ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/2669694

2-D Pole Balancing with Recurrent Evolutionary Networks

Article · August 1998

DOI: 10.1007/978-1-4471-1599-1_63 · Source: CiteSeer

CITATIONS
51

READS
27

2 authors, including:

Faustino Gomez
NNAISENSE SA
65 PUBLICATIONS **2,941** CITATIONS

SEE PROFILE

All content following this page was uploaded by Faustino Gomez on 29 March 2016.

The user has requested enhancement of the downloaded file.

In *Proceedings of the International Conference on Artificial Neural Networks (ICANN-98, Skovde, Sweden).*
New York: Elsevier, 1998.

2-D Pole Balancing with Recurrent Evolutionary Networks*

Faustino Gomez
Department of Computer Sciences
The University of Texas at Austin
Austin, TX 78712
inaki@cs.utexas.edu

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

Abstract

The success of evolutionary methods on standard control learning tasks has created a need for new benchmarks. The classic pole balancing problem is no longer difficult enough to serve as a viable yardstick for measuring the learning efficiency of these systems. In this paper we present a more difficult version to the classic problem where the cart and pole can move in a plane. We demonstrate a neuroevolution system (Enforced Sub-Populations, or ESP) that can solve this difficult problem without velocity information.

1 Introduction

The pole-balancing or inverted pendulum problem has long been established as a standard benchmark for artificial learning systems. For over 30 years researchers in fields ranging from control engineering to reinforcement learning have tested their systems on this task [1–3]. There are two primary reasons for this longevity: (1) Pole balancing has intuitive appeal. It is a real-world task that is easy to understand and visualize. It can be performed manually by humans and implemented on a physical robot. (2) It embodies many essential aspects of a whole class of learning tasks that involve *temporal credit assignment* [4]. In short, it is an elegant environment that is a good surrogate for more general problems.

Despite this long history, the relatively recent success of modern reinforcement learning methods on control learning tasks has rendered the basic pole balancing problem obsolete. It can now be solved so easily that it provides little or no insight about a system's ability. Neuroevolution (NE) systems (i.e. systems that evolve neural networks using genetic algorithms), for example, often find solutions in the initial random population [5, 6].

In response to this need for a new benchmark, a variety of ways to extend the basic pole-balancing task have been suggested. Wieland [7] presented a series of increasingly difficult variations on the standard pole balancing task

*This research was supported in part by National Science Foundation under grant #IRI-9504317.

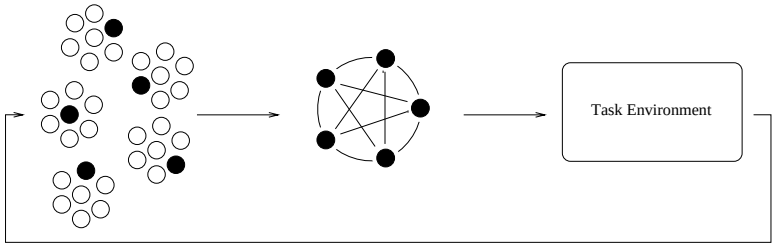

Figure 1: **The Enforced Sub-Populations Method (ESP)**. The population of neurons is segregated into sub-populations shown here as clusters of circles. The network is formed by randomly selecting one neuron from each subpopulation.

culminating in a two pole version. In Gomez and Miikkulainen [5], the Enforced Sub-Populations (ESP) method was shown to be significantly faster than other NE methods on this control problem. Here we present a different, more intuitive and realistic task, where the cart and pole can move in a plane instead of just a 1-D track. We demonstrate how ESP can evolve a fully recurrent network to solve this difficult task even when no velocity information is provided.

2 Enforced Sub-Populations (ESP)

ESP is a neuroevolution system based on Symbiotic, Adaptive Neuro-Evolution (SANE; [8]). Like SANE, it differs from other NE systems in that it evolves a population of neurons instead of complete networks (figure 1). These neurons are combined to form neural networks that are then evaluated on a given problem. ESP differs from SANE in that neurons for the different positions in the network are evolved in separate subpopulations.

Evolution in ESP proceeds as follows:

1. **Initialization**. The number of hidden units u in the networks that will be formed is specified and u subpopulations of neuron chromosomes are created. Each chromosome encodes the input and output connection weights of a neuron with a random string of floating point numbers.
2. **Evaluation**. A neuron is selected from each subpopulation at random to form a network. The network is submitted to a *trial* in which it is evaluated on the task and awarded a fitness score. The score is added to the *cumulative fitness* of each neuron that participated in the network. This process is repeated until each neuron has participated in an average of e.g. 10 trials.
3. **Recombination**. The average fitness of each neuron is calculated by dividing its cumulative fitness by the number of trials in which it participated. For each subpopulation, the neurons are then ranked by average fitness. Each neuron in the top quartile of its subpopulation is recombined with a higher-ranking neuron in the same subpopulation using crossover

and low-probability mutation. The offspring is used to replace the lowest-ranking half of the population.

4. Goto 2.

The key difference between ESP and SANE is that in SANE all of the neurons belong to a single population and are all allowed to mate with each other. This feature allows SANE to sustain diversity and prevent premature convergence. If one type of neuron genotype begins to take over the population, networks will often be formed that contain several copies of that genotype. Because difficult tasks usually require several different hidden neurons, such networks cannot perform well. They incur low fitness, and the dominant genotype will be selected against, bringing diversity back into the population. As a matter of fact, in the advanced stages of SANE evolution, instead of converging the population around a single individual like the standard GA approaches, the neuron population clusters into "species" or groups of individuals that perform specialized functions in the target behavior [8].

ESP builds on the SANE in two ways: (1) It accelerates evolution. The subpopulations that gradually form in SANE are already circumscribed by design in ESP. The "species" do not have to organize themselves out of a single large population, and their progressive specialization is not hindered by recombination across specializations that usually fulfill relatively orthogonal roles in the network. (2) It can evolve recurrent networks more easily. A neuron's behavior in a recurrent network is critically dependent upon the neurons to which it is connected. Since SANE forms networks by randomly selecting neurons from a single population, a neuron cannot rely on being combined with similar neurons across any two trials. A neuron that behaves one way in one trial may behave very differently in another, and SANE receives very noisy information about the fitness of recurrent neurons. The sup-population architecture of ESP makes the evaluation of the neurons more consistent. A neuron's recurrent connection weight r_i will always be associated with neurons from subpopulation S_i. As the sub-populations specialize, neurons evolve to expect, with increasing certainty, the kinds of neurons to which they will be connected. Therefore, the recurrent connections to those neurons can be adapted reliably.

3 The 2D Pole Balancing Problem

Figure 2 shows the two-degree-of-freedom pole balancing system. The objective is to apply force to the cart in both the x and y directions at regular time intervals such that the pole is balanced indefinitely and the cart stays within the track boundaries. The state of this system is defined by eight variables: the angle of the pole from vertical in the x and y directions (θ_x,θ_y), the angular velocities of the pole ($\dot{\theta_x}$,$\dot{\theta_y}$), the position of the cart in the plane (x,y), and the velocity of the cart ($\dot{x}$,$\dot{y}$). The equations of motion for this system are an extension of those found in [7] for the single pole problem with one equation for each principal axis. In all of the experiments we performed the networks were only provided with x, y, θ_x, and θ_y, and the network itself had to determine the

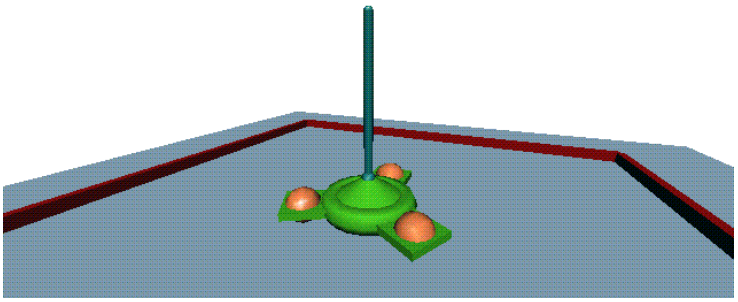

Figure 2: **The 2-degree-of-freedom pole balancing system**. The figure shows a snapshot of a 3D real-time display, available at `http://www.cs.utexas.edu/users/inaki/esp/2dpole-demo`.

Method	Generations	Failures
SANE	579	13
ESP	41	0

Table 1: Comparison between SANE and ESP on the the 2D pole problem without velocities. Results are the average of 50 simulations.

derivatives. This is a realistic version of the problem since only the positions can be observed easily in the real world.

The SANE approach has proven faster and more efficient than other reinforcement learning methods in the basic pole balancing task [6] and ESP has been shown to solve the double pole balancing problem very efficiently [5]. The two-dimensional problem examined here was found to be more difficult to evolve than the double pole problem. There are several factors that make this problem more challenging for NE systems: (1) The mechanical system has two degrees of freedom so that the network must have a vector output. (2) The state space is larger (8 variables instead of 6). (3) Because the networks are not provided with velocity information they need to be recurrent. (4) The networks must evolve to control the system by also evolving to compute the derivatives of the "visible" state variables.

4 Experimental Results

The pole balancing experiments were implemented using the Runge-Kutta fourth-order method with a time step of 0.01s. The pole was 1 meter long and was always started leaning 1 degree towards the northeast with a velocity of zero. This prevented the networks from finding a solution by simply outputting force values close to zero. The force was continuous in the range [-10, 10] N. Fitness was determined by the number of time steps a network could keep the pole within ± 15 degrees from vertical and keep the cart within a 3 meter area. A task was considered solved if a network could balance the poles for 180,000 time steps. Each generation, 600 fully recurrent networks were

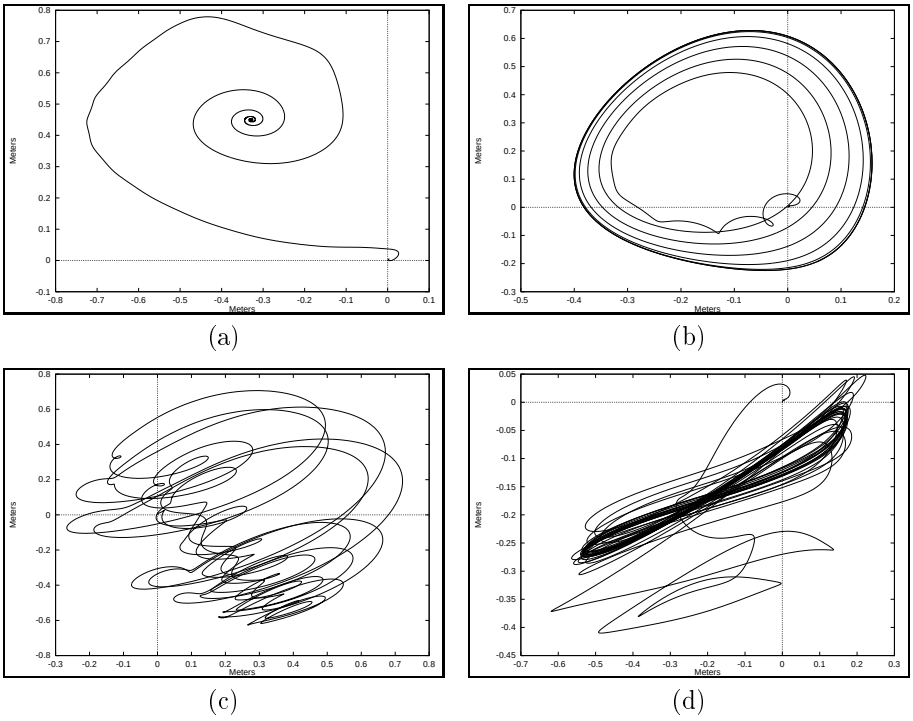

Figure 3: Sample solution trajectories for the 2D pole problem. Paths show the movement of the cart over time starting at the origin.

evaluated. At each time step the networks were relaxed once. Neuron chromosomes were encoded as strings of floating point numbers, and arithmetic crossover was used to generate new neurons. Each chromosome was mutated with probability 0.2, replacing a randomly chosen weight value with a random value within the range [-6.0, 6.0]. These techniques and parameters were found effective experimentally; small deviations from them produce roughly equivalent results.

For the ESP runs the system was restarted from a new random seed if performance ceased to improve over 15 generations. This was done because if ESP converged early it could spend a great deal of time evaluating suboptimal solutions–time which could be better spent retrying with a new population, especially given the speed with which ESP typically solves the task. The total number of generations over all retries was counted as the performance of each run.

Table 1 shows the results for SANE and ESP. A simulation was considered to have failed if a solution could not be found within 1000 generations. SANE failed to find a solution 26% of the time and always took many more generations than ESP when it did succeed. ESP requires 14 times fewer generations to reliably find a solution with restarts occurring 14% of the time.

Figure 3 shows the path of the cart for four different example solutions. All of these solutions begin by moving northeast—the direction the pole is leaning towards initially. Once the pole is brought back towards the vertical position, it has a non-zero velocity and the networks employ a number of different strategies to bring the system under control. In figure 3(a) we see the cart following a spiral path toward a stable fixed point. In (b) the cart spins outward from an unstable fixed point to a stable limit cycle. The paths in (c) and (d) appear chaotic: they follow a regular path that does not repeat itself even for prolonged observations.

5 Discussion and Conclusion

The 2-D pole balancing task without velocities is significant not only because it involves a realistic, non-linear, dynamic environment, but also because it requires memory. In the real-world, an agent rarely has direct access to sufficient state information to act optimally. Many tasks from game-playing to robotics require memory to disambiguate states.

The current task, therefore, can serve as a surrogate with which new methods can be tested. SANE has been shown effective in a variety of domains, including robot arm control, constraint satisfaction, and in controlling chaos [6, 8, 9]. The results presented in this paper show that ESP extends this powerful method by allowing the evolution of recurrent networks and therefore making it applicable to non-Markovian environments. In the future, we plan to apply this system to real-world tasks such as robot navigation and game playing.

References

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

[2] D. Michie and R. A. Chambers. BOXES: An experiment in adaptive control. In E. Dale and D. Michie, editors, *Machine Intelligence*. Oliver and Boyd, Edinburgh, UK, 1968.

[3] J. Schaffer and R. Cannon. On the control of unstable mechanincal systems. In *Automatic and Remote Control III: Proceedings of the Third Congress of the International Federation of Automatic Control*, 1966.

[4] R. Sutton. *Temporal Credit Assignment in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, MA, 1984.

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

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

[7] A. Wieland. Evolving neural network controllers for unstable systems. In *Proceedings of the International Joint Conference on Neural Networks* (Seattle, WA), volume II, pages 667–673, Piscataway, NJ, 1991. IEEE.

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

[9] D. E. Moriarty and R. Miikkulainen. Evolving obstacle avoidance behavior in a robot arm. In P. Maes, M. Mataric, J.-A. Meyer, and J. Pollack, editors, *From Animals to Animats 4: Proceedings of the 4th International Conference on Simulation of Adaptive Behavior*, pages 468–475, Cambridge, MA, 1996. MIT Press.

View publication stats