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:

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

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 (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. E1.1

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 (Baker 1987) is one of the more stable options. *Tournament* selection is another possibility, but special 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. C2.2 C2.3

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, μ—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