Evolving Populations of Expert Neural Networks

Joseph Bruce and Risto Miikkulainen
{jbruce|risto}@cs.utexas.edu
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712 USA

Abstract

In standard neuroevolution, the goal is to evolve one neural network that would compute the right answer most often. However, it often turns out that the population as a whole could perform even better, if we could only choose the right network for each input. One way to do this is to evolve networks that output not only the answer, but also an estimate of that answer's correctness. Experiments in the handwritten character recognition domain show that such an evolutionary process, combined with an effective technique for speciation, can create a population of networks that collectively performs better than any individual network.

1 Introduction

In a typical approach to problem solving with evolutionary methods, a genetic algorithm is used to evolve a population of individuals each attempting to solve the task. The most fit individual found during the evolution, the *champion*, is designated as the final result. For example, when neural networks are evolved for a decision task, the champion is the neural network that is most likely to produce the correct decision for any given input. The rest of the population, and the knowledge and expertise it encodes, is simply thrown away.

However, an analysis of the final population shows that there are often other individuals in the population that are able to produce correct decisions for inputs that the champion cannot handle. Figure 1 shows the fitness of the final population in the handwritten character recognition task. Although the champion only identifies 64% of the characters correctly, 98% of the characters are correctly identified by *at least one* individual in the population.

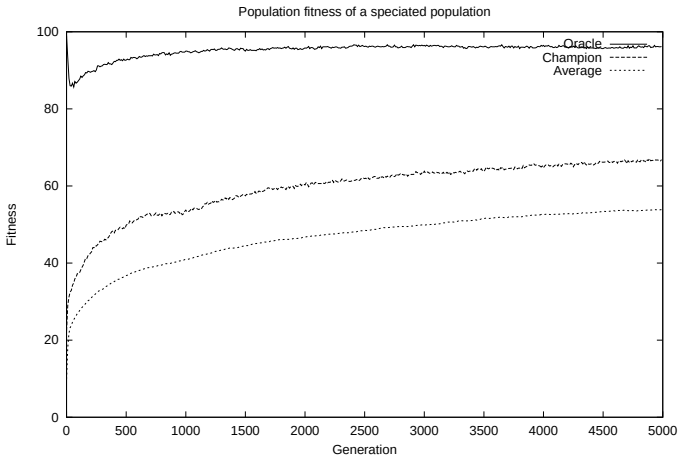

Figure 1: **Percent of correct decisions attainable from a standard neuroevolution population.** Three different measures are used, from top to bottom: the best answer found in the entire population (unrealizable in practice), the answer of the most fit individual, and the population average. Fitness indicates the percentage of correctly-identified test characters in the handwritten character recognition domain (section 4).

We could obtain this level of accuracy by simply choosing the best answer from those produced by the entire population. But how can we determine which individual most likely has the right answer for a given input? One solution is to evolve the individuals not only to produce the answer, but also a level of *confidence* that this answer is correct. The population's answer can be defined as the decision made by the most confident individual.

To appear in *Proceedings of the 2001 Genetic and Evolutionary Computation Conference.*

This idea is tested in the handwritten character recognition task. For the method to work, it is essential to maintain high diversity in the population. Several speciation methods are tested; the island model and its continuous version, the spatial model, are found to be the most effective. With such diversity, the method leads to populations that collectively perform better than any single individual. Confidence evolution of expert neural networks therefore constitutes a promising approach to utilizing the entire population as the result of the evolutionary algorithm.

2 The Method of Confidence Evolution

Figure 1 suggests that in order to make use of the knowledge and expertise in the entire population, the champion might be determined separately for each individual input. Such a selection can be made correctly only based on the correct answer, which is not available during performance. Therefore, although such a high level of performance exists in the population, it is unattainable in practice.

However, it is possible to approximate this selection in various ways. For example, it may be possible to train (or evolve) a meta-level neural network to decide which individual in the population is most likely to produce the right answer for a given input. This is the approach taken for example in the Mixtures of Experts approach [Jacobs et al., 1991], which works well in many supervised tasks. An interesting alternative is to require each individual to rate the quality of each answer it produces. If the population learns to do this effectively, one would be able to outperform the champion by choosing the decision of an individual reporting the highest level of confidence for each decision. This is the approach taken in this paper.

Confidence may be represented by an additional output unit on the neural network. Confidence is that unit's activation when the network is presented with an input. The range of confidence, therefore, is between 0 and 1. To encourage the network to output useful estimates through this unit, the fitness evaluation must be altered. Many fitness evaluations in decision tasks are sums of Booleans, counting the number of correct decisions an individual makes in trials on a training set:

$$f = \sum_i s(\vec{v}_i), \tag{1}$$

where $s(\vec{v}_i) = 1$ if the network's answer on trial i was correct, 0 otherwise.

Training with confidence changes this evaluation in a simple way. It treats each of these trials as a bet. Instead of simply winning 1 each time it is correct, it also stands to lose 1 if it is incorrect. Moreover, the size of the bet is determined by its confidence output $c(\vec{v}_i)$:

$$f = \sum_i s(\vec{v}_i)c(\vec{v}_i), \tag{2}$$

where $s(\vec{v}_i) = 1$ if the answer is correct, $s(\vec{v}_i) = -1$ if it is incorrect. So the network is penalized $s(\vec{v}_i)c(\vec{v}_i)$ for a wrong decision on input $\vec{v}_i$ and is awarded $s(\vec{v}_i)c(\vec{v}_i)$ for a correct decision. This process allows the network to unilaterally set the amount of the bet that its response is correct. It encourages networks to output high confidence only for decisions that are likely to be correct.

The fundamental change to the standard way of evolving neural networks is that in an evolution with confidence, the entire population is considered to be the product of evolution. Answers may be obtained from the population by simply choosing the answer provided by the most confident individual. The specific method for leveraging confidence to extract high-quality decisions is problem-dependent. For example, for some decision tasks the sum of the population's outputs weighted by their confidence might be a useful quantity. But in other domains, such as robotic controllers, it might be better to allow a single individual to provide each decision: weighted sums of different motor outputs could easily lead to a disaster.

In order to use confidence, the population must be diverse enough so that significantly different decisions are made by networks inhabiting distinct niches. A strong method of diversity maintenance (also called speciation or niching), is therefore crucial for success. On the other hand, the restrictions imposed by the speciation technique may adversely affect learning performance for all individuals in the population. A proper speciation technique should strike a balance between these two factors.

3 Methods of Speciation

Speciation is an important design principle in genetic algorithms. Genetic algorithms lose diversity over the course of evolution, converging to a point at which all of the genomes in the population are essentially the same. At this point, crossover operation between two nearly identical genomes is unlikely to create a more fit offspring, and the progress of the GA from that point will be slow, mostly through mutation. A properly speciated population, containing several "niches" of solutions to the task, can continue improving even

after some of the niches reach local maxima from which they are unable to improve. It provides for a more reliable search, with many approaches being explored in parallel throughout the evolution.

Speciation techniques are generally implemented in terms of genomes, rather than the structure implemented by the genomes, or that structure's performance, because a diversity of genomes is needed to search the solution space in parallel. However, this diversity of genomes also results in diversity of behaviors. A speciated population contains a wider range of answers—and is more likely to contain at least one correct response for a particular input—than a homogeneous population. Therefore, speciation can be used to create a population where different individuals are responsible for different inputs.

Methods for promoting diversity may involve changes to different parts of the canonical genetic algorithm. In this paper we will compare speciation techniques that modify the GA's selection scheme, the replacement scheme, and the fitness evaluation function. Also crucial to population diversity is the scaling scheme, i.e. the algorithm that converts the individuals' fitnesses into probabilities of being included in the next generation, either unchanged or combined with another genome by crossover. An aggressive scaling scheme that rewards slightly fitter individuals with much higher probabilities will quickly lead to convergence, as genetic material possessed by moderately-fit individuals will be lost in each generation. More subtle scaling schemes are desirable (and also used in this paper) to delay population convergence.

A very simple approach to speciation is to arbitrarily divide the population into non-interacting subpopulations, or islands. A genome cannot perform crossover with any genome of another island, and a newly created individual may replace only a genome in the island of its parents. In some versions, the island model provides for a small rate of migration between islands. Without migration, this approach is equivalent to running a genetic algorithm independently on each of the islands. Even this trivial approach to speciation can be useful; if the genomes in one island reach a plateau early, others may continue improving. This difference is not directly promoted; it is simply allowed to occur by chance. Under migration, populations on an island are allowed time to make small adjustments before competing with outside genomes [Mühlenbein, 1991].

A more general, continuous version of the island method is the spatial, or topological, method. Each individual may inhabit a vertex of an undirected graph, and it may only perform crossover with an individual connected to it through an edge. The resulting offspring may only replace the least fit of its parents, only if the offspring is more fit. This setup is more biologically plausible than the usual "panmictic" populations in which any individual may mate with any other. It prevents premature convergence since a particular genome can spread only to immediate neighbors in a single timestep [Kephart, 1994].

The implementation of a spatial population described above also incorporates the more general speciation strategy called preselection, which stipulates that a newly created individual in a population may only replace one of its parents. This protects against premature convergence because it ensures that at least some of an individual's genetic material will survive into the next generation [Goldberg, 1989].

Fitness sharing is a technique that penalizes genomes that inhabit neighborhoods of many other genomes. Generally, an individual's fitness evaluation is divided by a sharing factor that measures the genome's proximity to others in the population. Genomes in heavily populated peaks receive a high penalty, which translates into a lower probability of propagating to the next generation. This technique is intended to spread the population across several peaks in the solution space, with larger (wider or higher) peaks able to support more individuals. Implicit sharing is a variation in which, for each input, only the individual with the best response from a randomly-selected subset of the population is awarded fitness for that input [Darwen and Yao, 1996].

Crowding is a generalization of preselection, where an individual only replaces a genome to which it is similar (but not necessarily the parent). Under crowding, after a new genome is created, a subset of genomes is randomly selected from the population. The genome in the subset which is closest to the new genome is chosen to be replaced by the new genome [Goldberg, 1989].

The confidence method was tested in conjuction of each of the above speciation methods. Interestingly, their performance was found to differ a lot in the handwritten character recognition domain, which will be described next.

4 Experiments

The method of confidence evolution was applied to the standard benchmark task of recognizing handwritten digits. There are many methods developed specifically for this task. The present goal is not to compete with them, but rather to test the viability of the method and to refine it further. This task is well-suited for

such analysis because the correct decisions are readily available. After the method has been tested and refined, it will be possible to apply it to other task where the correct performance is not known.

The data set used was the freely-available subset of 2,992 examples of handwritten digits 0..9 in the NIST database, scaled to an accuracy of 8×8 pixels [Choe et al., 1996]. The networks had an input layer of 64 units to encode the 8×8 input representing a digit to be classified. The input layer was fully connected to a hidden layer of 20 units, which was fully connected to an output layer of 11 units, representing each of the ten possible digit classifications, and an additional unit to output the confidence in this classification. The output unit with the highest activation was chosen as the classification. The genome represented the real values of the weights and biases of the network. While the canonical GA operates only on binary strings, analogous operations of crossover and mutation were implemented for the real-valued genome: a uniform crossover could take place between weights, and each weight was mutated with a 0.01 probability by adding a normally distributed random value of 0 mean and 1.0 standard deviation to the weight.

The genetic algorithm proceeded on a population of 100 individuals for 5,000 generations. During each generation, the fitness for each network was calculated according to equation 2 (and equation 1 for those experiments where confidence was not used) on a training set of 2,000 patterns, selected randomly among the 2,992 for each experiment. Fitness scaling was done by sigma truncation scaling [Goldberg, 1989], which tolerates negative fitness values. Selection was fitness-proportionate. Throughout evolution, each population was tested on a randomly-chosen test set of 200 patterns not part of the training set.

Notice that a more accurate fitness evaluation could easily be designed, such as the sum of squared errors between the outputs and the target output. Such an evaluation would capture more information about the differences between individuals; however, Boolean values were chosen in order to emulate less well-understood decision tasks for which such detailed information is not available. Furthermore, the fitness evaluation seeks to reward only the desired outcome (the correct classification in the winner-takes-all sense), not any specific way of attaining the outcome. Having such an open-ended fitness evaluation allows the network to implement its own, possibly unexpected, method for achieving the result [Floreano and Urzelai, 2000].

The different speciation techniques outlined in Section 3 were each tested as part of the confidence evolution method. The island method was implemented by splitting the 100 genomes into 10 noninteracting islands of size 10, with no migration. A spatial population was laid out on a 10×10 grid with edges folded back to create a torus; an individual could mate only with one of its four neighbors, with the offspring replacing the parent with lower fitness, only if the offspring's fitness was higher. Fitness sharing was implemented by penalizing genomes according to Euclidean proximity to others in the population. Crowding was implemented such that a new individual was placed in the population in place of the closest Euclidean neighbor in a random subset of 10 from the population. In preselection, an offspring replaced the parent with the lowest fitness if the offspring was more fit than that parent.

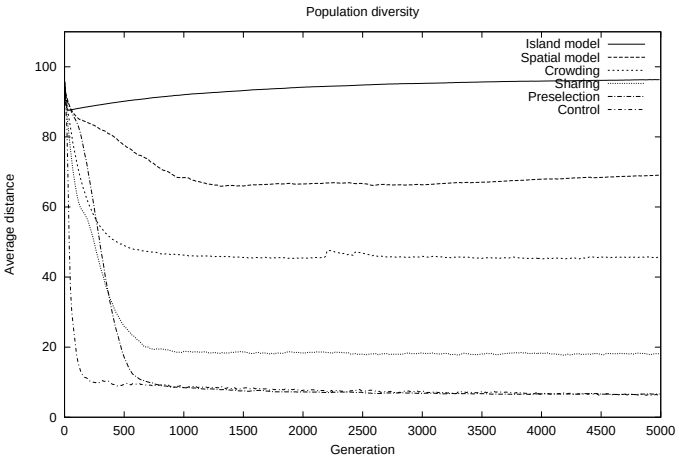

Figure 2: **Population diversity under different speciation methods.** The plots (from top to bottom) show the maximum, average and minimum Euclidean distances between genomes in the population as evolution progresses. Plots are each averaged over 10 independent evolutions.

5 Results

To varying degrees, each speciation method was able to maintain diversity in the evolution. For each of these methods, the average Euclidean distance between the 100 genomes in the population throughout an evolution without confidence is plotted in figure 2. The island model and the spatial model, i.e. those methods that directly restrict mating to a local neighborhood, resulted in particularly good genomic diversity. A control evolution with no speciation is shown as well. The

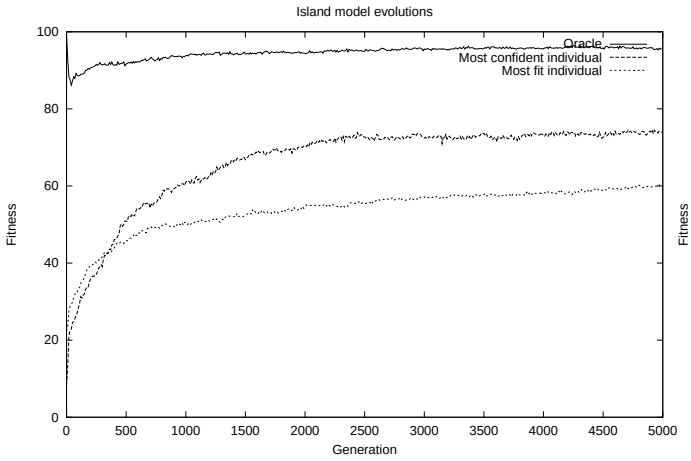

Figure 3: **Accuracy of confidence evolution with islands.** On top, the fitness obtained by choosing (through an unrealizable oracle) the best answer in the entire population is plotted. In the middle, the fitness obtained by choosing the answer of the most confident individual is shown. The fitness of the most fit individual is plotted in the bottom. The plots are averages over 10 runs.

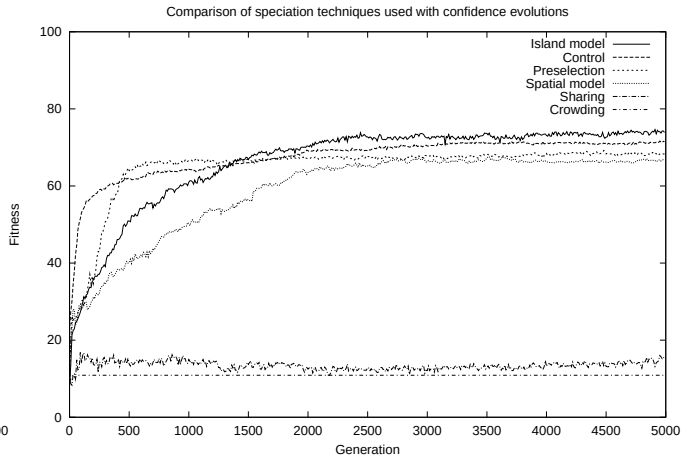

Figure 4: **Accuracy of confidence evolution with different speciation methods vs. standard evolution.** The island method performs the best; the other methods are weaker than the standard evolution, labeled "Control", in the order shown in the legend. The plots are averages over 10 runs.

control evolution quickly lost much genomic diversity, bottoming out at approximately generation 200. This result highlights the need for speciation.

Evolutions with crowding, sharing, and preselection each slowed convergence compared to the evolutions with no speciation, but were considerably less effective. These techniques do not restrict replacement as strongly as the island and the spatial models do; they involve a high degree of randomization in the choice of which population member a new genome should replace. Since this decision is randomized, sampling error can affect replacement, causing genetic drift. Mahfoud (1992) cites this stochastic error as a difficulty with crowding and preselection.

As expected, those speciation methods that maintained the highest diversity also provided the best advantage for confidence evolution. Populations evolved using the island model and the spatial model were diverse enough so that choosing the answer of the most confident individual resulted in better performance than could be obtained from the population's champion. Figure 3 compares the accuracy in the test set of the (unattainable) best answers, answers selected by confidence, and the most fit individual for the island model.

The accuracy of the different methods in the test set is compared in figure 4. The main result is that confidence evolution with islands resulted in a slightly higher level of performance than the control evolution. The other methods were worse, underlining the importance of an effective speciation method in confidence evolution. Crowding, in particular, did not perform above chance; apparently one or more high-bidding and wrong individuals persisted thoughout the evolution, never being replaced because they were too far from the other genomes. This result demonstrates the difference between random diversity (such as the population before the first generation), and useful diversity (niches of high-performing but different individuals). Only the latter kind of diversity is useful for confidence evolution.

Interestingly, when speciation methods were added to the standard evolution, the performance either was not affected or actually became worse. This result suggests that making full use of speciation requires a technique such as confidence. It also shows that the observed improvement over standard evolution is indeed due to confidence, and not speciation.

6 Discussion and Future Work

The experimental results in this paper show that confidence evolution can improve performance of neuroevolution in the handwritten digit recognition task. They also show that effective speciation is crucial for this

technique. How general are these results?

Speciation is generally used in a genetic algorithm to increase the overall rate of learning by slowing down population convergence. However, when speciation is used in conjunction with confidence evolution, the goal is instead to increase the variation in answers made. The existing speciation techniques used in this paper may not be the best for this new, slightly different goal. This issue is underscored by the fact that confidence provided the greatest advantage in evolutions with completely noninteracting subpopulations – a technique that would tend to harm overall fitness since small populations are being evolved in each island, leading to cruder solutions.

In the extreme, a speciation technique that even significantly decreases the overall fitness of the population would work well with confidence evolution if it maintains a large variety of correct answers. This is a tradeoff that is not acceptable in a standard genetic algorithm: if the goal of speciation is to increase the overall rate of learning, a technique that lowered the fitness of all individuals significantly would not be useful. But given this new set of criteria, perhaps such strong speciation techniques could be devised in the future, gaining even more benefit from confidence evolution than is possible with the current techniques [Ackley, 1987, Katila, 1987].

It is also possible that domains other than character recognition might be more amenable to the current speciation techniques. This domain benefits from a large number of individuals fine-tuning an approximate solution in a small space, which requires exactly the convergence that speciation attempts to avoid. Instead, genetic algorithms in general and speciation methods in particular are strongest at quickly finding approximate solutions. Therefore, problems that involve more global search may be more amenable to the current techniques.

In particular, genetic algorithms are the method of choice for sequential decision tasks where the correct answers are not known and the feedback is highly sporadic [Moriarty and Miikkulainen, 1997, Moriarty et al., 1999]. Given the promising results in the handwritten character recognition domain, confidence evolution should work well in such tasks. For example, imagine applying confidence to the training of a robotic controller. Each neural controller in the population would be presented with an encoding of the robot's sensory input, and it would output a motor action and a confidence level. The action recommended by the most highly confident controller would be selected. After several decisions were made, a fitness evaluation for the whole sequence of decisions would be obtained. This fitness would then be distributed to the controllers according to how confident they were of their outputs and how often they were selected.

Although at first it seems that such fitness information might be too noisy, the situation is very similar to those of SANE and ESP neuroevolution methods described in [Moriarty and Miikkulainen, 1997], [Moriarty, 1997], [Gomez and Miikkulainen, 1997], and [Gomez and Miikkulainen, 1999], where populations of neurons are evolved to form good neural networks. Each neuron receives a fitness based on how well the whole neural network performed in the task: in effect, the neurons are evolved to speciate into useful subtask that work well together. In reinforcement learning tasks with confidence evolution, similarly each network is rewarded based on how well the whole population did in the sequence of decisions. Given how powerful the SANE and ESP methods are, this same approach may also work well in confidence evolution.

In other learning tasks, it may be useful for an agent to express its confidence in a more direct form, by answering a more specific question about its performance. For example, a neural robotic controller might estimate the amount of time needed to reach a goal state, rather than estimating its probability of success. Such fitness functions might lead to more powerful evolution. Similarly, instead of always selecting the most confident individual's answer, the answer might be constructed by combining answers of the most fit individuals, or by at least not considering the answers of those with the lowest fitness. This method would for example solve the problem that occurred with crowding in the above experiments.

Different techniques of training individuals to output confidence could also be considered. Evolving networks to provide answers and confidence estimates is clearly a more difficult task than simply evolving networks to provide answers. Might this increased difficulty be offset by using a combination of evolution and learning, or by Lamarkian evolution? Might it be beneficial to evolve the confidence neuron or network in a separate phase of evolution? Or perhaps as a separate network entirely? These are some of the issues that will be explored in future work.

7 Conclusion

This paper shows how the knowledge and expertise encoded by the entire evolved population can be utilized to obtain a high level of performance. High-quality decisions may be extracted from the population if a

fitness evaluation rewards individuals that accurately output estimates of the quality of their decisions. To use this technique, a diverse population, capable of producing many different correct answers, is needed. This research motivates the development of techniques to ensure a high level of diversity throughout evolution, possibly even at the expense of overall fitness. The technique of confidence may have broad applicability in the domain of reinforcement learning tasks, which is the main direction of future work.

References

[Ackley, 1987] Ackley, D. (1987). A Connectionist Machine for Genetic Hillclimbing. Boston: Kluwer.

[Choe et al., 1996] Choe, Y., Sirosh, J., and Miikkulainen, R. (1996). Laterally interconnected self-organizing maps in hand-written digit recognition. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems 8*, pages 736–742. Cambridge, MA: MIT Press.

[Darwen and Yao, 1996] Darwen, P. and Yao, X. (1996). Every niching method has its niche: Fitness sharing and implicit sharing compared. *Parallel Problem Solving from Nature*, pages pp. 398–407.

[Floreano and Urzelai, 2000] Floreano, D. and Urzelai, J. (2000). Evolutionary robots with on-line self-organization and behavioral fitness. *Neural Networks*, 13:431–443.

[Goldberg, 1989] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

[Gomez and Miikkulainen, 1997] Gomez, F. and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342.

[Gomez and Miikkulainen, 1999] Gomez, F. and Miikkulainen, R. (1999). Solving non-Markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, Denver, CO. Morgan Kaufmann.

[Jacobs et al., 1991] Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

[Katila, 1987] Katila, R. (2000). In Search of Innovation: Search Determinants of New Product Introductions. Doctoral Dissertation, the University of Texas at Austin.

[Kephart, 1994] Kephart, J. O. (1994). How topology affects population dynamics. In Langton, C. G., editor, *Artificial Life III*, volume XVII. Addison-Wesley, Reading, MA.

[Mahfoud, 1992] Mahfoud, S. (1992). Crowding and preselection revisited. *Parallel Problem Solving from Nature*, 2:27–36.

[Moriarty, 1997] Moriarty, D. E. (1997). *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis, Department of Computer Sciences, The University of Texas at Austin. Technical Report UT-AI97-257.

[Moriarty and Miikkulainen, 1997] Moriarty, D. E. and Miikkulainen, R. (1997). Forming neural networks through efficient and adaptive co-evolution. *Evolutionary Computation*, 5:373–399.

[Moriarty et al., 1999] Moriarty, D. E., Schultz, A. C., and Grefenstette, J. J. (1999). Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research*, 11:199–229.

[Mühlenbein, 1991] Mühlenbein, H. (1991). Evolution in time and space - the parallel genetic algorithm. In Rawlins, G., editor, *Foundations of Genetic Algorithms*. Morgan-Kaufmann.