To appear in Proc. 7th Intl. Conf. on Genetic Algorithms (ICGA97), Morgan Kaufmann, San Francisco, CA.

Culling and Teaching in Neuro-evolution

Paul McQuesten and Risto Miikkulainen
Department of Computer Science
The University of Texas at Austin
Austin, TX 78712
{paulmcq, risto}@cs.utexas.edu

Abstract

The evolving population of neural nets contains information not only in terms of genes, but also in the collection of behaviors of the population members. Such information can be thought of as a kind of "culture" of the population. Two ways of exploiting that culture are explored in this paper: (1) Culling overlarge litters: Generate a large number of offspring with different crossovers, quickly evaluate them by comparing their performance to the population, and throw away those that appear poor. (2) Teaching: Use backpropagation to train offspring toward the performance of the population. Both techniques result in faster, more effective neuro-evolution, and they can be effectively combined, as is demonstrated on the inverted pendulum problem. Additional methods of cultural exploitation are possible and will be studied in future work. These results suggest that cultural exploitation is a powerful idea that allows leveraging several aspects of the genetic algorithm.

1. Introduction

In natural learning systems not all abilities are congenital. Instinctual behaviors are insufficient for all but the simplest natural organisms existing in static environments. Successful organisms are adaptive: their behaviors are not completely determined by their genome. In their struggle for survival they have available not only their genetic endowment, but also skills learnt since birth. In many cases successful adults have been taught skills by their parents. For example, many species of songbirds must learn their songs from their elders; human language is culturally transmitted. The reason is that culture can adapt to environmental change much more rapidly than a congenital behavior could. Adaptation during life also influences the course of evolution, via the Baldwin Effect. Could such effects be utilized in computational evolution?

The population in a genetic algorithm can be viewed as a repository of imperfect knowledge about the problem. While a standard GA exploits only fitness scores, evolving neural nets can also supply an estimate of the correct response in any situation. Many previous researchers have studied the combination of learning and evolution, but usually with either no teacher or an omniscient teacher. This work constructs a teacher from the common knowledge contained in the current population.

This work is aimed at sparse reinforcement learning problems where feedback is expensive or rarely available. It seeks to reduce the number of full fitness evaluations required to solve a problem. Like all genetic algorithms, the *only* information from the environment comes from the evaluation function. From a computational viewpoint, expensive fitness evaluations are just as bad as sparse reinforcement. Indeed, in a control problem where fitness can only be determined by a long simulation, sparseness and expense are identical.

Phenotypes in this paper are feed-forward neural networks. The output of a neural net anywhere in its input space can be computed without being charged for a fitness evaluation. Backpropagation can be used to train nets, again without being charged.

In this neuro-evolution approach two exploitations of the cultural knowledge in the population are devised: (1) Culling: operational similarity to elders is used to cull oversize litters. (2) Teaching: new organisms are taught to respond somewhat like an elder before they have to face a fitness evaluation.

The following two sections motivate and describe the culling and teaching algorithms. In section 4 the methods are tested experimentally on the pole-balancing task. Section 5 analyzes the relationship of learning and evolution, and outlines directions for further exploitation of culture.

2. Culling

From observation of nature comes the idea of overproduction of offspring: producing litters whose size is beyond the carrying capacity of the environment. Hundreds of turtles hatch for each one which survives to reproduce. Culling in nature can be done by the environment. In neuro-evolution we'll have to settle for something less complicated.

The following subsections develop our culling mechanism by attacking the dismal distribution of relative fitness from naïve crossover. Section 2.1 examines the distribution obtained by a standard GA. Section 2.2 shows how it can be improved by a magic, costless fitness estimator (a perfect oracle). Section 2.3 shows how to construct a less-than-perfect but implementable estimator from the behavior of the current population. (Section 4.2 shows the method to be practical and effective.)

2.1 Crossover distribution

Crossover has special hazards for evolving neural networks. With typical encodings, the same network can be represented in many ways, since changing the order of hidden units will not affect the output. That is, there are many different genotypes that specify indistinguishable phenotypes. When phenotypically similar individuals are chosen to mate, but their genotypes use a different permutation of hidden units, the result of crossover is most likely to be nonsense. Whitley, Dominic and Das (1991) describe these "disastrous crossovers" resulting from the symmetries in neural net representations. They use very small populations and high mutation rates to force the population to quickly settle on a single one of the equivalent permutations. Nolfi, Elman and Parisi (1994) eliminate crossover entirely, claiming that the problem must be better understood in order to devise appropriate encodings.

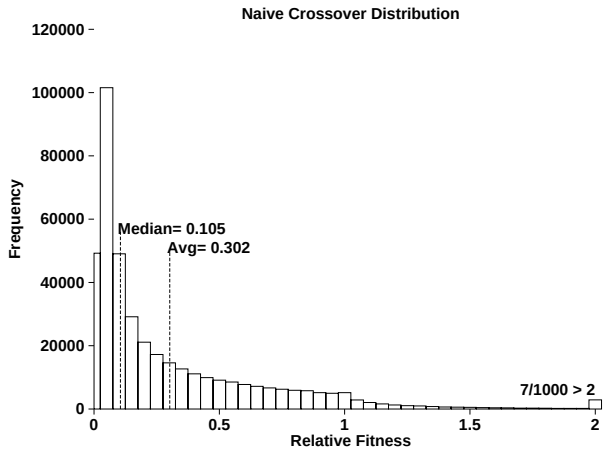

Figure 1: Fitness distribution of naïve crossover in 100 runs, 53 successful; over 400,000 individuals. The average offspring is only 30% as fit as its parents.

We believe that learning is unlikely to be competitive in simpler problem domains where mutation suffices, so crossover is the primary operator in this work. Pair-wise mating is the essential feature which distinguishes genetic algorithms from population-based hillclimbers (Eshelman and Shaffer 1995). The challenge is to discover how to properly utilize crossover.

To gain insight into crossover's operation, let us inspect the distribution with a standard GA. The task is the well-known control problem of the inverted pendulum. Fitness evaluations in this problem are very expensive, requiring thousands of simulation cycles. The equations for the simulated system are taken from (Whitley, Dominic and Das 1991), and are reproduced in Appendix A for convenience. The fitness function is the total number of simulator steps until the pole exceeds ±12°, taken over all eleven initial conditions from (Whitley, Gruau and Pyeatt 1995).

A pole-balance attempt is considered successful if the pole remains within limits for 1000 time steps, so a run is successful if a network with a score of 11,000 is found. Like many researchers, this work uses a stagnation criteria: A run is terminated as unsuccessful if there has not been at least 1% improvement since eight generations ago. (Further detail of the experimental setup is given in Section 4. The results shown in Figs 1 & 2 appear in this section in order to motivate and explain the culling attack.)

Figure 1 shows a histogram of relative fitness from all crossovers performed during 100 runs of the standard GA on the pole-balancing problem. The abscissa is the ratio of offspring fitness to average parental fitness: an individual crossed with itself would get a ratio of one. The absolute scale of the ordinate axis is immaterial. The shape is important: it motivates this attack. The distribution is multimodal and largely deleterious. The notation "Avg=0.302" in the figure indicates that the average offspring is only 30% as fit as its parents. Most crossovers are indeed awful, as observed by Nordin and Banzhaf (1995). The distribution is dominated by an enormous peak near zero fitness, with a lesser lump near the average of the parents. This distribution is intuitively plausible: in any interesting domain, there are simply many more ways to do things wrong than right.

In the 100 runs contained in Figure 1 over 400,000 individuals were generated. 53% of the runs succeeded. The average successful run required about 4000 fitness evaluations. Fitness of offspring was below 5% of average of parental fitness one hundred times as often as it was greater than 100%.

However, a few crossovers are outstanding. The notation at the right of Figure 1 shows that roughly seven of each one thousand offspring had fitness exceeding twice their parental average. Perhaps genetic algorithms work by simply waiting for these exceptional events?

This observation suggests a method of making crossover more productive. The effective crossover distribution can be improved by suppressing the lower lobe, which should raise the average, and the increased expectation should accelerate the GA. This is the main idea of culling.

2.2 A perfect oracle

To determine how effective culling could be in principle, a perfect oracle is employed. This "estimator" of the fitness function is a full fitness evaluation which is not charged for.

The computer implementation is straight-forward: to produce one new offspring for the population, first produce a litter of, say, eight. Grade each by consulting the oracle. Discard all but the one estimated as best, which will in fact be the best, since this oracle is perfect. The resulting distribution is shown in Figure 2.

Figure 2 represents 30 runs of culling with a perfect oracle, where 30,000 individuals were placed in the population out of 264,000 crossovers performed. All runs succeeded on the pole-balancing task, on average with 1300 chargeable evaluations. The crossover distribution is much better than in the standard GA. The average offspring is now 62% as fit as its parents (up from 30% in Figure 1). Over 3% are twice as good as the parental average (up from 0.7% in Figure 1).

Table 1 shows that overproduction with elite selection ("culling with a perfect oracle") reduces the number of generations dramatically. Of course such a perfect oracle is not usable in practice, since each offspring was actually evaluated by a full fitness test. The important question is can culling improve efficiency when it is based on a less than perfect oracle?

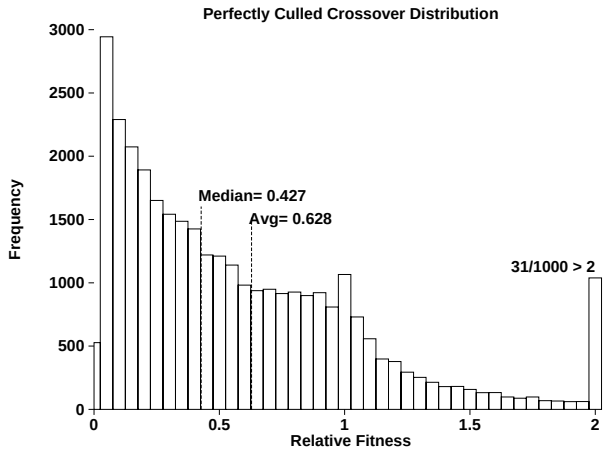

Figure 2: Culled crossover distribution: best of 8 by perfect oracle. Average is double naïve crossover.

2.3 A realizable oracle

As seen above, culling works with perfect information, but can it work without full fitness evaluations? The low performance peak is so dominant that all that is needed to make a significant change is to recognize abysmal individuals with reasonable probability. For instance, poor neural nets could be identified as those with no connections to their outputs or inputs. A more thorough evaluation can be done with the performance of the individuals in the population as a guide. The method chosen is to "quiz" the new offspring, and grade them with respect to the current population's knowledge of the answers. The theory is if one can answer simple questions as well as an above-average performer, one is probably better than the vast majority of potential offspring. For neural networks, administering the quiz is simple: just set the input vector per the "question," evaluate the network, and report the grade as the distance from the benchmark's output vector.

Since the benchmark is drawn from the current population, a true genius will (inadvertently) be penalized. So, if over-applied this technique would result in continual mediocrity. The low lobe of the crossover distribution, however, is so big that this danger does not arise in practice.

There are several ways of choosing the benchmark individual (see Discussion.) In the experiments reported in this paper a parent is the benchmark. Parents are most often above average due to reproductive selection in the GA. Further, parents and offspring are more likely to share similar architecture than unrelated individuals. Similarity of architecture increases the likelihood that similar responses are due to competence.

There are also many ways to determine the questions included in the culling "quiz." This initial investigation uses the most elementary method: questions are chosen randomly. Each element of the input vector is set to a value chosen uniformly from the interval [0.45, 0.55]. The notion behind the small interval is that this is only a qualifying exam. Neither the imperfect elder nor the child is likely to know how to answer sensibly in extreme circumstances.

3. Learn from elders

This section shows another way to exploit the improving behavior of the population. The benchmark and quiz of culling are metamorphosed into teacher and syllabus.

3.1 The Baldwin Effect

In 1896 the biologist Baldwin described how acquired abilities influence the course of biological, Darwinian, evolution. Hinton and Nolan (1987) first demonstrated the Baldwin Effect in an abstract genetic algorithm, thus confirming that the biological Baldwin Effect also occurs in evolutionary computation. In a more elaborate model, Nolfi, Elman & Parisi (1994) showed how training neural nets influenced their evolution.

Further studies of the Baldwin Effect (French & Messinger 1996; Hightower, et. al. 1995) have confirmed a U-shaped curve for inherited vs. acquired characteristics: Abilities which are either very simple or very crucial tend to be inherited while those in the middle remain plastic. This is true both in natural biology and in computational evolution. The advantage of plastic abilities is that they can adapt to environmental change in less than evolutionary time. Further, and important for learning, the genome does not have to be long enough to include all the details of a learned trait.

Many researchers have found learning to combine well with evolution. For example, (Braun and Zagorski 1994; Sebag and Schoenauer 1994; Ackley and Littman 1991). In sparse reinforcement tasks, however, in general, no targets are available for training.

Nolfi, Elman & Parisi (1994) harnessed the Baldwin Effect in a sparse reinforcement domain by adding a task to their neural nets which was related to, but distinct from, the primary performance task. Their secondary task was simple, ingenious and general: to predict the inputs that would occur in the next time step. Clearly, the ability to predict future inputs should be positively correlated with performance in almost any task. They found that the hidden-unit representations that developed to support the secondary task were being used to advantage by the primary task. Note that such a secondary task is always amenable to training: the actual inputs from the next time step provide the necessary error signal.

3.2 Teaching by elders

Our learning technique differs from all of these by extracting training targets from previous population members. The motivation comes from the natural world, where parental training of offspring is ubiquitous.

To emulate parental training in neuro-evolution is straightforward: Train the offspring with backpropagation to emulate the parents' output in a set of training examples, before the offspring is subjected to the complete, expensive fitness evaluation. The set of training examples is called the syllabus. As with culling, the teaching syllabus and the teacher could be chosen in various ways, and only the most straightforward choices are analyzed in this paper. (See the Discussion section for alternatives.) As mentioned in 2.3, parents are a good choice for teacher because they are relatively highly fit, and usually share structure with their offspring. The set of training examples consists of random input vectors (a new random number is produced each time). The error signal is Euclidean distance between the offspring and parental output vectors for a given input vector. In culling, the errors between several candidate offspring are compared; in teaching, the error is used to adjust weights of each offspring.

Excessive training could lead to a student becoming a near copy of the teacher. Clearly, this would stall any hope of progress via evolution, so the amount of training must be moderate. Thus the method is restricted to only incremental improvement over an untrained individual. We apply only a limited amount of backpropagation (amounts shown in Fig. 4), and do not iterate over the syllabus. Each training case is presented and backpropagated only once. (In many neural network studies a reported backprop epoch often means a presentation of *each* training case. In contrast, here there are only a fixed number of cases, and each is presented only once. There are no "epochs.")

As will be seen, learning from elders evolves offspring that are trainable, that is, when they are trained for a few steps toward a parent, they will perform well.

4. Empirical Evaluation

The culling and teaching techniques were tested on the pole-balancing problem. Table 1 reports the number of fitness evaluations required to find a solution for various methods. Note that while unsuccessful runs are reported, they are not used in calculating the mean, so methods with high success rates are actually better than the reported average indicates.

4.1 Experimental setup

All populations consist of 200 individuals. The initial 200 individuals are randomly generated. A "steady state" GA is used: each child is inserted into the population as it is generated, displacing the least fit individual found in four random probes (a 4-tournament). The population is not sorted. Mating parents are chosen by a 2-tournament. For reporting purposes a new "generation" is declared every 200^{th} individual.

The current implementation does one-point crossover 2/3 of the time, and two-point crossover 1/3 of the time, as recommended by Masters (1993). This results in a constant probability of disruption of schemata, independent of defining length, just like uniform crossover. A small amount of mutation is included: 0.1% probability of each bit flipping. The learning rate for

Method	Mean	S.D.	runs	Suc
Standard GA	4026	1430	100	53%
RBC	1523	1624	24	42%
Perfect Culling	1300	451	30	100
Culling	2105	722	20	85%
Teaching	1653	643	50	98%
Teach + cull	960	316	50	100

Table 1: Pole balancing performance. Mean is the average number of evaluations for successful runs, SD stands for rounded standard deviation, and Success indicates the percentage of runs which were successful.

backpropagation is 0.15, with no momentum. Unless otherwise noted, the length of the syllabus was 10. Each question is generated by a new random throw.

As a baseline comparison (and sanity check) Random Bit hill-Climbing, RBC, per Davis (1991) was implemented and tested. RBC is a form of next-ascent which flips bits in a random order. RBC does not use the "patience" test; it stops only at the top of a hill—when no single bit flip will improve the current score. In this problem RBC beats standard GA, suggesting that the error surface is quite smooth. However, RBC only finds a solution 80% as often as the standard GA does.

Table 1 summarizes the results. The standard GA and Perfect Oracle were described in Section 2. The following sections analyze the results for culling, teaching, and their combination.

4.2 Culling results

The main result is that culling by parental similarity takes less than 55% of the evaluations needed by the standard GA, and succeeds 28% more often. Culling with a parent achieves 85% of the success of perfect culling, but without the eight-fold increase in actual fitness evaluations.

Figure 3 shows the performance of culling with various litter sizes. The leftmost point (+) represents the standard GA. A litter size of one performs just like the standard GA, as it should. Larger litter sizes give better performance, but the improvement is very small after about 8. This is a nice result because is suggests litter size does not need to be optimized carefully; anything from 8 up will perform well.

A series of runs not shown investigated the length of the questionnaire. Over the range of four to sixteen questions performance was about the same.

4.3 Teaching results

Teaching by parents turned out to be even more effective than culling (Table 1): now 98% of the runs are

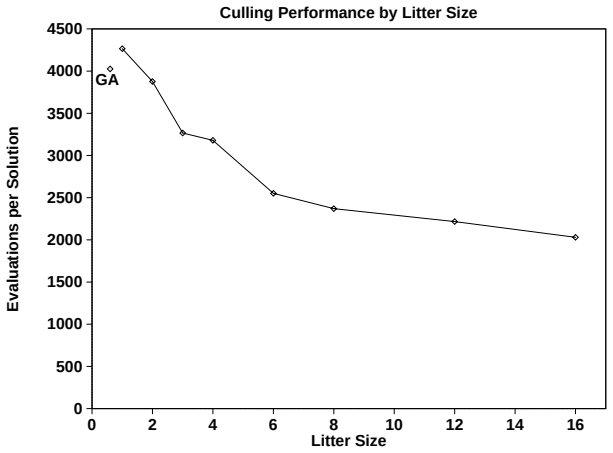

Figure 3: Effect of litter size on culling. Mean evaluations per solution, 20-50 runs per data point. Culling from a litter of 8 or more works well.

successful, and only 41% of the evaluations used by standard GA are needed.

These results were obtained with a fixed-sized syllabus of 20 vectors. Since backpropagation has a larger effect with more training vectors, the size of the syllabus is an important parameter, similar to litter size for culling.

Figure 4 shows how the average evaluations per solution varies with the length of the syllabus. Each data point is the average of 20–50 runs. Success rates were 98% or better when the syllabus contained 20 or more questions. This graph indicates that the size of the syllabus does not have to be very carefully set, as long as it is large enough (around 20).

The state of the network after teaching corresponds to the phenotype in biology: phenotypic fitness determines reproductive success, and is what the GA responds to. However the distribution of pre-training, "natal," fitness is interesting. It turns out that almost everyone is born with

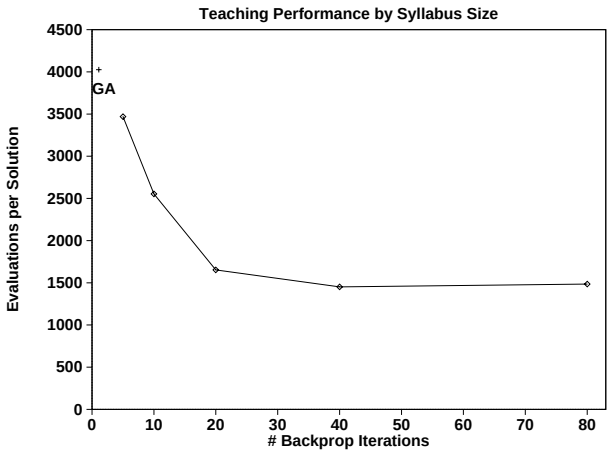

Figure 4: Effect of training time on performance. Most of the benefit is achieved with 20 questions.

poor fitness. Even the winners (those that achieve a score of 11,000) behave poorly before training, almost as poorly as random nets. The teaching GA does not evolve solutions directly, but rather evolves networks that will respond to the training regimen. Consider networks to be a point in weight space, and call the "solution region" any point representing a successful network. The genomes that win with teaching are not located in the solution region, but rather at some point which will be pulled to a solution region by an imperfect teacher. The performance increase of teaching over the standard GA (Table 1) indicates that the area of the set of points representing winning genomes is larger than the solution region.

These results could be seen as a mere affirmation that backpropagation is sensitive to initial conditions. Contrariwise, perhaps it succeeds by evolving extremely good teachers, that is, nets which can teach an arbitrary net to succeed. This possibility is tested by subjecting 10,000 randomly-generated (non-evolved) networks to teaching by selected winners, and by the teachers of those winners. No successes were found, indicating that "teachability" requires some specific attributes of the initial weights. The GA can find those combinations: evolution and learning are truly working in harmony.

4.4 Combining culling and teaching

Despite motivating both these techniques as "cultural," they are based on different mechanisms. Culling improves the distribution from which crossovers are selected. Teaching modifies the feedback provided to the genetic algorithm via the Baldwin Effect. Therefore it might be possible to combine them into an even stronger learning algorithm.

In the combination test, for each new offspring desired, eight are generated and trained with backprop on 20 test cases. The one with the lowest training error is chosen to enter the population. These training time and litter size parameters were found to be good in the previous sections.

As can be seen from Table 1, culling and teaching work well together. The combined technique achieves a 100% success rate with 25% of the evaluations used by the standard GA, and also has the smallest variance.

This work is aimed at problem domains where the evaluation function is expensive, therefore the primary criteria is number of evaluations, but it is still important to check how much overhead the culling and teaching methods impose. For the set of results in Table 1, including the time in unsuccessful runs, the standard GA took an average of 380 CPU seconds to produce each solution (200MHz Pentium-Pro), and the Perfect Oracle was roughly the same, when the CPU time for all the uncharged evaluations is included. RBC was slower than the standard GA, but only by a factor of 1.9, again indicating that the pole balancing problem is not very difficult. Culling by itself took 2.1 times as long per solution as the standard GA, so it would not be competitive unless the fitness function was more expensive. Teaching was the fastest at only one-tenth the time of the standard GA. Teaching and culling combined took one-sixth the time. These ratios would improve with a more expensive fitness function, thus these new methods are efficient and practical.

These results show that information stored implicitly in the behaviors of the population can be used effectively to enhance several different aspects of neuro-evolution.

5. Discussion and future work

Nolfi and Parisi (1994, 1995) studied neuro-evolution in rapidly changing environments. Networks that learn adapt to the current environment and therefore score higher. The teaching inputs are computed by the network itself. The teaching part of the network is evolved normally and does not change its weights during life. Nolfi and Parisi found that the targets that the teaching part evolves to generate do not specify the optimal output. Instead, they cause the network to learn more efficiently. We believe that this result translates directly into the importance of a good syllabus in our methods.

There is an interesting similarity between our teaching results and those of Nolfi and Parisi. Our networks do not evolve genes that are good for the task: before training the networks perform very poorly. A performance histogram of winners looks just like one of random networks. Unlike random nets, however, winning nets are trainable in context. This resembles the Nolfi and Parisi result that the nets do not evolve optimal weights, but weights that teach well. This principle appears fundamental to successful combination of learning and evolution, whether based on self-teaching or learning from a teacher.

How should Teachers be chosen or composed? The reported method is based on functional similarity to a single elder. The benchmark/teacher individual could be chosen in other ways For instance, as the current population champion, or even by averaging the responses of a committee. In preliminary experiments teaching by parent was found more effective than teaching by the current best individual in the population, and so therefore was reported in this paper. It appears that an unrelated teacher has a higher chance of having an architecture that is too different to give reliable estimates. A committee of the several most fit might be less parochial. Perhaps a new offspring (student) should be allowed to try several teachers and continue only with the higher scoring one(s).

Alternatively, perhaps a separate population of teaching specialists could be co-evolved. These would be evaluated according to the fitnesses of their pupils.

It is well known in both symbolic and connectionist machine learning that results depend crucially on the quality of the training corpus (syllabus, in our terms). An active area of our current research is how to build a better syllabus. For example, by adding memory to the phenotype, a teacher could remember some characteristics of actual problem instances encountered during its own evaluation. A teacher could remember "significant problems," or remember the range of most frequent input parameters. Perhaps the capability for such significance analysis could be evolvable, along the lines of an Adaptive Critic (Barto, Sutton and Anderson 1983).

Several researchers have found it useful to change the syllabus over time, as in the Incremental Learning of Elman (1991) and (Gomez and Miikkulainen 1997), but that requires modifying the evaluation function. A similar effect should be possible in these "cultural" methods by varying the length and/or content of the syllabus, *without* any access to the insides of the evaluation function.

In addition to culling and teaching, population culture could perhaps be utilized in other ways. For example, when a fitness function has severable components, each evaluation can be reported as a vector, giving fitness for each component separately. In the pole-balancing problem the time-to-fall on each of the eleven initial conditions can be returned. Each element is then considered to be a sub-goal, and the vectors are used to control mating. The first parent is selected by overall performance, as usual. For the second parent, however, an individual is found that solves most of the cases that the first one misses. This idea can be dubbed the *Jack Sprat* method of spousal selection. In preliminary experiments it leads to more efficient evolution, apparently by enhancing the odds of combining solved sub-problems.

There may be other forms of utilizing "culture" in the population as well. The results reported in this paper form a promising starting point.

6. Conclusion

This paper demonstrates that the current state of the population contains useful information (beyond its fitness scores) that can be tapped in a manner reminiscent of cultural transmission of skills. Two such novel methods were evaluated in this paper, and good values were found for the main parameters. Culling of oversize litters used only 55% of the evaluations required by a standard GA and succeeds 28% more often. Learning to respond somewhat like an elder used only 41% and succeeds 98% of the time. Since these techniques are operating on disjoint aspects of the GA, they can be combined. The combination achieves a 100% success rate with only 25% of the evaluations as the standard, and produces solutions six times as fast.

Other aspects of neuro-evolution could be handled by such methods as well, as will be explored in future work.

Acknowledgments

This research was supported in part by NSF under grant #IRI-9504317. Our thanks to the anonymous reviewers, whose comments improved the presentation.

References

Ackley, David, and Michael Littman (1991). Interactions between Learning and Evolution. In Langton, C.G., C. Taylor, J.D.Farmer and S. Rasmussen, eds., *Artificial Life II*. Addison-Wesley, Redwood City, CA.

Barto, A.G., Richard S. Sutton, and Charles W. Anderson (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13.

Braun, Heinrich and Peter Zagorski (1994). ENZO-M - A Hybrid Approach for Optimizing Neural Networks by Evolution and Learning. In Y. Davidor, H-P.Schwefel, and R.Manner, eds. *Parallel Problem Solving from Nature: PPSN-III*. Springer-Verlag, Berlin.

Davis, L.D. (1991). Bit-climbing, representational bias, and test suite design. In Belew, R.K. and L.B. Booker, eds, *Proceedings of the Fourth International Conference on Genetic Algorithms*. Morgan Kaufman, San Francisco.

Elman, Jeffrey L. (1991). Incremental Learning, or The Importance of Starting Small. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society.* Erlbaum, Hillsdale, NJ.

French, Robert M., and Adam Messinger (1995). Genes, Phenes and the Baldwin Effect: Learning and Evolution in a Simulated Population. In Brooks, R.A. and P. Maes, eds., *Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.* MIT Press, 1994 (Second Printing 1995).

Gomez, Faustino and Risto Miikkulainen (in press). Incremental Evolution of Complex General Behavior. *Adaptive Behavior*, MIT Press, Cambridge, MA.

Hightower, Ron R., Stephanie Forrest, and Alan S. Perelson (1995). The Evolution of Emergent Organization in Immune System Gene Libraries. In L.J.Eshelman, ed., *Sixth International Conference on Genetic Algorithms.* Morgan Kaufmann, San Francisco.

Hinton, Geoffrey E., and Steven J. Nowlan (1987). How Learning Can Guide Evolution. *Complex Systems* ***1***, 495-502.

Masters, Timothy (1993). *Practical Neural Network Recipes in C++*. Academic Press, San Diego.

Nolfi, Stefano, Jeffrey L. Elman, and Domenico Parisi (1994). Learning and evolution in neural networks. *Adaptive Behavior* ***2*** (1994): 5-28

Nolfi, Stefano, and Domenico Parisi (1994). Good teaching inputs do not correspond to desired responses in ecological neural networks. *Neural Processing Letters* ***1*** no. 2 (11/94) pp. 1-4.

Nolfi, Stefano, and Domenico Parisi (1995). Learning to adapt to changing environments in evolving neural networks. C.N.R.-Rome Technical Report 95-15.

Nordin, Peter and Wolfgang Banzhaf (1995). Complexity Compression and Evolution. In In L.J.Eshelman, ed., *Sixth International Conference on Genetic Algorithms.* Morgan Kaufmann, San Francisco.

Sebag, Michèle, and Marc Schoenauer (1994). Controlling Crossover through Inductive Learning. In Y.Davidor, H-P.Schwefel, and R.Manner, eds. *Parallel Problem Solving from Nature: PPSN-III.* Springer-Verlag, Berlin.

Whitley, Darrell, Stephen Dominic and Rajarshi Das (1991). Genetic Reinforcement Learning with Multilayer Neural Networks. In Belew, R.K. and L.B. Booker, eds., *Proceedings of the Fourth International Conference on Genetic Algorithms.* Morgan Kaufman, San Francisco.

Whitley, Darrell, Frederic Gruau and Larry Pyeatt (1995). Cellular Encoding Applied to Neurocontrol. In L.J.Eshelman, ed., *Sixth International Conference on Genetic Algorithms.* Morgan Kaufmann, San Francisco.

Appendix - System Equations.

The simulation follows the same definitions as those of Whitley, Dominic and Das (1991).

$$\ddot{\theta}_t = \frac{mg\sin\theta_t - \cos\theta_t\left[F_t + m_p l\dot{\theta}_t^2 \sin\theta_t\right]}{\left(4/3\right)ml - m_p l\cos^2\theta_t}$$

$$\ddot{x}_t = \frac{F_t + m_p l\left[\dot{\theta}_t^2 \sin\theta_t - \ddot{\theta}_t\cos\theta_t\right]}{m}$$

where:

x is the cart position, range ± 2.4 *meters*

$\dot{x}$ is the cart velocity, range ± 1.5 *m/sec*

θ is the pole angle

$\dot{\theta}$ is the angular velocity of the pole

m_p is the mass of the pole = 0.1 *kg*

m is the total mass of the system = 1.1 *kg*

l is the length of the pole = 0.5 *meter*

F is the control force = ± 10 *Newtons*

g is the acceleration due to gravity = 9.8 *m/sec*^{2}