Symbiotic Neuroevolution for Evolution of Neural Network's Architecture

Alireza rezaee

lecturer of Islamic azad university buin Zahra branch

Phd student of electrical engineering of Amirkabir university of technology

Tehran, Iran

arrezaee@aut.ac.ir

RÉSUMÉ. Alireza rezaee – received the B.Sc degrees in control engineering from Sharif University of Technology,Iran in 2002, and M.Sc degree in electrical engineering from Amirkabir University of Technology,Iran , he is PhD candidate in electrical engineering in amirkabir University of Technology. His main research interests include Dynamic Bayesian network, robotic, navigation,multi agent.

ABSTRACT. In this paper an extension of SANE that simultaneously evolves the weights and architecture of an MLP neural network is presented. The symbiotic adaptive neuroevolution (SANE) system coevolves a population of neurons that cooperate to form a functioning neural network. Evolutionary Strategies (ES) is applied to evolve the network weights. In order to increase the evolving system performance and achieving global optimum convergence the concept of age is introduced in calculating the fitness. The current investigation focuses on evolving neurocontrollers for a nonlinear unstable system (cart-pole problem). The results indicate the suitability for using SANE to evolve weights and architecture of a neurocontroller.

KEYWORDS: Index Terms – SANE, Coevolution, neural network, neurocontroller.

1. Introduction

Neural Networks are commonly used in many applications, due to their intrinsic characteristics, including: capability to capture nonlinear relationships between input and output, celebrated robustness and graceful degradation, etc. Choosing a particular neural network for an application concerns two considerations: first, the neural network architecture, and second, its learning algorithm. Conventional learning algorithms have drawbacks such as: local minima trapping, saturation, initial weight dependence, and over

fitting. Besides, gradient-based algorithms can not be applied in many problems because it does need the error in network outputs as a supervisory signal. In these domains, such information is unavailable or computationally costly to obtain. Moreover, most neural learning methods, being based on gradient descent, cannot search the nondifferentiable landscape of multilayer network's architectures.

the number of weights augments very rapidly with the number of hidden units.

The main advantage of evolved neural networks, as compared with traditional ones, is their ability to learn in spaces where no gradient information is available [4]. This is the case for the landscape of architectures. Some degree of architectural encoding will not only scale better than direct codifications, but will address the search for suitable architectures. This second means of evolution is called the indirect coding scheme, which, in its extreme form, could be defined as the evolutionary approach where only architectural parameters and not weights are coded into chromosomes. For example, connection matrices coding feedforward single hidden layer networks trained by backpropagation have been used successfully [10]. It is important to notice that here backpropagation is not substituted by another algorithm, but combined with it. Many other such combinations have been tried. For example, GAs have also been used for weight initialization. Backpropagation is used later for fine-tuning the initial weights [2].

The indirect coding scheme gets in trouble when assigning fitness to chromosomes since network performance is highly dependent on the initial weight configuration [11, 12]. Much care has to be taken to avoid this noise. The answer may be in schemes between the two extremes (direct and indirect) described above. Yao et al suggested EPNet to solve this problem [3]. The idea behind EPNet is to put more emphasis on evolving ANN behaviors, rather than just its circuitry. Its emphasis is on maintaining behavioral links between a parent and its offspring.

It might even be beneficial not to converge to a best individual, as happens in some control problems where maintaining adaptability is crucial. For example, SANE [1] coevolves a neuron-based population that cooperates to form a neural network. In the FLN [8], on the contrary, gives up hidden units from the very beginning. The computational burden moves from the hidden layer to the input layer. The GA task consists in finding the right subset of polynomial input attributes that leads to an effective separation of the given pattern classes.

Taking into account that, in many problems, the goal is to learn a nonlinear mapping, a feed-forward multi-layer architecture seems to be suitable, most approaches to the genetic design of neural architectures search the space of one-hidden-layer networks. Evolutionary techniques have proved to search efficiently this kind of huge, nondifferentiable, noisy, deceptive, and full of local minima spaces that defeat more conventional search techniques

This article demonstrates the advantages of cooperative, coevolutionary algorithms used in SANE [1]-symbiotic adaptive neuroevolution system- in evolving neural network's weights and architecture simultaneously. This paper is organized as follows. In the next section definition of symbiosis and extension of the SANE approach in evolving architecture is presented. Section three discusses cart-pole balancing problem. Finally section four draws conclusion.

2. symbiotic adaptive neuroevolution

2.1. Symbiosis

As a definition, Symbiosis is a close ecological relationship between the individuals of two (or more) different species. Sometimes a symbiotic relationship benefits both species (Mutualism), in other cases one of them is unaffected or harmed (Commensalism or Parasitism respectively) or both of them are unaffected (Neutralism). In symbiotic adaptive neuroevolution (SANE), symbiotic is defined as a type of coevolution where individuals explicitly cooperate with each other and rely on the presence of other individuals for survival. Hence, the relationship between individuals (neurons) is Mutualism.

2.2. SANE

In almost all approaches to neuroevolution, each individual in the population represents a complete neural network that is evaluated independently of other networks in the population. By treating each member as a separate, full solution, the EA focuses the search toward a single dominant individual. Such concentration can greatly impede search progress in both complex and dynamic tasks. In contrast, the symbiotic coevolutionary approach method restricts the scope of each individual to a single neuron. More specifically, each individual represents a hidden neuron in a two-layer neural network. In SANE, complete neural networks are built by combining several neurons. Since no single neuron can perform the whole task alone, the neurons must optimize one aspect of the neural network and connect with other neurons that optimize other aspects. Evolutionary pressures therefore exist to evolve several different types or specializations of neurons. In this way, the neurons will form a symbiotic relationship. It follows that the evolution performed in SANE can be characterized as symbiotic evolution. Symbiotic evolution is defined as a type of coevolution where individuals explicitly cooperate with each other and rely on the presence of other individuals for survival.

The advantages of symbiotic evolution are twofold. First, the neuron specializations ensure diversity, which discourages convergence of the population. A single neuron cannot "take over" a population, since to achieve high fitness values; there must be other specializations present. If a specialization

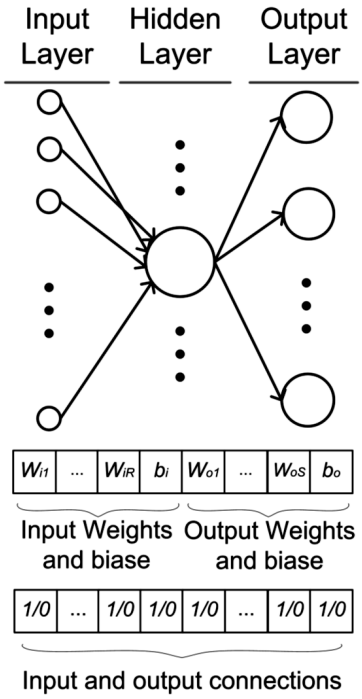

Figure 1. General structure of one hidden layer perceptron neural network. Each neuron in the hidden unit is coded by two vectors. The upper vector represents the weights and the lower one connections.

becomes too prevalent, its members will not always be combined with other specializations in the population. Thus, redundant partial solutions do not always receive the benefit of other specializations and will incur lower fitness evaluations. Evolutionary pressures are therefore present to select against members of dominant specializations. This is quite different from standard evolutionary approaches, which always converge the population, hopefully at the global optimum, but often at a local one. In symbiotic evolution, solutions are found in diverse, unconverged populations. By maintaining diverse populations, SANE can continue to use its recombination operators to build effective neural structures.

Two populations are evolved through SANE, Blueprints and neurons. Blueprints population represents the network chromosomes and neurons populations contain all members that, in combination with each other, build blueprint individuals. Symbiotic evolution is distinct from most coevolutionary methods, where individuals compete rather than cooperate to survive.

Evolution at the neuron level more accurately evaluates the genetic building blocks. We evolve each neuron by two ways, evolving weights and evolving connections. Evolving connections is done through mutation and recombination. In this phase neurons are recombined with many different neurons in the population, which produces a more accurate evaluation of the neural network building blocks. Weights are

updated by evolutionary strategy (described in subsection C). Fig.1 shows the general structure of a one-hidden-layer perceptron neural network. Two vectors assigned to each neuron, one of them represents weight values and the other one shows whether a connection exists or not. Fig.2 shows the main steps in evolution of one generation in SANE. In Standard SANE, no evolutionary pressure exists to build minimal structure. An Extension of this method is developed in order to make a minimal network. As mentioned above each individual in blueprints population represents a neural network by a network, in which each field corresponds to a neuron that cooperates in building that network. In order to evolve structure rather than only neurons' weights, another vector is assigned to neurons that demonstrate the probability of existence of each connection in the next generation. We called this auxiliary vector. This vector is evolved in each generation by means of evolutionary strategy. In each generation, a vector with random variables with uniform distribution between 0 and 1 for each neuron is produced. Having compared this vector with auxiliary vector, connection exist in each neuron will be assigned.

2.3. Evolutionary Strategies

Evolutionary Strategies (ES) are population-based search algorithms developed by Rechenberg and Schwefel [5].ES exploit a population of μ individuals to probe the search space. At each iteration of the algorithm, λ offspring are produced by stochastic variation, called mutation, of recombinations of a set of individuals (called the parents) from the current population. Mutation is typically carried out by adding a realization of a normally distributed random vector. After the creation of the offspring individuals, a selection phase takes place, where either the μ best individuals among the offspring population, or the μ best individuals among both the parent and the offspring populations are selected to form the population of the next generation. These two selection schemes are denoted as (μ, λ)-ES and $(\mu+ \lambda)$-ES, respectively. ES use a set of parameters, called strategy parameters, to parameterize the normal distribution used in

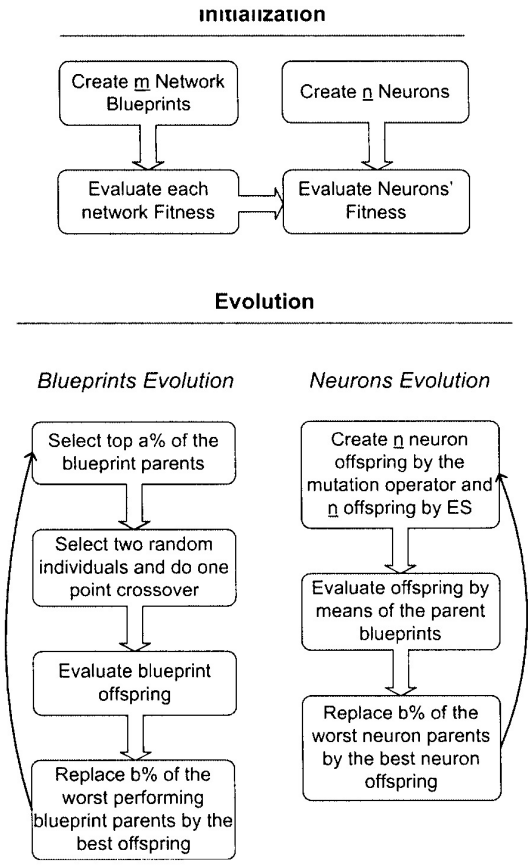

Figure 2. Main steps in evolution of one generation in SANE. Two populations are evolved in iteration: blueprints in which networks are coded and neurons.

the mutation procedure. These parameters can either be fixed, or evolve during the evolution process resulting in self-adaptive ES. Clearly, the parameters of the normal distribution play an important role in the performance of the ES algorithm. Evolutionary strategy described below is applied to evolve neuron weights ([6], [7], [9]).

1. Each individual is a pair of real-valued vectors (η and w). The w is neuron weights and biases vector and η corresponds to the adaptable standard deviations.

2. Create an offspring for each individual: Use the mutation operator which is defined below:

$$\eta_i'(j) = \eta_i(j)\exp\{\tau' N(0,1) + \tau N_j(0,1)\} \qquad (1)$$
$$w_i'(j) = w_i(j) + \eta_i'(j) N_j(0,1)$$

n is the length of η and w vectors and μ is the number of individuals. N(0,1) denotes a realization of a normally distributed one-dimensional random variable with mean zero and standard deviation one. Nj(0,1) indicates that the random variable is sampled a new for each value of the counter j. A realization of a normally distributed one-dimensional random variable having mean zero and standard deviation τ

is given by τN(0,1). The factors τ, τ' are robust exogenous parameters, which have been commonly set to $\sqrt{2\sqrt{n}}$ and $\sqrt{2n}^{-1}$ respectively.

3. Determine the fitness of every individual, including all parents and offspring. Each neuron's fitness is the mean of five best networks' fitness in which neuron cooperated (calculating network fitness is described in detail in subsection D).

2.4.Fitness

fitness function plays a key role in how efficiently and accurately the problem is solved. An improper fitness function may cause the system to completely malfunction, for example, if we use only "1/error" as fitness function in a manipulator control problem the evolutionary algorithm tends to make neural network just to make steady state error low and leave transition response which is crucial in many problems . In fact, with improper fitness function the neural network does not learn the rules governing the system. There is another problem in this evolving system. Algorithm starts with simple structure. The more complex structures exist in the population if the simpler networks can not make a good fitness. However a simple structure with evolved connection and weights may perform better than a complex structure which has not access to the correct weights during the evolution process yet. This problem leads to remain simple structures in the population and make the system to converge a suboptimal solution. To solve this problem the concept of age is introduced. Age of networks is involved in calculating each blueprint's fitness. Thus, more complex structures which are appropriate to find the optimal problem can exist. More discuss on how efficiently choose the fitness function is involved in next section for a real world problem. The fitness of a network blueprint is determined as follows

$$f = f_{perf} + \alpha . f_{comp} + \beta . f_{age} \quad (2)$$

where the terms are the complexity, performance, and the age of the blueprint respectively. Parameter α can be tuned in order to find a compromise between the complexity of the architecture and its corresponding performance. Any $\alpha << 1$ value will guarantee that performance will never be sacrificed for architectural simplicity. Parameter β shows how the age of the blueprint affects the overall fitness and is problem dependent. For a one-hidden layer perceptron complexity fitness can be determined as follows

$$f_{comp} = f_{comp,\min} + N_{connec} + (R+S).N_{neuron}$$
$$f_{comp,\min} = (R+S)N_{neuron,\min} \quad (3)$$

where the terms are minimum complexity assumed to be capable of solving the problem, the number of all connections in the network, and the number of neurons multiplied by the total of inputs and outputs.

Three terms in Eq.2 can be normalized before adding together represents the final fitness value.

3. CART-POLE PROBLEM

Balancing pole is a benchmark of nonlinear unstable systems. The task is to balance the system such that the pole does not fall beyond a predefined vertical angle (20 degrees) and cart remains within the bounds of the horizontal track (0.5 meters).

In this problem, the neural network produces output control signal which is the applied horizontal force to cart. Another alternative is to use a network which produces PID gains for a controller rather than producing the control signal itself.

As mentioned in preceding section designing appropriate fitness function for a specific problem play a vital role in determining how accurate and efficient the system work. The fitness function is chosen to be the time system can hold the pole beyond a predefined vertical angle and cart within the bounds of the horizontal track. At the first glance this fitness function may seem too simple to guide the evolutionary programming to find the desired solution. However evaluating networks with random initial condition (i.e. randomly choose cart and pole position and velocity), and then calculating fitness function as described causes evolutionary process to find the solution. More precisely, choosing random initial condition prevents networks which just learn specific function (for a particular initial condition) and not general rule governing the system from gaining high fitness.

The equations of motions for cart-pole problem are simulated. A one-hidden layer perceptron neural network is fed with the four states, cart and pole position and velocity. Fig.3 shows the fitness evaluation over the generation. Fig.4 depicts the performance of the system in balancing pole with random initial condition.

4. conclusion

This article demonstrates the advantages of cooperative, coevolutionary algorithms used in SANE-symbiotic adaptive neuroevolution system- in evolving neural network's weights and architecture simultaneously. The SANE system incorporates a coevolutionary search strategy called symbiotic evolution to form neural networks in

difficult decision tasks. This specialization is the heart of SANE's search efficiency. By reducing the solution space for each individual, SANE searches in parallel decompositions of the complete neural network space while maintaining high levels of diversity. To solve the problem of remaining simple structure in the population and allowing more complex structure entering the blueprint population the concept of age is introduced.

Controlling nonlinear unstable systems with neural networks often poses a difficult problem. While the proper behavior of the system may be describable, there may not be any training cases to provide for a supervised learning procedure. This paper suggests SANE approach that can simultaneously adjust both the weights and the architecture of a network. This provides a method for designing a network of sufficient size.

REFERENCES

[1] D. E. Moriarty and R. Miikkulainen, "Forming neural networks through efficient and adaptive coevolution," *Evol. Comput.*, vol. 5, no. 4, pp. 373–399, 1998.
[2] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, pp.1423–1447, Sept. 1999.
[3] X. Yao and Y. Liu, "A new evolutionary system for evolving artificial neural networks," *IEEE Trans. Neural Networks*, vol. 8, pp. 694–713, 1997.
[4] D. B. Fogel, L. J. Fogel, and V. W. Porto, "Evolving neural networks," *Biolog. Cybern.*, vol. 63, pp. 487–493, 1990.
[5] H.-P. Schwefel, *Evolution and Optimum Seeking*. New York: Wiley, 1995.
[6] graph generation system," *Complex Syst.*, vol. 4, no. 4, pp. 461–476, 1990.
[7] S. A. Harp, T. Samad, and A. Guha, "Toward the genetic synthesis of neural networks," in *Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 360–369.
[8] A. Sierra, J. A. Macías, and F. Corbacho," Evolution of Functional Link Networks" IEEE Trans. EC, vol. 5, no. 1, Feb 2001
[9] H.Talavati.Far, K.razi, M.B.Menhaj," A Self-tuning Controller for TeleoperationSystem using Evolutionary Learning Algorithms in Neural Networks", unpublished. 2005.
[10] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms Appl.*, 1989, pp. 379–384.
[11] H. Kitano, "Designing neural networks using genetic algorithms with
[12] graph generation system," *Complex Syst.*, vol. 4, no. 4, pp. 461–476, 1990.
[13] S. A. Harp, T. Samad, and A. Guha, "Toward the genetic synthesis of neural networks," in *Proc. 3rd Int. Conf. Genetic Algorithms and Their Applications*, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 360–369.

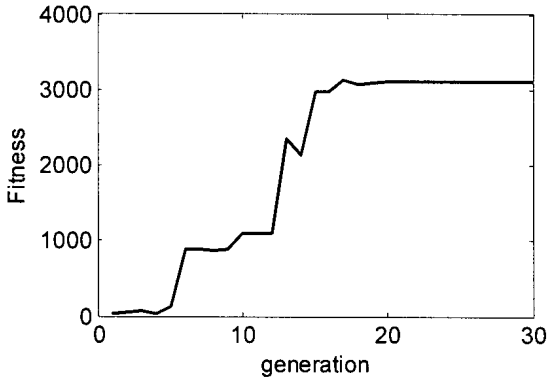

Figure 3. Fitness over generation

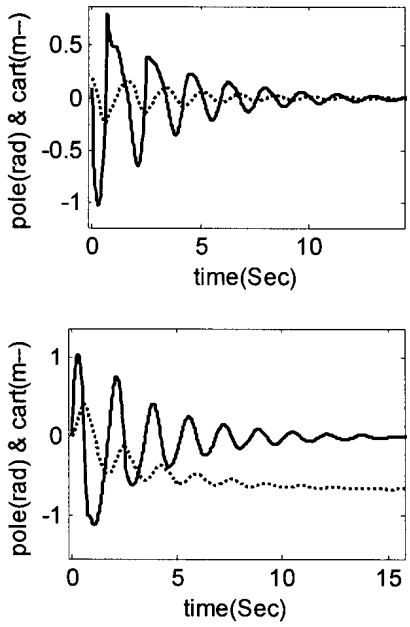

Figure 4. pole and Cart position and velocities