A Variable Radius Niching Technique for Speciation in Genetic Algorithms

Justin Gan

Dept. Cybernetics, Reading University,
Whiteknights, Reading, Berkshire,
RG6 6AY, UK.
Email: jgrg@cyber.reading.ac.uk
Phone: (+44) 0118 9316261

Kevin Warwick

Dept. Cybernetics, Reading University,
Whiteknights, Reading, Berkshire,
RG6 6AY, UK.
Email: K.Warwick@reading.ac.uk
Phone: (+44) 0118 9318210

Abstract

In this paper, a continuation of a variable radius niche technique called Dynamic Niche Clustering developed by (Gan & Warwick, 1999) is presented. The technique employs a separate dynamic population of overlapping niches that coexists alongside the normal population. An empirical analysis of the updated methodology on a large group of standard optimisation test-bed functions is also given. The technique is shown to perform almost as well as standard fitness sharing with regards to stability and the accuracy of peak identification, but it outperforms standard fitness sharing with regards to time complexity. It is also shown that the technique is capable of forming niches of varying size depending on the characteristics of the underlying peak that the niche is populating.

1 INTRODUCTION

Genetic Algorithms (GAs) have long been the target for multimodal optimisation research. The problem essentially being that a Simple Genetic Algorithm (SGA) is too simple. To clarify; a SGA will converge to a single peak within a fitness landscape, and depending on the ruggedness of the landscape, that peak will be the most fit optimum. This is not necessarily always the case as local optima can often mislead a SGA into populating the incorrect peak. This convergence to a single peak, even if there are other peaks of equal fitness present in the search space, is a result of *genetic drift* (DeJong, 1975). Genetic drift can be attributed to stochastic fluctuations within the selection process. For these reasons, it is desirable to encourage speciation, and hence diversity within the population. In addition to this, if multiple peaks are populated, then the GA has potentially found multiple solutions to the problem at hand.

Over the years there have been numerous proposals for techniques and methodologies that promote speciation within a GA population. The majority of these techniques employ fitness sharing in some form or other. The basic premise is to allocate to each peak in the fitness landscape a number of individuals proportional to the peak's fitness relative to the other peaks in the landscape. This is termed as a *niche proportionate* population (Miller & Shaw, 1995). In fitness sharing, this is achieved by reducing the fitness of an individual by an amount proportional to the number of other individuals in the immediate vicinity, defined by some threshold distance. On the fitness landscape, this can be seen to reduce the height of the populated peaks so that the individuals in each of the different peaks have the same fitness. Thus, they are all equally likely to be selected in the next generation, and diversity is maintained.

The vast majority of these techniques suffer from two main restrictions; the requirement for at least some foreknowledge of, or an accurate estimation of the number of peaks in the fitness landscape; and that the peaks are evenly spaced throughout the search space. These are both artificial constraints imposed on the inherent freedom exhibited by GAs.

This paper presents a subtly different approach to the fitness sharing mechanism, originally developed by (Gan & Warwick, 1999), which attempts to address the two restrictions described earlier. The formulated mechanism is compared to standard fitness sharing on a large group of classical optimisation test-bed functions. We also analyse a phenomenon that emerged in the original methodology that was termed *striation*, and show how and why it manifests itself, and how it can be eliminated. Firstly, however, we review some of the more well known niching methods.

1.1 NICHING METHODS

Perhaps the most well known and well quoted technique is *Fitness Sharing* (Goldberg & Richardson 1987). Here, a distance (similarity) metric between any two individuals is defined, d_{ij}. The shared fitness of an individual, i, is defined as the individual's raw fitness divided by its niche count, m_i. The niche count is defined by summing the sharing function, $sh(d_{ij})$, over all p members of the population.

$$m_i = \sum_{j=1}^{p} sh(d_{ij}) \qquad (1)$$

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{sh}} \right)^{\alpha} & \text{if } d_{ij} < \sigma_{sh} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

To determine the value of niche radius, σ_{sh}, the following equation (Deb, 1989 and Deb & Goldberg, 1989) based upon the assumptions that there are m maxima in the function, each surrounded by a hypersphere of radius σ_{sh}, the hyperspheres do not overlap and completely fill the k-dimensional problem space, and that each dimension is normalised. σ_{sh} is given by:

$$\sigma_{sh} = \frac{\sqrt{k}}{2 * \sqrt[k]{m}} \quad (3)$$

Arguably the most notorious aspect of this particular technique is its $O(p^2)$ complexity, but perhaps even more significant are the requirements of a single fixed value of σ_{sh}, knowledge of the number of maxima, and that the maxima be equally spread throughout the search space a minimum distance of $2\sigma_{sh}$ from one another. This is far too restrictive for any real-world problem where irregular peaks, high dimensionality problem spaces, and high multimodality require large population sizes. This is because a certain number of individuals are necessary to accurately locate, explore and maintain each maximum.

Several improvements to the original fitness sharing algorithm are described by (Oei *et al*, 1991) that reduce the complexity to $O(np)$ by sampling the population rather than computing the distance to every other individual. A niche-size parameter, n^*, is also suggested that is used to effectively limit the maximum number of individuals in each niche.

Another technique worthy of note is *Sequential Niching* (Beasley *et al*, 1993). The technique works by iterating over a SGA. On each iteration, the fitness landscape is modified according to the location of solutions found in previous iterations. The modified fitness function, $m_i(x)$, at iteration i is calculated from the raw fitness function, $f(x)$, multiplied by a number of single-peak derating functions. Initially, $m_0(x) = f(x)$. The best individual, s_i, found in the i^{th} GA run is used to determine the midpoint of the single-peak derating function $g(x,s_i)$. The modified fitness function is then given by:

$$m_{i+1} = m_i(x) * g(x, s_i) \quad (4)$$

One of the key problems with this technique was determining exactly when to terminate a particular GA iteration. A halting window of h generations was utilised, where the GA terminated if the fitness of any generation was not greater than the population h generations earlier. More importantly, the algorithm actually alters the fitness landscape from GA run to GA run, so after many iterations it is impossible to tell whether the peaks the algorithm is finding are real peaks, or 'phantom' peaks introduced by multiple applications of the derating functions.

An interesting technique for niche formation was described in (Yin & Germay, 1993). Here, MacQueen's *KMEAN* clustering algorithm is employed to divide the population into k clusters of individuals, corresponding to k niches. The shared fitness of an individual is given by dividing the raw fitness of the individual by the niche count, m_i, of the cluster to which the individual is a member.

$$m_i = n_c - n_c * \left(\frac{d_{ic}}{2d_{\max}} \right)^{\alpha} \qquad x_i \in C_x \quad (5)$$

Here, d_{ic} is the distance between individual i and the cluster's centroid. n_c is the number of individuals in cluster c. Two other parameters d_{min} and d_{max} are defined. After determining the positions of the clusters, each of the clusters are then compared. Two clusters are merged if the distance between their centroids is less than d_{min}. If an individual is further than d_{max} from all existing clusters, a new cluster is formed on that individual. This method allows the formation of stable subpopulations and has also been shown to perform well on irregular fitness landscapes, but, as with all other niche techniques that rely on a distance metric, it requires some *a priori* knowledge of the fitness function.

Two variations on a steady state evolution method called a *forking* GA have been described by (Tsutsui *et al*, 1997). Both variations rely on the same *forking* technique to subdivide the search space, but the first looks at the *salient schema* within the genotypic search space, and the second looks at *neighbourhood hypercubes* centred on the current best individual in the phenotypic search space. In both cases, the search space is divided into sub-spaces depending upon the status of convergence of the present population, and the solutions found so far. For a more detailed description of this technique, the reader should consult (Tsutsui *et al*, 1997).

GAS (Jelasity, 1998) is another method that explicitly employs a variable radius niching scheme. Here, a *species* (defined by its centre in phenotypic search space, and radius) describes a non-overlapping window on the search space. The radii of the species are determined by using a *radius function*. The method employs a *cooling* technique (similar to simulated annealing) which allows the search to focus in on promising regions of space, starting off with an initially large radius that decreases as the search progresses. For more details the reader should consult (Jelasity, 1998).

2 SO HOW DOES IT WORK?

Here we will provide a brief summary of the modified Dynamic Niche Clustering (DNC) scheme (Gan & Warwick, 1999), and highlight the alterations made in the interim. It should be noted at this point that the technique relies on a decoded phenotype distance metric. It should also be noted that unlike other niche techniques, DNC allows an individual to be a member of more than one niche, and that the niches themselves may overlap to a certain degree (this will be described in detail later). Furthermore, the original DNC method was optimised for 1-dimensional problems. The technique has since been

generalised to n-dimensions and as a result, all calculations described subsequently are vector arithmetic.

DNC maintains an explicit population of niches, henceforth referred to as the *nicheset*, that evolves alongside the normal GA population. The niches are persistent, that is; they are retained from generation to generation. Each niche consists of a number of variables; a vector describing its midpoint in the parameter space (mid_i), the niche radius (σ_{shi}), the generation and location at which the niche was spawned, and a list of references to the individuals that are currently members of the niche. An individual is considered to be a member of a niche if it falls within the hypersphere described by the midpoint and niche radius of that niche.

$$\left\| mid_j - v_i \right\| \le \sigma_{sh_j} \tag{6}$$

That is, if the euclidean distance between the individual and the midpoint is less than the niche radius, then individual *i* is a member of niche *j*. In order to prevent the unlimited growth or contraction of a niche, the niche radius is bounded by two values, σ_{max} and σ_{min}. The current population size, dimensionality, and the overall extents of the parameter space determine these bounding values and the initial niche radius. A number of different schemes were proposed for the initial choice of niche radius, but for this paper only the *Inverse Power Law* scheme (Gan & Warwick, 1999) will be considered. Note, however, that the original scheme did not take into consideration the dimensionality of the search space. We recommend the following amendment;

$$\sigma_{sh\,initial} = \frac{\sqrt{d}}{3 * \sqrt[d]{pop}} \tag{7}$$

Here, *pop* is the population size, *d* is the number of dimensions, and we assume that the parameter space is normalised. $\sigma_{max} = 2*\sigma_{sh}$, and $\sigma_{min} = \sigma_{sh}/4$. Initially, in the first generation, a niche is added to the nicheset for each individual in the population, with its midpoint centred on that individual and its niche radius calculated using equation (7). In the initial and each subsequent generation, the following process is executed, in place of any fitness scaling procedures.

1) Determine the current members of each niche by comparing every member of the population to each niche in the nicheset, using equation (6). If an individual is not a member of a niche, a new niche is spawned centred on that individual, and it is added to the nicheset.

2) Recalculate the midpoints of each of the niches in the current nicheset using the *Fitness Distribution from Midpoint* scheme (Gan & Warwick, 1999). Here, the midpoint of niche *j* is modified by using:

$$mid_j = mid_j + \frac{\sum_{i=1}^{n_j} (v_i - mid_j) * f_i}{\sum_{i=1}^{n_j} f_i} \tag{8}$$

Where mid_j is the midpoint vector of niche *j*, v_i is the vector phenotype value of individual *i*, f_i is the raw fitness of individual *i*, and n_j is the number of individuals within niche *j*. Thus, the midpoint is moved to the highest density of most fit individuals currently within the niche.

3) The niche members are recalculated. If a niche has no members it is removed from the nicheset.

4) Each niche in the nicheset is compared to every other niche and the euclidean distance between their midpoints is calculated. These values are stored along with references to the two niches that were compared in an array of nearest niche pairs. This array is then sorted on the distance using a quicksort algorithm.

5) The sorted nearest pair array is then cycled through and each pair of niches is analysed in the following way. If the midpoint of niche *i* is less than half the niche radius of niche *j* away from the midpoint of niche *j* (or vice versa), then the two niches are *merged* together and replace niche *i* (this is described in detail in section 2.1). The *merge* action may actually move the midpoint of niche *i*, so part of the nearest pair array may need to be recalculated and resorted. All references to niche *j* are removed from the nearest pair list. It is not necessary to analyse every single entry in the entire nearest pair array. Because they are sorted on distance, once a point is reached where the niche pairs are further than σ_{max} apart from each other, this subprocess can be terminated.

6) The niche members are recalculated once more and finally the sharing function is applied to each of the individuals within the population. See section 2.2 for further details.

2.1 MERGING TWO NICHES

As stated previously in step 5), two niches, *i* and *j*, will be merged into a single niche if the midpoint of niche *i* is less than half the niche radius of niche *j* away from the midpoint of niche *j* (or vice versa), i.e. if

$$\left\| mid_j - mid_i \right\| \le \frac{\sigma_{sh_j}}{2} \; OR \left\| mid_j - mid_i \right\| \le \frac{\sigma_{sh_i}}{2} \tag{9}$$

The new midpoint is calculated using the *Fitness Distribution from Midpoint* scheme (Gan & Warwick, 1999). This has the effect of moving the midpoint to the average weighted distance of all of the individuals within both niches, from the naïve midpoint. The naïve midpoint is defined as:

$$mid_{naive} = mid_i + \frac{mid_j - mid_i}{2} \tag{10}$$

The new midpoint is defined as:

$$\begin{aligned} mid_{new} &= mid_{naive} + \frac{w_i + w_j}{\sum_{a=1}^{n_i} f_a + \sum_{b=1}^{n_j} f_b} \\ w_i &= \sum_{a=1}^{n_i} (v_a - mid_{naive}) * f_a \qquad a \in i \\ w_j &= \sum_{b=1}^{n_j} (v_b - mid_{naive}) * f_b \qquad b \in j \end{aligned} \tag{11}$$

Here, v_i is the vector describing the position of individual *i* in parameter space, f_i is the fitness of individual *i*, and n_j is the niche count of niche *j*. The niche radius of the new niche is determined by comparing the two original niches. If niche *i* was completely encompassed by niche *j*, then the new niche radius is simply the old radius of niche *j*.

Conversely, if niche *j* was completely encompassed by niche *i*, then the new niche radius is simply the old radius of niche *i*. It is possible to determine if a niche completely encompasses another niche by using the equation for a line, and the equation for a sphere in n-dimensions.

$$L(t) = Q + t\vec{v}$$
$$(X - P) \bullet (X - P) = r^2 \qquad (12)$$

Here, *L(t)* is the line that passes through the midpoints of niche *i* and niche *j*, *P* is the midpoint of the niche in question and *r* is its niche radius. Substituting *L(t)* for *X* and solving the resultant quadratic equation for *t* yields two possible values of *t*. These two values are where the line *L(t)* intersects the boundaries of the niche with midpoint *P* and radius *r*. By comparing the relevant values of *t* for the intersection of *L(t)* with niche *i*, and the intersection of *L(t)* with niche *j*, it is straightforward to determine if one niche is completely encompassed by the other niche.

If neither niche is encompassed by the other, then the new midpoint is compared to the original midpoints of niche *i* and *j*. If the new midpoint is closer to niche *i*, then the new niche radius is calculated as being the distance from mid_{naive} to the furthest extent of niche *i*'s hypersphere, along *L(t)* away from niche *j*. Conversely, if the new midpoint is closer to niche *j*, then the new niche radius is calculated as being the distance from mid_{naive} to the furthest extent of niche *j*'s hypersphere, along *L(t)* away from niche *i*. If the new midpoint is equidistant from niche *i* and niche *j*, then the new niche radius is calculated as half the distance between the furthest extents of either niche's hypersphere, in both directions along *L(t)* from the naïve midpoint.

2.2 THE SHARING FUNCTION

The original function employed in (Gan & Warwick, 1999) and in *Dynamic Niche Sharing* (Miller & Shaw, 1995) was simply the number of individuals in the niche. A somewhat more complicated sharing function has since been adopted, and is very similar to the function used by (Yin & Germay, 1993), see equation (5).

$$m_i = n_j - n_j * \left(\frac{d_{ij}}{2\sigma_{sh_j}} \right)^{\alpha} \text{ if ind.} i \in \text{ niche } j \qquad (13)$$

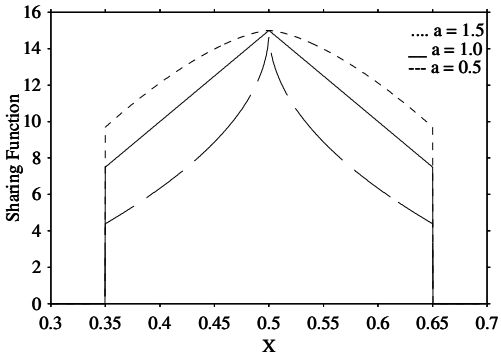

Figure 1: Triangular Sharing Function (1D)

Here, m_i is the niche count of individual *i*, n_j is the number of individuals in niche *j*, d_{ij} is the euclidean distance between individual *i* and the midpoint of niche *j*, σ_{shj} is the niche radius of niche *j* and α is a constant (set to 1.0 for this paper). Figure 1 shows the effects of α on the sharing function of a niche with midpoint 0.5, niche radius 0.15 and a niche count of 15. The shared fitness of an individual is calculated by dividing its raw fitness by its niche count, m_i. But what happens when an individual is a member of more than one niche? We provide an answer to this question in the next section.

2.2.1 Striation: The Effect of Overlapping Niches

In the original DNC scheme any individuals that were members of more than one niche would have their raw fitness divided by two or more sharing functions. The net result of this was to multiply all of the niche counts, m_i, together and divide the raw fitness by this value. This had the effect of setting the individuals raw fitness to a very low value relative to the other individuals, and hence that individual would not be selected in the next generation. Thus, where two niches overlap, we get the *striation* phenomenon that is evident in Figure 2. Figure 2 shows the population spread per generation of the original DNC scheme run on Deb F1 (Deb, 1989). Individuals are represented by the symbol ◇, and the niches in each generation are displayed slightly above the generations y-position. The niche midpoints are represented by an 'x' with x-errorbars to either side to show the niche radius. Here, one of the maxima is at *x=0.7* but it is quite obvious that there are two subpopulations to either side of this value. This is the *striation* effect; where two niches, traversing up the slope on either side of the peak, start to overlap but don't converge due to the heavy penalisation incurred by individuals from being a member of two niches. The GA then forms two stable subpopulations in each niche. This problem is further compounded as the dimensionality of the search space is increased, as it is more likely that an individual will be a member of at least more than one niche because niches will be traversing up the slopes of peaks in many different dimensions.

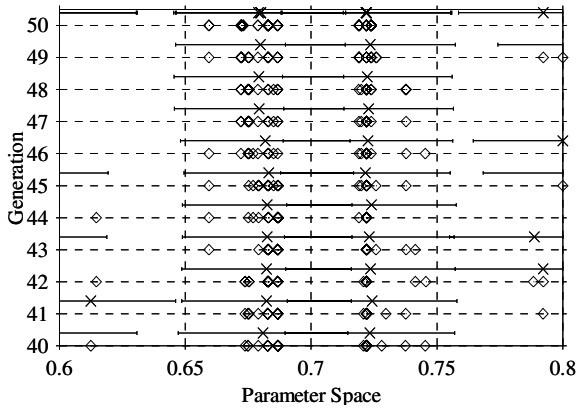

Figure 2: The Striation Effect

There are conceivably three possible ways of dealing with individuals that are members of more than one niche.

First, we can use the method employed in the original DNC scheme as described above, whereby we divide the raw fitness of an individual by the product of all the niche counts, m_i, of the niches to which the individual is a member. This, however, leads to the *striation* effect mentioned previously. Second, niches can be treated in exactly the same way as individual's niche counts are treated in standard fitness sharing. That is, the raw fitness of an individual is divided by the sum of all the niche counts, m_i, to which the individual is a member. Here, there is still a heavy penalty incurred from being a member of more than one niche, but less so than in the first approach. Third, there is no penalty incurred for being a member of more than one niche. An individual's raw fitness is only divided by the largest niche count, m_i, of all the niches to which the individual is a member. Because DNC relies on overlapping niches converging upon one another, the third approach was employed throughout this paper.

3 THE ISSUE OF COMPLEXITY

Complexity is a fundamental issue to all optimisation algorithms so we provide here a conservative estimate of the number of evaluations required to implement DNC per generation. The main contributor to the complexity of this algorithm is within stage 4) of the process described in section 2. Here, each niche in the nicheset is compared to every other niche and the euclidean distance between their midpoints is calculated. These values are stored along with references to the two niches that were compared in an array of nearest niche pairs. This array is then sorted on the distance using a quicksort algorithm. With a nicheset consisting of n niches, this subprocess has a complexity of $O((n^2\text{-}n)/2 + (n^2\text{-}n)/2 \log (n^2\text{-}n)/2)$ as it is not necessary to compare a niche to itself, nor is it necessary to compare niche j to niche i if niche i has already been compared to niche j. So the overall complexity becomes:

$$O(\frac{(n^2-n)}{2}+\frac{(n^2-n)}{2}\log\frac{(n^2-n)}{2}+np) \qquad (14)$$

Where p is the population size of the GA, and n is the number of niches currently in the nicheset. In the initial generation there is an equivalent number of niches and individuals. This actually results in a greater number of comparisons than standard sharing alone. However, in later generations, when both the populations of individuals and niches have stabilised, the number of comparisons will be at a minimum and actually several orders of magnitude faster than standard sharing. The rate at which DNC approaches this optimum number of comparisons is dependent on the fitness function and the current level of selection pressure. Higher selection pressure causes the population to converge towards the peaks more rapidly.

It is clear from Figure 3 that if there is a significantly low ratio of niches to population members (approximately 200 niches in a population of 500) then the complexity of standard sharing and DNC is the same. In the later generations, however, this case is very unlikely because it would require each niche to have a population of less than three individuals. In the light of experimentation, niche counts of 4-10 are the minimum size required for stability. If we are to assume that a niche size of 5 is the absolute minimum, then $p/5$ is the maximum number of niches that DNC is capable of sustaining. Comparing this ratio to Figure 3, DNC is approximately twice as fast as standard sharing.

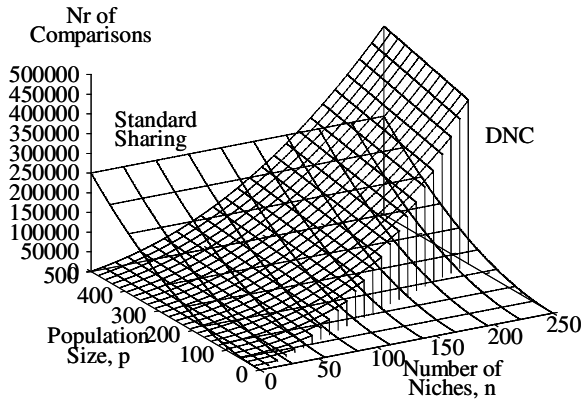

Figure 3: Complexity of Fitness Sharing vs DNC

4 THE FUNCTION TESTBED

The test-bed of functions used was collated from a number of sources, (DeJong, 1975; Deb, 1989; Patil, 1995; and Beasley *et al,* 1993). Detailed descriptions of all functions can be found in the relevant documentation. The test suite described in Table 1 incorporates a variety of functions that exhibit a number of different traits, including continuous, discontinuous, unimodal, multimodal, linear, non-linear, and low to mid dimensionality properties in order to illustrate that DNC performs consistently and reliably. In Table 1 *Pop* corresponds to the population size used in each GA run, and *D* is the number of dimensions in the search space.

Table 1: Test Function Suite

Fn	Description	Pop	D
F1	Deb F1 – Equal maxima	76	1
F2	Deb F2 – Decreasing maxima	76	1
F3	Deb F3 – Uneven maxima	76	1
F4	Deb F4 – Uneven Decreasing maxima	76	1
F5	Himmelblau function	200	2
F6	DeJong F1 – Sphere model	300	5
F7	DeJong F2 – Rosenbrock's Saddle	200	2
F8	DeJong F3 – Step function	400	5
F9	DeJong F4 – Quartic function	400	5
F10	DeJong F5 – Shekel's Foxholes	300	2
F11	Sines function	300	2
F12	Goldstein and Price function	200	2
F13	Schwefel's function	250	3
F14	Rastrigin's function	200	2
F15	Ackley's function	200	3

5 RESULTS

The following genetic parameters were used in all test runs. A mutation rate of p_m=0.0 was employed in order to allow analysis of DNC without the effects of noise. Crossover rate, p_c=0.66, with single point crossover. Remainder stochastic sampling with replacement (Goldberg, 1989) and no elitism was employed. G_{gap}=1. Parameters were encoded using 15 bits. DNC was compared to standard sharing (Goldberg & Richardson, 1987) with niche radius defined by equation (3).

Table 2: Average Performance of DNC

Fn	Success Rate	Std Dev.	RMS Error	Std Dev.	Time Taken
F1	1.0	0.0	3.7e-3	1.88e-6	3.8
F2	0.96	6.4e-3	5.2e-3	7.01e-6	3.8
F3	1.0	0.0	5.1e-3	4.38e-6	3.8
F4	0.98	3.6e-3	5.6e-3	2.21e-6	3.8
F5	1.0	0.0	0.4779	0.0107	59.7
F6	1.0	0.0	0.4489	0.0743	232.2
F7	1.0	0.0	0.9615	0.0858	70.4
F8	0.942	2.4e-3	0.843	0.091	180.0
F9	0.878	0.112	0.1523	0.04	422.3
F10	1.0	0.0	0.7144	6.1e-3	131.0
F11	0.9755	2.33e-4	0.2957	3.18e-4	198.9
F12	0.94	0.02	0.3728	0.0171	70.7
F13	0.923	8.4e-3	0.762	0.076	212.0
F14	0.942	0.051	0.0472	5.6e-3	72.1
F15	0.994	6.3e-3	0.352	0.021	174.0

Table 3: Average Performance of Standard Sharing

Fn	Success Rate	Std Dev.	RMS Error	Std Dev.	Time Taken
F1	1.0	0.0	4.1e-3	4.5e-6	18.2
F2	0.98	3.4e-3	4.7e-3	3.6e-6	17.5
F3	1.0	0.0	3.9e-3	3.1e-6	18.0
F4	1.0	0.0	4.5e-3	1.6e-6	18.0
F5	1.0	0.0	0.3605	8.3e-3	150.3
F6	1.0	0.0	0.8431	0.1187	762.1
F7	1.0	0.0	0.9794	0.0345	151.5
F8	0.988	2.8e-3	0.324	7.82e-3	1295.3
F9	1.0	0.0	0.1132	1.6e-3	1350.4
F10	1.0	0.0	0.82895	0.0216	316.2
F11	0.9959	6.7e-5	0.2638	2.48e-4	320.0
F12	0.97	2.2e-3	0.1036	1.8e-3	149.2
F13	0.96	0.089	0.921	0.054	306.0
F14	0.9	0.0159	0.153	8e-3	149.7
F15	0.94	0.016	0.623	0.021	212.3

Tables 2 and 3 show the overall performance of DNC and standard sharing averaged over 10 GA runs. To measure performance we considered the *success rate*, *accuracy*, *speed* and *consistency* (Beasley *et al*, 1993) of the two sharing schemes. We define *success rate* as the ratio of peaks discovered to actual peaks in the fitness landscape. *Accuracy* is defined as the RMS error of the distance between the best individual in the each of the niches and the nearest maxima. We measure *speed* by the length of time (in seconds) taken to complete 50 generations. *Consistency* is described by the standard deviations of the success rate and accuracy measures defined earlier, over the 10 runs. For standard sharing, an individual was considered to be a member of a niche if its fitness was within 80% of the peak fitness of the nearest peak.

5.1 ANALYSIS OF RESULTS

As can be seen from Tables 2 and 3, DNC does not quite perform as well as standard fitness sharing with regards to success rate and accuracy, but it outperforms standard sharing in speed by several orders of magnitude in all functions. It is also apparent that DNC is also able to consistently perform well.

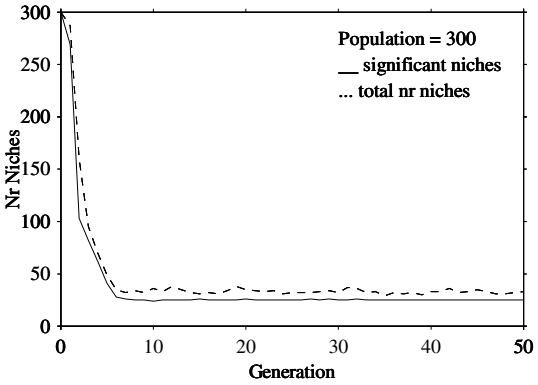

Figure 4: Number of Niches per Generation (F10)

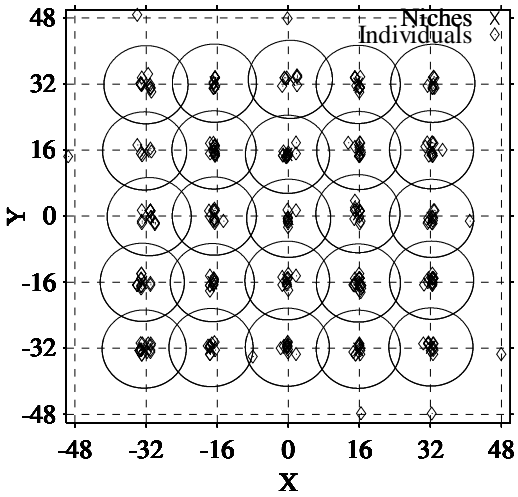

Figure 5: Final Niche & Population Spread (F10)

Figure 4 shows the number of niches per generation for a single GA run on function F10 (DeJong F5), Shekel's Foxholes. The diagram shows both the total number of niches and the number of *significant* niches. A *significant* niche is one that has a population size greater than 2. It is clear how, initially, there are an equivalent number of niches and individuals, but the nicheset rapidly approaches and maintains its optimum number of 25 *significant* niches. When we compare this ratio of 25 niches to 300 individuals to the complexity diagram in Figure 3, it is evident that DNC performs many orders of magnitude faster than standard sharing.

Figure 5 shows a plot of the niche population spread and the population spread in the final generation for F10 using DNC. A niche is described by a circle with midpoint at the niche midpoint and radius equal to the niche radius of the niche. Here, it is apparent that DNC has successfully identified all 25 peaks, and is maintaining populations of individuals clustered closely around the summit of each optimum.

5.2 SO IS IT TRULY VARIABLE RADIUS?

From Figure 5 it is not immediately evident that DNC does actually form niches of variable radius. There is little variation in the niche radii because each of the peaks in F10 have the same hypervolume. Figure 6 shows a simple 1-dimensional problem with two peaks of drastically different hypervolumes. The position and size of the final niche and population spread at generation 50 with population size 50 using DNC on this function are also shown in Figure 6. Individuals are shown by a ◊ symbol. A niche is described by 'x' at the midpoint, with x-errorbars to either side to describe the niche radius. The location of the niches in the y-axis are arbitrarily chosen to best contrast the relative positions of the niches and the individuals. With an initially calculated niche radius of 0.021, based on equation (7), it is apparent that DNC has successfully identified the two peaks in the fitness landscape, but the radii of the two niches are different and reflect the characteristics of the underlying peaks. The niche population sizes were 24 for the niche on the peak at 0.058, 24 for the niche on the peak at 0.159 and 2 for the niche at 0.0095.

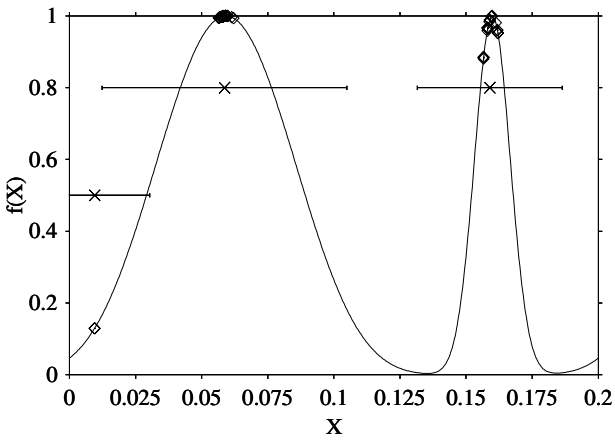

Figure 6: Final Niche & Population Spread

5.3 FUTURE WORK

In the original DNC scheme a *separate* function (Gan & Warwick, 1999) was described that has been omitted from this paper. The *separate* function was designed to reduce the niche radii of any niches that were close enough together so as to overlap, but not actually to be *merged*. This actually prevented the algorithm from working effectively and led to the premature contraction of niches to the minimum size of niche radius. The *merge* function can potentially increase or decrease the niche radius, but the tendency is towards an increase. A counteracting function is therefore required to prevent niches from expanding excessively. Research is currently underway into developing such a function. Either a weakened version of the original *separate* function, or a function that analyses the individuals within each niche and uses their position and spread from the midpoint to modify the radius could be viable solutions. This could also lead to the generation of a *split* function that will split a niche in two if there are two or more distinct populations of individuals within a single niche.

The triangular sharing function described previously in section 2.2 has large discontinuities at the edges of the niche. If the niche radius does not completely encompass a peak, then at the edges of the niche there can potentially be areas of higher fitness relative to the individuals within the niche. This can sometimes lead to small subpopulations forming at the edges of such niches. This phenomenon can be seen manifesting itself in Figure 6. The niche with midpoint at 0.0095 has actually formed at the edge of the niche with midpoint 0.058, on the slope of the larger peak. Figure 7 shows a plot of the modified fitness landscape as seen by the GA after DNC and sharing has been applied. The modified fitness of the two peaks is the same, as is to be expected, but the fitness landscape now has two large peaks at the edges of the niche at 0.058. Relative to the two actual peaks, the fitness of the spikes is much higher, so any individuals that land on them are very likely to form their own niche. In order to combat this and eliminate the discontinuities at the edges of niches, a possible solution could be to double the radius of each niche, but modify the triangular sharing function so that it extends from its population size down to 1, instead of the population size divided by 2. This would result in a much smoother modification of the fitness landscape and reduce the problem of 'phantom' peaks induced by fitness sharing.

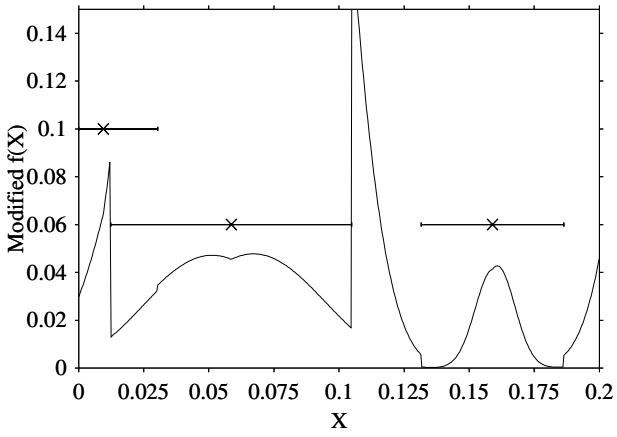

Figure 7: Modified Fitness Landscape after Sharing

A logical progression from DNC is in describing a niche by a hyperellipse rather than by a hypersphere. This would allow the formation of irregular shaped niches, and is a subject of future research for the authors. By the very

nature of DNC, local elitism and niche mating restriction schemes are very easy to implement. Initial tests show that significant improvements in performance are attainable.

Unfortunately, because DNC relies on a decoded parameter space distance metric, it has not been tested on any other kind of problem, for example the deceptive trap *unitation* functions (Ackley, 1987 and Goldberg *et al,* 1992). Research is currently underway on adapting DNC so that it uses a genotype hamming distance as a similarity metric. This would then allow the technique to be extended to problems that do not rely on euclidean space.

6 CONCLUSION

In this paper, we have described a novel and unique method of locating all the optima within a fitness landscape with very little prior knowledge of that landscape. This is achieved through use of a variable radius niching scheme that uses the population size to determine an initial niche radius. The principle behind this is the more individuals in the population there are, the more peaks we can expect to find and populate.

Implementing a non-overlapping, fixed radius niching scheme will only allow the formation of a fixed number of niches. In most cases, the number and size of the peaks in the fitness landscape must be known beforehand, so we can only ever expect to find p niches, irrespective of the population size. Utilising a variable radius niching scheme allows the GA to form as many niches as is required. This removes the necessity for knowledge of the fitness landscape, *a priori*. Dynamic Niche Clustering is a first attempt at providing a robust and reliable variable radius niching scheme for multimodal function optimisation.

Although it does not perform quite as well as standard fitness sharing with regards to accuracy, DNC clearly outperforms fitness sharing in terms of speed. It is the belief of the author that achieving good, robust, near-optimal solutions in a reasonable time is better than obtaining the optimal solutions (which may not even be attainable) over a prolonged and unrealistic time-scale.

It could be argued that the method presented in this paper is an overly complicated means of achieving multimodal function optimisation without *a priori* knowledge, but unless one is willing to submit to the unreasonable limitations of standard fitness sharing, it is necessary to implement a variable radius niching scheme.

Acknowledgements

I would like to thank the EPSRC (Engineering and Physical Sciences Research Council) for funding this research (award no: 97311359), and Prof. Kevin Warwick for his invaluable assistance in guiding this work in the right direction.

References

D.H.Ackley (1987). An Empirical Study of Bit Vector Function Optimization. In L.Davis (ed.) *Genetic Algorithms and Simulated Annealing*, 194-200, Pitman.

D.Beasley, D.R.Bull & R.R.Martin (1993). A Sequential Niche Technique for Multimodal Function Optimization. *Evolutionary Computation* **1**(2):101-125.

K.Deb (1989). Genetic Algorithms in Multimodal Function Optimization. Masters thesis, TCGA Report No 89002, Dept. Engineering Mathematics, Univ. Alabama.

K.Deb & D.Goldberg (1989). An Investigation of Niche and Species Formation in Genetic Optimization. In *3rd International Conference on Genetic Algorithms*, 42-50.

K.A.DeJong (1975). Analysis of the Behaviour of a Class of Genetic Adaptive Systems, Ph.D. thesis, Univ. Michigan.

J.Gan & K.Warwick (1999). A Genetic Algorithm with Dynamic Niche Clustering for Multimodal Function Optimisation, In *4th International Conference on Artificial Neural Networks and Genetic Algorithms*, 248-255.

D.E.Goldberg (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.

D.E.Goldberg & J.Richardson (1987). Genetic Algorithms with Sharing for Multimodal Function Optimization. In *2nd International Conference on Genetic Algorithms*, 41-49.

D.E.Goldberg, K.Deb & J.Horn (1992). Massive Multimodality, Deception and Genetic Algorithms. In R.Manner & B.Manderick (Eds.) *Parallel Problem Solving from Nature, 2*, 37-46, North-Holland.

M.Jelasity & J.Dombi (1998), GAS, a Concept on Modeling Species in Genetic Algorithms. In *Artificial Intelligence, 99(1)*, 1-19.

B.L.Miller & M.J.Shaw (1995). Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization. In *IEEE International Conference on Evolutionary Computation*, 786-791.

C.K.Oei, D.E.Goldberg & S.Chang (1991). Tournament Selection, Niching and the Preservation of Diversity. IllIGaL Report No 91011, Dept. General Engineering, Univ. Illinois.

R.B.Patil (1995). Intervals in Evolutionary Algorithms for Global Optimization. Supplement to *International Journal of Reliable Computing*.

S.Tsutsui,Y.Fujimoto & A.Ghosh (1997), Forking GAs: GAs with Search Space Division Schemes, In *Evolutionary Computation*, Vol.5, No.1, 61-80, MIT Press.

X.Yin & N.Germay (1993). A Fast Genetic Algorithm with Sharing Scheme using Cluster Anlaysis Methods in Multimodal Function Optimization. In *International Conference on Artificial Neural Networks and Genetic Algorithms*, 450-457.