Cultural Enhancement of Neuroevolution

Paul H. McQuesten

Report UT-AI-TR-02-295 August 2002

paulmcq@cs.utexas.edu
http://www.cs.utexas.edu/users/nn/

Artificial Intelligence Laboratory
The University of Texas at Austin
Austin, TX 78712

The Dissertation Committee for Paul Herbert McQuesten
certifies that this is the approved version of the following dissertation:

Cultural Enhancement of Neuroevolution

Committee:

Risto Miikkulainen, Supervisor

Joydeep Ghosh

Benjamin Kuipers

Raymond Mooney

Bruce Porter

Cultural Enhancement of Neuroevolution

by

Paul Herbert McQuesten, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

August 2002

to Pamela, Mitchell, and Leslie

Acknowledgments

Some of the notions of the functions of culture which lead to this work were the product of extensive discussions with Dr. Pamela McQuesten, to whom I have the honor and joy of being married. Her unstinting support was invaluable in the production of this dissertation.

Mitchell and Leslie helped Daddy more than they can ever know. Your love is an inspiration to all who know you.

I would never have gotten into college without the unconditional love and support of Beulah White McQuesten, my mother, who has very patiently waited for her son to finish college. Hi, Mom!

My advisor, Risto Miikkulainen, tried to force me to write a clear, coherent, and understandable dissertation. As all of his students know, Risto is a patient and dedicated mentor. His questions lead directly to two of the methods developed here, CE-MATE and CE-DIVERSE.

The source code of SANE was graciously and generously made available by David Moriarty. This research was supported in part by the National Science Foundation under grants IRI-9504317 and IIS-0083776, and by the Texas Higher Education Coordinating Board under grant ARP-003658-476-2001.

PAUL H. MCQUESTEN

The University of Texas at Austin
August 2002

Cultural Enhancement of Neuroevolution

Publication No. _____

Paul Herbert McQuesten, Ph.D.
The University of Texas at Austin, 2002

Supervisor: Risto Miikkulainen

Any transmission of behavior from one generation to the next via non–genetic means is a process of culture. Culture provides major advantages for survival in the biological world. This dissertation develops four methods that harness the mechanisms of culture to enhance the power of neuroevolution: culling overlarge litters, mate selection by complementary competence, phenotypic diversity maintenance, and teaching offspring to respond like an elder. The methods are efficient because they operate without requiring additional fitness evaluations, and because each method addresses a different aspect of neuroevolution, they also combine smoothly. The combined system balances diversity and selection pressure, and improves performance both in terms of learning speed and solution quality in sequential decision tasks.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Any transmission of traits from one generation to the next via non–genetic means is a process of culture. The goal of this dissertation is to harness the mechanisms of culture to enhance the power of neuroevolution in sequential decision tasks.

To make that endeavor plausible requires first explicating the advantages of culture in the natural world, and explaining how cultural mechanisms relate to neuroevolution. The second section characterizes sequential decision problem domains and why they are a worthwhile challenge. Next, the advantages of neuroevolution approaches to such tasks are explained. This introduction then finishes with a guide to the rest of the document.

1.1 Advantages of Culture

Culture provides major advantages in the biological world. This section decomposes those advantages, beginning by noting that genetically hard–wired behaviors are insufficient for creatures as sophisticated as, say, a predator mammal. Thus, sophisticated creatures must learn. That learning is cheaper when lead by a teacher. Justifying that chain of observations will expose attributes of culture that can be related to neuroevolution, as follows.

Even casual observation of nature reveals that instinctual, hard–wired behaviors are insufficient for all but the simplest organisms existing in static environments. Successful organisms are adaptive. Their behaviors are not completely determined by their genome. In the struggle for survival they have available not only their genetic endowment, but also skills and abilities learnt since birth. Sophisticated adaptive organisms not only learn from their own experience, but are also taught skills by their parents or other elder members of their species. In fact, there are many biological cases where adults can be successful only when they have been taught skills by their elders. For example, many species of songbirds learn their songs from older birds; training cubs to hunt is ubiquitous in predatory mammals; social primates learn group norms by observation of adults, sometimes with explicit teaching (Dawkins 1989).

Culture provides major advantages in the biological world (Hutchins and Hazlehurst 1991). First, culture accelerates learning, because immature organisms do not have to use trial–and–error for everything. They can take advantage of the learning and skills that have been acquired by elders as stepping–stones in their developmental path. Learning from actual experience can also be dangerous.

A second advantage of culture is that it enables the transmission of more information than the genome can contain. The entire behavioral repertoire of an organism does not have to be encoded in its genes.

Thirdly, culture provides organisms with both stability and flexibility. Stability, because culturally–transmitted traits can persist beyond the lifetime of a single organism. Flexibility, because culturally–transmitted traits can be adapted faster than genetically–determined traits. In other words, culture can change on a timescale that is intermediate between learning (during one lifetime) and evolution (during many generations).

Fourthly, shared culture makes communication possible in at least one extreme but definitive case: human language is a cultural construct. Language is not completely specified in the genome, neither is it completely reconstructed by each new human. Indeed, language is only effective to the extent it is shared, and thus it must be learned from elders.

None of these attributes of culture depend on artifacts, such as libraries and pyramids. The non–artifactual component of culture resides completely in the behaviors[1] of individuals.

The above observations of culture meshing with neuroevolution in nature, *in vivo*, motivate my search for powerful problem solving systems based on simulated evolution of neural networks, *in silico*. Neural networks can readily be trained by previous members of the population, and I will also show other ways that behavior can influence evolution. This dissertation will demonstrate how such cultural mechanisms can be exploited to enhance neuroevolution in terms of both speed of learning as well as quality of solution. My novel method is called Culture–Enhanced Neuro–Evolution, or CENE (pronounced to rhyme with gene).

The significance of the CENE contribution will be demonstrated by application to an important and challenging domain of machine learning, sequential decision tasks.

1.2 Sequential Decision Tasks

Sequential decision tasks (Barto 1990; Moriarty 1997) are some of the most difficult and most general problems in machine learning. A sequential decision task is one where an entire sequence of decisions must be made before the outcome becomes known. For instance, in games such as checkers, chess, or go, moves that appear to have near–term benefit may actually be responsible for an eventual loss. The win or lose signal is not available until many decisions have been completed, and there is no information about whether any particular decision was good, bad or did not make

[1] Producing sounds is also a behavior.

a difference. Therefore it is difficult to determine how to adjust decision–making policies in a sequential decision task. Minsky (1963) calls this the *credit assignment problem*.

Improving the computational performance on sequential decision problems can have significant real world consequences. For instance, automobile and air traffic control are two arenas where unknown and unpredictable delays occur between making a decision and realizing its full effect. Feedback in such cases cannot be attributed to any specific single decision. Real–world tasks such as these are a prime motivation for improving the speed of machine learning procedures. For example, even if a perfect solution to highway traffic signal control was available, it would be of no practical utility if the solution required multiple hours to compute.

Having described the tasks, next we turn to solutions.

1.3 Neuroevolution Approach

Reinforcement learning (Sutton and Barto 1998) is a robust symbolic approach to sequential decision tasks. Moriarty (1997, Chapters 2 and 5) compares evolutionary methods and reinforcement learning in some detail. He shows that evolutionary methods have an advantage due to a more robust credit assignment mechanism, at least in domains that are path–dependent. Many other investigators have likewise found neuroevolution, or the simulated evolution of artificial neural networks, to be an effective strategy for solving sequential decision tasks (Belew 1993; Nolfi, Elman, and Parisi 1994).

Artificial neural networks[2] are well–suited to complex problems for two reasons. First, they can recognize patterns in complex inputs, and substantial problems have complex inputs. Second, neural networks generalize well, which is a critical attribute because the available input states in sequential decision tasks are typically so complex that it is rare to encounter the exact same state with any frequency, and thus difficult to apply discrete statistical techniques. Evolutionary algorithms are well–suited to sequential decision tasks because they work with sparse reinforcement naturally and because they search globally, requiring few assumptions about the solution domain.

My goal is to improve the ability of neuroevolution to tackle sequential decision tasks by incorporating culture. In this dissertation the culture contained in an evolving population of neural networks will be used to improve the performance of neuroevolution in sequential decision tasks. The culture implicit in a population is the collection of behaviors of its members. The behavior of a neural network is simply its input–output mapping. I use samples of these mappings to effectively compare the behaviors of networks. Neural networks make an excellent subject for this cultural focus as a supervised training method, backpropagation (Rumelhart, Hinton, and Williams 1986), is available to implement teaching by elders, which is a primary mechanism of culture. I will show how these observations can be combined to modify the standard genetic algorithm to take advantage of culture exhibited by behavior.

[2] All neural networks in this work are artificial, but the adjective is generally dropped to avoid clutter.

CENE includes four unique methods of exploiting culture in neuroevolution. Because each method is carefully designed to leverage a different aspect of the genetic algorithm, the combination of all four methods is a powerful and significant enhancement to neuroevolution. The CENE methods do not examine the genome in any way, and have a relatively slender interface, with the intention that they could be used in neuroevolution systems of other workers.

Empirical comparison will provide clear evidence of improved performance on significant sequential decision tasks.

1.4 Guide for the Reader

This chapter has briefly introduced the problem domain of sequential decision tasks and my approach. The approach draws upon ideas of culture and observations of the natural world to develop principled methods of modifying genetic algorithms through the exploitation of culture.

The next chapter provides a deeper discussion of foundations, including genetic algorithms, neural networks, and learning and evolution. Prior studies of culture in evolutionary computation are reviewed, noting similarities and contrasts with my approach. The third chapter describes my experimental setup, including the performance metrics used to compare different techniques and the statistical methods necessary to determine the significance of the comparisons. Each individual method is then described in a separate chapter presenting the abstract opportunity, the rationale for the chosen approach, specific implementation details, and empirical results and analysis of operation. After the individual methods are covered, Chapter 8 combines them together and evaluates the combination on a more complex task. The discussion in Chapter 9 evaluates the significance of the results for each of the previously presented methods and suggests several possibilities for future work. Finally, the last chapter summarizes the contributions of this work to the advancement of neuroevolution, and to understanding the possibilities of culture *in silico*.

Chapter 2

Background and Related Work

My work focuses on problems where the target behavior is not known but must be learned through exploration, which are called reinforcement learning tasks. This chapter provides the foundations necessary to discuss such tasks, neuroevolution of solutions, and culture in neuroevolution. The chapter begins with a review of reinforcement learning, a more general name for sequential decision tasks, and the fundamental trade–off between exploration and exploitation. There follow brief introductions to genetic algorithms, my chosen approach to reinforcement learning, and neural networks as an appropriate way to express controllers for these tasks. Prior work utilizing culture in evolutionary computation is reviewed.

2.1 Sequential Decision Tasks

Reinforcement learning occupies a middle ground between supervised learning utilizing an omniscient teacher, and unsupervised learning where no teacher is present. In supervised learning, the student is given the correct answer after answering each problem, so the learning task is how to adjust the student to produce that output for that input. In a discrete domain where exhaustive enumeration is possible, supervised learning can be reduced to rote memorization. However, omniscient teachers are not available in many interesting domains,especially not in the natural world. Therefore the natural world cannot provide clues about how to construct or utilize an omniscient teacher.

In reinforcement learning the feedback for each question is not the correct answer, but a measure of goodness of the current answer. When such measure is available only occasionally, the reinforcement is said to be sparse. Sparse reinforcement means that a proposed solution will have to be evaluated many times before any feedback information becomes available. Games are a prime example of sparse domains: many moves must be made before the win or lose signal is available.

Control problems are typically sparse, as well. In control literature, the physical target system to be controlled is typically referred to as the “plant,” whereas in machine learning it is often

called the environment. In interesting problems, the plant must be operated for many cycles before the solution quality can be assessed. For instance, in the pole–balancing problem, solution quality is determined by how long the pole can be kept upright. The quality is unknown until the proposed solution fails. Then the *credit assignment problem* Minsky (1963) must be faced: How should credit and blame be apportioned among the many actions in the sequence leading to a reward or punishment signal? A genetic algorithm (GA) is the answer propounded in this work, as described next.

2.2 Genetic Algorithms

This goal of this dissertation is to investigate the interaction of culture and neuroevolution. That neuroevolution is an effective approach to sequential decision tasks has been previously established. For example, Moriarty and Miikkulainen (1996a) contrast evolutionary and TD methods applied to sequential decision tasks. Gomez and Miikkulainen (2001) perform more extensive comparisons. They find evolutionary algorithms to be superior in continuous and noisy domains, and in those with hidden state. Such problem domains are likely to occur in the natural world.

Genetic Algorithms are a form of search inspired by the processes of natural evolution. There is an extensive prior literature. The classic introductions to genetic algorithms include those of Holland (1975), Goldberg (1989), and Mitchell (1996). The related evolutionary computation methods of Evolution Strategies, Evolutionary Programming, and Genetic Programming are not applicable to this work, as the phenotypes they evolve necessarily have no independent training method. Specifics of the vanilla genetic algorithm, SOTA, used as a baseline in this work are given in Section 3.4.

Any reinforcement learning technique faces the trade–off between exploration and exploitation (Sutton and Barto 1998). The only way to obtain new information about the solution space is to generate a new candidate solution (called a population member in genetic algorithms) and submit it to the fitness evaluator. The central thesis of hill–climbing is that higher fitness values are likely to be found near points that have already been found to have high values. Indeed, if the fitness landscape is monotonic (has only one peak) then searching near the current champion will always succeed in finding that peak. Searching near known good regions is called exploitation, because it exploits current knowledge. Unless the fitness landscape is completely chaotic, points near a known good point have a much better chance of yielding some improvement than a random probe.

Interesting problems, however, are multi–modal. They contain many peaks, called local optima, some of which may be far below the true global optimum. Strict hill–climbing techniques, even if they make sophisticated use of local gradient information, are likely to converge to a local optimum simply because there are more of them (Hertz, Krogh, and Palmer 1991). Thus hill–climbers most often fall short of the global optimum. That is the hallmark of an overly–exploitative technique. One cure for this problem is to restart from a random position when convergence is

detected. A random restart is an exploratory move. Hill–climbers are purely exploitative, so are only intended for locating local optima.

There is an inherent tension between exploration and exploitation in any generate and test machine learning method. For each test a decision is required whether to explore new territory or to search near known good regions. Allocating an evaluation to explore a new region may reveal something promising, but continual random guessing is as bad as exhaustive enumeration. Ideal techniques would continually tune the balance between exploration and exploitation to match the problem at hand. For example, the technique known as simulated annealing has a principled schedule for changing gradually from exploratory to exploitative.

The exploration versus exploitation battle of machine learning is manifested by diversity and selection pressure in evolutionary computation. Diversity, the variation in the population, will concern us in Chapter 6. Selection pressure is simply a measure of how strongly the Darwinian principle of "survival of the fittest" is applied. Low selection pressure means that even individuals with low fitness are allowed to reproduce. High selection pressure means that high–fitness individuals have a much greater chance of generating offspring than individuals of lower fitness. Choosing a high–fitness individual to be a parent is exploitative, whereas letting a low–fitness individual mate is exploratory.

The task of a GA may be viewed as just "asking the right questions," that is, sampling the fitness surface in informative places. Each sample (an evaluation of the fitness function) is expensive, so the goal is to find the highest point with the fewest samples. Indeed, the design principle I have adopted is that none of my methods introduce any additional fitness evaluations. This rules out the entire class of "GA plus local search" hybrid methods (Braun and Zagorski 1994; Hart and Belew 1996), and may be overly extreme, but serves to tightly focus my design efforts. Future work could consider relaxing this prohibition, but, as will be seen in the body of this work, there are many fruitful avenues to explore within this restriction.

The population of a GA can be seen as a memory of a finite number of previous samples. Within evolutionary computing there are several different methods used to combine those memories to determine the next sample point. In contrast, most function optimization techniques, including simulated annealing, work with only one sample point at at time. Some of them can be made "population–based" by running many sample points in parallel, but that is just a speed–up, because there is no interaction between sample points.

The genetic material of an individual is called its genome, or genotype. The genome may be composed of multiple chromosomes, but this work uses only one chromosome, which is sufficient for the problems addressed. A chromosome is a string of functional units, or genes. The word "locus" means a particular position along a chromosome. An allele is a specific value of a gene at a locus. A schemata is a pattern of values that may contain wild–cards. Schemata 1*0 would match alleles 110 or 100. The genetic algorithm manipulates genotypes via crossover and mutation.

For computation, a genotype must be interpreted as a phenotype, which could be a classifier

system, a neural network, or even just the parameters of some given algorithm. Since my goal is to investigate culture, learning must be involved in addition to evolution. Neural Networks have a computationally convenient learning rule and are the phenotypes chosen for this study, as described next.

2.3 Neural Networks

Phenotypes in this work are multi–layer feed–forward neural networks. Neural networks are good phenotypes for sequential decision problems because they can recognize patterns in complex inputs, and substantial problems have complex inputs. Indeed, neural networks can represent any computable function (Hertz, Krogh, and Palmer 1991). Neural networks also generalize well, which is a critical attribute because the available input states in sequential decision tasks are typically so complex that it is rare to encounter the exact same state with any frequency, and thus difficult to apply discrete statistical techniques. The other reasons for adopting neural networks for this study is that they are easy to represent in a genetic algorithm, and a large prior literature exists on neuroevolution, some of which will be reviewed below. In order to study the interaction of learning and evolution, the phenotype must also have a supervised learning procedure, and backpropagation (Rumelhart et al. 1986) fills that role for neural networks.

The term "syllabus" is used herein to mean the collection of problem instances used for backprop training and comparing behavior. Each syllabus item consists of an input vector and the corresponding target output vector, as is normal for a training case in supervised learning. The issue to be addressed in following chapters is how to construct a syllabus in sparse reinforcement domains, where supervised training is normally not applicable.

A primary reason that neural networks are appropriate representations of solutions for this work is that the goal is to reduce the number of fitness evaluations and neural networks can be compared and trained without involving the fitness function. The output of a neural net anywhere in its input space can be computed without being charged for a fitness evaluation. In Chapter 7 backpropagation will be used to train networks, again with no reliance on the fitness function.

2.4 Culture

As is well known, nature is not a function optimizer (De Jong 1993). Natural evolution does not find optima; organisms adopt solutions that are adequate to allow them to reproduce. Nonetheless, the problem–solving abilities of natural systems are impressive enough to inspire many researchers to attempt to abstract the principles employed by biological learners, and incorporate them into machine learning (Belew 1990, 1993).

Indeed, the natural world provides the only known existence proof that complex problem–solvers can emerge from evolution. Researchers in evolutionary computation reason that perhaps

some of the phenomena of natural evolution can be captured in computer simulation. Culture is a natural phenomenon involved in sophisticated behaviors, such as bird songs and human language. My work extends that same perspective to learning and culture. Culture is a powerful force in natural evolution, so I seek techniques to enhance computational evolution with culture.

2.4.1 Culture in computation

Analogies between nature and computer are not perfect. Techniques suggested by observation of nature are not guaranteed to be available or effective in computer simulation. The supposition that "the natural way is the best way" has no objective content. These caveats notwithstanding, I will demonstrate that cultural effects can be effectively harnessed to accelerate artificial neuroevolution.

A reasonable view of culture is that it is composed of assets and behavioral conventions. A complete sociological study of culture would have to account for physical assets, such as food crops and ships, but in computer simulations only informational assets can be considered. This is not a restriction, however, because in neuroevolution the desired goal can be specified as a set of information, such as, what are the correct control outputs for each point in the input space of the pole–balancing task? Informational assets are also called knowledge.

An alternative view of culture is as the *process* of transmitting assets from one generation to the next. Hutchins and Hazlehurst (1991, p.354) characterizes culture as "an adaptive process that accumulates partial solutions to frequently encountered problems." In either view, any non–genetic transmission of knowledge, behavioral traits, conventions or beliefs from one generation to the next is a process of culture.

The advantages of culture in natural biology were mentioned in the Introduction. Two of these advantages have already been found operative in evolutionary computation. Belew (1989) pointed out that culture operates on an timescale intermediate between learning (in a single lifetime) and evolution (over many generations). He advocated "culture" as a third adaptive mechanism, so that all timescales could be covered. Belew (1990) observed that without culture, genetic transmission of good solutions can only go to a few offspring. Cultural transmission allows the same information to be available to many recipients. This increased "fan-out" can be harnessed to increase the speed with which a novel, favorable trait spreads in the population. Belew (1990) also demonstrated that adding culture to evolution and learning increases the speed of adaptation.

Cultural approaches may appear to resemble hybrid genetic algorithms, where evolution is combined with some form of local search or hill–climbing. In fact, the gradient ascent hill–climbing in Braun and Zagorski (1994) is described as learning. However, cultural techniques differ from a hybrid GA in a fundamental way. Hill–climbing uses fitness evaluations, whereas cultural manipulations are constructed without fitness evaluations. Thus, since the number of fitness evaluations consumed is the figure of merit in this dissertation, cultural techniques will be inherently more efficient, if they are effective at all.

2.4.2 Nomenclature

Culture, evolution and neural networks are phenomena of the natural world which have inspired computer simulations. This subsection cautions about the unreliability of biological metaphors in machine learning, and defines some nomenclature adopted for this work.

In order to inform our intuitions, this dissertation uses several terms from the natural world, including "culture," "offspring," and "artifact". It is well to keep in mind that any remaining biological basis is at most metaphoric—the connotations of these terms are not likely to hold in their entirety in machine learning. Only properties that are explicitly demonstrated to occur *in silico* can be relied upon.

For instance, in evolutionary computation we speak of "mating" two "parents" to form a new "child," and all of our biological knowledge of the real world will cause us to intuit a loyalty of parent toward child. But that is patently absurd: there is no loyalty, not even a simulation, in these computations.

Another kind of danger in following nature too closely is that the analogies may be too confining. For example, Lamarkian evolution is impossible in the real world (except in the immune system), so by holding too tightly to analogy one might miss the fact that Lamarkian evolution is quite possible in the computer.

Nonetheless, despite this metaphoric unreliability, these informal terms will be found to provoke interesting questions. For example, Craighurst and Martin (1995) could notice that the standard GA would permit "incest," and so could study the effect of prohibiting it. A terminological example from this work is culling of overlarge litters.

We call an "artifact" any trace that one individual can leave in the environment that can be sensed by other individual(s) who did not witness the act of modification and who can tell that the modification did not occur "naturally." When other individuals can witness actions there is the possibility of implicit communication, which is studied in some multi–agent scenarios but not further developed here. Section 9.6 will present a few thoughts on how culture could affect explicit communication.

Artifacts are an explicit representation of culture. This work explores implicit culture, with no artifacts. However, prior work utilizing explicit representations of culture is reviewed next.

2.4.3 Explicit culture

Most prior work with the combination of learning and evolution was either non–cultural, where each individual fends for itself, or used artifacts to represent culture. In biological systems, these representations would be physical artifacts. In computational systems, any information that exists apart from any individual in the population is an artifact. For instance, if there is any memory apart from the phenotypes of the individuals in the population, that would be an artifact. Such information has been known as a *gene bank*, a *blackboard*, and a *concept library*.

The following studies of explicit culture utilize a variety of memory representations. In Reynolds and Chung (1997) (Section 2.4.3) the meaning of the artifacts is predetermined by the experimenter, while in Spector and Luke (1996b) (Section 2.4.3) the individuals develop their own interpretation of the contents of memory. My work and related work described in Section 2.4.4 will address culture without artifacts.

Shared Memory

Spector and Luke (1996a,b) defined *culture* as "any information transmitted between individuals and between generations by non-genetic means." Their culture resides in an explicit artifact. It is represented as a global shared memory: a set of cells readable and writable by any individual at any time. Intuitively, this approach can succeed if individuals record good ideas in this memory, and later individuals exploit that knowledge. For example, location X might come to represent the best policy to follow in situation Y.

Spector and Luke (1996b) found such culture to be useful in problem domains that contain important regularities that can be taught and learned. They hold that the reasons for this utility are (a) more information may be transmitted than the capacity of the genome, and (b) useful information can spread faster than with evolution alone. Note the similarity to the conclusions of Belew (1990) cited above.

Sebag, Schoenauer, and Ravisé (1997), reversed the typical approach of concentrating on success. They recorded past *failures* in order to avoid them. Their mechanism maintained allele frequencies of the worst performing offspring, in order to bias future offspring. This mechanism is eugenic (operating on the genome); a cultural counterpart would be a global shared memory of traits to be avoided.

Haynes (1998) shows culture in Genetic Programming with a type of blackboard mechanism. He records good subtrees in an external memory available to all individuals. This "collective memory is the body of common knowledge that a group shares." He finds this memory being used to express partial solutions that are larger than the allowed chromosomes.

Long Term Memory

Louis and Johnson (1997) reasoned that when confronting a series of similar problems prior solutions might well provide useful information. Thus, they augmented a search algorithm with a memory of past experience. In particular, they report work where a Genetic Algorithm is assisted by a Case-Based Reasoning module. When a new problem is presented, the initial population is seeded with individuals who did well on similar previous problems, as determined by the CBR system.

Their approach is similar to mine, in that they exploit the partial domain knowledge that appears in the behaviors of the population. The CBR stores genotypes (solutions), but they are

indexed by which problems they performed best on. This basis for selection is in contrast with other systems that attempt to select candidates based on the representation of the phenotype or genotype: Louis and Johnson (1997) defined similarity by a metric on the *problem*, not on the solution.

Louis and Johnson (1997) found that it does not always pay to be overly greedy. Instead of trying to find maximally similar cases, they "inject a set of individuals with different fitnesses saved at different generations." Their set of cases corresponds to my syllabus. I read their result to say that the syllabus should not be overly concentrated on any single region of the problem space.

Normative Beliefs

Reynolds and Chung (1997); Reynolds, Michalewicz, and Cavaretta (1995) presented a model of culture as an explicit "belief space." The belief space stored a generalization of the problem solving experiences of superior individuals. The rules for constructing beliefs were explicitly programmed by the researcher, as were the rules on how beliefs affect the evolution of the population. Beliefs were represented as a vector of intervals, one for each (real-valued) parameter (gene). These intervals were widened and/or narrowed as new high-performing individuals were encountered. Intuitively, belief intervals are supposed to correspond to social norms, as in, "this culture honors those who are virtuous," where "honor" is implemented as increased reproductive success, and "virtue" is determined by proximity to the norms. Actually, Reynolds and Chung (1997) implemented eugenic breeding, whereby alleles were pushed toward the norms. This was possible because the phenotype and genotype are identical. Otherwise, the beliefs would have to be expressed in genotype space to allow eugenic engineering.

The intervals were used to control the amount and direction of mutation. This procedure can be a big win: if some parameters are found to have narrow ranges they would be effectively frozen, reducing the dimensionality of the search space.

On the other hand, this approach will have a hard time with highly epistatic problems (those with a large degree of parameter interaction). For instance, if the two schemas (gene-X:high gene-Y:low) and (gene-X:low gene-Y:high) both have high fitness, that means two disjoint regions of parameter space should be sampled. The belief-space intervals will either exclude one of these regions, or include almost all of the space. If there are two binary inputs this is exactly the exclusive-or problem. This scheme could be advantageous if applied to problems that do not have any such xor character.

Conceptually, the epistasis problem could be solved by extending the procedure to maintain a set of neighborhoods rather than an axis–aligned parallelepiped. While such neighborhoods might be good engineering, I think there is a fundamental problem with the belief–space approach: Procedures such as this are trying to directly determine promising regions of weight space. However, it is always the case that we already know promising points in weight space: the phenotypes of high-scoring individuals. Many different statistical decision procedures are available to generate regions from those points. Direct identification of regions of weight space should be left to these polished

and sophisticated statistical techniques.

Such statistics are effective in simple domains, but they tend to behave like hill–climbers, great at exploitation but weak on exploration. Moreover, if polished statistical procedures are not effective at generating optima from sample points, then the fixed algorithms of Reynolds and Chung (1997) and Rychtyckyj and Reynolds (2001) will not be either. I conclude that raw weight space is not the best place to look for culture. For neural networks there is even a further problem: since a permutation of hidden units does not change the behavior of a neural network, the weight space has many symmetries that can further confound the belief representation.

The belief space idea does suggest a possibility beyond the scope of this dissertation that might prove to be interesting: Instead of fixed generalization algorithms, might it might be possible to somehow *evolve* representations of belief and virtue?

Conclusion

These studies of explicit culture confirm that culture can enhance machine learning, but there seems to be no principled way to choose the most appropriate representation of artifacts. Indeed, to harness the full power of artifactual culture may turn out to require the interaction of several representations. The resolution of that issue seems far in the future. This dissertation focuses on implicit culture, which is reviewed next.

2.4.4 Implicit Culture

The above studies all included an explicit representation of the content of their culture. Let us now turn to a study of implicit culture, where there are no artifacts. Information is transmitted from one individual to another by direct observation or communication. When members of the current population are viewed as imperfect domain theories, using their behaviors to provide guidance to offspring is a cultural mechanism.

Cecconi, Menczer, and Belew (1996) arranged for inter–generational transmission of behavioral traits in a way that contrasts with my methods. They determined the syllabus by the happenstance of parental experience: For a time after an offspring is created, it notionally "lives on its parent's shoulders." That is, it is given the same sensory input that its parent receives. Its motor outputs are ineffective, rather the differences between its outputs and its parent's outputs are used as an error vector for backpropagation training of the offspring. Thus, the offspring is taught by its parent, just as in my technique. The difference is the content of the training, what I call the syllabus.

The offspring can only learn skills which the parent happens to exhibit, which will depend on environmental conditions. During the limited training time it may often happen that the parent is in a bland or benign part of the environment that limits the variety of skills which the offspring would be exposed to. Conversely, in a hostile part of the environment the birth rate is likely to be low and/or parental reactions may be cut short by death, except when the parent is already

extremely competent. Even then, all of the offspring's training is concentrated on an extreme set of circumstances which may not help general competence. Of course, the variety of the learning experience could be enhanced by simply increasing the training period, but that increases training costs. I hypothesize that the emphasis on the quotidian exhibited in Cecconi et al. (1996) will often lead to an impoverished learning experience.

In contrast, the syllabus in my technique can contain arbitrary input vectors, so it can sample any ranges of the input parameters. On the other hand, some parameter combinations may never in fact occur in the actual environment, which should be an advantage for Cecconi et al. (1996). I hypothesize that my propensity to unrealistic situations is easier to adjust than Cecconi et al. (1996)'s propensity to monotony, but that is beyond the scope of this dissertation.

Despite differing syllabi, both Cecconi et al. (1996) and my work, explained in Chapter 7, show that teaching immature offspring to respond somewhat like a parent is effective. Extra–genetic transmission of behavioral tendencies is a promising technique for evolutionary computation.

2.4.5 Summary of Culture in Neuroevolution

There are many aspects of full–blown culture, and thus many ways to represent artifacts have been studied. It seems likely that many different representations could capture some significant aspect of culture, but at present there seems to be no principled way to chose among them, except from the engineering perspective of which ones are empirically faster. Since I do not know how to choose a good representation of culture, this work is based on the minimum possible representation: just the behaviors of the population members themselves, with no artifacts.

Thus, this work is an attempt to bridge the gap between no culture and full culture by studying a restricted case: culture without artifacts. This simpler setting should be easier to analyze than arbitrary culture, and yet more effective than learning alone.

2.5 Conclusion

This chapter has characterized the tasks and solutions to be learned in this dissertation, as well as related previous work. Culture in the natural world is very powerful, and some of those advantages have been shown to obtain in neuroevolution. There have been several approaches to combining culture with evolutionary computation but none of them have the behavioral focus of my work. It appears to me that focusing on behavior without the confounding possibilities of artifacts is a promising route to further understanding of culture in neuroevolution.

Before beginning the chapters that develop my novel exploitations of culture, the specifics of the tasks and measurements that will permit my results to be evaluated are provided in the next chapter.

Chapter 3

Experimental method

This chapter sets forth the experimental methodology employed in this work. First, the specific learning tasks are described. Performance metrics, including both evolution speed and solution quality, are presented. Then resampling statistical methods and their necessity are explained. Finally, benchmarks to compare the new methods are presented.

3.1 Task Descriptions

The tasks chosen to be measured in this dissertation are well–known non–linear control problems with a rich prior literature. This section describes them, although the exact details needed to replicate the tasks are relegated to the Appendix. The tasks include the ubiquitous Pole Balancing problem, and the Acrobot swing–up task studied by Sutton and Barto (1998). Description of the task in the more complex Khepera robot navigation is deferred to Chapter 8 where it is explored.

The two simpler tasks are used to measure the performance of the four component methods to be developed in Chapters 4 through 7: Culling and teaching methods are tested in the Pole Balancing task in order to be consistent with the proposal and earlier work by McQuesten and Miikkulainen (1997). The fitness evaluations are very expensive in pole balancing, and therefore the Acrobot task was used to test mate selection and diversity enhancement. After these measurements, the implementations of the four component methods were frozen before they were exposed to the more complex Khepera domain (Chapter 8). This rather large jump in complexity helps to assure that the component techniques are general, that is, not overly adapted to the domains in which they were developed.

3.1.1 Pole–Balancing task

The Pole–Balancing problem, also known as the Inverted Pendulum, or the Cart–Pole system of Figure 3.1, is a well–known standard control problem which is often used in reinforcement learning

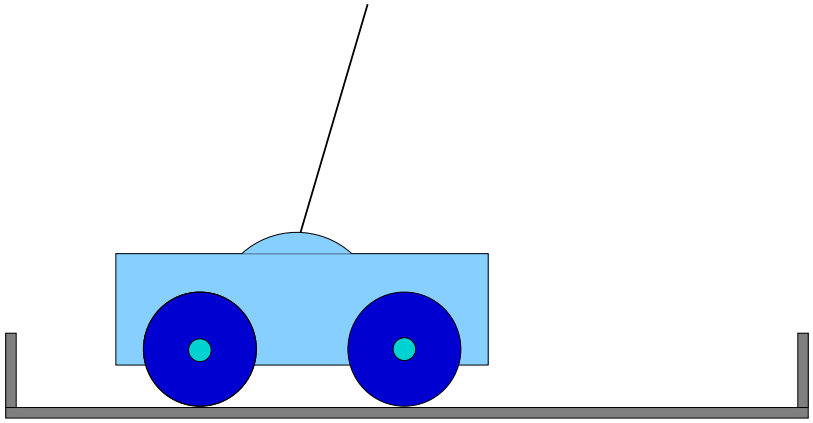


Figure 3.1: Schematic of Inverted Pendulum system (not to scale). The cart mass is 1 kg and the pole is 0.1 kg. The pole is 1 meter long. The track is long enough for the center of the cart to move 4.8 meters. The task is to keep the system within limits by application of a "bang–bang" force.

studies. The task is to keep the system within limits by choosing to apply a leftward or rightward impulse (a "bang–bang" force) at each time step.

Analytic solutions are available from engineering literature, but they are not made available to reinforcement learners. The only reinforcement available in this problem is the failure signal when the pole falls or the cart moves beyond given bounds. A candidate control algorithm must execute for many cycles before its quality can be assessed. Fitness evaluations in this problem are very expensive, requiring thousands of simulation cycles.

There are four degrees of freedom in this system: position and velocity of the cart, angle of the pole, and the angular velocity of the pole. Those are the four inputs to the neural network controller. The output controls whether a leftward or rightward impulse will be applied.

The details of the problem studied herein are as follows: The fitness function is the total number of simulator steps until the pole exceeds the limit angle ($\pm 12°$), taken over all eleven initial conditions from Whitley, Gruau, and Pyeatt (1995). A pole–balance attempt is considered successful if the system remains within limits for 1000 time steps, so a run is successful if a network with a score of 11,000 is found. The details of the equations of motion are presented in Appendix B for the sake of completeness.

3.1.2 Acrobot Swing-up problem

The Acrobot is a 2–joint robot shown in Figure 3.2. It is an abstraction of a human gymnast, but both joints are completely free to rotate unrealistically through a full 360 degrees. The anchor pivot is fixed in space. Torque is applied at the hip joint. The controller's output determines whether the applied torque will be clockwise or counter–clockwise at each timestep.

The goal is to swing the tip up above the fixed anchor joint to a height equal to one leg

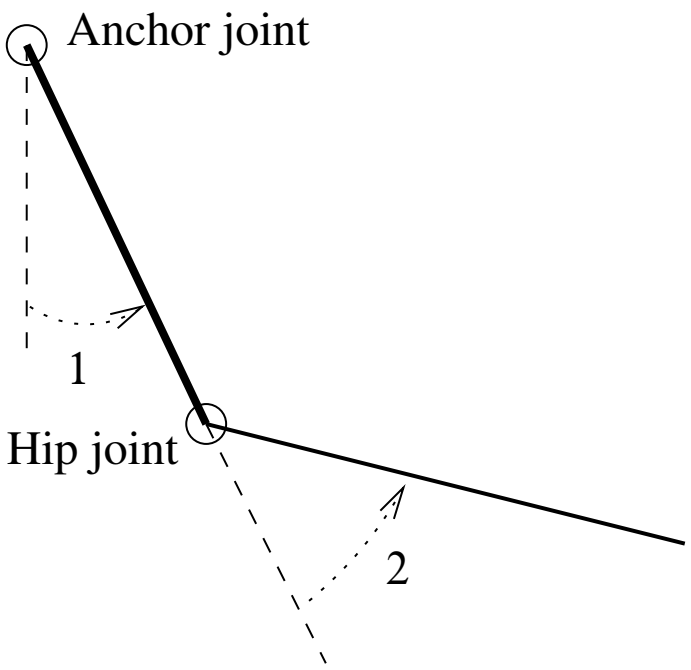


Figure 3.2: The Acrobot: an abstraction of a human gymnast. The joints are free to rotate. The anchor joint is fixed in space, torque is applied only at the hip joint. The task can only be accomplished by swinging back and forth to pump energy into the system.

length. The Acrobot is described by Sutton and Barto (1998, Sect. 11.3). The equations of motion and enegery are derived in Spong and Vidyasagar (1989, example 6.4.2). DeJong and Spong (1994) develops swing-up strategies in a symbolic form. The implementation is detailed in Appendix A for those interested in duplicating the acrobot.

This system has four degrees of freedom and can be parameterized by four state variables: θ_i and v_i, the angles and angular velocities of each leg. Control is exercised five times per second (5 Hz), but the Eulerian update would be too inaccurate at that rate, so the simulation code divides each controller timestep into four subslices, which gives the $delta = 1/5 \div 4 = 50\,msec$ shown above. Angular velocities are bounded by $\dot{\theta}_1 \in [-4\pi, 4\pi]$ and $\dot{\theta}_2 \in [-9\pi, 9\pi]$. The starting state on each trial is $\theta_i = v_i = 0$, that is, at rest hanging straight down.

The Q-learning techniques of Sutton and Barto (1998) require the investigator to specify a decomposition of the input space. For the acrobot task they used a tiling of 18648 tiles The Q-learning task is then to learn the quality values of these eighteen thousand regions. Reasonable neural networks for this problem have orders of magnitude fewer parameters than that.

3.2 Performance Metrics

For the genetic algorithms considered in this work all coupling with the task environment is supplied by a black–box fitness function. A fitness function takes as input a candidate controller produced by the GA, exercises that controller in the specified task environment, and returns a real number reply with the standard semantics of "bigger is better." For example, in the pole–balancing problem the fitness reply is the amount of time the network could keep the pole from falling. As in any reinforcement learning, no feedback is given as to what the network did wrong or could do to improve next time.

A fitness evaluation requires plant simulation and network activation for many iterations. This takes many more CPU cycles than used by the genetic manipulations, even for the relatively simple problems investigated in this dissertation. Simulation of more significant and realistic problems would be even more expensive.

In fact, the plant need not be simulated—in the extreme, the fitness evaluation could as well be done in the physical target system itself. Since the fitness function is a complete black box to the genetic algorithm, these techniques work just as well when fitness is evaluated in the real world. For example, a candidate solution can be evaluated by downloading it into a physical robot and measuring actual performance, as shown by Miglino, Lund, and Nolfi (1995) and Nolfi and Parisi (1995a).

Physical exercise of a plant is very much more expensive than a computer simulator, and sometimes even dangerous. Therefore, in complex domains it is important to learn from as few fitness evaluations as possible. Thus this work seeks to reduce the number of full fitness evaluations required to solve a problem.

Fitness evaluation functions can be arbitrarily expensive, so large amounts of computation performed outside the fitness function are justified. Thus quite elaborate inheritance, development, training, and learning mechanisms can be utilized, as long as they reduce the number of fitness evaluations.

Unless otherwise explicitly noted, and without loss of generality, all fitness values herein are non–negative real numbers, where larger is more fit. In many cases, fitnesses are scaled so that the maximum is unity for ease of presentation.

Speed versus Quality

Trade–offs emerge when choosing to measure speed of evolution or solution quality. Even when a natural metric of solution quality exists, using it may not be appropriate. For example, in Pole–Balancing the number of steps until failure could be called the solution quality. But in actuality, most networks that can avoid failure for 10,000 steps can keep going for a million steps, so that metric quickly begins to measure only differences of useless subtlety. A far better approach defines success as some specified solution quality (see Koza (1992, success predicate)).

Machine learning techniques are then rated by how much effort is required, on average, to generate a success. For example, in the Acrobot task a success is a network that can achieve one leg–height above the anchor joint. In pole–balancing success is defined as avoiding falling for 11,000 steps.

As stated above (Section 3.2,) the measure of effort in this dissertation is the number of fitness evaluations consumed. Different evolutionary methods are compared on the basis of the average number of individual networks evaluated to reach the specified success criteria. When comparing multiple methods, the success threshold must be set low enough that the weakest methods find some solutions. I prefer that there be at least twenty successes to average over.

Reported averages are useful to the extent that they are true expectations of future behavior. Since the true expectation would take an astronomical number of runs, confidence intervals are necessary to adaquately characterize the trustworthiness of the reported average. Section 3.3 describes the statistics necessary for satisfactory confidence intervals.

A learning curve plots solution quality as a function of the time expended to achieve it. While plots of learning curves do show both speed of evolution and quality of solution, the appearance of parallelism between two curves may hide significant differences, as will be shown in Section 8.2.3.

For many tasks a learning curve provides insight into the functioning of the solution generator, but learning curves are not universally applicable. For some problems the quality of the solution is inherently binary. The Acrobot either achieves the straight-up position or it does not, so a learning curve is a line along $x = 0$ until time runs out or success is achieved. Thus, learning curves are not useful for binary qualities. Further, learning curves are not sensible for RBHC, because it does not get better over time.

Like many studies, this work uses a stagnation criterion when speed of evolution is being measured. A run is terminated as unsuccessful if no improvement occurs over sixteen consecutive generations. When the purpose of an experiment is to measure the solution quality achieved over time, the stagnation criteria is disabled.

The standard performance metric in this work is the number of evaluations used per solution found. When appropriate, some performance tables include a column for relative performance, titled $\div$BASELINE, which gives the ratio of that row's performance to the performance of a baseline method.

3.3 Resampling Statistics

In order to report reliable measures of statistical significance and confidence intervals in this dissertation, the modern method of resampling is used. This section describes some pitfalls of more tradtional procedures based on the standard bell curve, and the requirement to carefully account for failure results. Since resampling is not so well known as traditional methods, a brief sketch is

provided.

Many standard statistical tests assume that the measurements have a normal Gaussian distribution. This assumption is often a fine approximation. Indeed, when the quantities of interest are averages of many components, the Central Limit Theorem guarantees they will be close to a normal distribution (Moore and McCabe 1993). However, strongly non–normal measurements with long tails and outliers abound in this dissertation. Standard tests of significance can be misleading or erroneous (Weiss and Kulikowski 1991).

There are "non-parametric" statistical procedures, so called because they assume nothing about the actual distributions, and therefore have no parameters, like average and standard deviation. Many of these techniques were known but impractical until the advent of cheap computing power, and therefore were not used in machine learning until recently.

The number of fitness evaluations required to find a solution is figure of merit (page 9) to compare different machine learning methods in this work. Since not every run will find a solution, a principled way to account for the evaluations expended on unsuccessful runs is required. Table 3.1 is a small artificial example of this point.

Evaluations	**Success?**	**Method**
10	pass	Standard
20	pass	Standard
50	FAIL	Standard
25	pass	Improved
30	pass	Improved
35	pass	Improved

Table 3.1: Artificial example comparing two methods, "Standard" and "Improved." The *Evaluations* column is the number of evaluations the method consumed in a particular simulation. *Success?* indicates whether the success criterion was met. The averages are misleading unless the FAILed run is included.

To report these example results, one plausible practice would be to give the success percentage along with the evaluations per solution of successful runs. The Standard method has an average of 15 with a success rate of 67% while the supposedly Improved method has an average of 30, which seems twice as expensive. That is misleading. The Standard method actually has an average of 40 evaluations per solution ($10 + 20 + 50 \div 2$), and so is expected to be more expensive in the long run.

When two sets of numbers are drawn from the same distribution their averages are likely to be somewhat different. Could such random noise account for the difference in the observed scores of Standard and Improved? The *null hypothesis* is that both sets of observations are random draws from the same underlying distribution. Resampling computes the probability that the null hypothesis could be true, given the actual observations.

Some works just report the success percentage along with the evaluations per solution of

successful runs. This is not too bad when the methods have similiar success rates. Reporting total evaluations expended divided by the number of solutions found is fine for comparing averages—the expectations of the methods involved.

However, one also needs to know how much to trust the averages (standard deviation, perhaps), and, most importantly, whether the difference between two averages is significant, or could be due to random chance. Thus, many works also report a statistical test of significance, often Student's *t*-test (Moore and McCabe 1993). Whereas the *average* of many runs tends to be normally distributed, there is no guarentee that the performance of each individual run will be so well–behaved, so significance tests based on the normal distribution are not applicable.

Furthermore, such tests take as input a simple set of numbers: there is no place to input the pass/fail indicator! A significance test on the data of Table 3.1 is not meaningful, because some of the runs failed. Eliminating the failed runs from the data submitted to the test understates the cost.

A machine learning method that sometimes fails can often be redefined to be a "restart" type, wherein it reinitializes to a random state when the failure predicate becomes true. It is then natural to add the unsuccessful evaluations to those of the following successful run. That requires that an experiment only be terminated immediately after a successful run, which can sometimes be practical. However, if the restarts are truly independent, it is equally likely that the sequence of runs could have occurred in any order, which means that the unsuccessful runs have no logical association with the following successful run to which their evaluations are being added. The computed average evaluations per solution will be correct under any ordering, but the distribution of averages will be multi–modal because some runs are combined with a failed run (or runs) and some are not. For a multi–modal distributions the calculated standard deviation may not have the statistical properties expected of it, for instance that 68% of the measurements are within $\pm 1\sigma$ of the average. Thus, conclusions based on reasoning about such a standard deviation are not sound.

Resampling offers an easy and principled way to solve the problem of non–normal distributions and also can handle failed runs accurately. For a comprehensive introduction to resampling see Lunneborg (1999) and David C. Howell's resampling page at
http://www.uvm.edu/ dhowell/StatPages/Resampling.

In order to compute the probability that two sets of measurements are not significantly different, begin with the standard *null hypothesis*, which is that the two sets are samples from the same distribution. According to the null hypothesis, any assignment of the actual measurements to the sets is equally likely. In principle, the exact method is then to exhaustively generate all possible combinations of the data, and compute the two groups' averages for each arrangement, counting the number of times the difference in the two averages exceeds the actual observed difference. That count is exactly the support in the data for the null hypothesis. Dividing by the number of permutations gives the frequency of cases that support the null, called the P–value. If P–value is small, say, $P < 0.01$, the null hypothesis is unlikely, so one concludes that the sets are drawn from different distributions. Statistical significance is a low probability of the null hypothesis, or low *P-value*.

Since the number of permutations grows exponentially with the size of the datasets, practical resampling performs a Monte Carlo estimate of the count by examining a large random subset of all possible combinations. Instead of a simple average any other statistic of interest can be computed, for example, the median or some quantile.

Since resampling can sample from sets of vectors as well as any other kind of set, failed runs are easy to account for, as follows: Each measurement, such as a run of some learning algorithm, is treated as a 2–vector consisting of (1) the number of fitness evaluations used, and (2) the pass/fail flag. The number of evaluations per solution is computed for each rearrangement of the data.

The only assumption of resampling is that the input samples be unbiased, and that an adaquate number be available to approximate the entire universe of possibilities. The use of quality random number generators prevents bias. The sample size requirement is not stringent: With only 20 unbiased samples the probability that they are all above or all below the true average is only about one in a million. The number of samples available in the results reported in this dissertation varies according to the computational load of the various experiments, but is always at least 20, and more often 50–100.

The resampling analysis takes the available N samples as a description of the underlying distribution, assuming that each sample is equally likely. That assumption is sound because each of the samples was in fact observed once in a series of unbiased experiments. Thus, another possible set of observations from that distribution is obtained by randomly drawing N values with replacement. For each such possible set the statistic of interest is calculated, for example, the mean of the set. Additional sets are drawn many thousands of times, for example 5000–fold resampling of the mean draws five thousand independent sets of N observations, calcuating the mean of each. The 95% confidence interval of the mean is then available by discarding the 250 lowest and 250 highest.

In the situation described above, where each observation is actually a 2–vector consisting of a number of evaluations accompanied by a pass/fail flag, the statistic of interest is number of evaluations per solution, which is just the sum of all evaluations consumed divided by the number of successful observation. The confidence interval calculation of the previous paragraph is just as valid for this statistic if "mean" is replaced throughout by "evaluations per solution."

Exact accuracy limits on resampling can be calculated when the distribution is normal. Specifically, if the exact P–value for some test is 0.050, then 5000–fold resampling will, 99% of the time, yield a P-value that falls between 0.042 and 0.058. When the exact P–value is 0.010, then 99% of the time a 10,000–fold resample will yield a value between 0.007 and 0.013 (Lunneborg 1999). These uncertainites are modest.

The confidence intervals and significance measurements reported in this dissertation (for example, Table 4.2) are computed by the resampling procedure summarized above.

3.4 SOTA: a State of the Art Genetic Algorithm

There is no one agreed–upon standard for all genetic algorithm studies. My intention is to develop enhancements that will benefit genetic algorithms in general, but my techniques need to be measured against some specific benchmark. A vanilla, but state of the art GA abbreviated as SOTA, for "State of the Art," is adopted as the concrete benchmark in this work. This section documents the SOTA implementation and common parameter values, and defines the interfaces to the advanced methods to be developed in subsequent chapters. The interfaces are designed so that one SOTA method can be precisely replaced by an alternative method. The contribution of a new method can thus be clearly determined by comparing performance with the new method intstalled and removed, confident that no other code differences exist.

This benchmark GA is intended to represent current best practice in the field, with the host of parameter choices set to reasonable values. SOTA is a steady state genetic algorithm, whereby each child is inserted into the population as it is generated. For reporting purposes a new generation is declared every P-th individual, where P is the population size. For more about steady-state GAs, see, for example, Whitley (1989), Syswerda (1991), or DeJong and Sarma (1993, 1995).

All populations consist of 200 individuals unless otherwise specified. The initial individuals are randomly generated. Since the population size is fixed after initialization, each new offspring must displace some existing individual. Classical genetic algorithms tended to replace a parent with its offspring (sometimes, only if the offspring was more fit). Function KILLSELECT in Figure 3.4 selects an individual to be discarded (decedent) as the least fit of a random sample of 4 individuals (a 4–tournament). This means that good parents will remain elegible to reproduce again, even if they once produce a modestly better offspring.

The population is not sorted, but the current best individual, called the champion, is always preserved in the population due to the method of selecting decedents. By definition, there is always an individual of lesser fitness than the champion. Preservation of the champion is referred to as "elitism" in the GA community. Leaving the population unsorted preserves the inherent parallelism of the basic genetic algorithm.

Many different mating strategies are possible, but binary tournament selection is used here because it requires no global calculation and thus is not a parallelization chokepoint, and because it spreads the mating opportunities across fitness ranks in a smooth and reasonable fashion (Blickle and Thiele 1995; DeJong and Sarma 1995; Goldberg and Deb 1991; Whitley 1989). In SOTA each mating parent is chosen by an independent binary tournament (function MATESELECT in Figure 3.4).

As recommended by Masters (1993), one–point crossover is used $\frac{2}{3}$ of the time, and two–point crossover $\frac{1}{3}$ of the time, This crossover blend results in a constant probability of disruption of schemata, independent of defining length, just like uniform crossover, but still permits building–block linkage (Schaffer, Eshelman, and Offutt 1991).

This work uses the same type of gene encoding as Moriarty (1997, Section 4.5), who demon-

```
program SOTA () {
   input: popSize = population size
   input: genLim = generation limit
   input: EVALFITNESS = the fitness evaluation function
   input: SUCCESSPREDICATE = function defining success
   initialize population array P[popSize] with random genomes
   evalLim= popSize * genLim
   numEvals = 0
   while numEvals < evalLim
         choose first parent A = SOTA.MATESELECT()
         choose second parent B = MATESELECT()
         choose individual to replace D = KILLSELECT()
         child C = BREED( A, B)
         replace D by C
         kidFit = EVALFITNESS( C ) evaluate fitness of new offspring
         numEvals += 1 count evaluations
         if SUCCESSPREDICATE( kidFit ) then exit
         if (numEvals mod popSize == 0) then report population attributes
}
```

Figure 3.3: SOTA, the State–Of–The–Art genetic algorithm used as the baseline for this work. SUCCESSPREDICATE always returns **false** when ultimate solution quality is to be measured. EVALFITNESS evaluates an individual in one of the tasks described in Section 3.1. MATESELECT, KILLSELECT, and BREED are functions which are replaced in advanced methods developed in succeeding chapters. The baseline SOTA subroutines are shown in Figure 3.4.

strated the advantages of such encodings. Each hidden unit has several genes, how many depends on the application. Each gene encodes a connection from or to its hidden unit, and consists of two parts: (1) tag: 8 bits designating the input or output unit on the other end of the connection, and (2) weight: 16 bits interpreted as the floating–point weight of the connection. This encoding can describe any feed–forward network with one layer of hidden units, which is sufficient for the tasks measured in this dissertation. A trivial change in interpretation of the tag to allow hidden units to be addressed permits any network to be described, including recurrent ones.

To the mutation and crossover operators in this work, a chromosome is manipulated as a binary bit–string[1] that is just the concatnation of the genes. A small amount of mutation is included: 0.1% probability of each bit flipping. The learning rate for backpropagation is 0.15, with no momentum. These are middle–of–the–road parameter values for neuroevolution.

[1]Some workers have found other representations to be more effective in some domains, but that is still an area of research.

```
function SOTA.MATESELECT () {
   binary tournament selection
   choose two individuals randomly without replacement
   return the individual of greater fitness in that sample
}
function SOTA.KILLSELECT () {
   the champion will never be killed
   four–way tournament selection
   choose four individuals randomly without replacement
   return the individual of least fitness in that sample
}
function SOTA.BREED ( A, B ) {
   standard breeding
   create new child C = CrossOver( A, B )
   apply mutation to C
   return C
}
```

Figure 3.4: Subroutines of the SOTA implementation. These functions called from Figure 3.3 are overridden in advanced methods developed in succeeding chapters.

3.5 RBHC: Random Bit Hill–Climber strawman

As a baseline comparison and sanity check, Random Bit Hill–Climbing, RBHC, per Davis (1991) was implemented and tested. RBHC is a form of next-ascent. It views the genome as a binary string, and flips bits in a random order. RBHC does not use the stagnation criteria; by design it stops only at the top of a hill, when no single bit flip will improve the current score. As a hill–climber, its performance is reasonable for uni–modal fitness landscapes.

RBHC does no learning—it is a strawman. However, RBHC is useful in the negative sense that any machine learning method that does not do better is surely deficient. Because each RBHC trial is independent of all others, learning curves are not appropriate.

3.6 Conclusion

This chapter described the experimental methodology of this study. Complex non–linear sequential decision tasks are chosen to evaluate neuroevolution techniques in this dissertation. Acrobot, Pole Balancing, and Khepera Chapter 8 are well–known and provide connections to the prior literature. Fitness evaluations are the appropriate measure of effort for these tasks. Speed of evolution is measured by the number of fitness evaluations required to reach a fixed solution quality. Ultimate

solution quality in a fixed number of fitness evaluations is an important alternate metric. Due to the highly non–normal distributions encountered in these studies and the requirement to account for unsuccessful runs, resampling statistical methods will be used to compute significance values and confidence intervals. Finally, SOTA is a serious benchmark for measuring the contributions of this work, while RBHC is a strawman to help detect fallacious measurement.

Having described the tasks and measurement tools, the next four chapters proceed to develop novel exploitations of behavior to improve neuroevolution.

Chapter 4

CE-CULL: Selection from overlarge litters

With the foundation laid down in the previous chapters, we can now turn to the first exploitation of culture devised in this work, CE-CULL. A litter is the set of offspring produced from each mating. In CE-CULL, unlike in standard neuroevolution, overlarge litters are produced, and one offspring is selected based on cultural techniques. This is found to increase the performance of neuroevolution.

In this chapter I first show the dismal distribution of fitness that results from the usual practice of crossover in standard neuroevolution. Then I show how that distribution could be improved by an imaginary, costless fitness estimator, that is, an unrealizable perfect oracle. Next I construct a less–than–perfect but implementable estimator from the behavior of the current population, resulting in the CE-CULL method. An empirical test shows the CE-CULL method to be practical and effective. Finally, the effects of litter size and questionnaire length are determined to be not critical over a broad range.

4.1 Culling motivation

In the standard GA the litter size is usually one or two[1]. Overly large litters or overproduction of offspring results from producing from each mating a litter whose size is beyond the carrying capacity of the environment. For example, hundreds of sea turtles hatch for each one that actually survives to reproduce. In this work, overlarge means more offspring are generated than are going to be added to the population. In fact, only one offspring from each litter will be added to the population. The issue is how to select that one in such a way as to best accelerate progress.

Culling in biology, *in vivo*, is accomplished by the environment, in parallel. Neuroevolution *in silico* has rather less compute power available, so reducing the number of fitness evaluations is

[1] A litter size of two is used when both complementary possibilities of crossover are retained. Whatever the size, a standard GA will insert the entire litter into the population, so it is not overly large by my terminology.

crucial. The novel discovery of this work is that an inaccurate and inexpensive *estimate* of fitness can be effectively used for culling. An effective estimate is constructed from cultural mechanisms.

The genetic operator of crossover poses 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, many different genotypes specify indistinguishable phenotypes. When phenotypically similar individuals mate, and their genotypes use a different permutation of hidden units, the result of crossover is most likely to be nonsense. Whitley, Dominic, and Das (1991) described these "disastrous crossovers" as resulting from the symmetries in neural net representations. To overcome this problem, they used very small populations to force early convergence to a single one of the equivalent permutations. Nolfi, Elman, and Parisi (1994) omitted crossover entirely, claiming that the problem must be better understood before crossover would be productive.

Such early convergence has a high cost: it compromises the exploratory ability of neuroevolution because crossover is ineffective in converged populations. When crossover is absent or ineffective, mutation bears the entire load of evolution, which requires very high mutation rates. As a matter of fact, pair–wise mating through crossover is the essential feature that distinguishes genetic algorithms from population-based hillclimbers (Eshelman and Schaffer 1993). I contend that mutation alone will be found to suffice only for relatively simple problem domains. In such simple domains learning is likely to be an overkill, and not competitive with less sophisticated techniques. Developing methods for more difficult domains demands that crossover be the primary genetic operator. Thus the challenge adopted in this work is to discover how to properly utilize crossover.

To gain insight into crossover's operation, let us first inspect the distribution of fitness of offspring in a standard Genetic Algorithm. The task is the well–known control problem of the inverted pendulum, described in Section 3.1.1. The histogram of Figure 4.1 reports the relative fitness from crossover observed during 100 runs of SOTA. Over 400,000 individuals were generated and 53% of the runs succeeded.

The ordinate in Figure 4.1 is the ratio of offspring fitness to average parental fitness: the offspring of an individual crossed with itself would be identical to its parent and therefore would have a ratio of one. Analyzing crossover in terms of this relative fitness is more appropriate than absolute fitness, because absolute fitness has an upward bias as the evolution proceeds[2]. Relative fitness, even when aggregated into a histogram, focuses on the incremental operation of the genetic algorithm. For instance, if we could obtain a distribution with an average relative fitness of 1.07, then the population fitness would be expected to double in eight generations. In practice the average relative fitness is far less than one, so without selection pressure fitness would continually drop. Nonetheless, higher average relative fitness does cause faster growth.

The distribution in Figure 4.1 is largely deleterious. Most offspring are much worse than their parent. In fact, the average offspring is only 30% as fit as its parents. That most crossovers are

[2] at least there is an upward bias in any run where the procedure is successful!

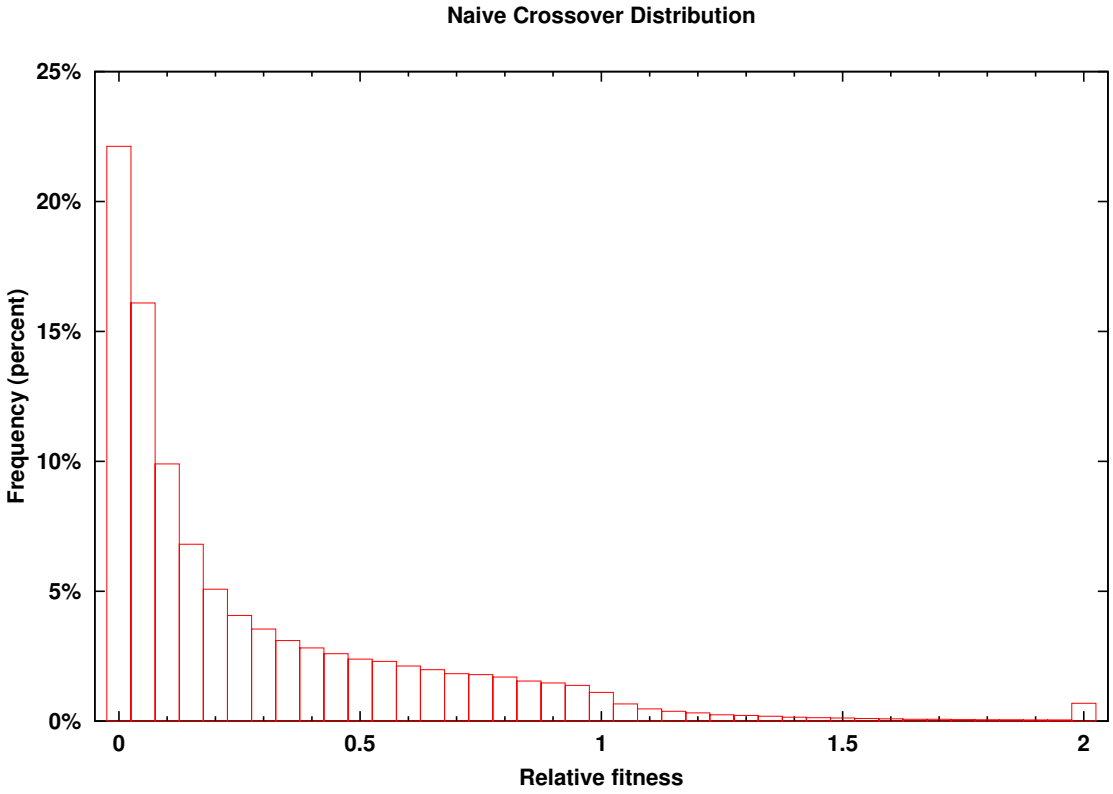


Figure 4.1: Fitness distribution of standard crossover in 100 runs containing over 400,000 individuals. The last bin contains all values ≥ 2. The average offspring is only 30% as fit as its parents. The lower lobe is so large that it is possible to suppress it, dramatically improving the quality of offspring.

poor has been previously observed by, for example, Nordin and Banzhaf (1995). The distribution is dominated by an enormous peak just above zero fitness, with a slight lump near the average of the parents. This leftward trend is intuitively plausible: in any difficult domain, there are simply many more ways to do things wrong than right.

To be sure, a few crossovers are outstanding. The lump at the right side of Figure 4.1 is due to the roughly 7 of each 1000 offspring that had fitness exceeding twice their parents' average. These rare high performers compensate for a lot of poor crossovers.

Note that Figure 4.1 shows the fitness of offspring was below 5% of average of parental fitness 100 times as often as it was greater than average parental fitness. That observation suggests a method of making crossover more productive. The effective crossover distribution could be improved by suppressing the lower lobe of the distribution. This is the main idea of culling. Before trying to find a method of suppression, however, it is instructive to first assume it can be done and determine whether it would really be useful. Such a study is in the next section.

```
function PerfectCulling.BREED ( Parent1, Parent2 ) {
   input: Parent1 and Parent2
   set highSoFar = −∞
   repeat litterSize times
         generate candidate offspring C = SOTA.BREED
         let F = fitness of C
         if F > highSoFar
               highSoFar = F
               bestKid = C
   insert bestKid into the population
}
```

Figure 4.2: Perfect Culling implementation. Candidate offspring are generated by standard breeding, Figure 3.4. Only one fitness evaluation per actual offspring is charged against performance. The exact best offspring from each litter is retained, so no culling method can do better.

4.2 A perfect oracle

To determine how effective culling could be in principle, a perfect oracle is employed. The perfect oracle actually computes a full fitness evaluation. The purpose of the perfect oracle is to determine the effectiveness of culling, while ignoring efficiency, so the fitness evaluations used to implement this "magic" oracle are not counted in reporting its performance. In other words, a perfect oracle knows exactly how good each offspring is, so such an oracle can retain the truly best ones.

Perfect culling is not realizable in practice because it requires a full fitness evaluation of each candidate offspring, but its performance is scored as if only one evaluation per actual offspring was used. However, the perfect oracle will later be replaced by an efficient, although imperfect, estimator of fitness based on culture. The performance of the perfect oracle

an efficient method because it allocates too many evaluations to each mating, while a standard GA would explore many more different matings with that many evaluations.

The implementation of Perfect Culling is shown in Figure 4.2. To produce each 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 4.3, which represents the 264,000 total crossovers performed in 30 runs of 8–fold culling with a perfect oracle, where 30,000 individuals were placed in the population. All runs succeeded on the pole–balancing task, on average taking 1300 chargeable evaluations.

The shape of the crossover distribution in Figure 4.3 is much better than that of the standard GA shown in Figure 4.1. The average offspring is now 62% as fit as its parents, compared to 30% for SOTA. Over 2% (21 per 1000) of the offspring are twice as good as the parental average, which

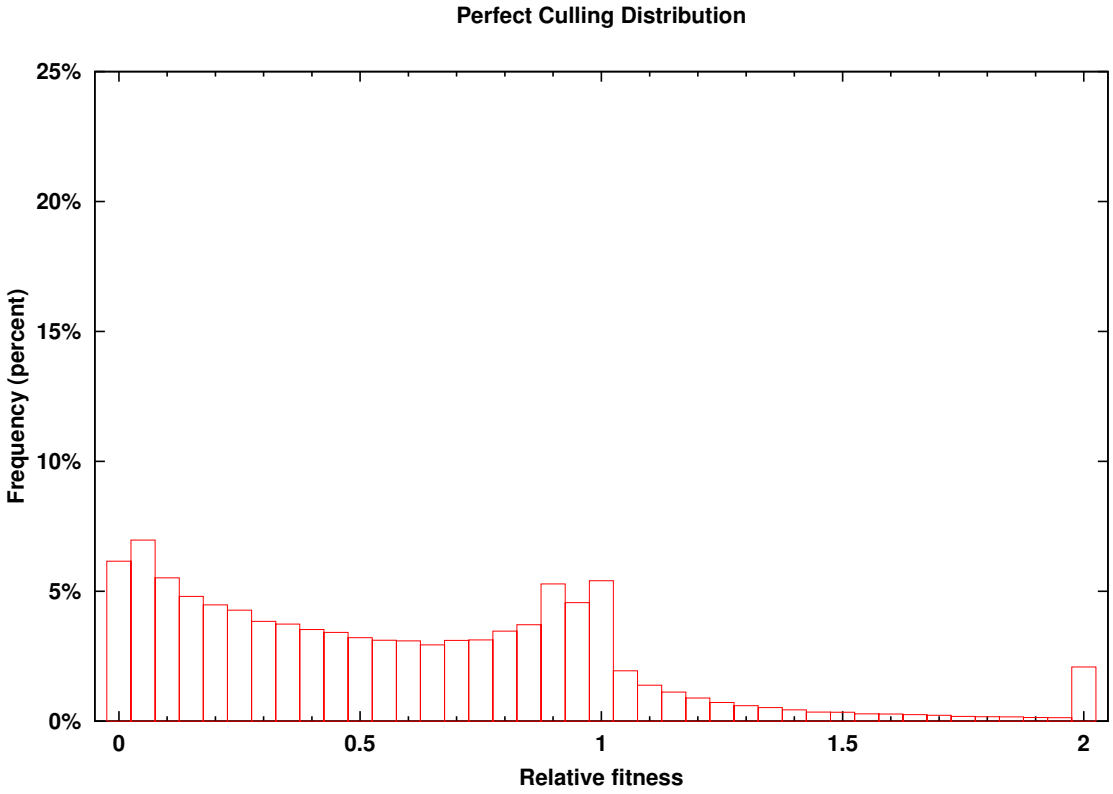


Figure 4.3: Distribution of fitness by crossover with perfect oracle. The abscissa scale matches Figure 4.1. Average fitness is 0.64, almost double that of standard crossover (Figure 4.1). The performance of Perfect Culling is driven by this distribution, which is much better than the offspring distribution of SOTA(Figure 4.1).

is triple ($21 : 7$) the rate of SOTAin Figure 4.1.

Table 4.1 (page 34) shows that culling with a perfect oracle improves performance dramatically. We have found what we want: a method to improve the yield from crossover. However, it cannot be constructed, so the next challenge is to devise a practical culling plan based on a less than perfect oracle that can preserve some of this improved efficiency.

4.3 CE-CULL implementation

To devise a practical culling scheme the perfect oracle used above must be replaced with a method that estimates fitness without making full fitness evaluations. The low performance peak of Figure 4.1 is so dominant that all that is needed to make crossover more productive is to recognize abysmal individuals with reasonable probability. For instance, poor neural nets could be identified as those with no connections to their inputs or outputs. Other such *ad hoc* rules to determine poor

quality by examining the genotype are possible as well, but domain–independent rules seem to be rather binary, as in the no–connection example. Determining relative performance of two networks requires rules specific to the problem domain, which are not available outside the fitness function. Instead, my cultural approach is to perform a preliminary evaluation guided by the behaviors of the individuals in the current population.

My practical culling method, CE-CULL, quizzes the new offspring and grades them with respect to the current population's knowledge of the answers. An individual that can answer simple questions as well as an above–average performer is probably better than the vast majority of potential offspring. In short, the method selects an elder conspecific as a benchmark and applies the syllabus (8) as a quiz. The algorithm is shown in Figure 4.4.

```
function CE-CULL.BREED ( Parent1, Parent2 ) {
   input: Parent1 and Parent2
   input: set of questions, Q[] // the culling syllabus
   input: a benchmark individual, B
   for each question q in Q[]
         activate benchmark B on input q
         let BV[q][] = resulting output vector
   set highSoFar = +∞
   repeat litterSize times
         generate candidate offspring C = SOTA.BREED( Parent1, Parent2 )
         for each question q in Q[]
               activate child C on input q
               let CV[] = resulting output vector
         compute dist = CV[] - BV[] // Euclidean distance
         if dist < highSoFar
               highSoFar = dist
               bestKid = C
   return bestKid
}
```

Figure 4.4: CE-CULL implementation. Candidate offspring are generated by standard breeding, Figure 3.4. See text for how questions and benchmark individual are composed. Selection is made without any fitness evaluations, by choosing the offspring whose answers are closest to the benchmark.

The syllabus contains a list of questions, which are just random input vectors. The target answers for the quiz are obtained by activating the benchmark network on each question and saving each resulting output vector. Then each candidate offspring is activated on each question, and the difference between its answer (output vector) and the benchmark answer is summed. The grade is the Euclidean distance between the candidate's output vector and the benchmark's output vector. Since the benchmark is drawn from the current population, a truly genius candidate will inadver-

tently be penalized. Thus, if over–applied, this technique would result in continual mediocrity. The low lobe of the crossover distribution (Figure 4.1), however, is so large that there is little possibility that culling will fail in the long run.

In the experiments reported in this work the benchmark is the highest scoring individual found so far. This current champion is guaranteed to have above–average performance. Other potential methods of constructing the benchmark will be discussed in Section 9.7.

This investigation uses an elementary method to determine the questions included in the culling quiz: they 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.

The work of Sebag et al. (1997), discussed in Section 2.4.3, with avoiding past failures suggests a negative variety of culling: favor those offspring with the least similarity to a loser. A similar generalization is a syllabus containing negative questions, that is, questions that will be rated higher when the answer is farther from the benchmark. The problem with negative questions is that it is too easy to generate wild answers. That type of selection pressure tends to reward networks that can merely generate large but otherwise inappropriate numbers. Preliminary experiments (not shown) confirmed this observation, so there are no negative questions in this work. (However, there are uses for diversity of opinion, and other ways to evaluate it, as will be found in Chapters 5 and 6.)

4.4 Culling results

The main result is that CE-CULL is about 3 times more efficient than SOTA, i.e., CE-CULL takes less than 35% of the evaluations needed by SOTA(Table 4.1). CE-CULL requires over twice as many evaluations per solution as perfect culling (2776 vs. 1300), but that is true only because the evaluations used by the perfect oracle are not counted. The Global Culling row represents the very same runs as Perfect Culling, except that its score counts those free fitness evaluations. The improvement of CE-CULL over Global Culling is 10,400:2776, equal to a factor of 3.7.. Thus, CE-CULL allocates its fitness evaluations so as to make them over three times as effective as overlarge litters with no questionnaire.

The *HighXO* column of Table 4.1 shows that CE-CULL has almost twice (0.012 : 0.007) as many offspring with fitness $\geq$ double the parental average as SOTA. The *AvgXO* column also shows great improvement (0.569 : 0.328) in average relative fitness, a substantial fraction of the improvement of Perfect Culling (0.642 : 0.328).

In this problem RBHC(Section 3.5) performs as well as SOTA, suggesting that the error surface is quite smooth and without too many deceptive local minima. In fact, the 4% performance difference between RBHC and SOTA is not statistically significant (Table 4.2). However, RBHC only finds a solution 80% as often as the standard GA does (42% vs. 53%).

Method	Eval/Soln	÷SOTA	Success%	AvgXO	HighXO
Global Culling	10,400	1.29	100%	0.642	0.021
RBHC	8,340	1.04	42%		
SOTA	8,037	1.00	53%	0.328	0.007
CE-CULL	2,776	0.35	85%	0.569	0.012
Perfect Culling	1,300	0.16	100%	0.642	0.021

Table 4.1: Performance of culling and standard methods on the pole–balancing task. Performance is shown as *Eval/Soln* = the number of evaluations used in all runs divided by the number of successful runs. The ÷SOTA column is the ratio of the score to SOTA's score. *Success%* is the percentage of runs that found a solution. *AvgXO* is the average relative fitness from crossover. *HighXO* is the fraction of offspring with relative fitness ≥ 2. At least 20 runs were averaged per row. CE-CULL takes only 35% of the evaluations needed by SOTA, which means it is a practical, effective implementation of culling based on culture.

Eval/soln	Method	Perfect	Cull	Sota
1,300	Perfect			
2,776	Cull	0.0		
8,038	SOTA	0.0	0.0	
8,341	RBHC	0.0	0.4	43.9

Table 4.2: Likelihood of null hypothesis: Probability of equal means (in %) for each pair of datasets. A small value implies true means are different. RBHC and SOTA are indistinguishable, but the rest of the differences are significant.

Table 4.2 shows that the performance differences of Table 4.1 are statistically significant, using the resampling method described in Section 3.3. For instance, the table says that if there is no difference between CE-CULL and RBHC, then the chance of the actual observations occurring is $P < 0.004$. This is a small chance, so the null hypothesis is rejected at the 1% level. The other differences are significant, as well, except that RBHC and SOTA are indistinguishable.

Figure 4.5 shows the histogram of relative fitness resulting from crossover in CE-CULL. Consistent with the performance averages, the CE-CULL distribution is not as good as the Perfect Culling distribution shown in Figure 4.3, but the overall shape is comparable. In contrast to both of these results, the crossover distribution of Figure 4.1 has almost no central peak, and its lower lobe is many times larger than the central value. An explicit comparison is provided in Figure 4.6, which plots the cumulative distribution functions of all three histograms. CE-CULL is much closer to Perfect Culling than to SOTA, so much of the benefit of Perfect Culling should be attained.

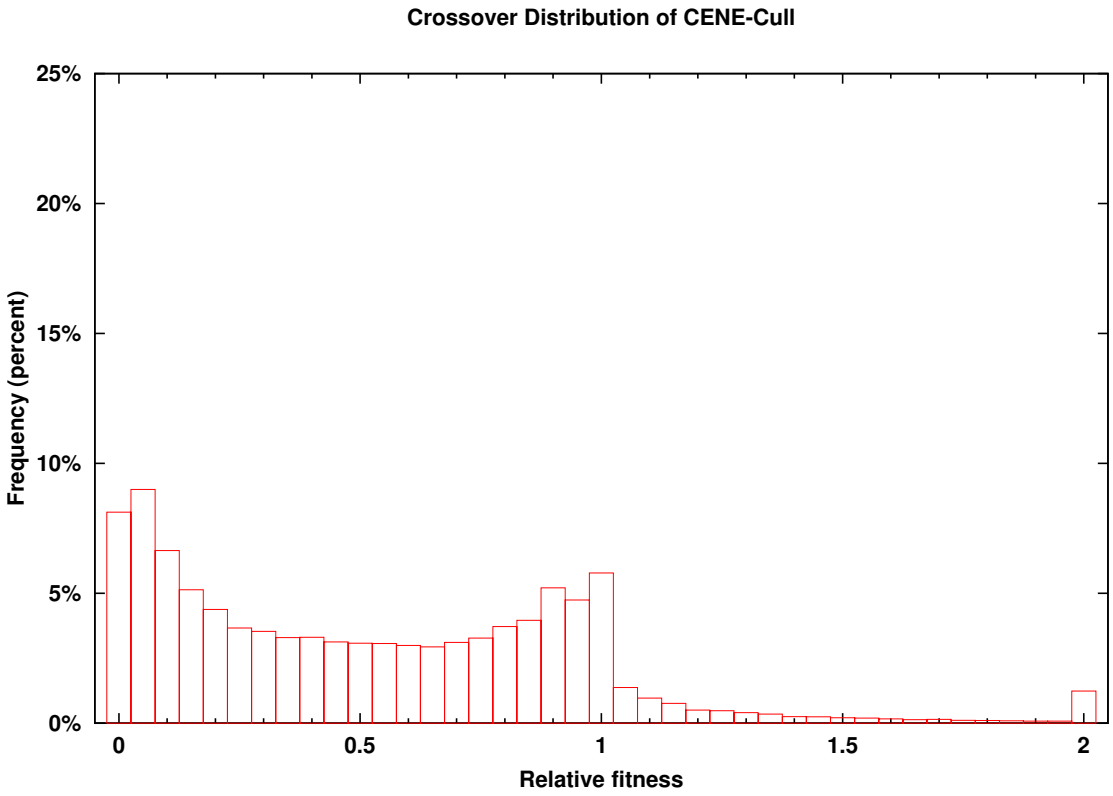


Figure 4.5: Distribution of relative fitness of crossover from CE-CULL. The abscissa scale matches Figures 4.1 and 4.3. Average fitness is 0.57. The shape nearly matches Perfect Culling, Figure 4.3, so much of the benefit of Perfect Culling should be attained.

4.5 Parameter effects

CE-CULL introduces two new parameters to neuroevolution. LitterSize is the number of offspring reproduced at each mating, and QueryLength is the number of questions asked of each offspring. It is important to know whether CE-CULL is sensitive to these parameters.

Larger litter sizes can be expected to tend to better performance because the expected maximum of any random sample becomes larger as the sample size increases. Figure 4.7 shows the performance of culling with various litter sizes. A litter size of one performs just like the standard GA, as it should. For Perfect Culling, performance continues to improve with larger litter sizes as expected, although the rate of improvement diminishes. The rate of improvement for practical culling, CE-CULL, cannot continue like Perfect Culling because it has only an estimate of fitness. The argmax of a noisy function will eventually become very noisy itself as sample size increases. Even the CPU time consumed by administration of the questionnaires will eventually become noticeable.

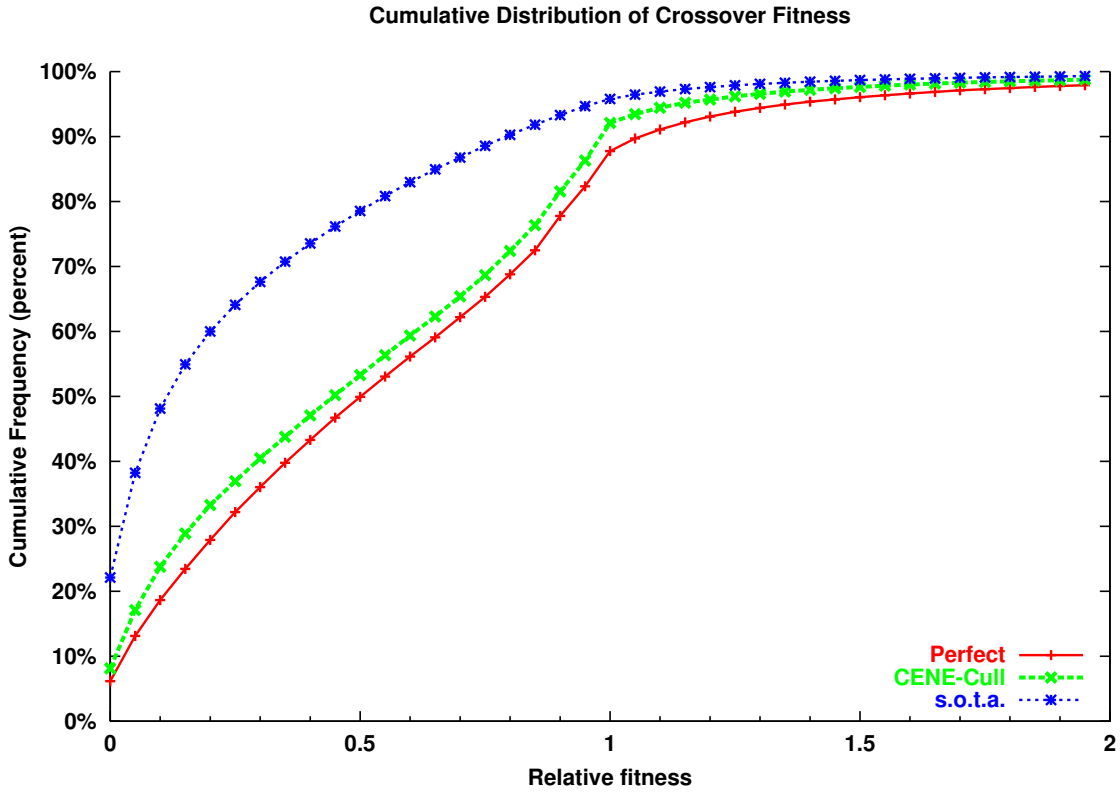


Figure 4.6: Comparison of cumulative distributions of relative fitness of crossover. CE-CULL is much closer to Perfect Culling than to standard crossover, indicating that it achieves much of the benefit of Perfect Culling.

The incremental improvement of CE-CULL is slight after 8 offspring per litter. These results suggests that litter size does not need to be optimized carefully. Anything from about 8 through 20 should perform satisfactorily.

It is clear that the number of items in the questionnaire can make a difference. An extremely short questionnaire is not reliable, because it might just happen to sample an input vector for which the benchmark and the candidate were atypically far apart. Obversely, a very large questionnaire would have the effect of requiring the offspring to be excessively similar to the imperfect benchmark, and that would stifle progress. Figure 4.8 plots performance of CE-CULL by questionnaire length. Performance varies little in the range of 4 to 16 questions. This is good because it means that the questionnaire size does not have to be carefully optimized.

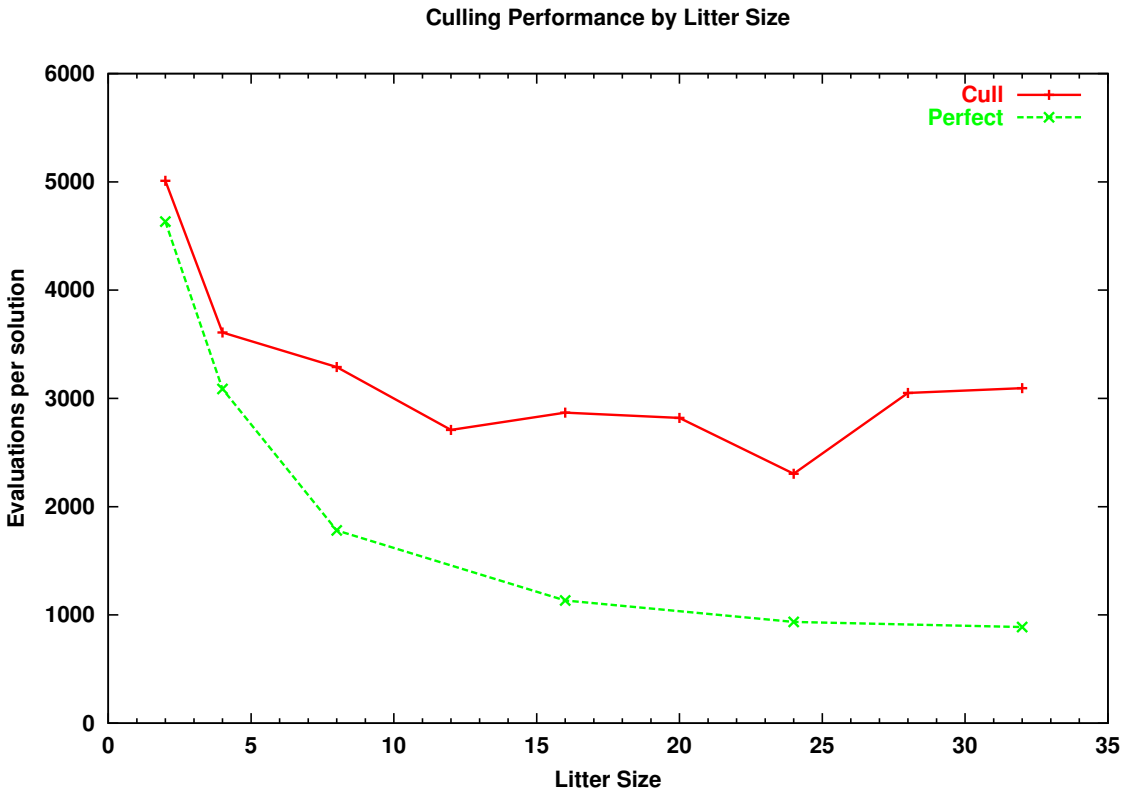


Figure 4.7: Effect of litter size on culling, plotted as evaluations per solution, with 25 runs per data point. Culling from a litter of 8 or more works well.

4.6 Conclusion

This chapter has demonstrated how culling over–large litters can be utilized to improve neuroevolution. A perfect oracle would provide a large improvement in the crossover distribution, but would cost too much. CE-CULL, uses an imperfect but increasingly competent elder to inexpensively select that offspring in a litter whose behavior most closely matches, and is therefore likely to be competent. The culling test should be thought of as only a general estimator of competence. It only needs to discriminate enough to reduce that huge lower lobe of Figure 4.1. Empirical tests verified that CE-CULL enhanced performance in the Cart-Pole domain, and demonstrated that parameter settings are not critical.

CE-CULL is based on selecting offspring that can be expected to perform better. The next chapter shifts focus from offspring to parents. It describes the development of a technique to choose mates by behavior.

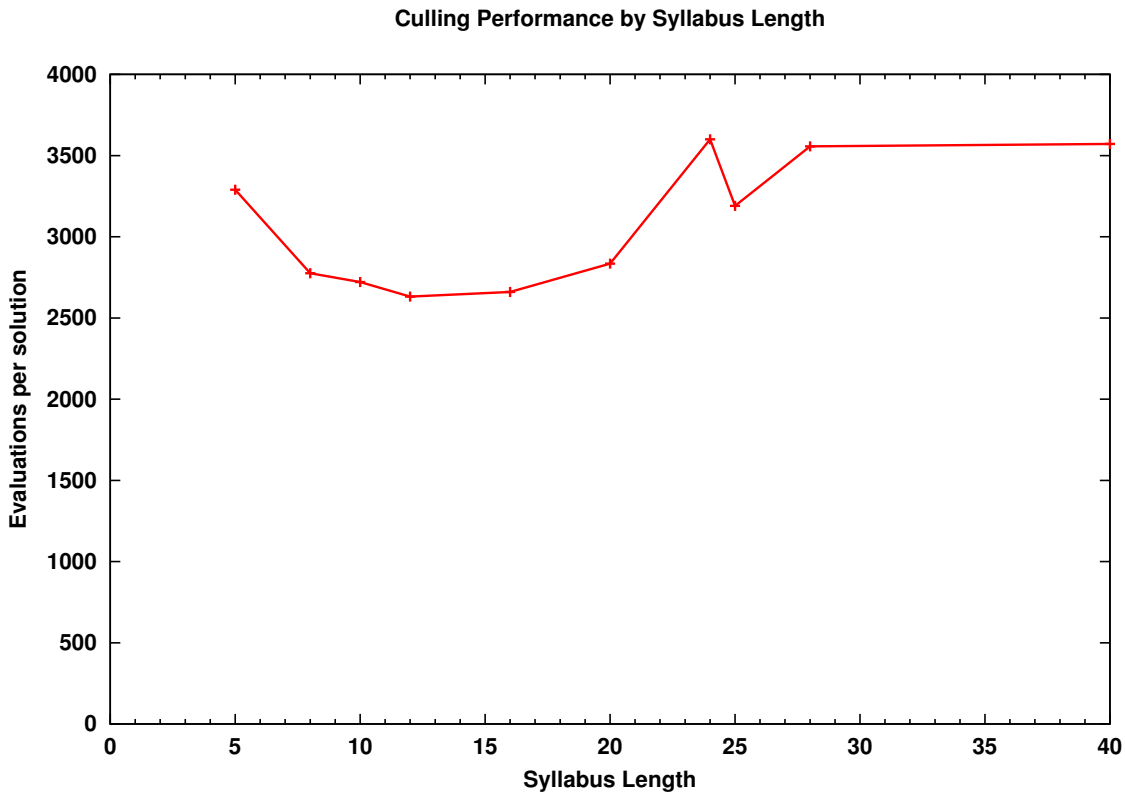


Figure 4.8: Effect of syllabus size on culling, plotted as evaluations per solution, with 25 runs per data point. Questionnaire length between 8 and 20 works well.

Chapter 5

CE-MATE: Mate Selection

The second exploitation of the behaviors of population members is CE-MATE, mate selection by complementary strength. This chapter begins with the idea that competency in a limited region of the input space is akin to a solved subproblem The vigor of hybrids in nature is a reflection of the fact that crossover can combine traits of the parents. After discussing how such complementary individuals can be recognized with non–vanishing probability, my novel CE-MATE method is detailed and potential pitfalls are identified. Empirical results show that CE-MATE yields a substantial improvement in performance over a range of parameters. Analysis of fitness distributions of mates and offspring show that CE-MATEś recognition of complementary behavior enables it to trade decreased fitness or mates for increased fitness of offspring.

5.1 Motivation

The machine learning view of why crossover is more powerful than mutation alone is that crossover can combine partial solutions, or solved subgoals—that is, the good traits of both parents. For example, given an individual that performs well in high–speed situations, it should be desirable to mate it with one that is competent in low–speed situations. It is much more likely that crossover of such complementary individuals could result in an offspring competent in both speed regimes than crossover of two individuals that both perform well in the same circumstance. How can we take advantage of this?

An individual of above average competence in only one region of the input space represents a partial solution of a subproblem. The goal of complementary mate selection is to enhance the odds of combining solved subproblems by mating pairs whose strengths cover each others' weaknesses. The standard genetic algorithm, Figure 3.3, chooses both parents by fitness, so they will often have similar behavior. When both parents have similar strengths and weaknesses the recombination of the two can be but little different. CE-MATE attempts to enhance the frequency with which mating pairs have complementary competencies, and thus the frequency with which crossover generates

good combinations. Of course, this will also increase the frequency of doubly–weak offspring, but survival of the fittest will weed them out. The genetic algorithm needs variation more than it needs a high average.

Since the fitness function is a black box, we have no *a priori* knowledge of a good decomposition of the problem at hand. That lack is not all bad, since the ultimate goal is machine learning methods that can handle problems for which there are no known good decompositions. Lacking any principled decomposition, the line of attack in this work is to rely on behavior, as represented by sampling the input space. My hypothesis is that the behavioral traits of an individual will prove to be an adequate proxy for decomposition—that behavioral strength will be correlated with solved subproblems.

Divide and conquer techniques are common in machine learning, but require knowledge of how to decompose the problem. The idea of mate selection by complementary strength is a way to encourage partial solutions to combine, assuming that behavioral differences are a satisfactory proxy for decomposition.

5.2 CE-MATE Implementation

This section describes the design issues and final implementation of CE-MATE. The first design choice is that one parent will continue to be chosen by fitness, as in the standard GA (Figure 3.3). This is so that fitness can continue to drive the evolution. CE-MATE differs from a standard genetic algorithm in how the second mate is chosen. The goal is that the behavior of second parent complement the first one's traits. This is accomplished by selecting an individual from a sample of the population whose behavior is most different from the behavior of the first parent. The sample is called a pool of suitors. With a perfect metric, the most different behavior would correlate with complementary competencies. With a feasible metric, Euclidean distance, the behavior measured as most different will at least be different, and will have a better chance of being complementary than when the behaviors are similar.

As always in this work, the concrete measure of behavior is obtained with a syllabus. Likewise, only the most simplistic syllabus construction is considered in this dissertation, a set of random input vectors. In similar vein, the population sample used as a mating pool is chosen at random. Hindsight indicates that the mating pool should have been chosen with some fitness bias, but (1) it is an interesting challenge for behavioral methods to determine if the simpler choice is workable, and (2) that hindsight arrived too late in the construction of this dissertation.

These considerations lead to the implementation shown in Figure 5.1. A list of questions is presented to the pool, as well as to the specified first parent. Each question is a vector from the input space of the problem. The behavior of each neural network is taken to be the resultant output vector. The difference in behavior is the Euclidean distance between the outputs of a suitor and the outputs of the specified first parent. The largest distance among the suitors represents the maximally unlike

```
function CE-MATE.MATESELECT () {
   input: Parent1 already selected
   input: a set of questions, Q[] the mate selection syllabus
   select a set of random individuals, W[] the lek, or pool of suitors
   clear distance accumulator array D[w]
   for each question q in Q[]
         activate Parent1 on input q
         let U[] = resulting output vector
         for each network w in W[]
               activate w on input q
               let V[w][] = resulting output vector
               compute H = V[w][] - U[] Euclidean distance
               add H into D[w]

   select that individual with the least total distance in D[w],
         as the second parent
   }
```

Figure 5.1: CE-MATE implementation.

behavior, so that is the mate chosen by CE-MATE.

SOTA chooses the second parent as the one of highest fitness in the mating pool. CE-MATE ignores the fitness of the second parent, so they will be weaker than those chosen by SOTA. The success of CE-MATE hinges on complementary behavior producing offspring enough better than their parents to overcome this handicap. An empirical comparison of performance will show that it does so.

5.3 Empirical analysis

As promised in Section 3.1, CE-MATE was tested on the Acrobot swing–up task of Section 3.1.2. In this section, empirical results first compare performance of CE-MATE and SOTA. Then examination of the fitness distribution of mates and offspring during these tests illuminate the worth of the trade–offs in CE-MATE.

5.3.1 CE-MATE performance

Performance results show that CE-MATE yields a substantial improvement in speed over plain SOTA. Table 5.1 reports the average number of fitness evaluations required to find a solution for CE-MATE with different syllabus lengths, versus SOTA. Longer questionnaires tend to result in better perfor-

Evals/soln	÷SOTA	Method	# queries	Success%
2,258	0.60	CE-MATE	40	98%
2,386	0.63	CE-MATE	10	98%
2,400	0.64	CE-MATE	20	98%
2,649	0.70	CE-MATE	5	98%
3,769	1.00	SOTA	–	84%

Table 5.1: Comparison of evolution speed in CE-MATE and SOTA. *Evals/soln* is the average number of evaluations required to generate each successful solution. The ÷SOTA column gives ratio of score to SOTA. *Success* is the percentage of runs that found a solution. *Queries* is the length of the syllabus. Each row of the table is an average of 50 runs. CE-MATE performs well, particularly when using 10 or more questions.

Score	Method	# queries	40	10	20	5	SOTA
2,258.4	Mate	40					
2,386.2	Mate	10	39.3				
2,400.3	Mate	20	38.0	47.5			
2,649.7	Mate	5	20.3	27.8	29.7		
3,769.5	SOTA	–	2.8	3.8	3.8	7.8	

Table 5.2: Probability of equal means (in %) for each pair of CE-MATE experiments. The differences between CE-MATE and SOTA are significant at the 95% level when the questionnaire length is ≥ 10.

mance, but even when only five answers are compared, CE-MATE is still 30% faster than SOTA.

To determine whether these speed differences are significant, a 10,000–fold resampling test was performed (see Section 3.3). The results are in Table 5.2. The bottom row shows less than a 4% chance that the differences between CE-MATE and SOTA observed in Table 5.1 are due to chance when the questionnaire length is ≥ 10.

5.3.2 Fitness of mates

Because it ignores fitness of the second parent, CE-MATE will necessarily select mates that are less fit than those chosen by SOTA. This subsection quantifies the result of that prediction. Figure 5.2 compares the fitness distribution of second parents for the best CE-MATE variant (a questionnaire length of 40) versus SOTA during the same 50 runs reported in Table 5.1. To normalize the graph a fitness ratio is plotted. Fitness relative to the then–current population champion is plotted, since that ratio completely determines what mates will be chosen by the binary tournament of SOTA. A reverse cumulative distribution is plotted so that higher lines represent better distributions.

The standard GA is choosing the second parent based solely on fitness, while CE-MATE ignores its fitness. Indeed, since the first parent is of above–average fitness, and CE-MATE deliberately selects for the behavior most unlike the first, there is an implicit bias for below average fitness in

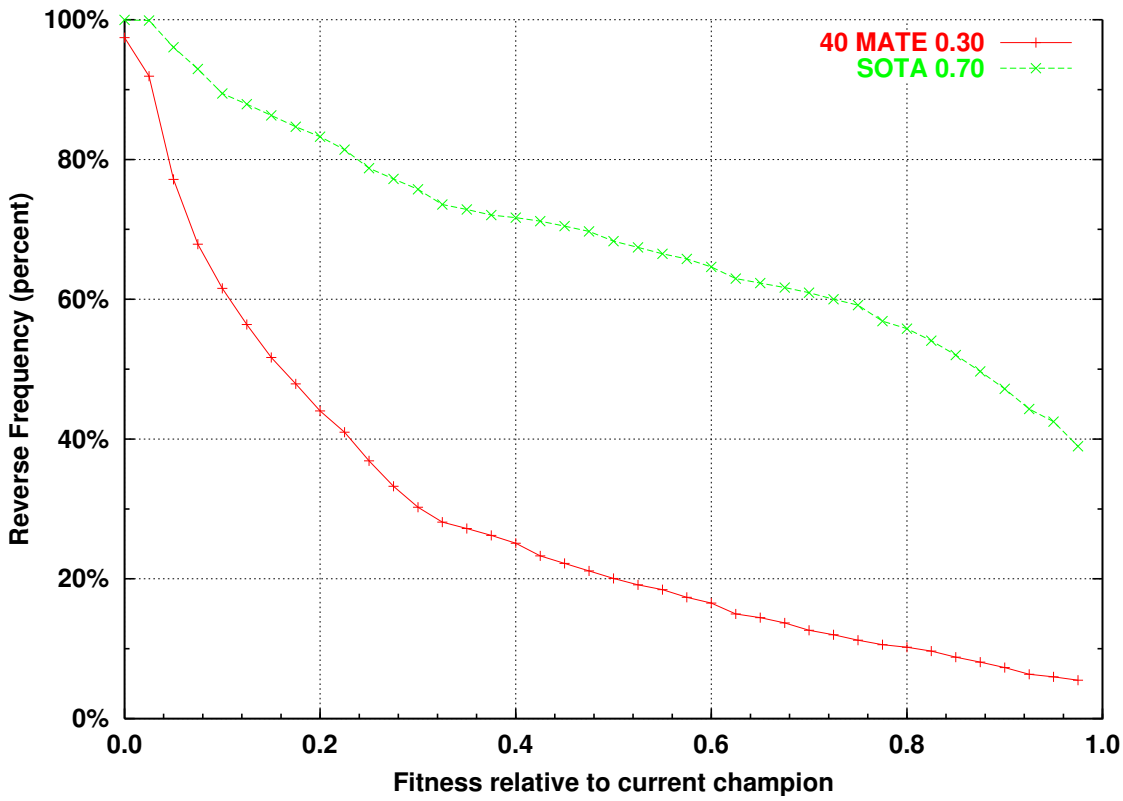


Figure 5.2: Distributions of relative fitness of second mates during the 50 runs summarized in Table 5.1 for CE-MATE-40 and SOTA. Fitness of current champion = 1.0. CE-MATE forgoes a lot of fitness.

the second parent. The lower curve shows that about 45% of the mates chosen by CE-MATE have relative fitness above 0.2, whereas SOTA picks mates at least that fit over 80% of the time.

The surprise contained in this chart is how much fitness is sacrificed. Nonetheless, CE-MATE did evolve solutions faster, as hypothesized above.

5.3.3 Fitness of offspring

In order to determine how CE-MATE can perform so well, the fitness distribution of offspring is compared in Figure 5.3. Again, it contains all the offspring generated by the runs summarized in Table 5.1. To normalize the graph a fitness ratio is plotted, but in this chart the ratio is fitness relative to the average of the parents, because that is what determines the rate at which the evolution can proceed.

Less than 7% of the offspring generated by SOTA have a fitness as great as their parents' average, whereas CE-MATE generates offspring better than the parental average twice as often (over 15% of the time). The advantage of CE-MATE gets more extreme for very high fitness ratios. At 1.5

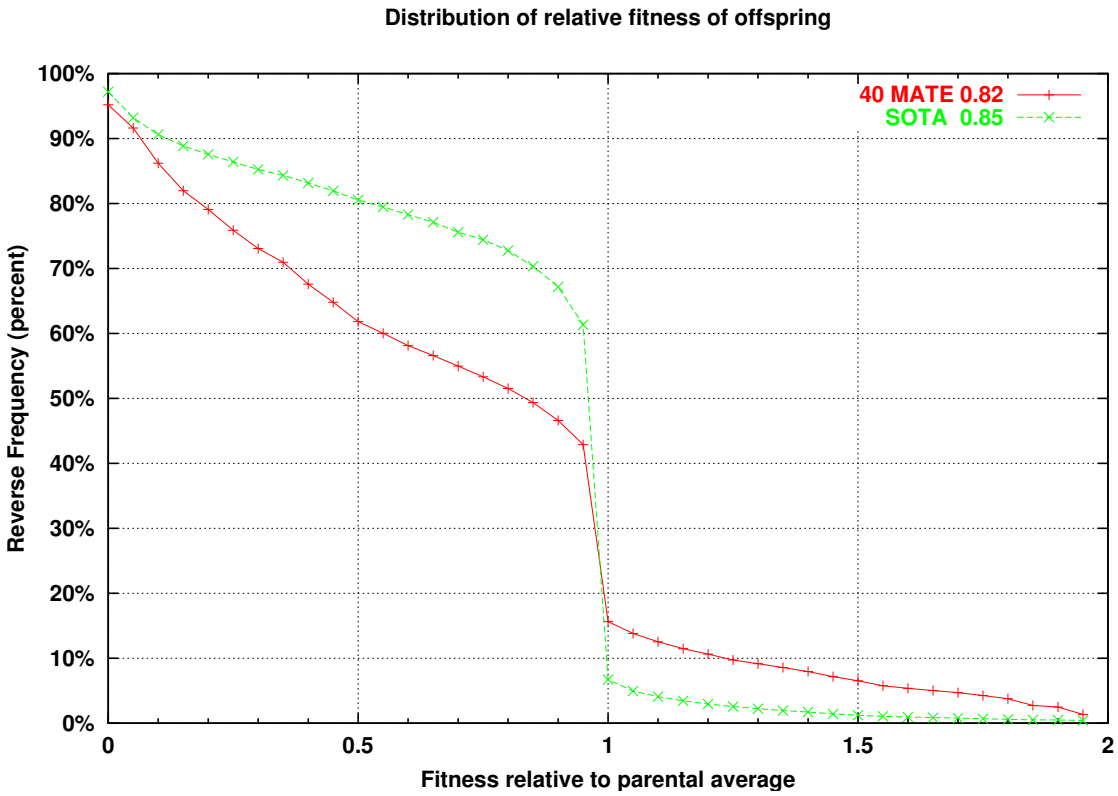


Figure 5.3: Distribution of fitness of offspring relative to parental average during the 50 runs summarized in Table 5.1 for CE-MATE–40 and SOTA. Average of parents = 1.0. The good performance of CE-MATE–40 is explained by the fact that it has much more area under the curve above 1.0 than does SOTA.

times parental average SOTA is below 1.3%, but CE-MATE generates fitness that high over 6% of the time—four times as often.

Combined with the evidence in Figure 5.2, which shows that the second parents are not themselves very good, this performance data suggest that CE-MATE must be making better combinations available to crossover.

5.4 Conclusion

This chapter has shown how intelligent selection of mates based on complementary behavior can be utilized to improve neuroevolution. The CE-MATE method selects mates of complementary behavior, instead of strictly by fitness as the standard GA does. That the mates chosen are necessarily of lower fitness than the mates chosen by SOTA is confirmed by the empirical distribution of fitnesses of second mates. CE-MATE sacrifices performance of second parents, but it evolves solutions faster

than the standard GA. Thus, the parental weakness is more than compensated by the improvement in offspring fitness, validating my assumption of the importance of hybridization. This non–obvious trade–off of complementary behavior for fitness is surprisingly good, given the measured weakness of the mates.

A secondary effect of CE-MATE is an improvement in population diversity, since the different phenotypes of the chosen mates tend to correlate with different genotypes. This result will be explicitly confirmed later in Section 8.3.4.

All my techniques involve relatively simple modifications to SOTA. CE-CULL inserted a choice between alternative offspring. CE-MATE is based on manipulating mate preference. The next chapter describes the development of a technique to manipulate the third type of individual selection in the genetic algorithm: Is there a better way to choose which individuals should be discarded to make room for new offspring?

Chapter 6

CE-DIVERSE: Diversity Enhancement

CE-DIVERSE is the third exploitation of population behavior devised for this dissertation. By enhancing phenotypic diversity premature convergence of the population is avoided, so progress can continue. First, I precisely define genetic diversity, describe its impact on evolution, and characterize traditional approaches to diversity management. Then phenotypic diversity and how it is utilized in this approach is described. Next, the CE-DIVERSE design and potential pitfalls are addressed, resulting in the concrete implementation. Finally, empirical results show that behavioral diversity enhancement yields a substantial improvement in performance, and its operation is analyzed to confirm its mechanism.

6.1 Genotypic Diversity

Diversity can help keep progress going in neuroevolution. In this section the collapse of diversity, called convergence, is first described, and then the formula for the numeric measure of diversity is defined. Finally, approaches to diversity promotion are categorized.

As evolution proceeds selection pressure causes alleles associated with high fitness increase in frequency in the population. When most or all of the population has the same allele (value) for a locus, the population is said to be *converged* at that locus. When the entire genome is converged change comes only from random mutation, and is unlikely to make any more progress. The most likely state for a converged population is very near some peak in the fitness landscape. If that peak is the highest one, the *global* optimum, the algorithm has succeeded (at least for a stationary problem, where the optimum does not move). Peaks with fitness values less than the global optimum are called *local* optima. Convergence to a local optimum is called *premature* convergence. In the language of machine learning, premature convergence is the consequence of too much exploitation.

Darwinian evolution operates by selection of variations. Thus, the members of the population must vary in fitness in order for the genetic algorithm to make progress. Crossover alone can never change a fully converged locus, so crossover is ineffective in a highly converged population.

With only mutation remaining operational, a genetic algorithm becomes a hill–climber, and is likely to be less efficient than algorithms specifically designed for hill–climbing. Progress is slow when the population is highly converged.

An intuitive view of the effect of convergence is that when the population is 90% converged, essentially only $\frac{1}{10}$ of the population is performing useful work, but the cost includes evaluating and reproducing the entire population, a highly inefficient situation.

The degree of convergence in a population is measured by the diversity of the alleles present in the population (see for example, Eshelman and Schaffer (1991) and Moriarty (1997, Sec. 4.2.3)). For the binary chromosomes in this work, the diversity of each locus is simply the proportion of zeroes and ones at that locus of the genotype in all the members of the population. Specifically, the diversity of each bit position, b is computed as:

$$\Phi_b = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} X_{i,b} \otimes X_{j,b}$$

where n is the population size, X_i is the bitstring representing the i-th individual, $X_{i,b}$ is the b-th bit of X_i, and $\otimes$ is the *exclusive or* operation. The leading factor of $2/(n(n-1))$ normalizes for the number of terms summed, such that Φ_b is the probability that two randomly chosen individuals would have the same value of bit b. For a random population all Φ_b = 0.5; i.e., it is 50/50 whether any two bits differ.

The average population diversity is the average over all bit positions,

$$\Phi = \frac{1}{l} \sum_{b=1}^{l} \Phi_b$$

where l is the number of bits in each individual. Since $H(i,j)$, the *Hamming distance* between two individuals i and j is

$$H(i,j) = \sum_{b=1}^{l} X_{i,b} \otimes X_{j,b},$$

we have

$$\Phi = \frac{2}{n(n-1)l} \sum_{i=1}^{n} \sum_{j=i+1}^{n} H(i,j)$$

Thus, Φ is the average Hamming distance between every two chromosomes.

Since lack of diversity means convergence means no further progress, several diversity enhancement techniques have been developed:

- Monitor the diversity level in the population, and take some action when it seems too low. One crude but sometimes effective method is simply to increase the mutation rate whenever the population diversity drops below some threshold (Cobb and Grefenstette 1993). Another example is to restart when the population converges.

- Island models like ESP (Gomez and Miikkulainen 1999) increase diversity by periodically injecting an individual from a different population.

- Take some action in the reproductive mechanism that tends to prevent diversity from ever becoming excessively low. The neuron populations of SANE (Moriarty and Miikkulainen 1996b) accomplish this because they are forced to cooperate and cannot become too similar. In another example Eshelman and Schaffer (1991) use a distance metric between genotypes and prohibit overly similar individuals from mating. Craighurst and Martin (1995) record kinship relations to enforce a prohibition against mating close kin. Crowding (De Jong 1975) and niching (Mahfoud 1995a,b) are also techniques of this class.

CE-DIVERSE is an instance of the second approach, because it does not directly manipulate the genome. Instead, by focusing on culture CE-DIVERSE obtains the advantages of genotypic diversity by enhancing phenotypic diversity, as will be shown below.

6.2 Phenotypic Diversity

The behavior of population members can be monitored to guard against loss of diversity without looking at their genomes. CE-DIVERSE enhances phenotypic diversity by increasing the reproductive odds of those individuals whose behaviors are least typical. To accomplish that objective requires an operational definition of typical behavior.

Two schemes, already mentioned in the previous section, crowding (De Jong 1975) and niching (Mahfoud 1995a,b), can also be based on phenotypic similarity, but in both of them the definition of phenotypic distance is based solely on fitness values. Thus, the entire population is represented by points on a single line, which is too crude for CE-DIVERSE. As we have seen in CE-MATE, a suitable characterization of behavior of a network is a sample of its input–output mapping. To emphasize the parallels with other CENE methods, this sample is called a diversity questionnaire, or syllabus.

In CE-DIVERSE the networks' responses to the questionnaire are combined to produce a measurement of diversity by regarding the *average* response of a crowd as representing *typical* response. Thus, the design goal of CE-DIVERSE is to increase the chance that the individuals with the most typical behavior will be discarded.

The geometric average of several output vectors, the centroid, is an appropriate measure of typicality, and is simple to compute. That individual with the least Euclidean distance from the centroid is regarded as the most typical. Thus that individual demonstrates the behavior CE-DIVERSE seeks to suppress. The chosen individual is deleted from the population to make room for a new offspring.

SOTA, the baseline genetic algorithm in this work, uses steady state breeding, where creating one new individual requires that one be discarded. As shown in Figure 3.4 on page 25, SOTA

```
function CE-DIVERSE.KILLSELECT () {
   input: a set of questions, Q[]
   select a set of individuals, W[]
   clear distance accumulator array D[w]
   for each question q in Q[]
        for each network w in W[]
             activate w on input q
             let V[w][] = resulting output vector
        compute K[]= centroid of V[][] average over all w
        for each network w in W[]
             compute H[w]= distance of V[w] from K
             add H into D
   ---------------------------------
   select those two individuals with the least total distance in D[w],
        let them be w_0 and w_1, the two most conventional
   kill the least fit of w_0 and w_1
}
```

Figure 6.1:

discards the least fit individual found in a 4–tournament. In CE-DIVERSE that tournament on fitness is replaced with an n-tournament ($n \geq 3$) on behavior, as detailed in the next section.

6.3 CE-DIVERSE implementation

In this section the design of CE-DIVERSE is presented. First, it was found that a diversity enhancement method that ignored fitness performed poorly. Thus, the actual CE-DIVERSE implementation is biased toward weaker individuals.

In an initial implementation of phenotypic diversity preservation (not shown), fitness was ignored when selecting the decedent. A sample chosen without regard to fitness will contain individuals with a range of fitnesses. If the population has converged to occupy a single fitness peak, individuals of higher fitness are nearer the peak, and so will behave more similarly than those of lower fitness, who are likely to be scattered around the base. Thus in the random sample of individuals, the high fitness individuals tend to resemble each other while the low ones do not, so that when they are all averaged to form the centroid the high fitness individuals tend to reinforce each other while those of low fitness tend to cancel each other. The centroid is therefore strongly influenced by individuals of higher fitness, and the closest individual may even be the most highly fit individual of the sample. This tendency toward discarding the best individual is detrimental to performance.

Preliminary experiments with the simple mechanism that ignored fitness confirmed that

argument. It does indeed preserve diversity, but increased diversity, by itself, does not increase performance. For instance, diversity can be enhanced by merely inserting random individuals into the population. However, inserting random individuals does not make progress toward increased fitness.

The actual implementation of CE-DIVERSE, shown in Figure 6.1, takes account of fitness in a simple but effective way. The fitnesses of the two individuals in the sample with the least distance are compared. The least fit of these two is eliminated from the population, which is a non–random 2–tournament.

CE-DIVERSE selects first by distance then by fitness. The opposite arrangement seems plausible, that is, select first by fitness, then by distance. For example, initially perform three individual two–tournaments on reverse fitness to obtain three weak individuals, and then discard the individual closest to the centroid. Because lower fitness individuals tend to be more dispersed in weight space, their response surfaces do not represent behavior that is actually typical. The centroid of a sample, therefore, is more likely to merely reflect wild disagreement, rather than representing a consensus. Thus, the individual who happens to be the closest to the centroid is not so much typical as just unlucky, and choosing it to discard is not significantly more effective than the binary tournament of plan SOTA.

There are at least two potential problems with the CE-DIVERSE approach to behavioral diversity enhancement. First, if nearby genotypes give rise to widely scattered phenotypes, behavioral diversity could appear high while the genotypic diversity was actually plummeting. Generally, phenotypic distance is correlated with genotypic distance, so this situation is unlikely. Second, since many genotypes map into indistinguishable phenotypes, phenotypic convergence could occur while the population was still genetically diverse. Such genetic diversity could be enough by itself to permit escape from the local minimum. This is not likely because selection pressure operates on phenotypes, so selection pressure is effectively disabled when phenotypic convergence occurs and the population performs a slow random walk, much as it does when it is genotypically converged.

The CE-DIVERSE implementation is supposed to increase both performance and diversity over SOTA. Empirical results in the next section verify that assumption.

6.4 CE-DIVERSE Results

This section presents empirical results comparing CE-DIVERSE and SOTA, in performance. The operation of the method is analyzed by further comparisons of diversity over time, and the fitness distribution of decedents.

6.4.1 CE-DIVERSE performance

Empirical results show that CE-DIVERSE yields a substantial improvement in performance over SOTA. The task is a modification of the Acrobot domain described in Section 3.1.2. The standard

Acrobot problem proved too simple to reveal definite differences between the methods. That is, SOTA is effective enough on the standard Acrobot task that enhanced diversity contributes little to performance. In the modified Acrobot problem in this chapter applied force is reduced to only 90% of its normal value in the previous chapter, which makes the swing–up much more difficult to achieve. The difficulty of this weak Acrobot task is seen by comparing SOTA results in Table 6.1 and Table 5.1. SOTA expends over three times the effort on this modified problem as on the standard Acrobot problem (12,998 vs. 3769 evaluations per solution).

Evals/soln	**÷SOTA**	**Method**	**Success%**
9,080	0.70	CE-DIVERSE	66%
12,998	1.00	SOTA	51%

Table 6.1: Comparison of the performance of CE-DIVERSE and SOTAon the more difficult Acrobot swing–up task. *Eval/Soln* is the number of evaluations used in all runs divided by the number of successful runs. *Success%* is the percentage of runs that found a solution. Each row of the table is an average of 200 runs. CE-DIVERSE is 30% faster than SOTA.

Table 6.1 compares the speed of evolution of CE-DIVERSE and SOTAon the modified problem. The main result is that CE-DIVERSE used 30% fewer evaluations than SOTA. This difference is statistically significant ($p < 0.006$).

6.4.2 Diversity over time

The performance of CE-DIVERSE is good, but my hypothesis was also that it would be achieved by enhancing diversity, so that needs to be analyzed. Figure 6.2 compares average population diversity over time during the experiments reported in Table 6.1 above. The curves show that CE-DIVERSE maintains more diverse populations than SOTA. Diversity in SOTA drops below 0.2 before 2000 evaluations, but CE-DIVERSE stays above that level three times as long. CE-DIVERSE still has a respectable diversity of 0.158 after 10,000 evaluations, whereas SOTA has dropped below that level very early, at only 2200 evaluations.

Together, these results show that behavioral diversity preservation, CE-DIVERSE, operates as desired: (1) Table 6.1 shows that CE-DIVERSE results in faster evolution, and (2) Figure 6.2 shows that it enhances genotypic diversity.

6.4.3 Fitness of decedents

Another useful analysis results from asking which individuals get discarded. Recall from Figure 6.1 that CE-DIVERSE finishes with a binary tournament on reverse fitness of two individuals. Thus, if the distances in behaviors had no systematic effect, CE-DIVERSE would have the same effect as a binary tournament, so CE-DIVERSE would discard individuals of higher average fitness than SOTA, which has a 4–tournament. Figure 6.3 shows the distribution of relative fitness of all individuals

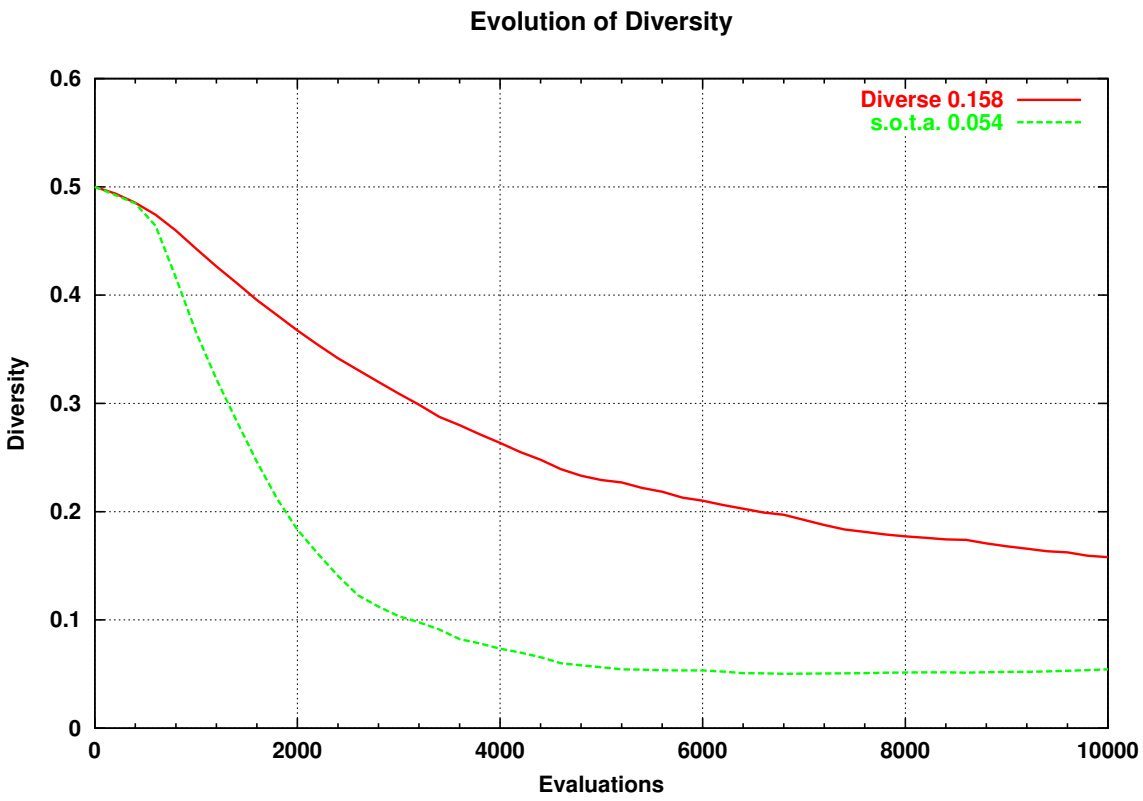


Figure 6.2: Comparison of population diversity by generation of CE-DIVERSE versus SOTA during the same 200 runs of Table 6.1. Numbers in legend are average final diversity. CE-DIVERSE maintains much more diversity than SOTA.

selected to be replaced during the same set of experiments summarized in Table 6.1. Fitness relative to the current population champion is shown, because that ratio is an effective proxy for fitness rank in the population, and tournament selection depends solely on rank.

The plots appear to show that individuals with a fitness ratio of 1.0 are being discarded, but that is just an artifact of round–off. The reverse fitness tournaments used herein can never select the actual champion. Nonetheless, SOTA is seen to discard many highly fit individuals. The reason is that SOTA failed to achieve the goal in almost half of its simulations. Failure to achieve the goal is usually associated with premature convergence, which means that later generations come to contain individuals of almost identical fitness. Thus, in a converged population the relative fitness of discarded individuals can be close to 1.0. The champion in a converged population is likely to have only slight random differences from the others.

Despite a smaller tournament size, CE-DIVERSE kills individuals of lower fitness than the standard GA does. This is especially evident in the peak just above zero, where CE-DIVERSE selects almost 25% of its decedents, whereas SOTA has less than 10% there. Thus, CE-DIVERSE does not

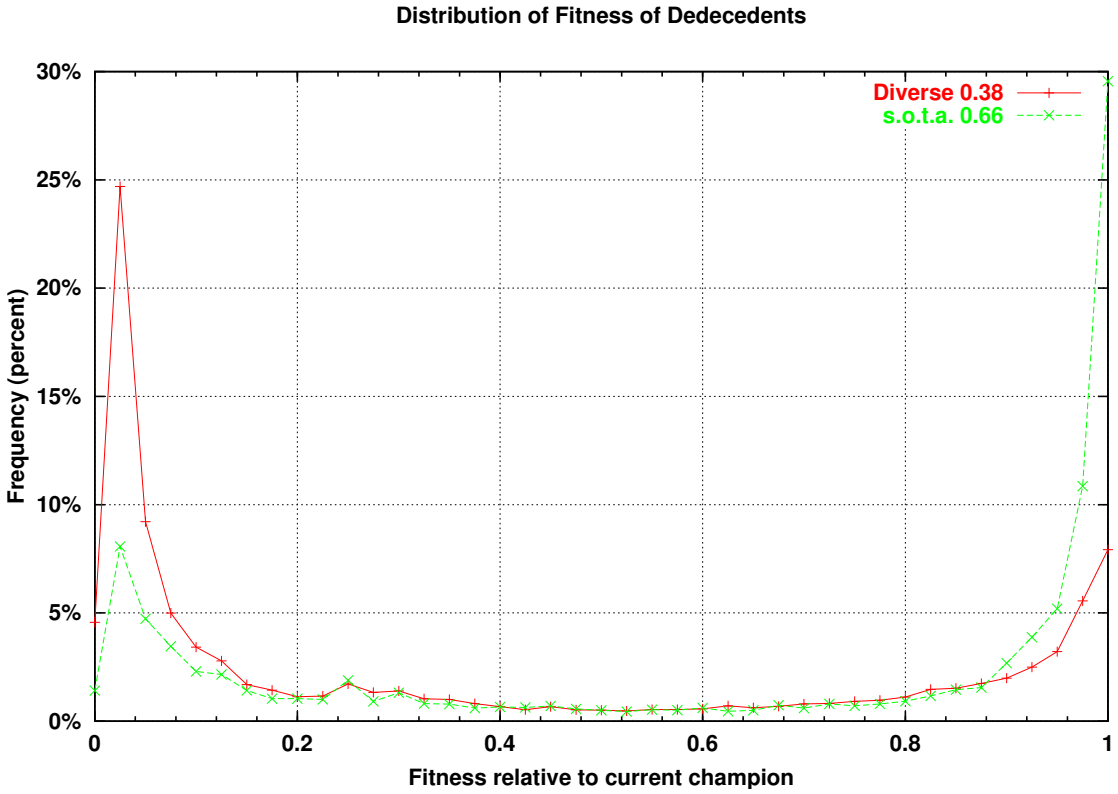


Figure 6.3: Comparison of distribution of relative fitness of discarded individuals. Numbers in legend indicate average of the distribution. The individuals discarded by CE-DIVERSE are of lesser fitness than those chosen by SOTA, so it is not behaving like a binary tournament.

behave like a binary tournament, and the distances in behaviors are having an effect.

The distributions are easier to compare when integrated, as in Figure 6.4 which shows the same data plotted as a cumulative distribution function. Higher curves are better because they mean that more of the discarded individuals are to the left, at lower fitness. For example, the ordinate for fitness = 0.6 shows that SOTA had less than 40% of its discards below that, which means that over 60% of the individuals it discarded were better than 0.6. In contrast, CE-DIVERSE discards individuals below 0.6 fitness fully 65% of the time.

A large difference exists in the fitness of discarded individuals. One possible explanation is that there are never any high–scoring individuals when CE-DIVERSE is used. However, that is not the case, because what is plotted in Figure 6.4 is *relative* fitness, where the current population champion is equal to one. Thus, the difference produced by CE-DIVERSE is real, and the chart indicates that it is due to a plausible approach: kill only the very weakest.

SOTA could be be modified to kill individuals of lesser fitness than it does now, merely by increasing the death tournament size. However, larger tournaments increase selection pressure

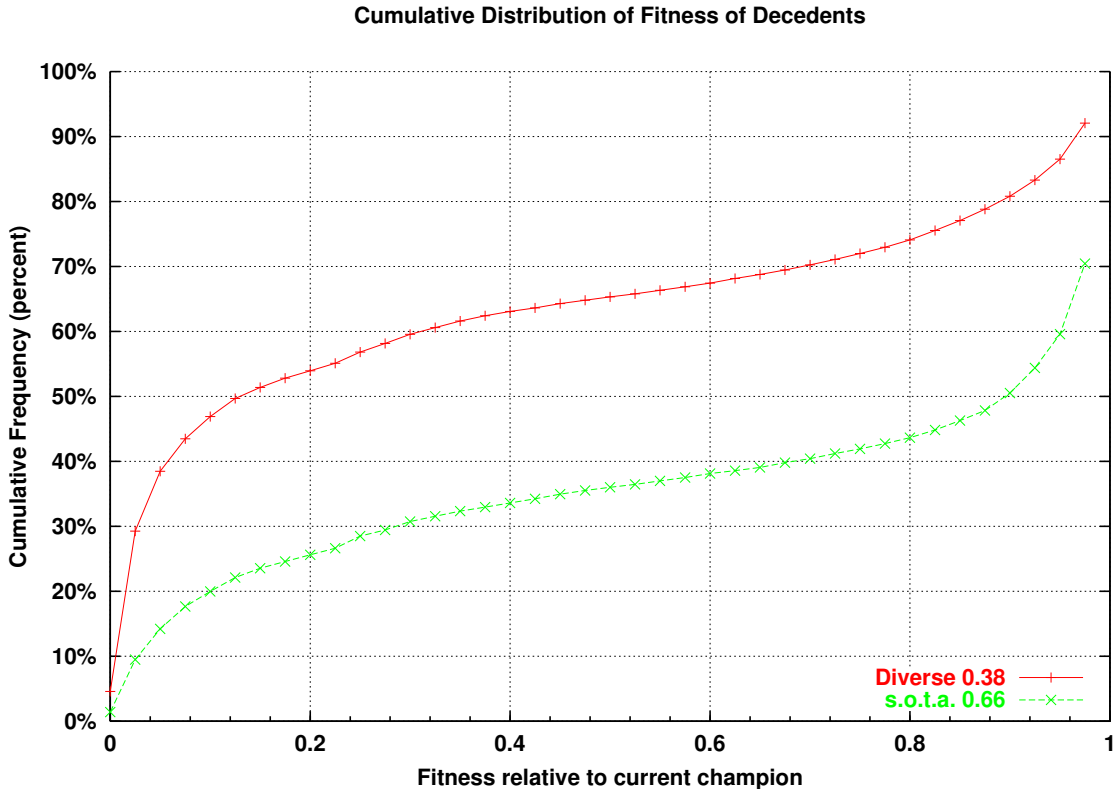


Figure 6.4: Comparison of cumulative distribution of relative fitness of discarded individuals for CE-DIVERSE versus SOTA. CE-DIVERSE discards individuals below 0.2 over twice as often as SOTA.

(Whitley 1989) and cause premature convergence. As shown by Figure 6.2, CE-DIVERSE does not suffer from premature convergence so it is not the equivalent of SOTA with a large tournament size. Thus, my twin design goals are met: kill the weakest while preserving diversity.

Comparing the success percentages in Table 6.1 show that SOTA finds a solution less than 80% as often as CE-DIVERSE(51 : 66). The diversity plot in Figure 6.2 shows that SOTA's diversity falls much faster than CE-DIVERSE. Thus, the diversity enhancement of CE-DIVERSE leads to substantially less premature convergence than SOTA, as was desired.

6.5 Conclusion

This chapter has shown that behavioral diversity can be utilized to improve neuroevolution. CE-DIVERSE, selects atypical but weak decedents and preserves diversity, and both portions of the strategy are necessary. Merely selecting very weak decedents leads to premature convergence. Discarding the most typical without regard to fitness leads to poor performance. CE-DIVERSE preserves diversity very well, and it is able to do so while discarding weaker individuals than those chosen by

SOTA, leading to good performance.

Each of the three novel techniques developed so far—CE-CULL, CE-MATE, and CE-DIVERSE—operate by examining the behavior of population members to select those with certain characteristics. In contrast, the technique constructed in the next chapter goes on to actually *modify* the behavior of evolved neural networks for even more powerful results.

Chapter 7

CE-TEACH: Educating Offspring

The three cultural techniques developed in previous chapters operate by examining the behavior of population members to select those with certain characteristics. In contrast, CE-TEACH, the technique presented in this chapter, actually *modifies* the behavior of the neural networks evolved. New offspring are taught to respond somewhat like an elder before they face a fitness examination. The inspiration comes from the natural world, where parental education of offspring is widespread.

After expanding on the rationale behind teaching, the implementation plan is presented and design issues resolved. Next, the CE-TEACH method is empirically determined to be effective on the Inverted Pendulum task. Finally, the surprising pattern of fitness before learning in CE-TEACH is found to explain how teaching widens the basins of attraction for the underlying GA.

7.1 Teaching rationale

As was reviewed in Section 2.4, culture is a powerful force in natural evolution, and so several researchers have been motivated to exploit culture in computational evolution as well. The CE-TEACH approach is unique because it focuses on behavior as the embodiment of culture, thus excluding artifacts. The behavioral focus also implies that a convenient source of teaching targets can be obtained from existing networks. The issue is how to find targets that will accelerate progress.

CE-TEACH constructs a teacher from the common knowledge contained in the current population. Such teachers are imperfect, but continually improve over the course of evolution. The important conclusion demonstrated in this chapter is that such imperfections are more than compensated for by the increased speed of supervised learning.

To discuss the rest of the issues involved in making this approach work, let us first make it concrete by proposing an implementation.

```
function CE-TEACH.BREED ( Parent1, Parent2 ) {
   input: Parent1 and Parent2
   input: J = a new offspring, not yet evaluated
   input: Q[] = a set of questions // the teaching syllabus
   input: T = an individual of the population // the teacher
   generate new offspring C = SOTA.BREED( Parent1, Parent2)
   for each question q in Q[]
         activate teacher T on input q
         let TV[q][] = resulting output vector
         activate student C on input q
         let CV[q][] = resulting output vector
         perform backprop on C with TV[q][] as the target vector
   return the now trained C
}
```

Figure 7.1: CE-TEACH implementation. A new offspring is generated by standard breeding (Figure 3.4), and trained via backpropagation to approximate target vectors produced by the teacher.

7.2 CE-TEACH implementation

The basic plan is to train offspring with backpropagation to emulate a teacher's output vectors over a set of training examples, before the offspring is subjected to the complete, expensive fitness evaluation. A concrete proposal for implementing CE-TEACH shown in Figure 7.1 draws attention to three issues: (1) how is the teacher specified, (2) what questions are to be taught, and (3) how much backpropagation is to be used.

The set of training examples is called the syllabus. The teacher and syllabus of CE-TEACH correspond to the benchmark and questionnaire of CE-CULL. As with the methods discussed earlier, the syllabus and the teacher could be chosen in various ways, and again only the most straightforward choice is analyzed in this work. The teacher is chosen like the benchmark of CE-CULL: the current population champion. Possible development of alternative methods of composing teachers is discussed in Section 9.7.

Just as in earlier CENE methods, the set of training examples consists of random input vectors. The target vector that drives backpropagation is the output vector produced by the teacher when activated on a given input vector. Whereas in CE-CULL errors between several candidates were compared, and in CE-MATE and CE-DIVERSE outputs of several candidates were compared, in CE-TEACH error is used to adjust the weights of an offspring.

The third issue remains: how much backpropagation should be used? Excessive training could lead to a student becoming a near copy of the teacher. That would stall any progress via evolution, so the amount of training must be moderate, and therefore the method is restricted to

only incremental improvement over an untrained individual. The design choice in CE-TEACH is that the amount of training is limited to a single pass over the syllabus, that is, each training case is presented and backpropagated only once. The effect of varying the length of the syllabus will be tested, however, and the results will be shown below in Figure 7.3.

To understand why CE-TEACH does not cause students to become overly similar to their teachers, it is useful to discuss the effect of backpropagation on the student's response surface. Since backpropagation is based on error surfaces in weight space the difference between these two spaces must first be clarified. For a network containing m weights[1], the error surface is plotted in a $m + 1$ dimensional space where the first m dimensions correspond to the weights. For every combination of weight values, the error value is plotted as height above the m dimensional subspace, forming a hyper–surface. In fact, backpropagation performs a gradient descent on that error surface (Hertz et al. 1991).

The response surface is similar to the error surface only in the mechanics of plotting. The response surface of a single output of some neural network is just a representation of the input–output mapping of that network. If there are n inputs and one output, then the result can be plotted as a surface in a space of dimensionality $n + 1$. The output value is plotted as a point above the input hyper–plane. Figure 7.2 shows example response curves for the case of a network with but a single input (1–dimensional stimulus, and no bias unit). In the case of multiple outputs, each output will have a separate response surface, each of which will individually behave as indicated.

Backpropagation does drive the total error of an offspring network toward zero. In CE-TEACH that error is measured at only a finite number of sample points, so backprop is not reacting to the teacher's entire response surface. Thus, CE-TEACH could run backpropagation until the error becomes very small, and the student's response surface would still not be identical to that of the teacher.

Figure 7.2 illustrates this finite sample effect schematically. Assume these two curves describe a student and a teacher in CE-TEACH, and that there are only two items in the syllabus, marked here as A and B. The arrows show the error signals resulting from the mismatch of student and target. Each backprop iteration will tend to try to push the student's response surface toward the teacher's, but only at one input point. Since there are only a limited number of sample points, small residual error does not imply that the response surfaces are similar anywhere else: they may touch or cross near the sample points but flutter disparately elsewhere. With higher dimensional input vectors, which are more typical, there is even more room for the two surfaces to diverge. Only if the number of samples approaches the number of weights in the networks can the response surfaces be perfectly aligned. In CE-TEACH the number of samples is only dozens, so the finite sample effect is pronounced.

A syllabus construction based on accurate knowledge of the teacher's response surface

[1] The error surface concept is actually valid for any machine learning method that is completely characterized by the values of its adjustable parameters. Just substitute such parameters for the weights in the rest of this discussion.

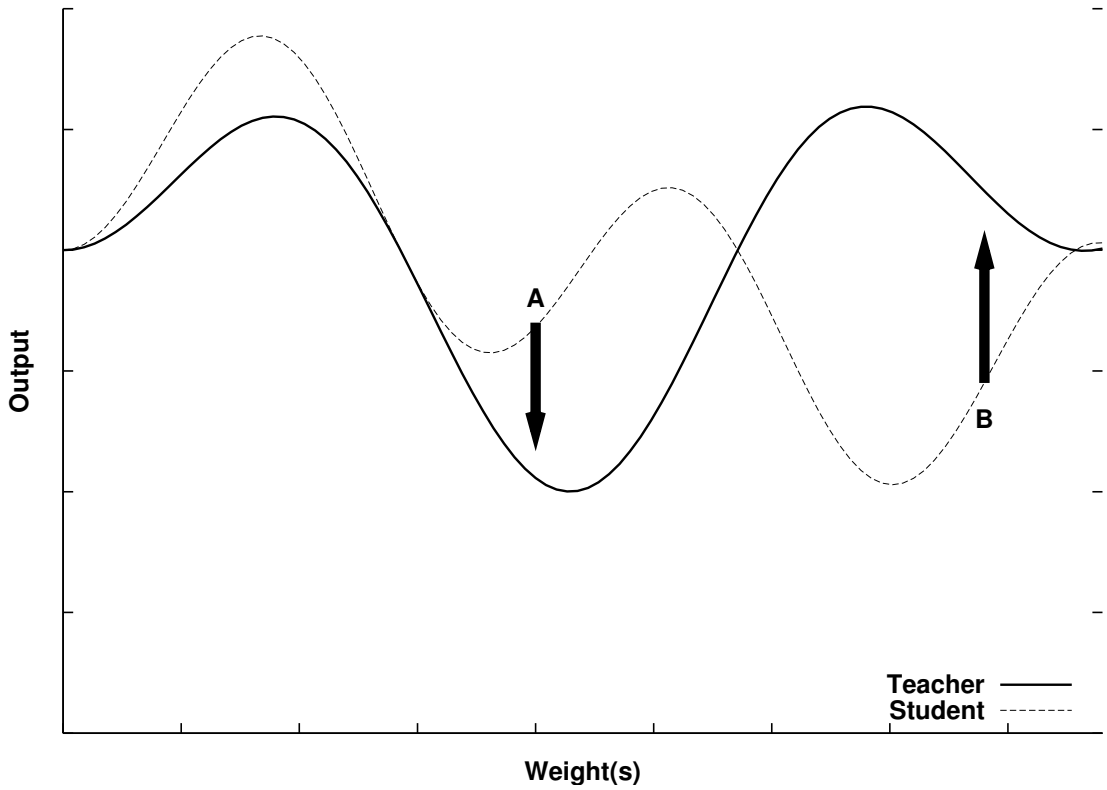


Figure 7.2: Teaching effect on student response surface with 1-dimensional input. The errors at two training examples are shown as arrows. Backpropagation will tend to move student's response curve toward the teacher's curve along the arrows.

would be very expensive. Fortunately, CE-TEACH does not need the exact response surface, as shown next.

7.3 CE-TEACH results

For empirical testing of CE-TEACH we return to the Inverted Pendulum task described in Section 3.1.1 and used to evaluate CE-CULL. Table 7.1 reports the average number of fitness evaluations required to find a solution for several methods. (RBHC and SOTA data is copied from Table 4.1.) Of 50 runs of CE-TEACH, only one did not succeed. On average, CE-TEACH required only one–fifth of the evaluations used by SOTA. Teaching provides a considerable performance boost.

The scores in Table 7.1 are significantly different, except for RBHC and SOTA. The significance P–values shown in Table 7.2 are computed by 10,000 resamples of the data points corresponding to each pair of methods.

Method	Eval/Soln	÷SOTA	sec/Soln	Success%
RBHC	8341	1.04	737.2	42%
SOTA	8037	1.00	380.5	53%
CE-TEACH	1710	0.21	35.5	98%
Perfect Culling	1300	0.16	262.6	100%

Table 7.1: Comparison of performance of CE-TEACH versus SOTA. *Eval/Soln* is the number of evaluations used in all runs divided by the number of successful runs. *Success%* is the percentage of runs that found a solution. *sec/Soln* is the average CPU time per solution. The RBHC and SOTA data previously appeared in Table 4.1. CE-TEACH finds solutions with only one–fifth the effort of SOTA.

Score	Method	Perfect	CE-TEACH	SOTA
1300.0	Perfect Culling			
1710.2	CE-TEACH	0.2		
8037.7	SOTA	0.0	0.0	
8340.9	RBHC	0.0	0.0	45.0

Table 7.2: Probability of equal means (in %) for each pair of CE-TEACH experiments. All differences are significant, except for SOTA versus RBHC.

Fitness evaluations in Pole Balancing are many times as expensive as a neural network activation. so it is a good example of an expensive sequential decision task for which the cultural methods were designed. It was arranged for the runs in this table to be performed on a single, unloaded machine, so comparative CPU times are available to evaluate the efficiency of the methods in practice. Table 7.1 shows that CE-TEACH was over ten times faster than SOTA. The speed–up in CPU time is better than the ratio of fitness evaluations because of the relatively high failure rate of SOTA, and the fact that failures tend to achieve relatively high scores. High scores in this task are achieved by keeping the pole up longer, which uses CPU time. Runs which do not find a solution continue for 50 generations, i.e. 10,000 evaluations, whereas successful runs in SOTA averaged only 4026 evaluations, and CE-TEACH only 1653. While almost half the SOTA runs failed, the CE-TEACH time was burdened with only one failed run CE-TEACH is simply better at finding solutions. Nonetheless, it is clear that CE-TEACH is several times faster than SOTA, and that ratio will be even higher for tasks with more expensive fitness functions.

These results were obtained with a fixed–size syllabus of 20 vectors. Since backpropagation has a larger effect with more training vectors, the size of the syllabus is an important parameter. Figure 7.3 shows how the average evaluations per solution varies with the length of the syllabus.

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). Too large a syllabus tends to train the offspring to be excessively similar to the imperfect teacher, which stifles progress (not shown in the chart).

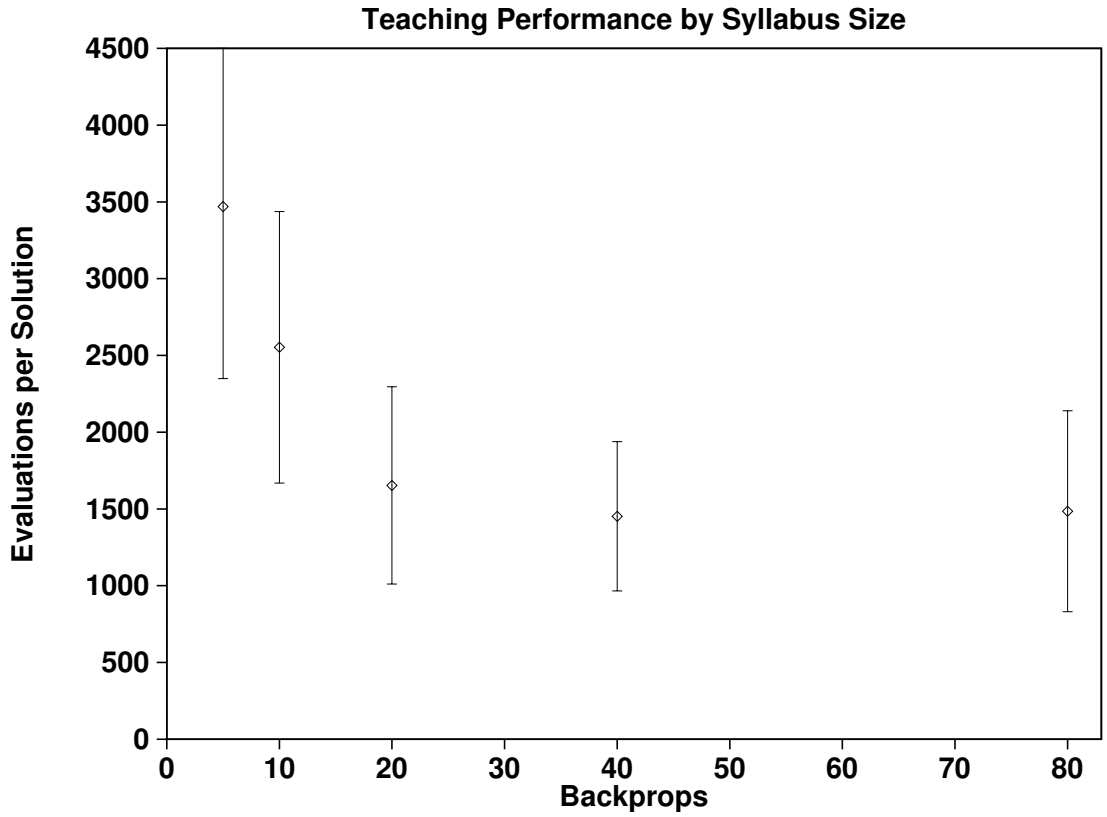


Figure 7.3: Effect of training time (syllabus length) on performance. Most of the benefit is achieved with 20 questions, and larger syllabi are not necessary.

7.4 Why is CE-TEACH successful?

The state of the network after teaching corresponds to the adult phenotype in biology. Phenotypic fitness determines reproductive success, and is what drives a genetic algorithm. However, the distribution of pre–training fitness in CE-TEACH (called "natal" fitness) is interesting. Almost every individual is born with surprisingly low fitness.

Even the winners (the networks that achieve the full score) behave poorly before training, almost as poorly as random nets. In fact, the median natal fitness of the winners in a set of 185 teaching runs was less than 2% greater than the median fitness of a large set of random networks, which was 88.4[2]. Although not shown here, a performance histogram of winners before training has essentially the same shape as a histogram of random networks.

The distribution of fitness relative to parental average in Figure 7.4 provides another useful insight into the operation of CE-TEACH. The figure compares offspring in CE-TEACH before and

[2] The score for random networks is greater than zero because it takes some time for the pole to fall.

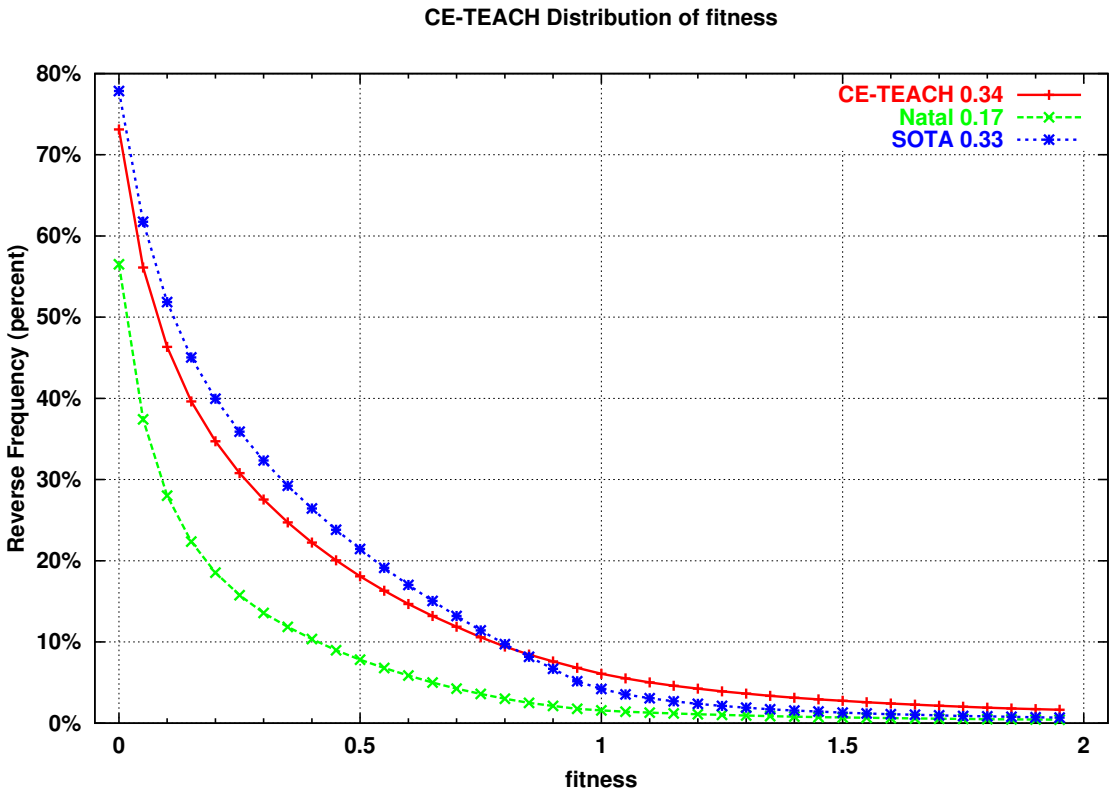


Figure 7.4: Comparison of relative fitness distributions before and after teaching, along with SOTA. Average fitness of parents = 1.0. Average natal fitness is only half that of SOTA, but after training they are much better.

after training by showing the reverse cumulative distribution of the fitness of offspring relative to their parents' average. The curve labeled CE-TEACH is the phenotypes actually produced. Natal labels the genotypes which became those phenotypes after training. For perspective, SOTA is also shown. The natal fitness distribution is significantly below even SOTA.

The ordinate is the ratio of offspring fitness to the average of its parents. Natal fitness evaluation is not needed by the algorithm, and is expensive, so this set of runs was not included in the timing data. Table 7.3 shows numeric data for selected ordinates of the chart. It shows that 6.09% of CE-TEACH offspring are better than the average of their parents versus 4.22% of SOTA, and only 1.58% of natal fitnesses. The 1.87% difference between CE-TEACH and SOTA may seem slight, but it is a ratio to parental fitness, so it works like compound interest. For example, during 30 generations a 4.22% growth rate yields a 3.45 growth factor, while a 6.09% rate gives an increase of 5.87. Those factors (5.87 : 3.45) still understate the advantage of CE-TEACH over SOTA, because they do not account for the higher fitness ratios. At a ratio of double parental fitness, the CE-TEACH advantage is overwhelming (1.65 : 0.69).

Average	Method	≥ 0.5	≥ 1.0	≥ 1.5	≥ 2.0
0.336	CE-TEACH	18.08	6.09%	2.75%	1.65%
0.328	SOTA	21.45	4.22%	1.31%	0.69%
0.174	Natal	7.81	1.58%	0.70%	0.43%

Table 7.3: Percentage of offspring exceeding several relative fitnesses. CE-TEACH produces over twice as many offspring that are twice as good as their parents as does SOTA (1.65 : 0.69), even though the natal fitness is much lower.

Table 7.3 also shows that the overall average natal fitness ratio is only about half of CE-TEACH or SOTA. The genomes evolved by CE-TEACH are worse than those of SOTA. The rate of improvement of natal fitness is so small that the agglomerated histogram of many runs is difficult to distinguish from a histogram of random individuals.

These results of CE-TEACH are surprising and counterintuitive. The normal pattern for natal fitness in previous genetic algorithms, including the prior CENE methods, is to for fitness to increase over the generations until some individual crosses the success threshold. In CE-TEACH, however, natal fitness remains poor. Why is that? The answer lies in the observation that CE-TEACH does not evolve solutions directly, but rather evolves networks that will respond to the training regimen. Consider a network to be a point in weight space, and call the "solution region" any point representing a successful network. The genomes that win with CE-TEACH are not located in the solution region, but rather at points that will be pulled into the solution region by an imperfect teacher.

The performance increase of teaching over the standard GA shown in Table 7.1 indicates that the area of the set of points representing winning genomes is larger than the solution region. In effect CE-TEACH widens the basins of attraction so that the underlying genetic algorithm has a larger target, and so faces an easier task. It is true that hybrid combinations of GA plus hill–climber have the same effect of broadening the basins. However, CE-TEACH does so without the additional fitness evaluations required by hill–climbers.

A possible alternative explanation of the distribution of natal fitness in CE-TEACH is that perhaps the method is evolving extremely effective teachers—teachers so powerful that they can train random networks. I performed an experiment to answer this question. Many random networks were subjected to training by selected winners. None of them learned the task, indicating that "trainability" requires some specific attributes of the initial weights. CE-TEACH can find such weight combinations: evolution and learning are working in harmony.

There is an interesting similarity between these teaching results and those of Nolfi and Parisi (1993a, 1994). CE-TEACH does not evolve genes that are immediately effective for the task: before training the networks perform very poorly. As remarked above, natal fitnesses match the fitnesses of random networks. Unlike random networks, however, winning nets are trainable in context. This resembles the Nolfi and Parisi (1994) result that their networks do not evolve optimal weights, but

rather weights that learn well. In Section 9.2 I suggest that these observations support the primacy of learning over doing when learning and evolution are successfully combined, whether based on self–teaching or learning from a teacher.

7.5 Conclusion

This chapter has shown how education of offspring can be utilized to improve neuroevolution. Many previous researchers have studied the combination of learning and evolution, but usually with no teacher. Unsupervised learning is slower than supervised learning, but supervised learning is not possible without output targets, as is the case for the sequential decision tasks targeted by this dissertation. The CE-TEACH method presented in this chapter takes advantage of supervised learning in this unsupervised context by constructing teaching targets from the behavior of the prior population. Such targets are imperfect, but improve over the course of evolution. This chapter showed that the targets improve fast enough so that CE-TEACH has a positive impact on performance.

CE-TEACH has a resemblance to the arrangement of Cecconi et al. (1996), reviewed in Section 2.4.4, where offspring were trained with situations encountered by their parents. The major difference is that their syllabus was determined by parental experience, whereas my syllabus is under independent control. Their scenario is more biologically plausible, but mine is more flexible, and should be capable of better performance. Further discussion is postponed to Section 9.2.

This concludes the development of my four novel techniques. Since they all address separate aspects of the genetic algorithm, they should be able to work well together. That expectation is fulfilled in the next chapter.

Chapter 8

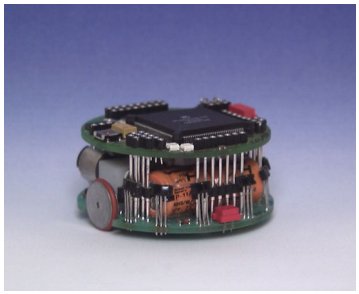
CENE: Combined Cultural Enhancements

The previous four chapters developed and tested each new method separately. Those presentations are rather disjoint, because each method is targeted at a separate aspect of the genetic algorithm. Because each of the four has a different target, they can operate together in synergy, as this chapter will demonstrate. In this chapter, all four methods are utilized together in a combined method called Culture–Enhanced NeuroEvolution, CENE, which is tested in a robot control task. After introducing the Khepera task, the chapter has three main analytic sections: first the performance of CENE is compared with other modern benchmarks; second, the contributions of the individual component methods are evaluated; and third, the interactions among the individual components are analyzed by removing one at a time. All these analyses taken together support the conclusion that CENE is a very effective mix where diversity and selection pressure are balanced well.

Within each analytic part of the chapter, performance is first compared by ultimate solution quality, then learning curves are presented to compare speed of evolution. Third, the time course of diversity in the evolving population is analyzed. The benchmark comparisons of the first part will show that CENE excels on both performance metrics, and also has excellent diversity. Before proceeding with the analyses, let us review the Khepera task domain.

8.1 The Khepera Domain

This section presents the Khepera robot and its simulator, and then details the task to be measured in this chapter.



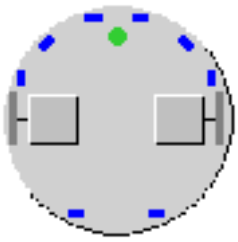


Figure 8.1: The Khepera robot, designed, manufactured and distributed by K–Team S.A., Switzerland. www.k–team.com Used by permission. The body diameter is about two inches (55 mm), and the wheels are 50 mm apart. (left) A picture of the physical robot. (right) A diagram of Khepera's floor plan showing placement of motors and wheels at left and right, and eight photosensors (blue rectangles) around the perimeter. Khepera is inexpensive and appropriate for teaching and research. Figures used with permission, copyright K–Team S.A.

8.1.1 The Khepera Robot and Simulator

The physical Khepera robot (Figure 8.1) is a small, two–wheeled vehicle with an embedded controller. The main purpose of Khepera is research and education, as described by Nolfi and Floreano (2000) and Murphy (2000). A widely used software simulator of Khepera is also available (Michel 1996). Importantly, the simulator allows perturbing both sensor inputs and effector outputs with noise. The simulator has been found to be faithful enough to the physical robot that control algorithms trained only in the simulator behave well when downloaded into a physical Khepera (Miglino et al. 1995; Nolfi and Parisi 1995a). Thus, machine learning tasks with the Khepera simulator are nearly real–world tasks. The simulator was chosen for this dissertation because experimenting with the simulator is both faster and less expensive than using a physical robot.

Khepera has eight infrared emitter–sensors that report the amount of reflected light in their field of view. A high value means an object is very close nearby, but due to the limited range of the sensors, a low value only implies that no obstacle is detected within about 60 mm, which is only somewhat more than one body diameter. The Khepera floorplan is bilaterally symmetric. Six of the eight sensors are oriented forward, sideways, and diagonally forward, while the other two are aimed backwards, as shown in Figure 8.1. These eight sensor readings (normalized to a range of 0 to +1) are the inputs to the neural network.

Although there are other optional modules available for Khepera, such as grippers, the basic Khepera in this simulator has only two effectors: the motors. The activation applied to each motor is a signed integer with a magnitude up to 10 (scaled from neural network outputs). Full activation of the same sign on both motors gives a maximum forward or reverse motion of 5 mm per timestep. The simulator is tuned for a control frequency of 20 Hertz, so one timestep is 50 milliseconds, and the maximum speed is 100 millimeters per second, or almost two body diameters per second. Full activation with opposite signs causes the maximum turn rate of 0.1 radian per timestep, or 2 radians

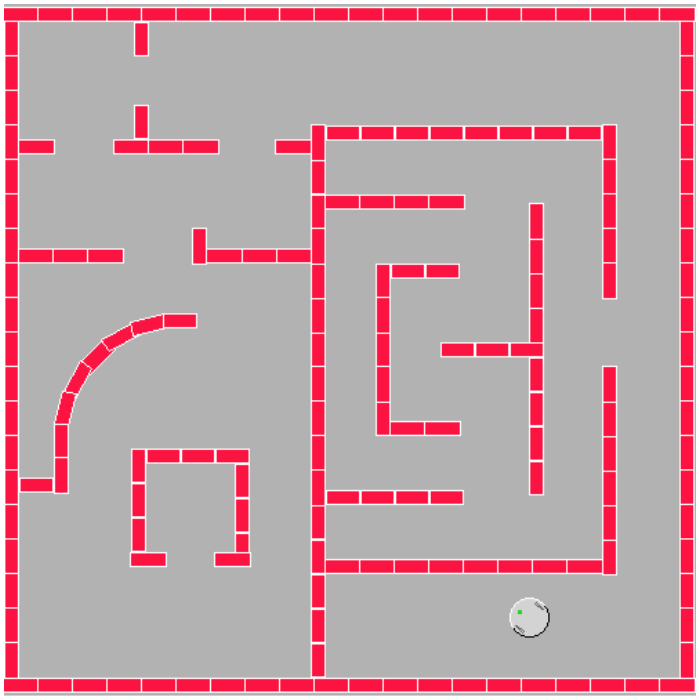


Figure 8.2: Maze–world, a complex Khepera environment. Each side of the square is one meter, so this world is about $1000 \div 55 = 18$ body diameters across, and takes about 200 timesteps to cross at maximum speed. The variety of geometries in this environment force Khepera to learn general strategies. Figure from (Michel 1996), used with permission.

per second, so a $90\circ$ turn in place can be done in $\frac{\pi}{4} rad \div 2 rad/sec = 393 ms$, or about 8 timesteps. Since each motor can be driven independently, Khepera can exhibit a wide variety of turning rates, including spinning in one spot. However, since the effectors are subject to noise, perfect turns and straight lines require the controller to continually adjust the activation level applied to the motors.

The simulator includes a facility for defining walls, and computes sensor activations from the specified geometry. The slightly complex world shown in Figure 8.2 was chosen for the experiments described in this chapter. This world exhibits a variety of geometries to the robot. It contains narrow corridors, wider areas, dead ends, and curved walls. The narrow corridors and doors are so close to the body diameter that rather precise motor control is required to navigate them.

The Khepera robot domain is suitable for a comprehensive evaluation of advanced neuroevolution techniques due to its complexity, and the wide availability of a free simulator which has been shown to have near real–world behavior. Also, a concrete comparison with an existing advanced neuroevolution method, SANE (Moriarty 1997), is available.

Analysis of results will be presented after the task is described.

8.1.2 The Race task

This section provides details of the specific task chosen for this analysis. The task is one elementary behavior that should be included in the repertoire of a successful physical robot: to achieve as much linear motion as possible in a fixed time, turning only where necessary to avoid contact with obstacles.

Khepera is small enough that its motion is dominated by friction, so momentum planning is not a necessary issue. Indeed, the simulator assumes the task is Markovian; only the current state matters. The task requires only local information, so it is compatible with the standard Khepera sensors. Since neither the specific task nor the Khepera platform require memory, a feedforward network is an appropriate controller. That is the phenotype used in this dissertation.

High scores are obtained by traveling at high speed in a straight line, while avoiding contact with any wall. Running into a wall extracts a heavy penalty: the trial stops at that point. This forces the controller to learn to steer to avoid walls. Thus, the score for this task consists of two factors: (1) The total distance traveled in a trial, discounted by (2) the amount of turning done.

The turning penalty is necessary since without it high scores can be achieved by continually turning in a small circle. That behavior is too easy to learn, and is not particularly useful for a real robot. Turns are necessary in this environment to achieve more than minimal scores, but steps with turns are penalized, so the network must learn to execute turns only when necessary.

The total score is the sum of the scores on each time step. The score for the i–th step is:

$$s_i = |F| * (1 - T^2)$$

where F is the forward (or backward) movement, and T is the amount of turning as a fraction of the maximum turn rate.

Even though Khepera is capable of moving just as fast in reverse as forward, and the scoring function treats backward motion as just as valuable as forward motion, the asymmetric placement of sensors makes achieving a high score moving backward very difficult. Since the initial networks are random, reversers may initially, by happenstance, have the highest scores, and concomitant mating success. This deceptiveness of the fitness function is just another challenge that a successful evolution has to contend with, and they do, as will be shown below.

8.2 Comparison with other methods

To determine how well CENE performs, test results were compared to other modern evolutionary methods, SOTA and SANE, and to Perfect Culling as well. SOTA is the standard GA benchmark used in all testing in this research (Section 3.4). It reflects current best practices in the field with reasonable parameter values. CENE is built from SOTA by adding the cultural mechanisms of the preceding four chapters, so the comparison to SOTA indicates how much they contribute. Comparison with SANE (Moriarty 1997) provides a connection with the prior literature, as he found that

Score	÷SOTA	Method
196	100%	SOTA
227	115%	SANE
240	122%	CENE
266	135%	Perfect Culling

Table 8.1: Comparison of solution quality attained in a large fixed number of evaluations (8200). The scores are averages of 50 runs each. The ÷SOTA column contains the ratio of the score to SOTA's score. CENE is the best real method (recall that Perfect Culling is not implementable.)

SANE was superior to Genitor and the Adaptive Heuristic Critic in the Khepera domain. Perfect Culling is the nondeterministic, unrealizable method derived in Section 4.2.

The first comparison examines results when solution quality is paramount, so a large number of evaluations is desirable to ensure that all methods have reached a plateau. CENE is seen to be the best real method. Following that, learning curves reveal larger differences during the earlier stages of evolution. As a result the speed of evolution metric shows even larger differences. Finally, CENE is shown to have excellent diversity as the evolution proceeds. Overall, CENE is the best of the methods examined.

8.2.1 Ultimate scores

CENE achieved the best solution quality in a large, fixed number of of evaluations. The comparison is shown in Table 8.1, the solution quality attained in a fixed number of evaluations. The number of evaluations is 8200—large enough that all the methods have stopped making progress[1]. The 8200 evaluations include 81 generations for SANE, and 40 generations for the other methods, since SANE uses a population size of 100 and my populations have 200 individuals.

In Table 8.1 it is clear that CENE attains a higher score than SANE and SOTA, but it is not obvious whether the differences are large. Applying the resampling methods of Section 3.3 to this data gives Table 8.2, which shows the probability of the null hypothesis for each pair of methods. The observed difference between CENE and SANE is significant at the 95% level.

At the end of evolution CENE is significantly better than the other methods in this task. In order to determine if the performance relationships are the same at all stages of evolution the next section contains learning curves, which provide details of the course of the evolution and allow comparing learning speed.

Score	Method	SOTA	SANE	CENE
196.41	SOTA			
226.77	SANE	0.1		
240.17	CENE	0.0	4.1	
265.76	Perfect Culling	0.0	0.0	0.1

Table 8.2: Percentage probability that difference in observed scores from Table 8.1 is due to chance. A small value implies that the true means are probably different. All of these differences are significant at the 95% level.

8.2.2 Learning Curves

While the final performances are significantly different, even larger differences exist earlier during the course of evolution, as shown in Figure 8.3. The fact that all the curves flatten considerably toward the right is an indication that the differences that remain are likely to be real. The very top line is the unattainable Perfect Culling. It raises steeply and plateaus early. By 1400 evaluations it has reached 90% of its final value. By 2000 evaluations its score exceeds what any other method can do with 8000.

SANE paces SOTA initially, but keeps rising longer. SOTA rises almost as fast as SANE until about 1800 evaluations, but then it slows down considerably and by 4000 has been overtaken by the other methods. CENE has a commanding lead early on, but SANE narrows the gap somewhat by the end. It is clear that CENE has better performance at all times except the very earliest. The discussion so far has concentrated on the solution quality at equal times. The next section examines a complementary metric, the time taken to achieve a specified test score.

8.2.3 Speed of evolution

There are many applications where ultimate performance is all that matters—where one percent improvement could have an economic benefit greater than the cost of a large amount of computer time. In such applications it is entirely appropriate to focus only on the highest score that can be achieved, as in the previous section. But for domains where it is not worthwhile or practical to run all methods until their learning curves flatten out, it is important to compare speed of learning. Speed is also the useful metric for domains where several techniques can reach the maximum quality. In addition, it is useful to report the speed metric in order to permit comparison with results presented in previous chapters.

The speed of evolution is shown in Table 8.3 as the average number of evaluations taken to reach a score of 200. A target of 200 was chosen for illustration, because it is the largest round

[1]Moriarty (1997) used 8100 evaluations in his Khepera task. Since my populations contain 200 individuals the limit needs to be a multiple of 200.

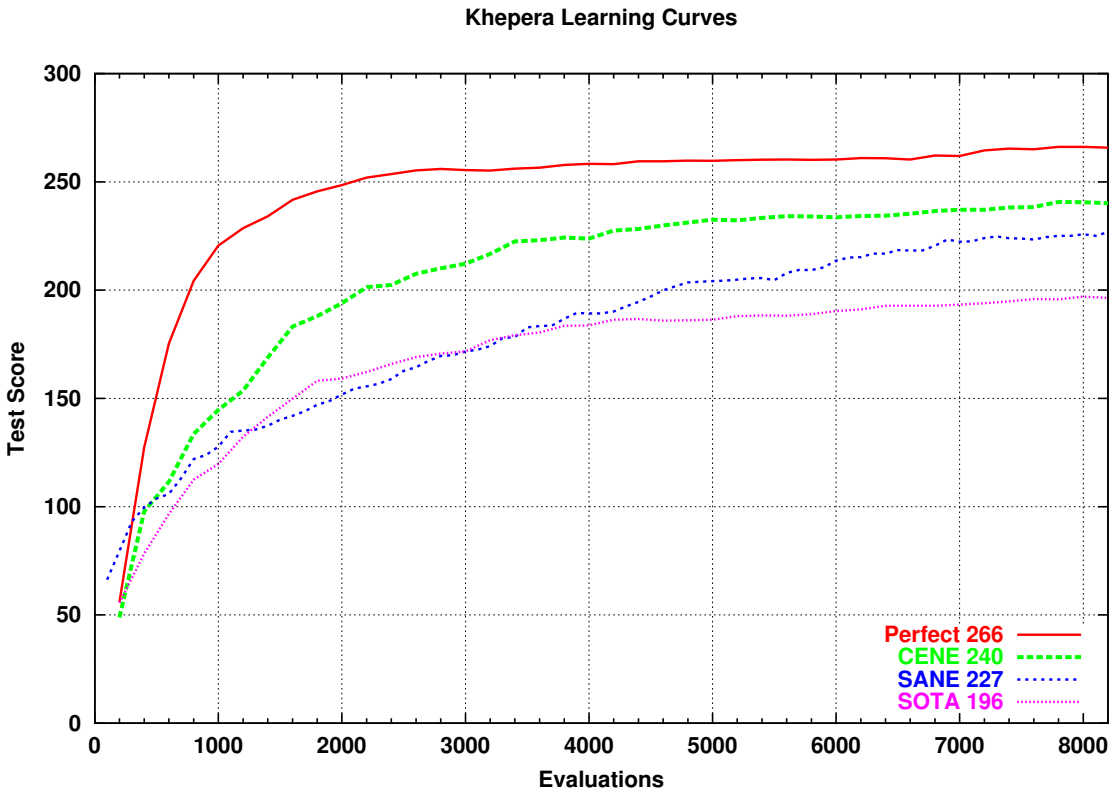


Figure 8.3: Comparison of learning curves over the course of evolution. Numbers in legend are final scores. Although all curves flatten considerably by the end, there are are differences in how fast they progress.

score that is routinely achieved by all methods. The unrealizable Perfect Culling is six times faster than SOTA by this metric. SANE takes half again as long as CENE (62 : 41%).

The relative order of the methods is the same as in Table 8.1, but the percentage differences are much more pronounced. CENE is much more efficient than SANE. For example, if each fitness evaluation cost a penny, then each solution would be expected to cost over $40 using SANE, but could be had for around $27 from CENE.

The resampling procedure of Section 3.3 produces Table 8.4, which shows that all these differences in speed are significant. This table is a case where parametric statistics such as Student's *t*-test are not useful. SOTA succeeded in attaining 200 only about two–thirds as often as the other methods, and a significance test based on ignoring the failed runs is just as misleading as a test that assumes all the runs succeeded[2].

[2]Section 3.3 discusses yet other methods of averaging the successful and failed runs, but concludes that resampling is the only reasonable significance computation.

Evals/soln	÷SOTA	Success%	Method
1048	0.16	98	Perfect Culling
2710	0.41	90	CENE
4079	0.62	94	SANE
6621	1.00	66	SOTA

Table 8.3: Comparison of average number of evaluations required to achieve a score of 200, averaged over 50 runs. *Success%* is the fraction of runs that reached 200. The percentage differences in speed shown here are larger than the differences in ultimate quality shown in Table 8.1.

Score	Method	Perfect	CENE	SANE
1048	Perfect			
2710	CENE	0.0		
4079	SANE	0.0	2.0	
6621	SOTA	0.0	0.1	0.6

Table 8.4: Probability of equal means (in %) for each pair of datasets from Table 8.3. Resampling shows that all these differences are significant.

8.2.4 Diversity

As shown above, CENE has excellent performance, both on quality and speed. This section begins to analyze how that performance is achieved by comparing diversity over the course of evolution with SANE. Diversity is shown not to completely determine final performance, but is needed in abundance for realistic methods (i.e., everything except Perfect Culling).

A primary design goal of SANE was automatic preservation of diversity (Moriarty 1997, Section 3.3.1), and it excels at that. The CE-DIVERSE component of CENE is a more active approach to the same end. Genetic diversity of the population is defined in Section 6.1. Population diversity over time during the execution of the runs shown in Figure 8.3 is shown in Figure 8.4. The SANE curve is virtually the same as the one shown in (Moriarty 1997, Figure 4.4), even though the task is slightly different.

SANE's ability to preserve diversity is remarkable. Even after 8000 evaluations, when the population contains very competent networks, SANE's diversity is almost 0.35, which is 70% of the diversity of a purely random population[3]. A diversity of 0.35 represents a lot of randomness in individual bit positions, yet the competence of the population members must tend to require patterns in the chromosomes. A strong pattern carried by bits of such diversity implies that the bits in the population genomes have been allocated in a very effective fashion.

In contrast to SANE, the diversities of Perfect Culling and SOTA fall precipitously, resembling the "Standard Elite" method of Moriarty (1997). Both the highest and lowest scoring methods

[3]Section 6.1 explains that a random population has an expected diversity of 0.5

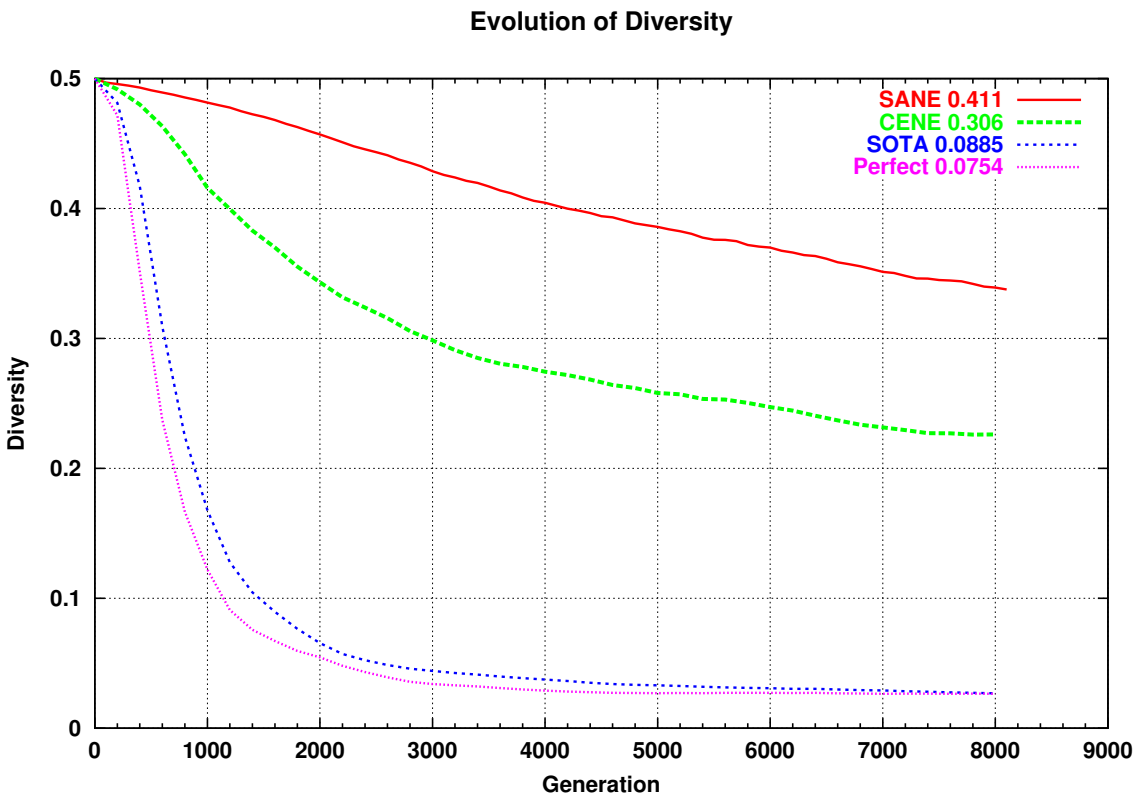


Figure 8.4: Diversity over time. The numeric suffixes in the legend indicate average percent diversity over all times. SANE and CENE remain diversified long after the others have converged.

exhibit this crash in diversity. It does not mean that diversity is a useless measure, however. Loss of diversity is not harmful when it is in an excellent direction. Such early convergence to low diversity means that both Perfect Culling and SOTA quickly lose the ability to make much progress, but for Perfect Culling that does not matter because its quality rises so quickly that it has already achieved an excellent score by the time the loss of diversity converges the population. Deterministic, realizable methods need to be smarter.

CENE has excellent diversity, even though not as extreme as SANE. It takes almost 8000 evaluations for its diversity to drop to half the initial level. This decline is gentle enough to permit evolution to continue making progress. Diversity's contribution to CENE will unfold further in Section 8.3.4 which presents diversity for the component methods, but finally Section 8.4.3 will reveal that the interaction of diversity and selection pressure is responsible for the success of CENE.

This section has shown that CENE has excellent performance, both on quality and speed, and it maintains diversity well. In order to understand this performance in Section 8.4 we will analyze contributions by removing components. Let us first, however, examine the performance of

the individual component methods by themselves.

8.3 Components of CENE performance

When the component methods of CENE were developed in the previous four chapters they were tested in isolation on several simple problems. This section provides a uniform comparison between all four on the same task, controlling the Khepera robot. This task is more difficult than the tasks reported in the individual component chapters, and, indeed, one method, CE-CULL, is clearly worse than the others. More importantly, the combination of all four, CENE, performs better than any one of its components.

This section follows the same overall plan as the previous section. First, solution quality is described, followed by learning curves. Then speed of evolution is compared, and finally the evolution of population diversity is shown. The results in this section confirm the findings of the individual component chapters that the components faithfully implement their design goals, although the performance of CE-CULLis found to be rather problem–dependent. CENE, the combination of the components, can still be effective because its components leverage different aspects of the genetic algorithm.

8.3.1 Component solution quality

The performance of each component method in isolation on the Khepera robot task is shown in Table 8.5. Results from Table 8.1 are included to make comparisons easier, but the rest of this section shows only CENE and its components, in order to avoid clutter. Comparing with the methods of Table 8.1 reveals that CE-MATE and CE-TEACH are competitive with SANE, and nearly as good as CENE. Evidently, CE-CULL is too greedy to be competitive in a domain as complex as Khepera. The result of eliminating CE-CULL will be examined in Section 8.4.

In order to determine if the performance differences above are significant, the resampling analysis of Section 3.3 was used to generate Table 8.6, which shows the probability that the differences could be due to chance alone. This table does not rule out that CE-DIVERSE, CE-MATE and CE-TEACH could all be equal to each other. Further, there is not enough difference in the actual observations to distinguish CENE from CE-TEACH, and CENE is only distinguished from CE-MATE at the 90% level. So, at the end of evolution the component methods are roughly comparable, except for CE-CULL. In the next subsection a more detailed look at the course of evolution over time will expose more significant differences.

8.3.2 Component learning curves

Table 8.5 above shows that the best component methods eventually achieve comparable solution quality. The learning curves which detail the paths taken to those destinations are presented in

Score	÷SOTA	÷CENE	Method
160	0.81	0.67	CE-CULL
196	1.00	0.82	SOTA
220	1.12	0.92	CE-DIVERSE
227	1.15	0.95	SANE
229	1.17	0.95	CE-MATE
233	1.19	0.97	CE-TEACH
240	1.22	1.00	CENE
266	1.35	1.11	Perfect

Table 8.5: Comparison of solution quality attained in a fixed number of evaluations (8200) for each component. Scores are average of 50 runs each. Columns labeled by ÷ show ratio of scores. The components are remarkably close together, except for Culling.

Score	Method	Cull	Diverse	Mate	Teach
159.8	Cull				
219.6	Diverse	0.0			
229.3	Mate	0.0	14.0		
233.4	Teach	0.0	9.0	30.9	
240.2	CENE	0.0	2.4	9.2	22.3

Table 8.6: Percentage probability that difference in observed means is due to chance. A small value implies true means are different. CE-TEACH, CE-MATE, and CE-DIVERSE are roughly equivalent, and CENE is not well separated from CE-TEACH nor CE-MATE.

Figure 8.5, and reveal that larger differences are present earlier in evolution. Section 8.2.2 compared CENE to Perfect Culling, SANE, and SOTA, so only the components and CENE are shown here.

CE-TEACH and CENE finish just ahead of the other real methods, but have a commanding lead for a substantial fraction of earlier times (between 2500 and 5000, for instance). CE-MATE and CE-DIVERSE do not rise as fast as the others initially, but they are steadier, still improving long after the other methods converge. They appear to be the least flattened methods at the end, which means they have not yet converged. That observation makes sense, as they are both anti–convergence techniques.

CE-CULL has good initial performance, but eventually stagnates. Khepera is one problem domain where culling actually performs worse than the standard GA (compare to SOTA in Figure 8.3). CE-CULL has good initial performance, and is competitive until about 1000 evaluations, but then slows down. It achieves 90% of its final value by 3000 evaluations, so it does not make much progress for the last five–eighths of this scenario. CE-CULL is a very greedy method, which accounts for the quick initial acceleration. As we saw in Chapter 4, for some problem domains that is a good strategy. However, for domains like Khepera, being too greedy does not pay off in the long

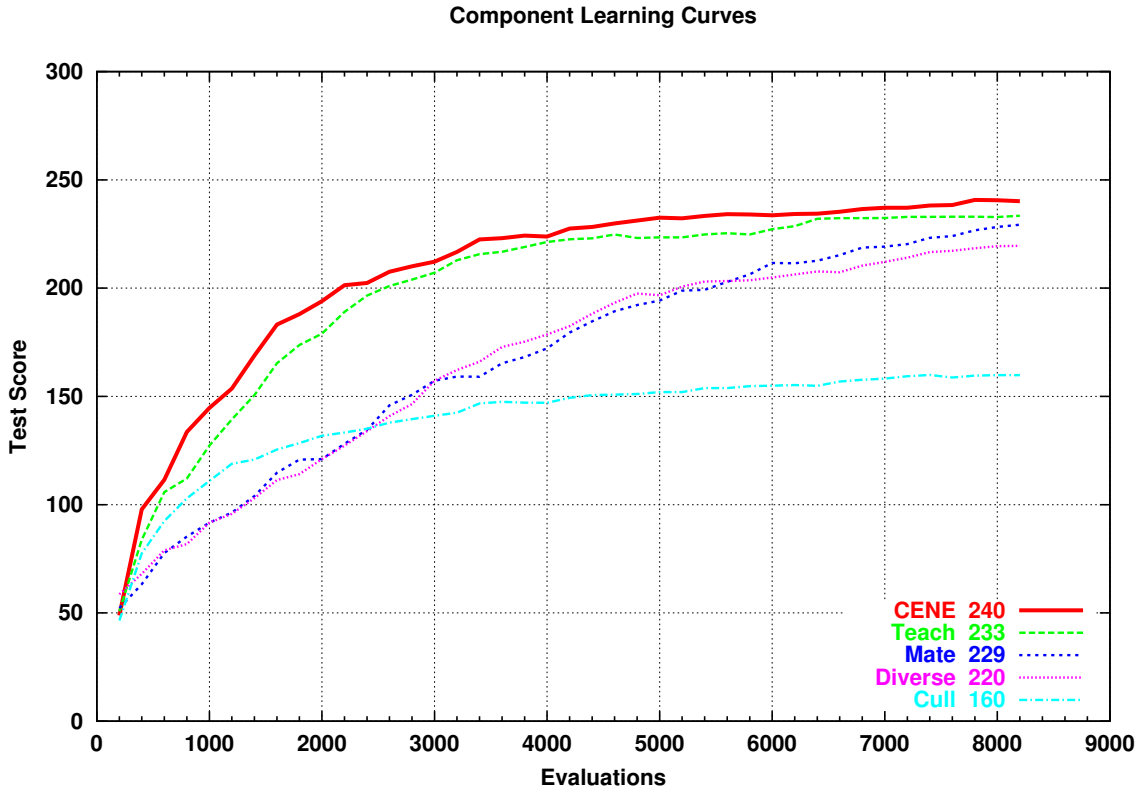


Figure 8.5: Comparison of component learning curves. The scale of the abscissa is deliberately the same as in Figure 8.3, to facilitate comparison. Despite ultimately converging, there are considerable differences in how fast the components progress.

run. It leads to premature convergence and eventual stagnation, as will be seen in Section 8.3.4.

The disappointing performance of CE-CULL raises the question of whether CENE would be improved by removing CE-CULL. Does the early good performance of CE-CULL contribute enough to counterbalance its ultimate poor performance? That question will be answered in Section 8.4.

Having examined the ultimate solution quality in a long evolution, the next subsection presents the complementary metric, the number of evaluations needed to attain a fixed score.

8.3.3 Component speed

The evolution speed of the component techniques in isolation, measured by the average number of evaluations needed to attain a score of 200, is shown in Table 8.7, which can be compared with scores for the benchmark methods found in Table 8.3. Once again, the differences in speeds are more pronounced than the differences in ultimate scores.

As was previously seen in Table 8.5, CE-CULL has a major problem with this task. CE-

Evals/soln	÷SOTA	Success%	Method
2710	0.41	90%	CENE
3407	0.52	88%	CE-TEACH
5853	0.88	83%	CE-MATE
6304	0.95	76%	CE-DIVERSE
14182	2.14	44%	CE-CULL

Table 8.7: Comparison of speed of evolution of component methods as average number of evaluations required to achieve a score of 200 within 8200 evaluations, averaged over 50 runs. The ÷SOTA column contains ratio of the score to SOTA's score. *Success%* is the fraction of runs that reached 200. The differences between methods are more pronounced than those based on the ultimate quality scores in Table 8.5.

Score	Method	CENE	Teach	Mate	Diverse
2710	CENE				
3407	Teach	16.4			
5853	Mate	0.0	0.1		
6304	Diverse	0.0	0.3	31.2	
14182	Cull	0.0	0.0	0.0	0.1

Table 8.8: Probability of equal means (in %) for each pair of datasets. CE-TEACH and CENE are not significantly different, and neither are CE-MATE and CE-DIVERSE.

MATE is twice as slow as CENE, with CE-DIVERSE lagging relatively close behind it, but the ÷SOTA column shows that CE-MATE and CE-DIVERSE are both better than SOTA. CE-TEACH is the best of the component techniques, being only 25% slower than CENE.

To determine whether the performance differences shown in Table 8.7 are significant, Table 8.8 shows the possibility of the null hypothesis given the actual observations for all pairs of methods. CE-DIVERSE and CE-MATE are not proved different, and there is about a one in six chance that CE-TEACH and CENE have the same performance[4] but the other differences are significant. On both performance metrics discussed so far, ultimate quality and speed, CE-TEACH is competitive with CENE. The picture diverges when diversity is examined.

8.3.4 Component diversity

Figure 8.6 shows population diversity over time of component methods on the Khepera task. Comparing Figure 8.4 shows that CE-CULL loses diversity about as fast as Perfect Culling, once again showing that greedy methods lose diversity quickly.

CE-TEACH has good diversity at first, but falls to very low levels. However, CE-TEACH has reached 90% of its final score at 3200 evaluations (Figure 8.5), and its diversity is still above 0.1

[4]Although not reported in this table, resampling showed that CE-DIVERSE and CE-MATE also have a reasonable chance of being equivalent to SOTA, and CE-TEACH has a 17% chance of matching SANE.

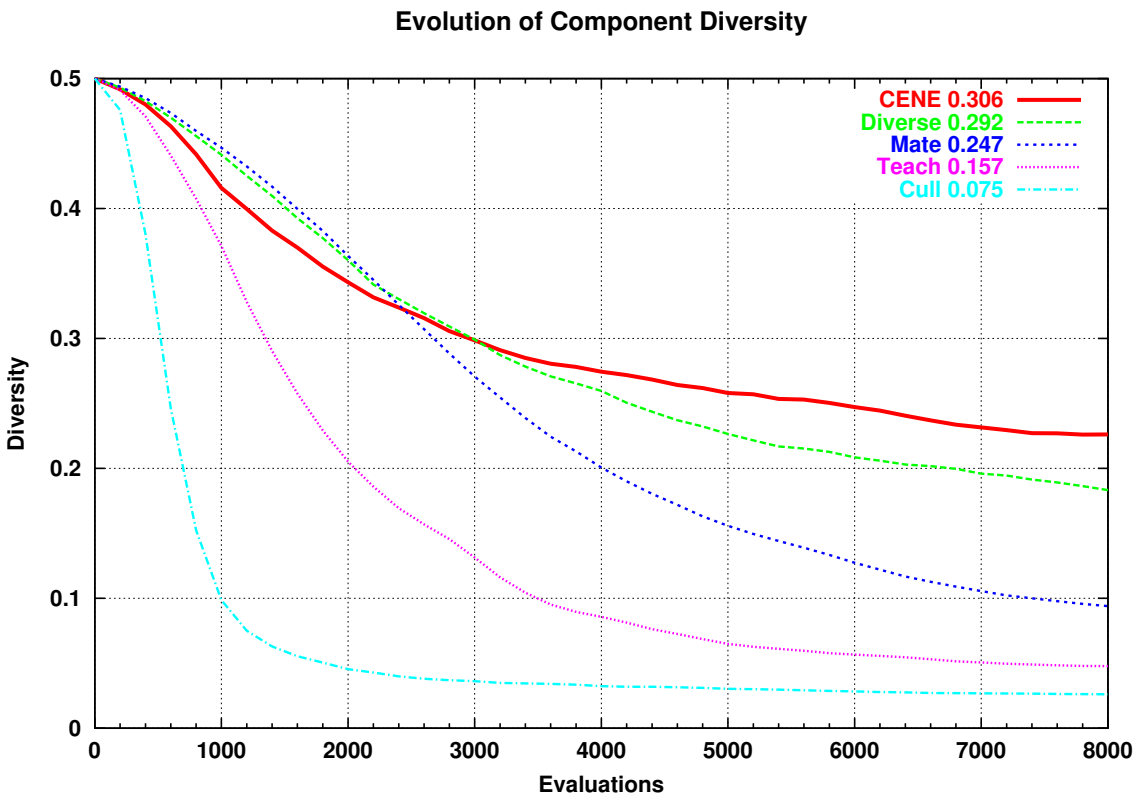


Figure 8.6: Diversity over time of component methods. The numeric suffixes in the legend indicate the average percent diversity. Because of the secondary diversity enhancement effects of CE-MATE and CE-TEACH, the diversity of CENE eventually surpasses even CE-DIVERSE.

at that point. CE-TEACH makes very little progress after its diversity falls below 0.1, an illustration of the general point that converged populations do not evolve well. CE-TEACH reaches high scores quickly enough to beat the other component methods, illustrating the other side of that point (i.e., high diversity is not the only attribute of a successful method). The diversity enhancement of CE-TEACH over SOTA can be understood as wider basins of attractions (see Discussion Section 7.4). Thus, the underlying GA has less selection pressure, and diversity falls more slowly.

Both CE-MATE and CE-DIVERSE maintain good diversity. They stay above 0.2 for over 7000 evaluations. CE-DIVERSE was designed to preserve diversity, so this is additional confirmation that the implementation is sound. The longevity of diversity in CE-MATE is explained by the fact that the complementary spousal behaviors promoted by CE-MATE also tend to be diverse genotypes.

CENE does better even than CE-DIVERSE, due to the combined benefits of CE-TEACH and CE-MATE. CENE shows that the greed of CE-CULL is overcome, perhaps even utilized, by the combination of the other three components. Section 8.4.3 below will reveal more about the source of CENE's diversity.

Score	÷CENE	Include	Omit
198	83%	CD–T	–Mate
200	83%	C–MT	–Diverse
215	90%	–DMT	–Cull
221	92%	CDM–	–Teach
240	100%	CDMT	CENE

Table 8.9: Comparison of CENE with omitting one constituent at a time. The omission is marked as "–". The ÷ CENE column shows the ratio of the score to CENE. All components contribute to the success of CENE.

The results in this section confirm the findings of the individual component chapters: the components faithfully implement their design goals. However, the performance of CE-CULL on the Khepera task is disappointing, suggesting that CE-CULL be removed from the combination. Such an ablation, and others, will be examined in the next section. CENE, the combination of the components, is effective because the components leverage different aspects of the genetic algorithm, as will be substantiated by the ablation study.

8.4 Dissecting component interactions

The previous section exposed strengths and weakness of each individual CENE component method in isolation in order to determine what they could contribute. This section analyzes the interactions of the components to determine what they actually contribute, and whether there are any unexpected redundancies or conflicts. The plan for this ablation study of the combined method, CENE, is to remove each of its constituents, one at a time.

This study will reveal overlap between secondary effects of the components, but in no case does one component render another inoperative. The component methods were designed to work together, and they do, as will be shown by the fact that removing any one of them causes a significant drop in performance.

After examining the performance of the ablated methods, diversity analysis will provide sufficient evidence to conclude that CENE is built on a solid foundation. But first, performance will show that all four components are necessary for CENE's success.

8.4.1 Ablated scores

Table 8.9 compares CENE with the result of omitting one constituent at a time. A label prefixed with a dash in the last column tells which technique is omitted for that row. The *Include* column is an almost–graphic depiction of the constituents included in that row, by using their initial letters.

All components are seen to contribute to the performance of CENE. Somewhat surprisingly, omitting CE-MATE (CD–T) has the largest impact, dropping the solution quality by 17%, but omit-

Score	Method	CD–T	C–MT	–DMT	CDM–
197.8	CD–T				
200.0	C–MT	41.2			
215.4	–DMT	2.9	5.9		
220.8	CDM–	1.2	2.2	29.1	
240.2	CDMT	0.0	0.0	0.2	1.8

Table 8.10: Probability of equal means (in %) for all pairs of ablated datasets. CDMT is CENE, which is significantly different from all others.

ting CE-DIVERSE (C–MT) is almost as bad. This implies that diversity, whether of mates or of decedents, is crucial for CENE.

To omit Culling (10%) or Teaching (8%) has ponderable but lesser impact. Thus, the weak performance of Culling by itself, shown in Table 8.5, does not harm the other techniques. It appears that the diversity–enhancing properties of the other techniques combine well with the excessive greed of Culling. It is sensible to conclude that high selection pressure is good when its diversity reduction effect is counterbalanced.

The effect of omitting Teaching is another surprise. Table 8.5 (page 75) shows that CE-TEACH is the most powerful of the four component techniques, but the ablation Table 8.9 says that leaving Teaching out (CDM–) is the least damaging choice, causing only an 8% degradation. The other three techniques together (CDM–) are about as powerful as CE-TEACH. In particular, it is necessary to combine all four techniques to beat SANE in this task.

To determine which performance differences are significant, resampling was employed to generate Table 8.10, which once again shows the probability that the methods have equal means given the actual observations. The performance advantage of CENE over all the partial methods is significant at the 98% level. This table also confirms what the comparatively close scores imply: (CD–T) and (C–MT) are not significantly different, and neither are (–DMT) and (CDM–).

8.4.2 Ablated speed

As seen in the analyses of the two preceding sections, it is also instructive to contrast performance measured by speed of evolution. The relative performance of the ablated techniques, measured by the average number of evaluations needed to attain a score of 200 is shown in Table 8.11, which can be compared with speed scores in Table 8.3 and Table 8.7. The differences in speeds are much larger than the differences in ultimate quality.

However, the pattern of Table 8.7 still obtains: Omitting CE-MATE (CD–T) has the largest impact, tripling the effort required to achieve a score of 200, and omitting CE-DIVERSE (C–MT) is nearly as harmful. To omit CE-CULL (–DMT, 204%) only doubles the effort, and removing CE-TEACH (CDM–, 180%) is almost as bad. Thus, measuring along the speed dimension reinforces the

evals/soln	÷CENE	Success%	Method	
2,709.6	1.00	90%	CDMT	CENE
4,876.2	1.80	78%	CDM–	–Teach
5,538.7	2.04	80%	–DMT	–Cull
7,194.3	2.65	64%	C–MT	–Diverse
8,399.9	3.10	60%	CD–T	–Mate

Table 8.11: Comparison of speed of evolution of ablated methods as average number of evaluations required to achieve a score of 200, averaged over 50 runs. The ÷CENE column contains ratio of the score to CENE's score. *Success%* is the fraction of runs that reached 200. The scores are much more separated than the ultimate quality scores in Table 8.9.

conclusions reached along the quality dimension, Section 8.4.1.

8.4.3 Ablated Diversity

Diversity analysis of the ablated methods will supply enough observations to conclude how CENE operates and what its components contribute. Figure 8.7 shows diversity over time for the ablated methods. It can be compared with Figure 8.4 for the benchmark methods. In particular, the CENE curve is the same in both figures.

CENE preserves diversity well, as we have already seen. When culling is omitted (–DMT) diversity is even higher than CENE. This is consistent with the extremely steep drop of CE-CULL by itself shown in Figure 8.6. It serves to reinforce the observation that CE-CULL is a very greedy method that converges quickly. Therefore CE-CULL should only be expected to work well in domains where there are few local optima. However, despite this caveat, in complex domains such as Khepera, CE-CULL still helps performance when combined with other methods and therefore contributes to forming an efficient general optimization method.

When CE-DIVERSE is omitted (C–MT), diversity falls almost as fast as that of SOTA. This result indicates that CE-DIVERSE is most strongly responsible for maintaining the high diversity seen in CENE, When Mate Selection is omitted (CD–T) the diversity loss is significant but smaller.

When Teaching is omitted (CDM–) diversity is almost as high as in CENE, confirming that Diversity Preservation and Mate Selection are indeed the main components responsible for diversity in CENE.

Diversity analysis of the ablated methods illuminates the relationship between diversity and selection pressure. There is overlap between secondary effects of the components, in that CE-TEACH and CE-MATE both contribute to diversity, and almost completely offset the diversity loss due to CE-CULL. The component methods were designed to work together, and they do, as was shown by the fact that removing any one of them causes a significant drop in performance. Therefore, CENE is built on a solid foundation.

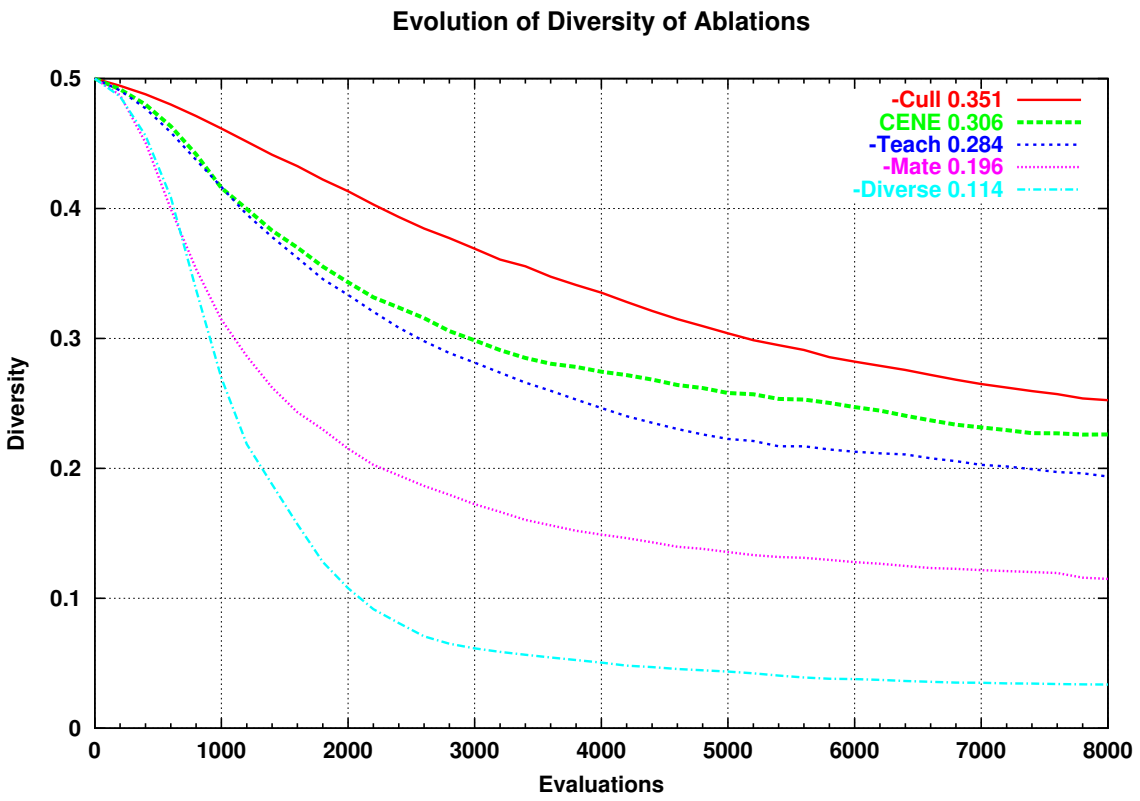


Figure 8.7: Comparison of diversity of ablated methods over time. In the legend, a preceding dash indicates which component was removed from CENE to generate that curve. The numeric suffixes give average diversity over the entire course of evolution. These curves can be compared with the benchmarks in Figure 8.4 and the components in Figure 8.6. Both CE-DIVERSE and CE-MATE contribute significantly to diversity, and almost completely offset the loss of diversity due to CE-CULL.

8.5 Conclusion

The results presented in this chapter demonstrate that the four cultural methods of CENE combine to form an effective neuroevolution procedure. Each individual component is effective because its design is derived from firm principles of evolution. The combination is effective because the components leverage different aspects of the genetic algorithm. The fact that the combined method, CENE, worked so well on the Khepera task after the component methods had been tested only on simpler tasks (see chapters 4–7) suggests that this result is robust and general as well.

An ablation study of CENE revealed overlap between secondary effects (i.e., diversity preservation by CE-MATE) of the components, but no cases where one component would prevent another from functioning. Further, removing any one of the four constituents caused a significant drop in performance. The interaction of the components exposes an interesting relationship between diver-

sity and selection pressure. CE-MATE contributes to diversity, even though it was designed for a different purpose (to assist crossover). CE-TEACH also contributes to diversity, because it widens the basins of attraction due to the plasticity of the individuals it evolves, but its diversity effect is considerably smaller.

CENE bettered SANE's performance even though SANE remains the diversity champion. This result underscores that diversity is not the only factor that contributes to the success of neuroevolution. The performance of Perfect Culling shows that selection pressure in a good direction is more important than the absolute best diversity. Diversity prevents high selection pressure from causing stagnation, so that high selection pressure can perform its job of accelerating progress toward the goal. Thus, diversity is seen to be the handmaiden of selection pressure. SANE combines very high diversity with very high selection pressure (Moriarty 1997). CENE is lower than SANE in both diversity and selection pressure, but since CENE has better performance than SANE, I contend that CENE has a better balance between diversity and selection pressure.

Interestingly, while the high selection pressure of CE-CULL is detrimental to diversity, the method still makes an important contribution to the combined performance. The diversity enhancement effects of the other components allow the exploitive tendency of CE-CULL to flourish safely. Thus, the components are leveraging each other.

The component methods were tested based on a SOTA implementation in order that their contributions could be measured consistently and reliably. However, all components can in principle be applied to more advanced neuroevolution algorithms, a prospect discussed in Section 9.3.

While CENE significantly exceeds other advanced methods on all metrics measured in this task, the difference is most pronounced when speed of evolution is more important than ultimate long–term solution quality, which will have important consequences when the methods are applied to harder problems. For problems that are so difficult that it is impractical to continue evolution until the learning curve levels out, speed of evolution is the most important criterion. Indeed, the definition of speed adopted in this work is precisely the number of fitness evaluations, and CE-CULL and CE-TEACH work directly to avoid fitness evaluations. CE-CULL substitutes behavior comparison for decisions about offspring that would otherwise require fitness evaluation. CE-TEACH uses behavior modification to improve the fitness of otherwise unfit offspring, which increases the number of target genotypes. More targets require fewer evaluations to find.

Having demonstrated that my cultural focus leads to effective and original neuroevolution methods, the next chapter integrates my findings and discusses future directions made possible by this work.

Chapter 9

Discussion and Future Directions

This dissertation contributes practical techniques that enhance the performance of neuroevolution in sequential decision tasks. This chapter summarizes these contributions and details the scope of the findings. Several of these issues lead to directions for future work.

The discussion begins with a recap of the performance enhancements developed herein, with an elaboration of their operation and interactions. Next, how the CENE methods could be combined with other neuroevolution substrates is discussed. Section 9.2 explores the interaction between learning and evolution. Then Section 9.4 elaborates on artifacts in culture, and Section 9.5 proposes a lifelong learning extension. Section 9.6 argues that communicating cooperative agents might benefit from the techniques developed in this work. Section 9.7 proposes alternative methods to chose or compose a teacher. Lastly, Section 9.8 shows why my CENE techniques are likely to be fruitful in non–stationary environments. I conclude that the work described in this dissertation is a significant step toward incorporating new aspects of learning into neuroevolution research.

9.1 Performance contribution

Culture provides major advantages in the natural world. It enables skills to be acquired by offspring without the cost and danger of learning those skills from scratch. This dissertation exploits culture expressed by behavior to contribute enhanced performance techniques to neuroevolution. Behavior of neural networks was found to be adequately characterized by samples of their input–output mappings. The CENE techniques compare and manipulate behaviors using such mappings to effect cultural mechanisms that enhance the performance of new generations. This section elaborates on the performance, effects, and interactions of those methods.

- CE-CULL: culling selects offspring that are more likely to succeed from an overlarge litter by comparing performance of candidate offspring with a known above–average individual.
- CE-MATE: mate selection by complementary behavior provides crossover with more chances

to combine solved subproblems by promoting the mating of individuals with complementary behaviors.

- CE-DIVERSE: phenotypic diversity preservation increases diversity without weakening the population.

- CE-TEACH: learning to respond like a somewhat competent elder leverages evolution by widening basins of attraction.

- CENE: all four techniques combine gracefully since each one leverages a different aspect of the genetic algorithm. The increased diversity due to three of the components permit the high selection pressure of CE-CULL to flourish. CENE both learns faster than SANE, an existing advanced method, and achieves higher ultimate solution quality.

All CENE methods are based on behavior represented by a syllabus that is a set of input–output samples. In CE-CULL, the errors between several candidate offspring are compared. In CE-MATE, the outputs of several candidate suitors are compared. In CE-DIVERSE, the outputs of several candidate decedents are compared. in CE-TEACH, training error is used to adjust the weights of an offspring. Since none of the methods introduce additional fitness evaluations they will be efficient if they are at all effective, as demonstrated in previous chapters.

The component methods are well–designed by principles of evolution to not interfere with each other, but can have secondary effects. The following table categorizes these effects as exploratory or exploitative.

Method	**Primary effect**	**Secondary effect**
Mate	Exploitative: enhance the productivity of crossover	Exploratory: encourage diversity of mates
Cull	Exploitative: find offspring of higher estimated fitness	
Diverse	Exploratory: increase population diversity	Exploitative: kill lower fitness candidate
Teach	Exploitative: improve fitness of offspring	Exploratory: makes bigger target for GA

Table 9.1: Comparison of primary and secondary effects of individual methods. CENE methods complement each other well.

The exploration versus exploitation battle of machine learning is manifested by diversity and selection pressure in evolutionary computation. The individual CENE methods enhance one of these factors without detracting from the other. The combined method, CENE, benefits from the dovetailing of the effects of the components. As Moriarty (1997) demonstrated with SANE, good

diversity allows higher selection pressure to be used. Diversity is thus seen not to compete with selection pressure, but rather to enable it.

All CENE techniques are efficient because they operate with no fitness evaluations beyond those that the underlying genetic algorithm would perform in any event. Rather, the methods are designed to increase the utility of each fitness evaluation.

CE-MATE and CE-DIVERSE can be viewed as principled engineering enhancements of the genetic algorithm. CE-MATE encourages the mating of individuals with complementary competencies, which provides crossover with more opportunities to combine solved subproblems. CE-MATE also has a positive effect on diversity, because it selects mates with disparate behaviors, and different behavior is strongly correlated with diverse genotypes. CE-DIVERSE directly preserves diversity in the population by choosing to remove individuals that are most like others. Population diversity by itself is antithetical to progress, as can be seen by considering that a population kept in a completely random state would be no more effective than random guessing. CE-DIVERSE can accomplish its diversity preservation without detracting from performance because it only removes weak individuals.

In contrast to the clean engineering approaches of CE-MATE and CE-DIVERSE, the other two component techniques rely on a form of boot–strapping. Overly aggressive boot–straps can quickly converge to mediocre performance, while timid boot–strapping may achieve no leverage whatsoever. CE-CULL and CE-TEACH both rely on the increasingly competent population to provide target outputs for constructing new individuals. The success of CE-CULL is based on the ease of recognizing abysmal performance, and so it can be viewed as an engineering approach to the large lower lobe in the standard fitness distribution of Figure 4.1.

CE-CULL is a greedy technique that can have problems with deceptive domains, as shown in Section 8.3. However, as discovered in Section 8.4, CE-CULL is a valuable component when combined with sufficient diversity preservation, as done in CENE. An analogy with culling in biology is suggestive: the smarter a species is, the less overproduction of offspring it does. Fish lay millions of eggs, but primates produce small or singular litters and invest a large amount of time and resources in each offspring. Such differences could be seen to suggest that the effectiveness of culling might fall off in more sophisticated problem domains. The biological analogy is not dispositive, however, as we can engineer a fix for the simplistic greed of culling, for example CENE. As was mentioned in the introduction, biology can provide suggestions for neuroevolution but does not constrain the programs we write.

CE-TEACH is also a boot–strapping approach inspired by observation of the effectiveness of educating offspring in the real world. Diversity preservation with CE-TEACH is generally good (Figure 8.6) because fitness does not depend just on the raw fitness of the genotype, but also on its *teachability*: how well it responds to backpropagation learning. These trainable networks have more diverse genotypes because they are located outside the basins of attraction of non–educated networks.

Empirical tests showed that CENE techniques perform well, with diversity and selection pressure in balance. In CE-TEACH learning, evolution, and culture exhibit non–obvious interactions which deserve more attention, as in the next section.

9.2 Learning and evolution

Due to the interaction of learning and evolution, the effects of CE-TEACH are more complex than those of the other CENE methods. This work has revealed a novel interaction between evolution and learning. As described in Section 7.4 genomes produced by evolution did not improve as fast as the trained phenotypes. That is, the evolved individuals were not particularly competent before training, but rather were capable of being taught competent behavior. Prior evolutionary algorithms result in genotypes that improve more–or–less continually to the desired performance level, when they work at all, and that is the pattern in phenotypic fitness in CE-TEACH, too. But natal fitness in CE-TEACH evolves not to better performance but to better trainability.

Three previous studies have reported unexpected interactions between evolution and learning whose root cause seems to me to be related to my finding. First, Nolfi and Parisi (1993b, 1994) found that their evolved networks were better learners than doers, a result that was not the intention of the investigators. Second, Sasaki and Tokoro (1997) found that learning ability can be more important than doing ability. Third, Ackley and Littman (1991) showed that it is easier to generate a good evaluation function than a good action function. The genotypes in my results resemble the first two: they are good learners and poor performers.

The similarity of those three studies and this work is that situations arise where learning ability is more important than doing ability. The learnability is useful is not that surprising. However, the fact that a learning/doing differential surfaced so directly in such simple yet disparate settings suggests that the primacy of learning ability over doing ability may be a quite general principle.

For instance, if that general principle holds, then I would predict it to manifest in a related simulation of Cecconi et al. (1996), reviewed in Section 2.4.4. The parents in that work carried their offspring around for some time after birth. During this time the immature offspring were exposed to the same input stimuli as the parent, and were trained via backprop. Successful individuals adopted the strategy of performing like an optimal teacher rather than an optimal forager.

If the parent in the Cecconi et al. (1996) work were to be given a boolean input which specified that a child was currently on board, then an additional strategy becomes available: When a child is present, act like an optimal teacher, else act like an optimal forager. There is no guarantee that evolution would find that strategy, but if it did, I would regard that as a successful prediction of the general principle of the primacy of learning. The ability to learn a task is so powerful that four different systems have evolved networks that sacrifice fitness in order to improve learning ability.

9.3 Generalization of CENE

SOTA was designed so that it would be easy to combine the CENE methods with it. However, CENE is not specific to SOTA, and could be used in other neuroevolution variants as well. The fact that CENE does not examine the genotype in any way makes such integration easy. This section describes what variants are compatible with CENE methods.

In general, a steady–state GA is an easier match than a strictly generational GA, which sometimes discard entire chunks of the population at a time instead of calling a routine like KILLSELECT (see the interface in Figure 3.3). CE-DIVERSE assumes that the underlying GA is able to discard any member of the population except the current champion.

All CENE methods require that it is possible to sample the behavior of individuals without invoking the fitness function. In most cases such sampling is straightforward, but how to do it is less obvious for cooperative coevolution methods like SANE. In such methods, the sampling can be performed at the level of partial solutions. Since CENE often computes distances between output vectors, the outputs should be floating–point. If outputs are integers or strongly quantized, CENE will probably need longer syllabi in order to develop useful discrimination.

CE-CULL requires the ability to create a temporary pool of individuals outside of the current population. If the mate selection strategy is a crucial part of a GA, then CE-MATE is not likely to be appropriate. CE-TEACH requires that the phenotype have a learning mechanism, which rules out most popular phenotypes except neural networks.

Finally, a GA which is already very good at some aspect may get only limited profit from some CENE components. For instance, SANE already preserves such a high level of diversity that it seems unlikely that CE-DIVERSE would be useful within it. ESP (Gomez and Miikkulainen 1999) and other island methods as well as Genitor (Whitley and Kauth 1988) also have explicit diversity mechanisms, so CE-DIVERSE might not improve those methods either. On the other hand, NEAT (Stanley and Miikkulainen 2002), where several species of neural networks with gradually more complex topologies are evolved in parallel should be amenable to CENE methods.

Such applications of CENE to other neuroevolution algorithms constitute a most interesting direction for future work.

9.4 Artifacts

One apparent limitation of this work is that the culture explored does not include artifacts. Such primitive culture does exist in the natural world. For example, some songbirds are known to pass on their songs without artifacts, for example, Dawkins (1989, Chap. 11) describes non–genetic transmission of song repertoires among New Zealand saddleback birds. Predatory mammals teach their cubs to hunt. Primates learn conventions of behavior from their elders.

The omission of artifacts from this study is deliberate and appropriate. At this stage of

our understanding the interaction of learning and evolution is still nascent. The study of non–artifactual culture allows focused work aimed at identifying principles without the confounding effects of some particular representation of artifacts. However, full–blown culture is so complex that it is extremely likely that it entails phenomena that do not manifest without explicit artifacts. Thus, while my omission of artifacts is beneficial at this stage, it is a limitation, that should be addressed in future work. My approach to culture could be extended to include simulated artifacts, using any of representations reviewed in Section 2.4. Indeed, because CENE techniques are based on behavior and so have no dependence on the genotype, the phenotype nor the task, any representation whatsoever of artifacts is compatible with CENE.

I hope that this study of a restricted case—culture *sans* artifacts—will provide a firmer foundation for a future understanding of how artifacts impact culture.

9.5 Lifelong learning

All learning in CENE takes place before any exposure to the environment—before fitness is evaluated. That is, individuals cease learning upon becoming adults. In the natural world, any being that is sophisticated enough to learn or to be educated during development will also be capable of learning from the experience of its adult life. The ongoing interaction of learning and evolution in the natural world is therefore not utilized in CENE. Thus CENE cannot create the full range of phenomena that occur in nature. However, since the model is strictly less powerful than nature, it should only be missing advanced interactions, not introducing artificial capabilities.

The interaction of lifelong learning and culture in neuroevolution is unclear at this point, but is very powerful in nature, and so is likely to be a fruitful area to explore.

9.6 Communicating Cooperative Agents

CE-TEACH resulted in evolving low–performance but trainable networks. The training performed transferred a behavioral tendency from the teacher to the student. Such a behavioral tendency can also be viewed as a proxy for a concept of how to cope with the world. Further extending that viewpoint suggests that sharing concepts in that fashion could be useful for cooperative agents to communicate concepts not designed beforehand. The possibility that back–propagation could be used as a communication method is non–traditional, so an elaboration is provided.

Primitive cooperative agents coordinate their actions via pre-specified strategies or by observing each others' actions. To achieve more sophisticated cooperation, or when observation is insufficient, cooperative agents need to communicate. Communication takes place when the meaning of a message is coherent among the sender and receiver(s). Let a signal be some physical disturbance used to carry a message (sound waves, marks on paper, etc). Groups of signals denote symbols. A message is then a collection of symbols constituting an intended communication. In

simple communication systems there is a bi-unique correspondence between the message and the signal (to within channel noise), so in discussions of cognition the difference between message and signal is usually insignificant.

Communication involves shared context. Sophisticated communication does not include all relevant context explicitly in each message. For instance, in the English language the word "dog" is a referent to a rather large description that the recipient already possesses. The meaning of any text can only be defined with respect to some agent. Even in the most primitive systems the total meaning is not absolute. For example, the voltage sent by a thermostat to a furnace seems to have a completely unambiguous meaning, but actually involves a lot of context.

Coherence between sender (thermostat) and receiver (furnace) is achieved by constructing them to share conventions about voltage levels, polarity, and perhaps other channel characteristics. Further, the designer of the devices had a model of the relevant parts of the real world in her/his head. Even this trivial communication has different meanings to different agents: The furnace, of course, has only a mechanistic response. The designer sees a host of physical implications. The user (in the unlikely event they noticed any communication between their appliances) would frame the meaning in terms of comfort and budget. Each agent grounds the meaning in its own universe. But there can be no communication unless those universes cohere.

Many animals that fight conspecifics for resources (food, territory, mates) start out exchanging threats, which is a negotiation. If one convinces the other that there is no doubt of the outcome, they are both better off to avoid physical altercation and damage. These threats are a communication of the form, "I'm bigger/stronger/braver/more determined than you." Honestly advertising your strength is only optimal for the strongest, so there is opportunity here for deception and bluff. Such sophisticated communication is only possible because the parties already share an enormous amount of information, concepts and world-views. In order to even attempt a bluff, you must feel you can make a very good estimate of the other's motivation and commitment.

These messages are highly situated. There is more information in the context than in the signal. The meaning of a message has only an indirect relation to the "text" used to represent it. Sophisticated cooperative agents also need to share substantial context. They would be handicapped if each of their messages needed to contain explicit representations of shared concepts.

Shared concepts can be acquired via several routes. For non–adaptive agents the shared concepts can be built in when they are designed or constructed. Indeed, a large part of the job of designing communicating cooperative agents today is creating concepts that will permit the desired solution, and parameterizing them in a way compatible with the signaling system available.

However, I believe that highly sophisticated adaptive agents will invent new concepts as they learn and evolve. They will need to communicate these emerging concepts. Any language prespecified by the implementor is unlikely to be optimal. Thus, I expect that agents can be more effective by letting them create their own language. Communicating cooperative agents will be handicapped if they can only communicate about concepts the system designer enumerated. They

will need to invent their own concepts and a vocabulary to discuss them with. In fact, Steels (1996) found that a cultural mechanism can create a coherent vocabulary. Culture is inevitably involved when learning and communication unite, because

The fact that human language spreads culturally also leads me to suspect that cultural evolution techniques may work better than genetically specified languages. In support of this view, Werner and Dyer (1991) held that since cooperation in the natural world depends so totally on communication, that the requirements of effective cooperation place important constraints upon how communication can evolve.

MacLennan (1991) argued that "cultural phenomena will be central to understanding the interaction of learning and communication." Culture without artifacts is a good place to begin to study the communication of cooperating adaptive agents. The closest this dissertation has approached communication is in teaching one network to respond somewhat like another. The "signal" containing the message is an input and output vector. The "concept" has only concrete representations: the arrangement of weights in the networks. Those weights may differ between sender and receiver, but if they respond similarly to an input, it seems reasonable to say that they share a concept. Coherence is approached when the backprop error decreases.

The techniques developed in this dissertation may form a basis for tasks where communication must be evolved. While beyond the scope of this dissertation, my teaching and syllabus construction techniques may shed light on how to evolve languages for communicating cooperative agents. For instance, in my teaching method, the information flow is one–directional, from teacher to student. If the roles are alternately swapped, a series of backprops would resemble a conversation.

9.7 Composing a Teacher

In CE-TEACH the teacher has been a single elder, either a parent or the current population champion. The quality of teachers is critical to the success of my teaching method. Future work should include study of additional methods of choosing or composing teachers.

In the tasks addressed in this dissertation, a single teacher has been sufficient. In more complex domains more variety may be useful. Perhaps a new offspring (student) should be allowed to try several teachers and continue only with the higher scoring one(s). A student might have several teachers, each one concentrating on a subset of the input space, that is teaching a separate subject. Another way to utilize several teachers appears to have no analog in human affairs: The responses of a committee could be averaged and used as backprop targets. While providing several teachers should promote a wider viewpoint, averaging output vectors could just as easily result in nonsense.

Alternatively, teaching specialists could be co–evolved. These specialists would be a separate population, evaluated according to the ultimate fitness of their pupils, as is sometimes done in animal breeding.

9.8 Non-stationary environments

My methods may have application in non–stationary environments. This section first characterizes such environments and classifies their characteristic time–scales. Then I argue that machine learning techniques that can deal directly with non–stationary environments could become more effective than a "one controller for all seasons" strategy. Finally, I indicate why cultural techniques in general, and CENE techniques in particular, should be advantageous in non–stationary environments.

The problem domains considered in this work are all *stationary*. In a stationary domain the rules remain constant over time. All changes of state are due to actions taken by the individual. In contrast, a non–stationary environment is one with spontaneous changes. The changes may be cyclical, as in day and night, or there may be a rough monotonicity, as in a wheel whose tread is wearing off. There may even be no discernible pattern, as when the changes are due to the presence of other agents. In fact, Sutton and Barto (1998, Sect. 2.6) notes that many reinforcement learning problems are effectively non–stationary.

Change in a non–stationary environment may occur on any time–scale. The time–scale dimension of environmental change is crucial in determining possible coping mechanisms. The extreme frequencies of change have clear requirements. When change occurs on short time–scales, shorter than an individual life–span, then each individual must be capable of coping with the change. In contrast, when the change is slower than many individual life–spans, the reservoir of genetic material in the population can evolve adapted solutions. Cultural mechanisms are uniquely well–suited to change on intermediate time–scales, as noted by, for example, Belew (1990); Belew and Mitchell (1996); Nolfi and Parisi (1995b).

For natural organisms the time–scales are fixed, but in machine learning applications the effective life–span of a controller can be deliberately controlled. Consider, for instance, a wheeled robot whose mission is to explore Mars for several seasons. The environmental cycles important to the robot might include photo–sensor and battery effectiveness as light and temperature change diurnally and annually. The current approach is to account for the predicted environmental changes when designing the initial control program. For instance, to program the controller with appropriate strategies for both high and low states of light and temperature. This one controller for all seasons approach is actually a redefinition of the task to make it a stationary task with parameters. However, viewing the task as stationary can result in combinatorial explosion of controller complexity as additional environmental changes, for example, tread wear and sensor aging, are added to the mix. Thus, it may become advantageous to deal directly with the non–stationary nature of the task. For instance, a background process in the deployed system could continually evolve new control programs scored on actual experience, and let one of them take over when performance warrants.

With the current state of the art in machine learning, such self–adaptive systems are not appropriate for something as dangerous as a nuclear reactor controller, but a multi–year deep–space probe might become an appropriate application.

There are two aspects of this dissertation that are good matches for long term on–line learn-

ing in non–stationary environments:

- Diversity preservation: Due to the perpetual novelty of the environment, the population cannot be allowed to converge, even in an indefinite number of generations, and yet introducing random genetic material (mutations) must stay limited enough to keep good performance. CE-DIVERSE (Chapter 6) is a good step in this direction.
- Flexible time-scale: Culture operates on a timescale intermediate between individual learning and multi–generational evolution. A neuroevolution system designed to cope with non–stationarity should include culture to ensure that unexpected frequencies can be covered.

9.9 Conclusion

This dissertation has developed novel methods of manipulating the behavior of population members to improve neuroevolution. The CENE methods are practical and effective, but they suggest several interesting extensions. The interaction of culture, learning and evolution in CE-TEACH affirms that learning is more important than doing. In all, the work described in this dissertation is an important step toward incorporating additional aspects of learning into neuroevolution research.

Chapter 10

Conclusion

The goal of this dissertation was to show that cultural mechanisms could be harnessed to enhance the power of neuroevolution in sequential decision tasks. I developed four techniques that take advantage of the information contained in the current state of the population. Analysis of each showed it to be effective at its facet of the genetic algorithm, even though CE-CULL is too greedy for some applications. Because the four component techniques address different aspects of the genetic algorithm they work together well when combined in CENE. Because none of the techniques use additional fitness evaluations they are also efficient. The combined method has a good balance between diversity and selection pressure.

This chapter first summarizes the contributions of each component and their combination, and then concludes with a perspective on what has been learned.

10.1 Contributions

Each of the contributions reviewed below illuminates a different aspect of the genetic algorithm. They are all based on samples of input–output behavior and need no knowledge of how the genome is represented. Therefore they should all work with other neuroevolution systems as well, in addition to the state of the art GA (SOTA) used as a basis for the developments in this dissertation. Analysis of CENE shows how their exploration versus. exploitation trade–offs combine effectively.

CE-CULL

CE-CULL shows how culling over–large litters can be utilized to reduce the tendency of crossover to produce ineffective individuals. A perfect oracle would provide a large improvement in offspring fitness, but would cost too much. Instead, CE-CULL uses behaviors in the population to inexpensively select those offspring that are most likely to be competent. To be effective, the culling test only needs to discriminate enough to reduce the huge lower lobe of incompetence. Culling leads

to significant improvement in learning speed in some domains, whereas in others it is too greedy to be used by itself. It also contributes positively to CENE in combination with other less greedy techniques.

CE-MATE

CE-MATE improves the chances of combining partially solved subproblems in crossover by choosing mates with complementary traits. The CE-MATE method selects mates of complementary behavior, so they are necessarily of lower fitness than the mates chosen by a standard GA. Such a selection reduces greediness and fosters diversity, and therefore combines well with culling. The complementary behaviors of the mates more than compensates for their lower fitness and as a result, CE-MATE allows evolving solutions faster.

CE-DIVERSE

Whereas merely selecting very weak decedents would accelerate premature convergence, CE-DIVERSE enhances diversity by discarding the most typical of a sample of weak individuals, As a result, CE-DIVERSE is able to preserve diversity while still discarding weaker individuals than SOTA. Phenotypic diversity proved to be an adequate proxy for genotypic diversity, allowing CE-DIVERSE to be efficiently implemented as well.

CE-TEACH

CE-TEACH takes advantage of supervised learning in an unsupervised context by constructing teaching targets from the behavior of the prior population. Such targets are imperfect, but improve over the course of evolution. CE-TEACH widens the effective basins of attraction seen by the underlying GA by teaching new individuals to respond somewhat like a partially competent elder. The natal fitness of new individuals is low, but they train well, suggesting that learning and evolution are working in harmony. CE-TEACH shows how teaching can be utilized to improve neuroevolution.

CENE

The cultural methods above combine to form an effective, novel neuroevolution procedure, CENE. The combination is effective because the components leverage different aspects of the genetic algorithm. The fact that the combined method worked so well on the Khepera task (Chapter 8) after the component methods had been tested only on simpler tasks (chapters 4–7) suggests that CENEis general and would be effective in various applications.

CENE bettered SANE's performance even though SANE populations are more diverse. This result affirms that diversity is not the only success criterion for neuroevolution. The performance of Perfect Culling shows that selection pressure in a good direction is more important than the absolute

best diversity. Diversity prevents stagnation, so that high selection pressure can accelerate progress toward the goal. SANE combines very high diversity with very high selection pressure (Moriarty 1997). CENE is lower than SANE in both diversity and selection pressure, but since CENE has better performance than SANE, I contend that CENE has a better balance between diversity and selection pressure.

Interestingly, while the high selection pressure of CE-CULL is detrimental to diversity, the method still makes an important contribution to the combined method. Diversity enhancement by the other components allows the exploitive tendency of CE-CULL to flourish safely. Thus, the components are leveraging each other.

While CENE significantly exceeds other advanced methods on all metrics, the difference is most pronounced in terms of evolution speed. In problems where it is impractical to continue evolution until the learning curve levels out, speed of evolution is the most important criterion. Indeed, CE-CULL and CE-TEACH work directly to speed up evolution by avoiding fitness evaluations. CE-CULL substitutes behavior comparison for decisions about offspring that would otherwise require such evaluation. CE-TEACH uses behavior modification to improve the fitness of otherwise unfit offspring, which increases the number of target genotypes, widening the basins of attraction, and thus requiring fewer evaluations.

Resampling

Finally, without the resampling methods utilized in this work, measures of statistical significance would not have been reliable. A worthwhile side–contribution of this dissertation is the demonstration that the resampling methods of Section 3.3 are able to produce reliable measures of statistical significance even for multi–modal distributions with long tails, where the assumptions of the standard techniques, such as Student's *t*-test, are not met.

10.2 Conclusion

This dissertation has presented a number of techniques and analyses, all of which follow the same unifying theme: engaging the power of culture through analysis of behavior.

Behavior is the non–artifactual expression of culture. Transmission of behavioral traits from one generation to the next by non–genetic means is a cultural mechanism. Behavior of the neural network phenotypes in this work was consistently represented as sampled input–output mappings, and those representations were found to be sufficient to shape the behavior of offspring in four disparate ways. The resulting CENE methods reduce CPU time even for relatively modest fitness functions. Because the figure of merit measured in this work was the number of fitness evaluations it is clear that CPU efficiency will be even better for the more expensive fitness functions expected from more complex problem domains.

Culture *in vivo* is a complicated and powerful phenomenon. This dissertation has shown that the power of cultural transmission of traits can be harnessed in evolutionary computation, and works well with evolution. This work is a significant step toward incorporating new facets of culture into neuroevolution research. I view these methods as steps on the road to understanding culture *in silico*.

Appendix A

Acrobot Task Detail

The Acrobot is a 2–joint robot described in Section 3.1.2 This appendix provides the details of the Acrobot implementation, for the convenience of implementors.

The Acrobot is described by Sutton and Barto (1998, Sect. 11.3). The equations of motion and enegery are derived in Spong and Vidyasagar (1989, example 6.4.2). DeJong and Spong (1994) develop swing–up strategies in a symbolic form. Friction terms are given in Spong, Lewis, and Abdallah (1993, Introduction, eq. 4 and 5), which is comparable with Slotine and Li (1993, App. I).

The equations of motion of this system are given in a form useful for implementation:

$$
\begin{array}{rcl}
 & & \forall i \in 1..2, \quad \text{let} \\
v_i & \equiv & \dot{\theta}_i \quad \text{angular velocity} \\
a_i & \equiv & \ddot{\theta}_i \quad \text{angular acceleration} \\
S_i & \equiv & \text{coefficient of static friction} \\
T_i & \equiv & \text{coefficient of viscous friction} \\
k & = & m_2 r_2^2 + I_2 \quad \text{constant} \\
 & & \text{calculate} \\
w_i & = & v_i T_i + \operatorname{sgn}(v_i) S_i \\
c & = & l_1 r_2 \cos\theta_2 \\
d_1 & = & m_1 r_1^2 + m_2(l_1^2 + r_2^2 + 2c) + I_1 + I_2 \\
d_2 & = & m_2(r_2^2 + c) + I_2 \\
e_2 & = & g m_2 r_2 \cos(\theta_2 + \theta_1 - \pi/2) \\
e_1 & = & g(m_1 r_1 + m_2 l_1)\cos(\theta_1 - \pi/2) + e_2 \\
h & = & m_2 l_1 r_2 \sin\theta_2 \\
j_1 & = & e_1 + w_1 - h(v_2^2 + 2 v_2 v_1) \\
j_2 & = & e_2 + w_2 + h v_1^2
\end{array}
$$

so the current accelerations are:

$$a_2 = \frac{d_1(\tau - j_2) + d_2 j_1}{d_1 k - d_2^2}$$

$$a_1 = -(d_2 a_2 + j_1)/d_1$$

apply the standard Eulerian update rule:

$$v_i \mathrel{+}= \delta a_i$$

$$\theta_i \mathrel{+}= \delta v_i$$

where:

$\tau \in \{+1, -1, 0\} \equiv$ torque applied at the second joint

$\delta = 0.05\ seconds \equiv$ time increment

$m_i = 1$ mass of link

$l_i = 1$ length of link

$r_i = 0.5$ length to center of mass of link

$I_i = 1$ moment of inertia of link

$g = 9.8$ acceleration of gravity

Appendix B

Inverted Pendulum Task Detail

The Inverted Pendulum is descibed in Section 3.1.1. This appendix provides implementation details. The equations for the simulated system are taken from Whitley, Dominic, and Das (1991):

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

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

where:

$x \equiv$ the cart position; range $\pm\, 2.4\ meters$

$\dot{x} \equiv$ the cart velocity: range $\pm\, 1.5\ meters/second$

$\theta \equiv$ the pole angle

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

$m_p \equiv$ the mass of the pole $= 0.1\ kg$

$m_c \equiv$ the mass of the cart $= 1.0\ kg$

$m \equiv$ the total mass of the system $= 1.1\ kg$

$l \equiv$ half the length length of the pole $= 0.5\ meter$

$F \equiv$ the control force $= \pm 10 Newtons$

$g \equiv$ the acceleration due to gravity $= 9.8\ m/sec^2$

See Ronco (1998, Section 3.4) for a noise spectrum, and Ronco, Arsan, Gawthrop, and Hill (1998) for friction terms.

Bibliography

Ackley, D., and Littman, M. (1991). Interactions between learning and evolution. In (Langton, Taylor, Farmer, and Rasmussen 1991).

Bäck, T., editor (1997). *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA-97, East Lansing, MI)*. San Francisco, CA: Morgan Kaufmann.

Barto, A. G. (1990). Connectionist learning for control. In Miller III, W. T., Sutton, R. S., and Werbos, P. J., editors, *Neural Networks for Control*, chapter 1, 5–58. Cambridge, MA: MIT Press.

Belew, R. K. (1989). When both individuals and populations search: Adding simple learning to the genetic algorithm. In (Schaffer 1989), 34–41.

Belew, R. K. (1990). Evolution, learning and culture: Computational metaphors for adaptive algorithms. *Complex Systems*, 4:11–49.

Belew, R. K. (1993). Interposing an ontogenic model between genetic algorithms and neural networks. In Giles, C. L., Hanson, S. J., and Cowan, J. D., editors, *Advances in Neural Information Processing Systems 5*, 99–106. San Francisco, CA: Morgan Kaufmann.

Belew, R. K., and Booker, L. B., editors (1991). *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

Belew, R. K., and Mitchell, M., editors (1996). *Adaptive Individuals in Evolving Populations: Models and Algorithms*. Reading, MA: Addison-Wesley.

Blickle, T., and Thiele, L. (1995). A mathematical analysis of tournament selection. In (Eshelman 1995), 9–16.

Braun, H., and Zagorski, P. (1994). ENZO-M: A hybrid approach for optimizing neural networks by evolution and learning. In Davidor, Y., Schwefel, H.-P., and Manner, R., editors, *Parallel Problem Solving from Nature: PPSN-III*, 441–451. Berlin; New York: Springer-Verlag.

Cecconi, F., Menczer, F., and Belew, R. K. (1996). Maturation and the evolution of imitative learning in artificialorganisms. *Adaptive Behavior*, 4:27–48.

Cobb, H. G., and Grefenstette, J. J. (1993). Genetic algorithms for tracking changing environments. In (Forrest 1993), 523–530.

Craighurst, R., and Martin, W. (1995). Enhancing GA performance through crossover prohibitions based on ancestry. In (Eshelman 1995), 130–135.

Davis, L. D. (1991). Bit-climbing, representational bias and test suite design. In (Belew and Booker 1991), 18–23.

Dawkins, R. (1989). *The Selfish Gene*. Oxford: Oxford University Press. Third edition.

De Jong, K. A. (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, The University of Michigan, Ann Arbor, MI. University Microfilms No. 76-09381.

De Jong, K. A. (1993). Genetic algorithms are NOT function optimizers. In (Whitley 1993), 5–17.

DeJong, G., and Spong, M. W. (1994). Swinging up the acrobot: An example of intelligent control. In *Proceedings of the 1994 American Control Conference*, 2158–2162.

DeJong, K., and Sarma, J. (1993). Generation gaps revisited. In (Whitley 1993).

DeJong, K., and Sarma, J. (1995). On decentralizing selection algorithms. In (Eshelman 1995), 17–23.

Eshelman, L. J., editor (1995). *Proceedings of the Sixth International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

Eshelman, L. J., and Schaffer, J. D. (1991). Preventing premature convergence in genetic algorithms by preventing incest. In (Belew and Booker 1991), 115–122.

Eshelman, L. J., and Schaffer, J. D. (1993). Crossover's niche. In (Forrest 1993), 9–14.

Forrest, S., editor (1993). *Proceedings of the Fifth International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

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

Goldberg, D. E., and Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. In (Rawlins 1991), 69–93.

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.

Gomez, F., and Miikkulainen, R. (2001). Learning robust nonlinear control with neuroevolution. Technical Report AI01-292, Department of Computer Sciences, The University of Texas at Austin.

Hart, W. E., and Belew, R. K. (1996). Optimization with genetic algorithm hybrids that use local search. In Belew, R. K., and Mitchell, M., editors, *Adaptive Individuals in Evolving Populations*, vol. XXVI of *SFI Studies in the Sciences of Complexity*, 483–496. Reading, MA: Addison-Wesley.

Haynes, T. (1998). Collective adaptation: The exchange of coding segments. *Evolutionary Computation*, 6(4):311–338.

Hertz, J., Krogh, A., and Palmer, R. G. (1991). *Introduction to the Theory of Neural Computation*. Reading, MA: Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Ann Arbor, MI: University of Michigan Press.

Hutchins, E., and Hazlehurst, B. (1991). Learning in the cultural process. In (Langton et al. 1991), 689–706.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press.

Langton, C. G., Taylor, C., Farmer, J. D., and Rasmussen, S., editors (1991). *Proceedings of the Workshop on Artificial Life (ALIFE '90)*. Reading, MA: Addison-Wesley.

Louis, S. J., and Johnson, J. (1997). Solving similar problems using genetic algorithms and case-based memory.. In (Bäck 1997), 283–290.

Lunneborg, C. E. (1999). *Data analysis by resampling: Concepts and applications*. Pacific Grove, CA: Brooks-Cole.

MacLennan, B. (1991). Synthetic ethology: An approach to the study of communication. In (Langton et al. 1991), 631–658.

Mahfoud, S. W. (1995a). A comparison of parallel and sequential niching methods. In (Eshelman 1995), 136–143.

Mahfoud, S. W. (1995b). *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL.

Masters, T. (1993). *Practical Neural Network Recipes in C++*. New York: Academic Press.

McQuesten, P., and Miikkulainen, R. (1997). Culling and teaching in neuro-evolution. In (Bäck 1997), 760–767.

Michel, O. (1996). Khepera simulator version 2.0 user manual. http://diwww.epfl.ch/lami/team/michel/khep-sim/SIM2.tar.gz.

Miglino, O., Lund, H. H., and Nolfi, S. (1995). Evolving mobile robots in simulated and real environments. Technical report, Institute of Psychology, C.N.R, Rome, Rome, Italy.

Minsky, M. (1963). Steps toward artificial intelligence. In Feigenbaum, E. A., and Feldman, J. A., editors, *Computers and Thought*, 406–450. New York: McGraw-Hill.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press.

Moore, D. S., and McCabe, G. P. (1993). *Introduction to the Practice of Statistics*. New York: Freeman. Second edition.

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, D. E., and Miikkulainen, R. (1996a). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, 22:11–32.

Moriarty, D. E., and Miikkulainen, R. (1996b). Efficient reinforcement learning through symbiotic evolution. In Kaelbling, L. P., editor, *Recent Advances in Reinforcement Learning*. Dordrecht; Boston: Kluwer.

Murphy, R. R. (2000). *An Introduction to AI Robotics*. Cambridge, MAMIT Press.

Nolfi, S., Elman, J. L., and Parisi, D. (1994). Learning and evolution in neural networks. *Adaptive Behavior*, 2:5–28.

Nolfi, S., and Floreano, D. (2000). *Evolutionary Robotics.* Cambridge: MIT Press.

Nolfi, S., and Parisi, D. (1993a). Auto-teaching: Networks that develop their own teaching input. In Deneubourg, J. L., Bersini, H., Goss, S., Nicolis, G., and Dagonnier, R., editors, *Proceedings of the Second European Conference on Artificial Life*.

Nolfi, S., and Parisi, D. (1993b). Self-selection of input stimuli for improving performance. In Bekey, G. A., and Goldberg, K. Y., editors, *Neural Networks in Robotics*, 403–418. Dordrecht; Boston: Kluwer.

Nolfi, S., and Parisi, D. (1994). Desired answers do not correspond to good teaching inputs in ecological neural networks. *Neural Processing Letters*, 1(2):1–4.

Nolfi, S., and Parisi, D. (1995a). Genotypes for neural networks. In Arbib, M. A., editor, *The Handbook of Brain Theory and Neural Networks*, 431–434. Cambridge, MA: MIT Press.

Nolfi, S., and Parisi, D. (1995b). Learning to adapt to changing environments in evolving neural networks. Technical Report 95–15, Department of Neural Systems and Artificial Life, Institute of Psychology, CNR – Rome.

Nordin, P., and Banzhaf, W. (1995). Complexity compression and evolution. In (Eshelman 1995), 310–317.

Rawlins, G. J. E., editor (1991). *Foundations of Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

Reynolds, R. G., and Chung, C.-J. (1997). Regulating the amount of information used for self-adaptation in cultural algorithms. In (Bäck 1997), 401–408.

Reynolds, R. G., Michalewicz, Z., and Cavaretta, M. J. (1995). Using cultural algorithms for constraint handling in GENOCOP. In McDonnell, J. R., Reynolds, R. G., and Fogel, D. B., editors, *Proceedings of the Fourth Annual Conference on Evolutionary Programming*, 289–305. Cambridge, MA: MIT Press.

Ronco, E. (1998). Open-loop intermittent feedback optimal control: A probable human motor control strategy. Technical Report EE-98005, Systems and Control Laboratory, University of Sydney. Submitted to IEEE Transactions on Systems, Man and Cybernetics.

Ronco, E., Arsan, T., Gawthrop, P. J., and Hill, D. J. (1998). A globally valid continuous-time GPC through successive system linearisations. Technical Report EE-98003, Systems and Control Laboratory, University of Sydney. Submitted to Automatica.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, 318–362. Cambridge, MA: MIT Press.

Rychtyckyj, N., and Reynolds, R. G. (2001). Using cultural algorithms to improve knowledge base maintainability. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, 367–374. San Francisco, CA: Morgan Kaufmann.

Sasaki, T., and Tokoro, M. (1997). Adaptation toward changing environments: Why Darwinian in nature? In Husbands, P., and Harvey, I., editors, *Proceedings of the Fourth European Conference on Artificial Life*, 145–153. Cambridge, MA: MIT Press.

Schaffer, J. D., editor (1989). *Proceedings of the Third International Conference on Genetic Algorithms*. San Francisco, CA: Morgan Kaufmann.

Schaffer, J. D., Eshelman, L. J., and Offutt, D. (1991). Spurious correlations and premature convergence in genetic algorithms. In (Rawlins 1991), 265–283.

Sebag, M., Schoenauer, M., and Ravisé, C. (1997). Toward civilized evolution: Developing inhibitions. In (Bäck 1997), 291–298.

Slotine, J.-J. E., and Li, W. (1993). On the adaptive control of robot manipulators. In (Spong et al. 1993), 147–157.

Spector, L., and Luke, S. (1996a). Cultural transmission of information in genetic programming. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996*, 209–214. Cambridge, MA: MIT Press.

Spector, L., and Luke, S. (1996b). Culture enhances the evolvability of cognition. In Cottrell, G. W., editor, *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, 672–677. Hillsdale, NJ: Erlbaum.

Spong, M. W., Lewis, F. L., and Abdallah, C. T., editors (1993). *Robot Control: Dynamics, Motion Planning, and Analysis*. Piscataway, NJ: IEEE.

Spong, M. W., and Vidyasagar, M., editors (1989). *Robot Dynamics and Control*. New York: Wiley.

Stanley, K. O., and Miikkulainen, R. (2002). Efficient evolution of neural network topologies. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*. IEEE. In press.

Steels, L. (1996). Self-organising vocabularies. In Langton, C. G., and Shimohara, K., editors, *Proceedings of the 5th International Workshop on Artificial Life: Synthesis and Simulation of Living Systems (ALIFE-96)*, 179–184. Cambridge, MA: MIT Press.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Syswerda, G. (1991). A study of reproduction in generational and steady state genetic algorithms. In (Rawlins 1991), 94–101.

Weiss, S. M., and Kulikowski, C. A. (1991). *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning and Expert Systems*. San Francisco, CA: Morgan Kaufmann.

Werner, G. M., and Dyer, M. G. (1991). Evolution of communication in artificial organisms. In (Langton et al. 1991), 659–687.

Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In (Schaffer 1989), 116–121.

Whitley, D., Dominic, S., and Das, R. (1991). Genetic reinforcement learning with multilayer neural networks. In (Belew and Booker 1991), 562–569.

Whitley, D., Gruau, F., and Pyeatt, L. (1995). Cellular encoding applied to neurocontrol. In (Eshelman 1995), 460–469.

Whitley, D., and Kauth, J. (1988). GENITOR: A different genetic algorithm. In *Proceedings of the 1988 Rocky Mountain Conference on Artificial Intelligence*, 118–130. Computer Science Department, Colorado State University.

Whitley, L. D., editor (1993). *Foundations of Genetic Algorithms 2*. San Francisco, CA: Morgan Kaufmann.

Vita

Paul Herbert McQuesten was born in Cincinnati, Ohio on November 3, 1946, the son of Beulah White McQuesten and George William McQuesten. After completing McKell High School in South Shore, Kentucky, he attended Michigan State University in East Lansing, Michigan on a National Merit Scholarship. Enlisting in the U.S. Air Force for four years, he had the pleasure of maintaining the system software for an early supercomputer, a Control Data 6600, serial number six. He was honorably discharged as a Staff Sergeant.

For the next eight years Paul did software development and customer support for Control Data Corporation in San Jose and Canada, mostly alternating between operating systems and compilers. He was a co–founder of Sorcim Corporation, where he lead the development of the SuperCalc spreadsheet. After Sorcim was sold in 1984, Paul did software consulting until going to work for Software Publishing in 1990.

In 1992, he moved to Austin, Texas. There he earned the Bachelor of Science degree, *summa cum laude*, in Computer Science at St. Edward's University in 1994.

Also in 1994, he was awarded a Microelectronics and Computer Development Fellowship in the Ph.D. program at the University of Texas. As a Teaching Assistant, Paul taught two sections of the undergraduate introductory programming course, and later became the Head T.A., responsible for proctors who graded all 500 students in the course. He was granted a Master of Science degree in Computer Science in 1996, with a minor in Economics.

In 1999–2001 Paul worked as a data mining specialist at Intelligent Technologies Corporation in Austin.

Paul attended ICGA–95 and FOGA–96 and presented two papers at ICGA–97. He has reviewed papers for FOGA, IEEE Transactions on Neural Networks, Adaptive Behavior Journal, and the 2002 European Conference on Parallel Computing.

Permanent Address: 1170 Glendale Road
Upland, California 91784 USA
http://www.cs.utexas.edu/users/paulmcq
paulmcq@alumni.utexas.net
Department of Computer Sciences
University of Texas at Austin

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay and James A. Bednar.