an environmental niche (however, it should be noted that in the work of Perry (1984) the domination of an environment by a species was not undesirable behavior).

The use of tag bits makes restricted mating and fitness sharing more efficient because distance comparisons do not have to be computed. Interestingly, it is also possible to make Goldberg's implementation of sharing more efficient by sampling (Goldberg *et al* 1992). In other words the distance of each individual from the rest is estimated by using a subset of the remaining individuals.

C6.2.6 Relationship with parallel algorithms

William M Spears

Clearly this work has similarities to the EA research performed on *parallel architectures*. In a parallel EA, a topology is imposed on the EA population, resulting in species. However, there are some important differences between the parallel approaches and the sequential approach. For example, with the fitness sharing approaches the fitness of an individual and the species size are dynamic, based on the other individuals (and species). This concentrates effort on more promising peaks, while still maintaining individuals in other areas of the search space. This is typically not true for parallel EAs implemented on MIMD or SIMD architectures. When using a MIMD architecture, species are dedicated to particular processors and the species remain a constant size. In SIMD implementations, one or two individuals reside on a processor, and species are formed by defining overlapping neighborhoods. However, due to the overlap, one particular species will eventually take over the whole population. C6.3.1, C6.4.2

References

Booker L B 1982 *Intelligent Behavior as an Adaptation to the Task Environment* Doctoral Dissertation, University of Michigan; *Dissertation Abstracts Int.* **43** 469B

Deb K 1989 *Genetic Algorithms in Multimodal Function Optimization* Master's Thesis, University of Alabama; TCGA Report 89002

Deb K and Goldberg D E 1989 An investigation of niche and species formation in genetic function optimization *Proc. 3rd Int. Conf. on Genetic Algorithms (Fairfax, VA, 1989)* ed J D Schaffer (San Mateo, CA: Morgan Kaufmann) pp 42–50

Eldredge N 1989 *Macro-evolutionary Dynamics: Species, Niches and Adaptive Peaks* (New York: McGraw-Hill)

Goldberg D E 1989 *Genetic Algorithms in Search, Optimization, and Machine Learning* (Reading, MA: Addision-Wesley)

Goldberg D E and Richardson J 1987 Genetic algorithms with sharing for multimodal function optimization *Proc. 2nd Int. Conf. on Genetic Algorithms (Cambridge, MA, 1987)* ed J J Grefenstette (Hillsdale, NJ: Erlbaum) pp 41–9

Goldberg D E, Deb K and Horn J 1992 Massive multimodality, deception, and genetic algorithms *Proc. Parallel Problem Solving from Nature Conf.* (Amsterdam: North-Holland) pp 37–46

Hollstien R B 1971 *Artificial Genetic Adaptation in Computer Control Systems* Doctoral Dissertation, University of Michigan; *Dissertation Abstracts Int.* **32** 1510B

Perry Z A 1984 *Experimental Study of Speciation in Ecological Niche Theory using Genetic Algorithms* Doctoral Dissertation, University of Michigan; *Dissertation Abstracts Int.* **45** 3870B

Sannier A V and Goodman E D 1987 Genetic learning procedures in distributed environments *Proc. 2nd Int. Conf. on Genetic Algorithms (Cambridge, MA, 1987)* ed J J Grefenstette (Hillsdale, NJ: Erlbaum) pp 162–9

Spears W M 1994 Simple subpopulation schemes *Proc. Conf. on Evolutionary Programming* (Singapore: World Scientific) pp 296–307