Every Niching Method has its Niche: Fitness Sharing and Implicit Sharing Compared

Paul Darwen and Xin Yao
Computational Intelligence Group
School of Computer Science
University College, The University of New South Wales
Australian Defence Force Academy
Canberra ACT 2600 AUSTRALIA
Email: darwen@cs.adfa.oz.au, xin@cs.adfa.oz.au

ABSTRACT

Several extensions to the GA attempt to find all or most optima in a search space containing multiple optima. Many of these methods emulate speciation in natural evolution. Such methods must find all or most optima, in order for co-evolutionary learning to succeed a range of management and control problems, such as learning game strategies. However, studies on which method is suitable for what kind of problems are ver few. In this paper, we compare two similar GA speciation methods, fitness sharing and implicit sharing. Using a realistic letter classification problem, we find they have advantages under different circumstances. Implicit fitness sharing covers optima more comprehensively, even when those optima have small basins of attraction, when the population is large enough for a species to form at each optimum. With a population not large enough to do this, fitness sharing can find the optima with larger basins of attraction, and ignore the peaks with narrow bases, while implicit sharing is more easily distracted. This indicates that for a speciated GA trying to find as many near-global optima as possible, then implicit sharing works well only if the population is large enough. This requires prior knowledge of how many peaks exist.

1. Introduction

1.1. Speciation as automatic modularisation

Real-world problems are often too complex for a single monolithic system. To reduce this complexity, a modular system uses specialized parts (or modules) to solve different aspects of the problem. Many natural and artificial systems show that a modular approach can solve a difficult problem satisfactorily. For example, separate parts of the brain deal with different aspects of vision: colour, depth, movement, detail, and shape [16]. Modular artificial neural networks perform well in speech [17] and image processing [11] [26].

A speciated GA can automatically create different species, each embodying a sub-solution (or module) [6]. However, this approach relies heavily on the speciated GA to find all (or most) near-global optima. If the GA misses some peaks, then the GA's expertise will generalise poorly [5].

A co-evolutionary GA is one where each individual is evaluated by how well it performs against every other individual This is a promising way to learn game strategies [1] [5] [21] [22] [25].

Without speciation, a GA will converge to only one high-fitness solution, due to genetic drift caused by a small population. A co-evolutionary GA will converge and overspecialise to only one high-quality strategy for the game being learned. The overspecialised expertise thus created will generalise poorly, and be naïvely vulnerable to other strategies not in the converged population [5].

Speciation can greatly delay genetic drift [18] [20, Section 8.2], and prevent the convergence and overspecialisation of co-evolutionary learning. Also, speciation lets us combine the diverse expertise embodied in the different species, by using a gating algorithm to choose which high-quality strategy to use when. This approach gives improved generalisation ability from co-evolutionary learning [6]. Each species' high-quality strategy can be used as a separate module, to deal with a different potential opponent's strategy.

This demonstrates that speciation as automatic modularisation can improve evolutionary learning, and promises to be useful in a wide range of problems of management and control.

1.2. This paper

We compare two GA speciation methods in their ability to form species at as many near-optimal peaks as possible, when those peaks have similar fitness, but varying basins of attraction and inter-peak distances.

No comparative results are currently available. A complete mathematical analysis can make use of previous work for both speciation methods, fitness sharing [20] and implicit sharing [24]. However, this would require a number of simplifying assumptions in order to be tractable. These assumptions may not be satisfied in practice. Here, we present empirical results, and find that implicit sharing works better when the population is large enough to form a species at each peak. With too many peaks, fitness sharing is less distracted by peaks with small basins of attraction. So implicit sharing works better *only if* the population is large enough to form a species at each peak of interest. A practical way to deal with this problem might be to do several runs, each with a larger population. When no more peaks are found, you have enough.

With this caveat, it is feasible for a GA with speciation to find all the near-global optima, and thus be suitable for speciation as automatic modularisation [6].

In Section 2, we describe two speciation methods, fitness sharing and implicit sharing. In Section 3, we describe the experimental setup, whose results are shown in Section 4, and discussed in Section 5. Section 6 concludes the paper.

2. Speciation Methods Based on Sharing

2.1. Fitness Sharing

Until recently, fitness sharing was the only successful GA technique for finding multiple optima [20, page 84]. Fitness sharing modifies a search landscape by reducing payoff in densely-populated regions. This encourages search in unexplored regions, and causes subpopulations to form.

Consider an individual i with fitness f_i. Its niche count m_i [15, page 191] [18] measures the number of other individuals with whom i shares fitness. The shared fitness is simply $f_i^s = \frac{f_i}{m_i}$. The niche count m_i is calculated with a distance metric d_{ij} that describes the difference between individuals i and j, usually their Hamming distance:

$$m_i = \sum_{j=1}^{\text{population size}} \mathrm{sh}(d_{ij}) \qquad (1)$$

The *sharing function* $\mathrm{sh}(d_{ij})$ makes distant individuals share less:

$$\mathrm{sh}(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_s}\right)^{\alpha} & \text{for } 0 \leq d_{ij} < \sigma_s \\ 0 & \text{for } d_{ij} \geq \sigma_s \end{cases} \quad (2)$$

Parameter α changes the sharing function's shape: $\alpha = 1$ gives a linear sharing function. More important is σ_s, the *sharing radius* or cutoff distance: two strings σ_s or further apart do not share fitness.

One way to find multiple optima is to make several runs of an ordinary GA. Each run, the GA converges to an optimum, but different runs will probably converge to different optima. Thus, several optima are found. Beasley *et al.* [2] improved this: each time the GA converges to an optimum, the payoff is reduced around that optimum to avoid converging there again. So n runs should find n different optima.

Fitness sharing similarly reduces payoff, but at heavily-populated regions of the search space, and it does it dynamically during a single run. It is a parallel version of the sequential scheme of Beasley *et al.* [2]. Mahfoud [19] [20, chapter 10] finds that the parallel approach works better than the sequential scheme. Although, fitness sharing has achieved remarkable results in difficult search problems [14], it is not without flaws. Firstly, σ_s is fixed, so all optima in the search space must be nearly equidistant [24]. Secondly, to set σ_s, you need to know *a priori* how far apart optima are, and their (unshared) fitness. Until you search the space, this information is unknown. This important flaw is often overlooked by simply assuming that we have perfect discrimination between different peaks [20, pages 106, 158].

These flaws can cause fitness sharing to fail to find all optima if they are not equidistant, or if the estimated distance between optima is incorrect. Also, even in a search space with nearly equidistant peaks of equal fitness, fitness sharing can still fail to find all peaks [4].

2.2. Implicit Fitness Sharing

An extension to the original fitness sharing is implicit sharing [12] [24]. Like the original, implicit sharing also modifies the fitness function. It does this with the algorithm in Table 2.1. In implicit sharing individuals must "match" discrete objects. Originally, in an immune system simulation, antibodies which best matched an invading antigen received the payoff for that antigen [24] [12]. Another use is in learning a game: a strategy receives payoff when it achieves the best score against a sample test strategy [22] [6].

At first glance, Table 2.1 looks completely different to ordinary fitness sharing. Surprisingly, they share the same theoretical basis [24]. The sample size σ resembles the sharing radius σ_s: both control species size, and regulate the number of individuals who obtain payoff by specializing to the same part of the search space.

However, in implicit sharing, the individual

```
For each data point $i$ to be matched, do the following $C$ times:
  1. From the GA population, select a sample of $\sigma$ individuals.
  2. Find the individual in that sample which achieves the highest match score against
     the single data point $i$.
  3. The best in the sample receives payoff. In the case of a tie, payoff is shared
     equally among the tie-breakers.
```

Table: 1: Payoff function for implicit sharing [24].

in the sample which achieves the highest score (no matter how low) receives payoff. This allows implicit sharing to reward the individual closest to a peak, even if it's not particularly close to it, and when another individual is almost as close. This contrasts with fitness sharing. A fixed sharing radius σ_s means the closest individual to a peak shares its payoff with other individuals that are almost as close This reduces the relative selection pressure. We therefore expect implicit sharing to be better at finding such peaks.

3. Experimental Setup

3.1. The Data

We use Frey and Slate's [13] character recognition data set. It has about 770 examples each of all 26 capital letters. Each data point consists of one class (out of 26 letters), plus 16 integer values between 0 and 15 inclusive. These measure certain attributes of the characters. What these attributes are is not important for these results. We choose this set because it is a hard classification task, and because it was used with a Holland-style adaptive classifier [13]. Accurate classification is not our primary interest — our main focus is to compare two GA speciation methods.

We want the speciated GA is to form a species at each cluster of same-letter data points. That is, if the data contains a group of "A" points, close together with few non-A points intermingled, then we would like a species to form that "covers" this cluster, as in Figure 1. We know such clusters exist, because nearest neighbour classification works well on this data set [10]. The test phase will classify points by seeing the letter of the species that covers it.

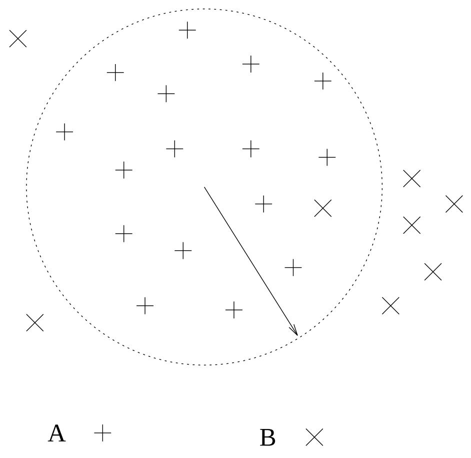

Fig. 1: A "cluster" of data points is when almost all the data near a certain point are of the same letter.

Consider a subset of the data, by randomly taking only 2 letters out of 26. Table 2 shows how scattered this data is. Most points (61.5% of S, 75.8% of V) are in a single cluster, i.e., there is a spot in the 16-dimensional space, such that 61.5% of the S data are closer to this spot than any non-S datum is. This spot is where we want a species. Other clusters exist containing fewer S data.

We represent GA individuals as points in the 16-dimensional integer space. We need 4 bits to represent the 16 values of each of the 16 ranges, making a GA genotype of 64 bits. Our payoff function, described below, gives a fitness landscape with these clusters appearing as peaks of similar fitness.

Letter	Average shortest distance to a different class	Total letters	Average cluster	Largest cluster
S	7.93 ± 1.66	748	189.3 ± 225.3	460 61.5%
V	8.02 ± 1.45	764	278.7 ± 324.0	579 75.8%

Table: 2: How scattered these two randomly-chosen classes are.

Letter	Average shortest distance to a different class	Total letters	Average cluster	Largest cluster
D	4.74 ± 1.13	805	70.5 ± 100.5	251 31.2%
I	5.24 ± 1.00	755	92.2 ± 104.5	204 27.0%
S	4.51 ± 0.97	748	34.7 ± 28.0	138 18.4%
U	5.84 ± 2.03	813	81.1 ± 101.5	226 27.8%
V	5.13 ± 1.08	764	85.7 ± 105.0	257 33.6%
W	5.15 ± 1.08	752	60.4 ± 42.0	213 28.3%
X	4.17 ± 0.69	787	41.0 ± 30.5	125 15.9%

Table: 3: How scattered these randomly-chosen seven classes are.

Table 3 shows what we find for more classes, again chosen at random. Clusters of same-class data points get smaller, because the data points are more intermingled with each other. This requires more species (and thus a larger population) to successfully cover each cluster. Note that the number of *clusters* of data points is different from the number of *classes* of data points.

There are many statistical and neural network approaches to "clustering" data, which can give high classification accuracy. Again, we are not primarily interested in classification accuracy in this paper, but in comparing two GA speciation methods. For this reason, we will not follow the usual approach to classification problems of having separate data sets for the learning and testing phases. We will learn on the same data that we test with. However, generalisation has been considered in our definition of the payoff function.

How well GA methods compare with non-GA methods as covering or clustering algorithms remains an open question. For example, Frey and Slate [13] obtained *worse* accuracy with a Holland-style adaptive classifier than with a simple nearest neighbour classifier [10], which suggests the evolutionary approach needs further development for this application.

3.2. Payoff Function

There are two desirable attributes we want reflected in the payoff function:

1. **Quantity**: Each species should classify as many data points as possible. One species per data point needs a huge population.
2. **Quality**: Each species should classify data points as accurately as possible. We do not want a species to cover many data points with low accuracy.

Although we didn't use a separate testing set, in terms of generalisation, increasing the importance of the the quality criterion has the effect of Occam's Razor allowing a more complicated solution, but which generalises more accurately. To balance these two desires, the payoff function assigns equal fitness to an individual that covers many data points with reasonable accuracy, as to an individual that covers fewer points with higher accuracy. This is analogous to Occam's Razor [3]. We do this by making the payoff proportional to both, raised to a certain power. Let c_i be the number of *correct* data points that are "covered" (in some sense) by individual i, and let n_i be the *total* number of data points covered. Then we let (unshared) fitness f_i be:

$$f_i = \left(\frac{c_i}{n_i}\right)^{\alpha} (c_i)^{\beta} \qquad (3)$$

To set the parameters α and β, imagine two individuals: one that "covers" (in some sense) 100 data points, of which 96 are of the same class with 4 others; and another that "covers" 200 data points, but only 180 are the same letter. We would like these two individuals to have the same fitness. This requires that:

$$\begin{aligned}\left(\frac{96}{100}\right)^{\alpha}(96)^{\beta} &= \left(\frac{180}{200}\right)^{\alpha}(180)^{\beta} \\ \Rightarrow \frac{\alpha}{\beta} &= 9.740 \approx 10\end{aligned} \tag{4}$$

From Equation 4, we take $\alpha = 10$ and $\beta = 1$. We use linear ranked selection, so only the relative values of α and β matter.

Note that the payoff function means that smaller clusters of same-class data points do not necessarily form sub-optimal peaks in the search space. A small number of data points with low errors gives the same payoff as a large cluster of data points with more errors. The only difference is the small cluster's peak in the fitness landscape would be surrounded by a smaller basin of attraction, and be more difficult for the GA to discover, or maintain a species at.

What do we mean when an individual "covers" a data point? We tried having a radius as a parameter of every individual, so that some might have a large radius and others a small radius. This caused positive feedback, with the whole population quickly evolving the maximum radius. So we used a nearest-individual approach. An individual "covers" a data point if it is closer to that data point than any other data point.

Other parameters were set as follows:

1. The sharing radius σ_s was 5, using the Euclidean metric in the 16-dimensional space. Setting σ_s is usually guesswork, as one does not know inter-peak distances before searching. But we allow ourselves a glance at Tables 2 and 3.
2. We used assortative mating, so like mates with like. This can improve a speciated GA [7] [8, page 49]. Also, incest prevention [9] was used, i.e., individuals mate with the most similar non-identical partner left in the population.
3. The mutation rate was 0.01, so a 64-bit genotype has a 52.6% chance of escaping unmutated. This is high, but assortative crossover reduces disruption.
4. The crossover rate was 1. The entire population was replaced each generation, except for the best individual in each generation which was kept, i.e., elitism was used.
5. Linear ranked selection was used — the highest scoring individual expects to receive 2 offspring, declining linearly to the worst individual expecting 0 offspring.
6. Population size was 50 for all runs.

4. Results

For both speciation methods, the GA had usually substantially converged by 50 generations. At the fiftieth generation, the test phase takes the best individuals of the final generation (down to a cutoff, described below). For every data point used in the learning phase, we see which of those individuals is closest to (i.e., covers) that data point. If the point is of a different letter to the individual, it is counted as an error. If they are the same, it is a correct classification.

We only want the best individuals from the final generation to be tested. Otherwise, we may include bad mutations, which would bring down the result. How much of the final generation we reject is found in the next two tables. Table 4 shows the results for fitness sharing, and in Table 5 for implicit sharing. The "Cutoff" columns indicate how many low-performing individuals were not used in the test phase: a "Cutoff" of 10% means the test phase did not include any individuals who scored 10% or less of the average score of the final generation. All individuals who score zero are also excluded, so if the "Cutoff" is 0% we accept all individuals with non-zero fitness.

In Tables 4 and 5, the best cutoff values are asterisked. Using these in the test phase, we find which speciation method got a better classification rate, as shown in Table 6. For each of the 100 runs, we show which of the two

Classes	0% Cutoff	10% Cutoff	25% Cutoff	50% Cutoff	100% Cutoff
2	0.9368	0.9451	0.9452	0.9455*	0.9350
3	0.8404	0.8464*	0.8409	0.8338	0.8094
4	0.7445	0.7467*	0.7389	0.7254	0.6956
5	0.6528*	0.6339	0.6180	0.5964	0.5618
6	0.5756*	0.5401	0.5165	0.4982	0.4698
7	0.5391*	0.4975	0.4836	0.4669	0.4345

Table: 4: Classification rates for 100 runs of fitness sharing, having rejected those who score less than the above fraction of the average payoff. Best are asterisked.

methods had the better classification rate. Is there a significant difference in the number of wins in Table 6? We use a binomial test, as each set of classes are drawn randomly from the alphabet. This needs the number of times one method is better, and ignores the ties. We try two different definitions of a tie in Table 6. The first is the usual one, where a method is better even if it classifies only one extra letter correctly. As this can be a narrow margin, we also consider a tie to be when the difference in classification rate is less than 2% of the total number of letters. Each class contains about 770 letters. This avoids counting one method as better when it narrowly won. Consider n non-ties, with s of those n wins were method A beating method B, so that method A seems to be better. If there is actually no differences between the two methods, then $p = q = 0.5$ in Equation 5, and the probability that A and B are equally good is [23]:

$$\text{Chance they're same} = \frac{n!}{s!(n-s)!}p^s q^{n-s} \quad (5)$$

Equation 5 lets us find the confidence that one method is better, as it seems to be. The results are shown in Table 6.

5. Discussion

Returning to Tables 4 and 5, note that for few classes, the best results ignore poor performers. With more classes, we can no longer afford to ignore poorer performers, and the best results are with 0% cutoff. Also note that the two methods can afford to ignore poor performers at different times: from Table 4, fitness sharing needs to use all individuals with non-zero payoff when classifying 5 classes or more; but from Table 5, on the same data, implicit sharing performs best by ignoring poorly performing individuals. That is, it takes longer for the asterisks to move left in Table 5.

Classes	0% Cutoff	10% Cutoff	25% Cutoff	50% Cutoff	100% Cutoff
2	0.9449	0.9505	0.9521	0.9528*	0.9513
3	0.8458	0.8648	0.8663*	0.8640	0.8514
4	0.7370	0.7529*	0.7454	0.7390	0.7163
5	0.6266	0.6541*	0.6450	0.6285	0.5909
6	0.5349	0.5468*	0.5292	0.5140	0.4757
7	0.4955*	0.4930	0.4776	0.4549	0.4117

Table: 5: Classification rates for 100 runs of implicit sharing, having rejected those who score less than the above fraction of the average payoff. Best are asterisked.

No.	Wins		Ties	Prob	Tie
lets	FS	IS		same	margin
2	43	57	0	9.67e-02	0%
3	40	60	0	2.84e-02	0%
4	47	53	0	3.09e-01	0%
5	48	52	0	3.82e-01	0%
6	60	40	0	2.84e-02	0%
7	72	28	0	6.29e-06	0%
2	8	17	75	5.39e-02	2%
3	21	37	42	2.40e-02	2%
4	41	44	15	4.14e-01	2%
5	38	46	16	2.23e-01	2%
6	52	33	15	2.51e-02	2%
7	58	18	24	2.36e-06	2%

Table: 6: This shows how many runs (out of 100) in which one speciation method was better, out of fitness sharing (FS) and implicit sharing (IS). The right-hand column shows the probability that the two methods are in fact equal. When these probabilities are around five percent, we can confidently say that it's not a fluke.

From Table 6, we see an interesting transition. With enough population for a species to cover each cluster, implicit sharing does a better job of comprehensive coverage. With more classes — and thus with more clusters to cover — fitness sharing does not degrade as rapidly. Figure 2 shows this transition.

This is because implicit sharing's selection is more discriminating. Consider the relative payoffs of the two individuals in Figure 3. Under fitness sharing, these very similar individuals will have a niche count of almost 2, causing the shared fitness to be halved. The only benefit that the closer individual receives is a slightly higher payoff. Thus there is little relative selection pressure.

Under implicit sharing, the probability that w individuals from this subset (of 2) are in a sample of size σ taken without replacement, from a population of size N, is given by [24]:

$$p(w, \sigma, N, 2) = \frac{C_w^2 C_{\sigma-w}^{N-2}}{C_\sigma^N} \tag{6}$$

In these experiments, $N = 50$ and $\sigma = 38$.

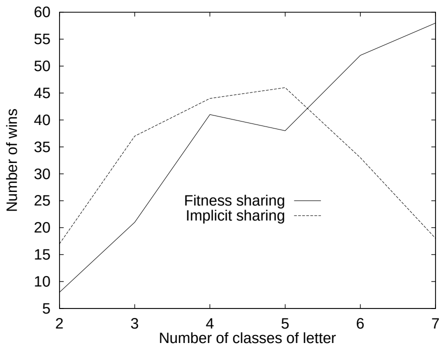

Fig. 2: Implicit sharing is better for few classes, fitness sharing is better for many classes.

Equation 6 gives:

Chance both in sample	$w = 2$	$p = 0.574$	
Only near ind sampled	$w = 1$	$p = 0.186$	
Only far ind sampled	$w = 1$	$p = 0.186$	
Neither in sample	$w = 0$	$p = 0.054$	
			(7)

Only the best individual in the sample receives payoff. For our two individuals in Figure 3, the closer individual will receive payoff when both are sampled ($w = 2$), and when only it is sampled. The farther individual will only receive payoff when its neighbour is not in the sample. There are C samples for each data point. From Equation 7 the expected fitness is:

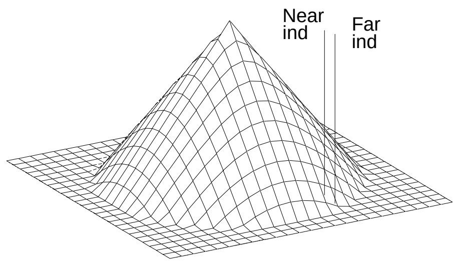

Fig. 3: Consider two similar individuals near a peak.

$$\begin{array}{rcl} f_{near} & = & (0.574 + 0.186)C \\ f_{far} & = & 0.186C \end{array} \Rightarrow \frac{f_{near}}{f_{far}} \approx 4$$

So there is greater *relative* selection pressure for the nearer individual. This is why implicit sharing covers peaks more comprehensively, even peaks with small basins of attraction (i.e., with few data points at high accuracy), given enough population. With a large enough population this is good, but with a small population it still tries to cover every single peak. Fitness sharing is not so easily sidetracked. In a previous study on a massively multi-optima space, fitness sharing worked very well [14], and we conjecture that implicit sharing would not work as well.

So what method is best for a given problem? It depends on how many interesting optima exist in the fitness landscape (which is usually not known in advance), and how large a population is computationally feasible.

Consider our original problem, of using a co-evolutionary GA with speciation to find *all* expert-level strategies in a game [6]. With a large enough population, implicit sharing would be best. Some previous attempts to learn game strategies with a co-evolutionary GA used this method [22] [6].

However, from Figure 2, if the game has more strategies than we expected or our population is too small, then implicit sharing will miss strategies it should have found, and our GA will generalise poorly. Normally, we don't know *a priori* how many near-global optima exist in a search space, so we don't know how large the population must be. A practical way to deal with this problem might be to do several runs, each with a larger population. When a larger population does not find any more peaks of interest, it is big enough.

6. Conclusion

Both speciation methods are good in different ways. As shown in Figure 2, implicit sharing performs better when the population is more than large enough for a species to cover each optima of interest. When the population is not large enough to do this, both methods degrade, but implicit sharing degrades considerably faster. This is because implicit sharing exerts a more discriminating evolutionary attraction between similar individuals, allowing it to covers peaks more comprehensively. This is fine with a large enough population, but with a small population it still tries to cover every single peak. In this situation, fitness sharing is not as easily distracted.

References

[1] Robert M. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In *Genetic Algorithms and Simulated Annealing*, chapter 3, pages 32–41. Morgan Kaufmann, 1987.

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

[3] A. Blumer, A. Ehrenfeucht, D. Haussler, and M.K. Warmuth. Occam's razor. *Information Processing Letters*, 24(6):377–380, 1987.

[4] Paul Darwen and Xin Yao. A dilemma for fitness sharing with a scaling function. In *Proceedings of the 1995 IEEE Conference on Evolutionary Computing*, pages 166–171, 1995.

[5] Paul Darwen and Xin Yao. On evolving robust strategies for iterated prisoner's dilemma. In *Progress in Evolutionary Computation*, pages 276–292. Springer, 1995.

[6] Paul Darwen and Xin Yao. Automatic modularization with speciation. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computing.* IEEE Press, May 1996. Submitted.

[7] Paul Darwen and Xin Yao. A dilemma for fitness sharing with a scaling function.

IEEE Transactions on Neural Networks, 1996. Submitted.

[8] Kalyanmoy Deb and David Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50, 1989.

[9] Larry J. Eshelman and J. David Schaffer. Preventing premature confergence in genetic algorithms by preventing incest. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 115–122, 1991.

[10] Terence C. Fogarty. First nearest neighbor classification problem on Frey and Slate's letter recognition problem. *Machine Learning*, 9(4):387–388, 1992.

[11] T. C. Folsom. A modular hierarchical neural network for machine vision. In *International Joint Conference on Neural Networks*, volume 2, pages 897–902. IEEE, 1990.

[12] Stephanie Forrest, Brenda Javornik, Robert Smith, and Alan Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211, 1993.

[13] P.W. Frey and D.J. Slate. Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2):161–182, March 1991.

[14] David Goldberg, Kalyanmoy Deb, and Jeffrey Horn. Massive multimodality, deception, and genetic algorithms. In *Parallel Problem Solving from Nature 2*, pages 37–46. North-Holland, 1992.

[15] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[16] Denise Grady. The vision thing: Mainly in the brain. *Discover*, 14:57–66, June 1993.

[17] F.S. Gurgen, J.M. Song, and R.W. King. A continuous HMM based preprocessor for modular speech recognition neural networks. In *Proceedings of 1994 International Conference on Spoken Language Processing*, pages 1507–1510, 1994.

[18] Samir W. Mahfoud. Genetic drift in sharing methods. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, volume 1, pages 67–72, 1994.

[19] Samir W. Mahfoud. A comparison of parallel and sequential niching methods. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 136–143. Morgan Kaufmann, 1995.

[20] Samir W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, 1995.

[21] Craig W. Reynolds. Competition, coevolution and the game of tag. In *Artificial Life 4*, pages 59–69. MIT Press, 1994.

[22] Christopher D. Rosin and Richard K. Belew. Methods for competitive coevolution: Finding opponents worth beating. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380. Morgan Kaufmann, 1995.

[23] Steven L. Salzberg. On comparing classifiers: A critique of current research and methods. Technical Report CS-1995-06, John Hopkins University, May 1995.

[24] Robert Smith, Stephanie Forrest, and Alan Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1992.

[25] Robert Smith and Brian Gray. Coadaptive genetic algorithms: An example in Othello strategy. In *Proceedings of the 1994 Florida Artificial Intelligence Research Symposium*, pages 259–264. Florida AI Research Society, 1994.

[26] M.M. van Hulle and G.A. Orban. The EDANN concept: a modular artificial neural network model for biological vision and image processing. In *World Congress on Neural Networks - San Diego*, volume 4, page 320, 1994.