Fast Reinforcement Learning through Eugenic Neuro-Evolution

Technical Report AI 99-277

Daniel Polani
Institut für Informatik
Johannes Gutenberg-Universität
D-55099 Mainz, Germany
`polani@informatik.uni-mainz.de`

Risto Miikkulainen
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
`risto@cs.utexas.edu`

January 25, 1999

Abstract

In this paper we introduce EuSANE, a novel reinforcement learning algorithm based on the SANE neuro-evolution method. It uses a global search algorithm, the *Eugenic Algorithm*, to optimize the selection of neurons to the hidden layer of SANE networks. The performance of EuSANE is evaluated in the two-pole balancing benchmark task, showing that EuSANE is significantly stronger than other reinforcement learning methods to date in this task.

1 Introduction

Reinforcement learning has been a paradigmatic topic in Artificial Intelligence research for a long time. The ability to learn using just a reinforcement signal is important in many real world applications like robotics, game playing, and traffic and process control, where the correct actions are not known but must be found through exploration.

There are currently two main approaches to reinforcement learning: the value-function approach utilizes incremental dynamic programming to learn evaluations for possible actions in each state (Sutton and Barto, 1998). The evolutionary reinforcement learning approach is based on evolving decision making systems such as neural networks to associate the best possible action directly to each state (Moriarty, 1997). The value-function methods are theoretically well understood, and also have some biological validity (Schultz et al., 1997). However, the evolutionary approach has proven more powerful in a number of benchmark tasks, especially in continuous domains and in domains where the state is incompletely specified. For example, the SANE neuro-evolution method has been shown to be more than twice as fast as Q-learning in the pole-balancing task (Moriarty, 1997; Moriarty and Miikkulainen, 1996).

This paper focuses on further developing techniques for evolutionary reinforcement learning. We introduce EuSANE, a method based on a two-level evolution of hidden neurons and network *blueprints* similar to SANE. However, the blueprint evolution takes place through a *eugenic* evolution algorithm (EuA (Prior, 1998)) which is not restricted to local interaction between chromosomes like the standard GA, but utilizes full population statistics about the allele fitnesses to generate offspring.

Pole balancing has established itself as the standard benchmark for reinforcement learning methods. Unlike many other dynamical systems, it is conceptually simple and intuitive to humans, yet a good representative of real-world control tasks (Barto et al., 1983; Anderson, 1989; Whitley et al., 1993; Pendrith, 1994; Moriarty and Miikkulainen, 1996). However, it is no longer challenging enough for modern reinforcement learning methods and more difficult variants need to be found. One particularly difficult variant is a cart with two poles of different lengths that have to be balanced simultaneously. The 2-pole problem can be solved with direct neuro-evolution methods (Wieland, 1990, 1991), but is still hard enough to pose a challenge to even the more recent composite neuro-evolution methods, such as SANE and Cellular Encoding (Whitley et al., 1995; Gruau et al., 1996a,b).

One particularly difficult variant is a cart with two poles of different lengths that have to be balanced simultaneously. This problem has been first solved successfully with direct neuro-evolution methods (Wieland, 1990, 1991), but composite neuro-evolution methods like Cellular Encoding (Gruau et al., 1996a) and symbiotic evolution (SANE) have proved to be far superior to the direct approaches. This paper uses the SANE neuroevolution algorithm as a starting point, because it appears to be currently among the strongest reinforcement learning method in the pole balancing domain (Moriarty and Miikkulainen, 1996).

In this paper we will first review the SANE method in Sec. 2, comprising the evolution on neuron level and the evolution of network blueprints. We will then present the Eugenic Algorithm (EuA) as a general optimization approach and specialize it to the optimization of the network blueprints in EuSANE. In Sec. 3 we will apply EuSANE to the two-pole-balancing problem and compare the results to other reinforcement learning methods. Finally, the lessons learned and potential for future work are discussed in Sec. 4.

2 The EuSANE Approach

2.1 The SANE Neuro-Evolution Method

Unlike other neuro-evolution approaches, where neural networks are realized either via direct weight encoding (Miller et al., 1989) or via a genotype-phenotype mapping (Kitano, 1990; Polani and Uthmann, 1992, 1993; Gruau et al., 1996a), SANE is not based on evolution of complete networks. Instead, individual neurons evolve, while fitness evaluation takes place only for complete networks. In a way, in SANE a network is a *team* of neurons that is assembled to solve a task. Thus good neurons are those that are able to cooperate with other neurons well enough to solve the problem at hand. This also leads to the view of SANE as a mechanism to support development of *symbiotic* relationships between subsets of the neuron population (Moriarty, 1997; Moriarty and Miikkulainen, 1996, 1997).

In SANE, neurons are evolved for the hidden layer of a three-layer feed-forward network. Each neuron carries its own input and output connections (encoded as a binary string of neuron label/weight pairs). In addition, a population of *network blueprints* is also maintained in SANE, each blueprint specifying those hidden neurons that together form a network (Fig. 1).

Starting with a population of randomly initialized neurons and blueprints, subsequent SANE steps are performed until a network is found that solves the task. In a SANE step, a set of neurons (a team) is selected from the population according to each blueprint to form the hidden layer of a feedforward network. The network is then used to solve the task in a *trial* and attains a fitness reward, depending on its performance.

For every neuron in the current team a trial counter is incremented and the network fitness is added to the neuron's fitness count. After enough trials have been performed that a neuron participates in – say, an average of 10 trials – each neuron's average fitness is evaluated dividing its fitness count by its trial count. This calculation assigns appropriate credit to each neuron in the population. The neurons are then sorted according to their average fitness and each neuron in the top quartile is recombined with a higher-ranking neuron by one-point crossover. The offspring replace the lower-ranking half of the population.

In addition to the neuron evolution, a second level of evolution takes place in a population of network blueprints. From the team point of view one would prefer not to select random neurons to form a hidden layer, but to assemble neuron teams that – according to experience – manage to work well together. Blueprints are chromosomes that specify good neuron team configurations.

If k is the required number of neurons in the hidden layer (a quantity fixed before optimization), a blueprint is an array containing pointers to k neurons from the population. Its fitness is directly the performance of the neural network it defines. Therefore, a GA can be performed on blueprint level in parallel to the GA on neuron level. Because of the particular semantics of the blueprints, a specialized mutation is

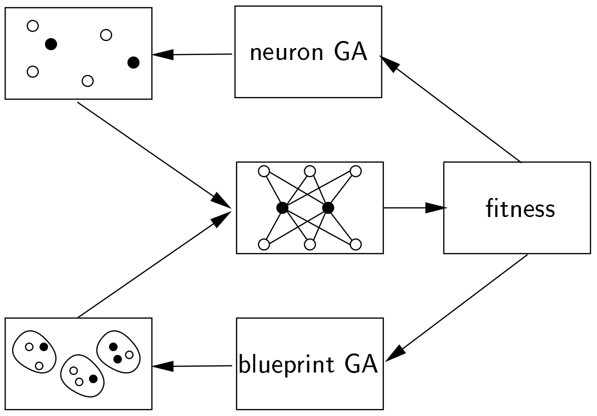

Figure 1: The SANE neuro-evolution method. Network blueprints are evolved to pick good combinations of neurons from the evolving neuron population. The resulting network is evaluated in the task, and the fitness is distributed to the blueprint and the participating neurons.

used that shifts a pointer from a neuron to one of its offspring. This kind of mutation only takes place in offspring blueprints to avoid disruption of well-performing blueprints.

The power of the SANE method originates from the fact that good networks require several different kinds of neurons. The genetic algorithm therefore automatically maintains diversity, which in turn results in more efficient search. SANE has been shown to be a very powerful reinforcement learning method, especially in continuous domains and domains with hidden state information like game playing and robotic control (Moriarty, 1997; Moriarty and Miikkulainen, 1996, 1997). It also forms a good platform for further development of reinforcement learning methods.

2.2 The Eugenic Algorithm

In this paper we present an alternative way to select the neuron teams in SANE. We use a search mechanism inspired by the *Eugenic Algorithm* (EuA) (Prior, 1998). Unlike in the standard GA, where recombination takes place between two chromosomes of the population, in EuA the offspring is constructed based on a statistics over the entire population. In this section we present the general EuA algorithm, and in the next we will apply it to the optimization of the SANE neuron teams.

The core idea of EuA is to select every allele of the offspring separately, based on explicit analysis of the allele fitness distributions in the population. It furthermore contains a restriction operator that focuses the analysis on members of the population most relevant for determining the next allele. In every generation only one new individual is generated, implementing a steady-state replacement.

The EuA algorithm works in the following way: An initial population of random chromosomes is first created[1]. EuA steps are then performed until the termination condition holds. An EuA step has the following structure: It starts with the set of all genes $U = \{x_1, x_2, \ldots, x_n\}$, where x_g denotes the gene g. From this set the *most significant gene* x_{sig} is then selected, that is, the gene most strongly correlated with the fitness of the individuals. The most significant gene is found by computing $|\overline{f(x_{g,1})} - \overline{f(x_{g,0})}|$ for each gene x_g, where $\overline{f(x_{g,a})}$ is the average fitness of all those individuals having allele a at gene x_g. The assumption is that the larger this difference, the stronger the gene's influence on fitness is. For x_{sig}, the algorithm selects an allele value $x_{\text{sig},a}$ according to the selection probability $\overline{f(x_{g,a})}/(\overline{f(x_{g,0})} + \overline{f(x_{g,1})})$ that favors those values that are more likely to lead to good individuals. In addition, the allele selection is subject to certain degree of mutation noise to maintain diversity. After selecting the allele the current gene is removed from U.

Next, a value E roughly approximating *epistasis* is computed for the population. Epistasis is a measure

[1]We will implicitly assume that these are bitstrings, although, in principle, the alphabet used is not limited to $\{0, 1\}$.

of how strongly the fitness of each allele depends on the alleles of the other genes. If the epistasis is high, the alleles for the different genes should not be chosen independently. Therefore, with a probability proportional to the epistasis, the population is restricted to all individuals having the allele $x_{\text{sig},a}$ at gene x_{sig}. When calculating the most significant gene and the selection probabilities for the remaining genes in U, this restricted population is used. Gene and allele selection is repeated until all genes are set, that is, until U is empty. At this point, the specification for the new individual is complete. The EuA step is completed by replacing the individual with the worst fitness in the population by this newly defined one.

Although other methods for constructing individuals based on population fitness statistics have been proposed (Syswerda, 1992; Baluja, 1994), EuA is unique in its restriction operation. A small epistasis indicates that the alleles are not very selective; in this case using the full population to determine allele averages will yield good estimates for the right allele choice. In the case of large epistasis the population should be restricted to the already selected alleles so that more accurate estimates for the fitness averages *conditional to the alleles* already set can be obtained. The restriction operator therefore allows EuA to make more efficient use of population statistics.

The epistasis is estimated in EuA by the value E which is based on the maximum difference $D_{\max}$ of selection probabilities for 0 and 1 alleles over all genes. If it is small, the selection probability differences are small and all allele choices seem equally good. This indicates high epistasis, since in this case it is necessary to look at several genes at once to determine whether fitness will be high or low. If $D_{\max}$ is large, at least one gene has a very clear selectivity, suggesting low dependency on other genes, i.e. low epistasis (Prior, 1998). A rough measure for epistasis can therefore be obtained as $E = 1 - D_{\max}$. Though E is only an approximate epistasis measure (the dominance of certain allele patterns in the population can modify the meaning of E), nevertheless this choice proves to be practical and results in a powerful restriction operator.

EuA has been tested in a number of classical optimization benchmarks such as the 2-D Rosenbrock, subset sum, multiple knapsack, maximum cut, minimum tardy task, and multiple local minima. Comparisons with several variants of hillclimbing, simulated annealing, and GA methods show that EuA is superior to them in many tasks, and generally more consistent (Prior, 1998). It therefore forms a good starting point for improving the neuron selection in the SANE neuroevolution method as well.

2.3 EuSANE

In EuSANE, EuA is applied to the task of optimizing the neuron teams in SANE. In the current implementation of EuSANE, each neuron of the population is represented by one gene in the blueprints. A neuron is included in the hidden layer of the network if the corresponding gene in the blueprint is set to one. This coding requires a slight modification of EuA to ensure that exactly (say) k neurons are selected in the hidden layer. After k alleles in the offspring are set to 1, the rest of the alleles are simply set to 0. If there are fewer than k 1-alleles after all genes have been set by EuA, randomly chosen alleles are set to 1. A blueprint mutation is introduced similar to that of SANE, where randomly chosen neurons are replaced by their offspring.

3 Experiments

3.1 Experimental Setup

The effectiveness of EuSANE was evaluated experimentally in the two-pole balancing task. To be able to compare its performance to a method of known strength, we ran the same simulations also with the SANE algorithm. As inputs, the network was given the six state variables: the angle θ_i of the poles $i = 1, 2$ ($\theta_i = 0^\circ$ indicates an upright pole), the angular velocity $\dot{\theta}_i$, position x ($x = 0$ m is the center of the track) and the velocity $\dot{x}$ of the cart. Gravity was $g = -9.8$ m/s^2, the mass of poles $m_1 = 0.1$ kg and $m_2 = 0.01$ kg, the mass of the cart $m = 1$ kg, the half length of the poles $l_1 = 0.5$ m and $l_2 = 0.05$ m. Output of the network consisted of a single neuron, whose value within $[0, 1]$ was scaled to $[-10\text{ N}, 10\text{ N}]$ to control the system. The simulation of the physical system was based on Euler's method with a time step of 0.01 s. The number of time steps the pole angles stayed within 15^o (i.e. $|\theta_i| \leq 15^o$) and the cart within the track ($[-1.5, 1.5]$ m) was taken as the fitness of the network[2].

Ten different initial positions were used, chosen randomly from an interval of width 0.2 m around zero for the length variables and $0.2 \cdot 180^o/\pi$ for the angular variables. A single trial was considered successful when the network could balance the poles from the given initial position for 120,000 time steps (corresponding to a simulation of 1200 seconds). A network was then considered successful if it was successful in *all* ten positions, which is a very demanding criterion.

3.2 Results

Hundred runs with SANE and EuSANE were performed with the same set of initial conditions. For every run the number of network evaluations required to solve the problem were determined. The results are compared in Fig. 2. The plot shows the cumulative distribution function of the number of evaluations required for the solution obtained during the 100 runs; in other words, y shows that fraction of the 100 runs that were solved in fewer than x evaluations[3].

Fig. 2 shows clearly that EuSANE (solid line) solves the two-pole balancing problem much faster than SANE and to a higher percentage. In fact, 99 from 100 runs are solved by 100,000 evaluations (requiring $\approx$ 19000 evaluations on average), when SANE solved only 30% of them in that time. Only slightly less than 50% of the SANE runs solved the problem at all in the allocated simulation time.

Comparisons to other reinforcement learning techniques on this problem confirm the conclusion that EuSANE is a powerful method. The neuro-evolution approach of Wieland (Wieland, 1990, 1991), requires 150 regular GA generations with a population size of 2048. It is unclear what generation gap has been used there, but, assuming a generation gap of 0.5, Wieland requires approximately 150,000 evaluations to find a network balancing the 2 poles from a single starting position. (Gruau et al., 1996a,b) use only single initial positions and a more complicated fitness function, which makes a comparison somewhat difficult. To get a rough idea, Gruau et al. report an average of 34,000 evaluations to find a successful network, whereas EuSANE required $19,000$ evaluations on average. So EuSANE appears to be performing better on a harder version of the problem.

EuSANE is robust with respect to parameter values and does not require much optimization. In a number of simulations, the number of hidden units and the mutation rates were varied, and found to have little effect, as long as they were in a reasonable range. For example, hidden layers of 5 to 15 hidden units were tried; the performance was somewhat weaker below 8 hidden units, but very similar for larger hidden layer sizes. Such robustness is very useful in applying EuSANE to new applications.

4 Future Work

Enforced Subpopulations (ESP) is another extension of SANE that is currently being developed (Gomez and Miikkulainen, 1997). It is based on enforcing speciation among the SANE neurons. First comparisons with ESP indicate that it is stronger on long runs needing many evaluations; EuSANE, however, has a clear advantage on shorter runs. This result suggests utilizing a *restarting policy* (Mössinger, 1995) with EuSANE: it is possible to calculate an optimal time for termination and restart of an EuSANE run if a solution is not found quickly enough, thereby minimizing the expected value for the total time until solution. This approach gives a clear advantage to methods that find quick solutions in a reasonable percentage of the runs, but need longer time for the rest of the runs, and therefore could improve EuSANE performance even further. We plan to study this approach in future work. Another possibility is to combine EuSANE with ESP; EuA would be used to select neurons from the ESP subpopulations. Finally, many improvements of the EuA itself are

[2]Except for our slightly smaller track size, the settings are the same as in (Wieland, 1990, 1991).

[3]Notice that this plot is different from the standard learning curves that show fitness over time, averaged over a number of runs. The fitness curves of individual runs often vary a lot, because large jumps in fitness take place at different times. Averaging them does not give an accurate picture of the progress of the algorithm. On the other hand, the distribution of evaluations until solution (such as those in Fig. 2) clearly indicate how long the solution can be expected to take.

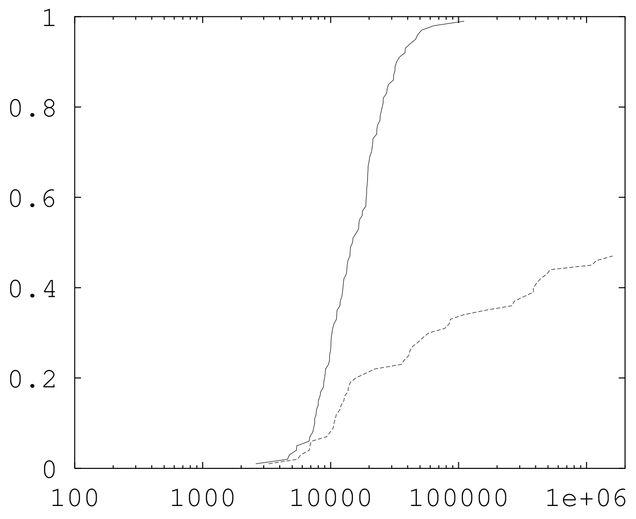

Figure 2: Probability of finding a 2-pole balancing solution by SANE and EuSANE in a given number of evaluations, estimated from 100 runs of each type. The y-value of a curve gives the probability that the task is solved in fewer than x evaluations. The dashed line shows the result for SANE, the solid line for EuSANE. A hidden layer size of $k = 10$ was used in both types of runs.

possible, including gene significance and epistasis based on the quantification of the mutual information of the allele distributions, thereby allowing even better interpretation of the optimization dynamics.

5 Conclusions

In this paper, we presented EuSANE, a variant of the SANE method using the Eugenic Algorithm as a strong combinatorial search method for the blueprint-level optimization. Experimental comparisons showed that it is capable at solving very difficult reinforcement learning problems, such as the two-pole balancing problem, faster than other current reinforcement learning approaches. In future work, we aim at further improving the strategy by introducing a restart policy and extending EuA by information-theoretic methods.

Acknowledgements

We wish to thank Faustino Gomez and John Prior for fruitful discussions. This research was supported in part by NSF under grant #IRI-9504317 and by the B5 program of the University of Mainz.

References

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, 9:31–37.

Baluja, S. (1994). Population-based incremental learning. Technical Report CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213.

Barto, A. G., Sutton, R. S., and Anderson, C. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Trans. Syst. Man., Cybern.*, SMC-13:834–846.

Gomez, F., and Miikkulainen, R. (1997). Incremental evolution of complex general behavior. *Adaptive Behavior*, 5(3-4):317–342.

Gruau, F., Whitley, D., and Pyeatt, L. (1996a). A comparison between cellular encoding and direct encoding for genetic neural networks. In Koza, J. R., Goldberg, D. E., Fogel, D. B., and Riolo, R. L., editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, 81–89. Stanford University, CA, USA: MIT Press.

Gruau, F., Whitley, D., and Pyeatt, L. (1996b). A comparison between cellular encoding and direct encoding for genetic neural networks. Technical Report NC-TR-96-048, NeuroCOLT.

Kitano, H. (1990). Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems*, 4:461–476.

Miller, G. F., Todd, P. M., and Hedge, S. U. (1989). Designing neural networks using genetic algorithms. In Schaffer, D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, 379–384. San Mateo, California.

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

Moriarty, D. E. (1997). *Symbiotic Evolution of Neural Networks in Sequential Decision Tasks*. PhD thesis, The University of Texas at Austin. TR UT-AI97-257.

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

Mössinger, P. (1995). *Vergleich und Optimierung von Genetischen Algorithmen am Beispiel des symmetrischen Subset-Sum-Problems*. Diploma thesis, Johannes Gutenberg-Universität, Mainz.

Pendrith, M. (1994). On reinforcement learning of control actions in noisy and non-markovian domains. Technical Report TR UNSW-CSE-TR-9410, School of Computer Science and Engineering, The University of New South Wales.

Polani, D., and Uthmann, T. (1992). Adaptation of Kohonen Feature Map Topologies by Genetic Algorithms. In R. Männer, and Manderick, B., editors, *Parallel Problem Solving from Nature, 2*, 421–429. Elsevier Science Publishers B.V.

Polani, D., and Uthmann, T. (1993). Training Kohonen Feature Maps in different Topologies: an Analysis using Genetic Algorithms. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, 326–333. San Mateo, CA: Morgan Kaufmann.

Prior, J. W. (1998). *Eugenic Evolution for Combinatorial Optimization*. Master's thesis, The University of Texas at Austin. TR AI98-268.

Schultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1599.

Sutton, R. S., and Barto, A. G. (1998). *Reinforcement Learning*. Bradford.

Syswerda, G. (1992). Simulated crossover in genetic algorithms. In *Foundations of Genetic Algorithms 2*, 239–255. Morgan Kaufmann.

Whitley, D., Dominic, S., Das, R., and Anderson, C. W. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, 13:259–284.

Whitley, D., Gruau, F., and Pyeatt, L. (1995). Cellular encoding applied to neurocontrol. In Eshelman, L., editor, *Genetic Algorithms: Proceedings of the Sixth International Conference (ICGA95)*, 460–467. Pittsburgh, PA, USA.

Wieland, A. (1990). Evolving controls for unstable systems. In D. S. Touretzky, J. L. Elman, T. J. S., editor, *Proc. of the 1990 Connectionist Model Summer School*, 91–102. San Mateo, CA: Morgan Kaufmann.

Wieland, A. (1991). Evolving neural network controllers for unstable systems. In *Proc. IJCNN (Seattle, WA)*, vol. II, 667–673. Piscataway, NJ: IEEE.