Comparison between Lamarckian evolution and Baldwin evolution of Neural Network

Dr. Imad F. T. Yaseen
Head of Dept. Computer information systems,
Al-Rafidain University College.

Qabas abdul Z.Inazy
Computer Science Department, Al-Mustansiriah University
Qabas_a@yahoo.com

ABSTRACT

Genetic algorithms are very efficient at exploring the entire search space; however, they are relatively poor at finding the precise local optimal solution in the region at which the algorithm converges. Hybrid genetic algorithms are the combination of learning algorithm (Backpropagation), usually working as evaluation functions, and genetic algorithms. There are two basic strategies in using hybrid GAs, Lamarckian and Baldwinian evolution. Traditional schema theory does not support Lamarckian learning, i.e., forcing the genetic representation to match the solution found by the learning algorithm. However, Lamarckian learning does alleviate the problem of multiple genotypes mapping to the same phenotype. Baldwinian learning uses learning algorithm to change the fitness landscape, but the solution that is found is not encoded back into the genetic string. We presented hybrid genetic algorithms for optimizing weights as well as the topology of artificial neural networks, by introducing the concepts of Lamarckian and Baldwin evolution effects. Experimental results with extensive set of experiments show that the hybrid genetic algorithm exploiting the Baldwin effect more effect than Lamarckian evolution but is slow in convergence, and The results of the proposed algorithms outperformed those of the previous algorithms.

Keywords: Lamarckian Evolution, Baldwin Effect, Artificial Neural Networks, Hybrid Genetic Algorithm.

I. Introduction:

The remarkable adaptation of some living creatures to their environments comes as result of the interaction of two processes, working at different time scales: evolution and lifetime learning. Evolution is a slow stochastic process that takes place at the population level and determines the basic structures of an organism. Lifetime learning works by tuning up the structures of an individual, by a process of a gradual improvement of the individual's capability of adaptation to its surrounding environment. Several breakthroughs in the Artificial Intelligence area have occurred, namely in terms of new optimization procedures, based on analogies with the evolution of natural living systems. Problem solving techniques, such as the Artificial Neural Networks (ANNs) and Genetic and Evolutionary Algorithms (GAs), have already on their shoulders interesting results on a broad set of scientific and engineering tasks, such as Combinatorial and Numerical Optimization, Pattern Recognition, Computer Vision or Robotics. In terms of a computational procedure, evolution seems suitable for global search, while learning should be used to perform local search [Bia94].Feed forward Neural Networks (FNNs) are one of the most popular ANN architectures, where neurons are grouped in layers and only forward connections exist. This provides a powerful connectionist model that can learn any kind of continuous nonlinear mapping, with successful applications such as Time Series Forecasting, Medical Diagnostics or Handwritten Recognition, just to name a few. The interest in supervised learning to problem solving and FNNs was stimulated by the advent of the Backpropagation algorithm [Rum86]. However, these procedures are not free from escaping to local minima when the error surface is rugged. Moreover, most of these algorithms strongly depend on problem specific parameter settings.

On the other hand, GAs are suited for combinatorial tasks, where the exhaustion of all possible solutions requires a huge computation. GAs perform a global multi-point search, quickly locating high quality solutions, being able to escape from undesired local minima [Ima00]. The use of evolutionary search as an ANN learning method may overcome gradient-based handicaps, but convergence is in general much slower, since these are general-purpose methods. In the past, GAs have presented difficulties in fine-tuning, being over-taken by other techniques, like the gradient descent methods) [Sha96] .Nevertheless, GAs have also shown advantages in training, under domains where gradient information is difficult to obtain (e.g., Recurrent Networks) [Mak99]. When using gradient-based local search methods to train neural networks has

difficulties in (a) escaping from local optima when the search surface is rugged, (b) finding better solutions when the surface has many plateaus (gradient is zero, for example), and (c) deciding the search direction when gradient information is not readily available (lack of teacher signals, for example).

It is widely believed that evolutionary search [Whi94], [Yao93] and simulated annealing [Coh97] can overcome the above difficulties, but these algorithms may require a large number of iterations in order to obtain an acceptable solution. However, this difficulty could be overcome by combining the efforts of local search (learning) and evolutionary search as they could complement each other. The idea of combining neural and evolutionary learning has received much attention in the last few years. The paper is organized as follows. We first give an overview of tow basic strategies. We then present the results of experiments on some classical problems. Finally, we conclude with some pointers to related work and a summary of our contributions.

The concepts of Lamarckian evolution and the Baldwin Effect which are explained in Section 2, Section 3 Evolutionary Training of Artificial Neural Networks, Section 4 described the our proposed hybrid genetic algorithm structures while last Section tested on three different problems: XOR problem, Adder problem and 6 input parity problem.

II. Lamarckian Evolution versus the Baldwin Effect

Many of the hybrid genetic algorithms that combine local search and genetic search use what is in effect a form of Lamarckian evolution and Baldwin Effect.

A. Lamarckian Evolution:

Learning is a process that involves the interaction between an individual and its environment. Through the experience of this interaction, the behavior (expressed by the phenotype) of an individual is adapted accordingly. In Lamarckian evolution, an individual can pass the knowledge (observed in the phenotype) acquired through learning to its offspring genetically (encoded in the genotype) [Bel92].where the genetic information itself, is called the genotype of the individual. The result, the individual, is called phenotype. The same genotype may result in different phenotypes.

As learning takes place in phenotype space**,** Lamarckian evolution requires an inverse mapping from the phenotype space to the genotype space, which is impossible in biological systems.

Lamarckian evolution was adopted in the hybrid algorithm [Yao93] to evolve feed forward networks. In each iteration of the evolution process, the hybrid algorithm selects a network from the population and trains it by back propagation for a fixed number of

epochs. If the training improves the network's performance, the trained network together with its associated fitness will be put back into the population for further evolution. The process is similar to Lamarckian evolution because the genotype and fitness of a chromosome are modified by learning, resulting in a transfer of knowledge to the offspring.

B. Baldwin Effect:

As Lamarckism cannot be found in biological systems, another school of thought is to use a more biologically plausible mechanism based on the Baldwin effect. In contrast with Lamarckian learning, Baldwinian learning cannot modify the genotypes directly. Only the fitness is replaced by the 'learned' fitness (fitness after learning).

Hinton and Nowlan [Hin87] are the first to use Baldwinian learning for evolving neural networks. In their experiments, random search is applied to every neural network generated by evolutionary search. The random search does not change the network; rather, only its fitness is updated to reflect the distance from the global optimum. Their experimental results show that using evolutionary search alone cannot find the global optimum, while the hybrid algorithm is able to do so.

III. Evolutionary Training of Artificial Neural Networks:

One of the recent developments in the field of artificial neural learning is that of using the technique of evolutionary search instead of conventional learning procedures for training artificial neural nets (ANNs)[Whi95].

The major idea underlying these approaches is to interpret the weight matrices (or vectors) of the ANNs as genotypes and to change the connection weights by means of specific evolutionary operations. Typically, Holland's genetic algorithm or variants of it have been used.

Perhaps the most striking argument for evolutionary network training is that this technique, in contrast to the conventional gradient descent learning procedures, inherently tends to avoid getting stuck in local minima of the error surface over the weight space (where the error is defined as the difference between the actual and the desired network outputs). Evolutionary training was successfully applied to tasks like the XOR problem / N input parity /adder problem.

Generally, all the approaches mentioned above (no matter which type of representational scheme they are using) realize a crossing of neural and evolutionary learning according to the hybrid genetic algorithm shown in Figure (1). Of course, the individual approaches differ greatly in detail; this concerns, in particular, the population size (ranging from one upwards), the parent-offspring replacement strategy (e. g. with/without substitution), the

evolutionary operators selection, cross-sover and mutation), the learning procedures (typically back propagation, the performance criteria (which can be distinguished into learning criteria speed and accuracy and cost criteria), and the desired results with regard to these criteria (e. g. maximal speed in a minimal network").

1. Creation of the next population of ANNs by means of fitness oriented reproduction. This inc-ludes both selection, mutation and recombi-neation. (The initial population is created at random.)
2. Training of the ANNs by conve ntional neural learning procedures.
3. Evaluation of the ANNs' fitness values according to some given performance criteria.
4. If the desired result is obtained then stop, otherwise go to step 1.

Figure (1) the hybrid genetic algorithm.

IV. The Proposed hybrid genetic algorithm:

The two models can be used in evolutionary computation applied to neural networks. Learning is part of the fitness evaluation when searching for a good set of weights for a given architecture, a significant speed-up and final quality of solution can be achieved. In addition, when using Hybrid Genetic Algorithm (HGA) to optimize architectures, learning can increase performance, the general structure of Hybrid Genetic Algorithm as Lamarckian (HGAL) and Hybrid Genetic Algorithm as Baldwin (HGAB) is shown in figure (2). For learning, apply the Backpropagation algorithm.

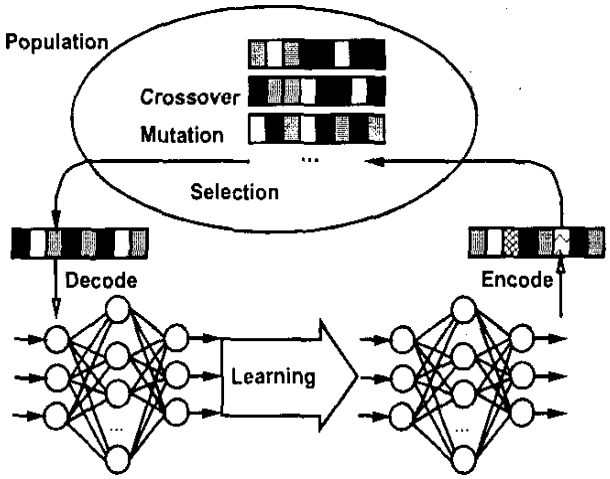

Figure (2) the general structure for hybrid genetic algorithms

The algorithm here proposed for belief revision extends the standard genetic algorithm in two ways:

GA (Fitness, max.gen, p, m)
Fitness: a function that assigns an evaluation coded as a chromosome
max-gen: the maximum number of generations before termination
p: the number of individuals in the population
m: the fraction of the population to be mutated at each step

1. (Hybrid Genetic Algorithm as Lamarckian HGAL):

Initialize population: P <— generate p hypotheses at random
Evaluate: for each h in P, compute Fitness[h])
gen <— 0
While gen < max-gen
Create a new population by
Selection (Roulette Wheel).
Crossover.
Mutate.
Learning : by the backpropagation procedure
Replacement NewP<---OldP
Return P with the highest fitness.

2. (Hybrid Genetic Algorithm as Baldwin HGAB):

Initialize population: P <— generate p hypotheses at random
Evaluate: for each h in P, compute Fitness[h])
gen <— 0
While gen < max-gen
Create a new population by
Selection (Roulette Wheel).
Crossover.
Mutate.
Replacement New P<---Old P
Learning: by the backpropagation procedure.
Return P with the highest fitness

V. Problems Description and Results:

a) For the XOR problem, we used a three-layer network consisting of 2 input, 2 hidden, and 1 output neurons. We applied sigmoid activation functions with output in the range [0,1].The weights were randomly initialized within the range [-1,1].

b) For the Adder problems we used a three-layer network consisting of 4 input, 3 hidden and 3 output neurons. Here, we used sigmoid activation functions with output in the range of [0, 1].The weights were randomly initialized within the range [0.7,-0.7].

c) For the 6-input- parity problem, we used a three-layer network consisting of 6 input, 6 hidden, and 1 output neurons. We applied sigmoid acti-veeation functions with output in the range [0,1].The weights were randomly initialized within the range [0.5,-0.5].

These parameters are summarized in Tables (1), (2) and (3). Table (4) give information about two HGAs proposed and We can notice from the results of this tables that average cost error of Baldwin learning is more effective than the Lamarckian learning but is slow in convergence because it take more number of generations than the Lamarckian learning does (high fitness is lowest cost error) and the HGA is successful (100%) for each model and that correct ratio for Baldwin learning is more effective (more optimal solutions).

The results of the proposed algorithms outperformed those of the previous algorithms. The proposed algorithms took much less iterations to converge.

Parametric study	*HGAL*	*HGAB*
Generation number	17	28
Population size	20	20
Mutation rate	0.001	0.09
Crossover rate	0.8	0.6
Max hidden units	5	5
Learning rate α	0.2	0.7
Momentum term η	0.7	0.8
Maximum cost	0.0025	0.0025
Learning cycle	100	100

Table (1) parametric study for XOR problem

Parametric study	*HGAL*	*HGAB*
Generation number	49	60
Population size	40	45
Mutation rate	0.1	0.003
Crossover rate	0.99	0.8
Max hidden units	9	9
Learning rate α	0.8	0.5
Momentum term η	0.8	0.9
Maximum cost	0.0025	0.0025
Learning cycle	500	600

Table (2) parametric study of Adder problem

Parametric study	*HGAL*	*HGAB*
Generation number	37	45
Population size	40	40
Mutation rate	0.009	0.05
Crossover rate	0.7	0.7
Max hidden units	13	13
Learning rate α	0.4	0.9
Momentum term η	0.9	0.9
Maximum cost	0.0025	0.0025
Learning cycle	900	1000

Table (3) parametric study of 6 input parity problem

Problems	*Proposed Method*	*Average cost error*	*Network topology*	*Generation no.*	*Correct ratio*
XOR	***HGAL***	0.008	(2-2-1)	17	0.55
	HGAB	0.004	(2-2-1)	28	0.85
	Ichikawa	0.08	(2-2-1)	30	-
Adder	***HGAL***	0.004	(4-3-3)	49	0.4
	HGAB	0.004	(4-3-3)	60	0.55
	Whitly	2.4	-	100000	-
6 input parity	***HGAL***	0.2	(6-7-1)	37	0.7
	HGAB	0.04	(6-6-1)	45	1.0
	Safaa	0.52	(6-7-1)	300	-

Table (4) Comparison of Lamarckian and Baldwin learning of three problems

The HGA is shown to be a very efficient and effective optimization method, especially when using Baldwin evolution HGAB, which learned the all tasks in less generation number on average and succeeded in all runs.
As is shown in table (4) the following is results and a comparative study between our HGAs method is better than each of Ichikawa's [Ich92] method Whitly's experiment [Whi95], Safaa's method [Saf98] and when we got a proper design with hidden units using proper Average cost error better than that Whitly which requires large cost value in addition to the number of generation . It is also clear that HGAs outperforms others exponents when used with the Lamarckian and Baldwin learning.

VI. Conclusions and Future Research:

The effect and efficiency of Lamarckian and Baldwin evolution methods used in a hybrid genetic algorithm to optimize weights and topology of artificial neural networks were studied in this research. The results of the research proved that the Baldwin evolution method (HGAB) has more effect and was more successful. Lamarckian evolution method requires an inverse mapping from phenotype to genotype, which is (in general) intractable.The results of the proposed algorithms outperformed those of the previous algorithms. The proposed algorithms took much less iterations to converge. The proposed algorithms also proved to have the ability to overcome the effect of the noise to a good extent.

In Future work, one intends to enlarge the experimental domain, by looking at some real-world applications, such as those of *system's control*, *time-series forecasting* or *medical diagnosis.* Some of these problems are typically embedded in dynamic environments, where the learning tasks evolve over time. In addition Comparison between Lamarckian and Baldwin evolution for recurrent neural networks can be applied in future work.

References:

Belew, R. K. J. McInerney, and N. N. Schraudolph (1992) . Evolving networks: Using the genetic algorithm with connectionist learning. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, Artificial Life 2, pages 511–547. Redwood City, CA: Addison-Wesley,.

Bianchini, M. M. Gori, and M. Maggini. (1994) On the problem of local minima in recurrent neural networks. IEEE Transactions on Neural Networks, 5(2):167-177.

Cohen, B. D. Saad, and E. Marom. (1997) Efficient training of recurrent neural network with time delays. Neural Networks, 10(l):ol-59, Connectionist learning. In C. G. Langton, C. Taylor,

Hinton, G.E., and Nowlan, S.J. (1987). How learning can guide

evolution. Complex Systems, 1, 495-502.

Ichikawa Y., Sawa T. *(1992). Neural Network Application for Dir Feedback Controllers.* IEEE Transaction on Neural Networks, Vol. IEEE ASSP Magazine, Vol. 4, No. 2, ppl-22. In Proceedings of the Second Conference on Artificial Life, C.

Imad F.T .Y (2000) .Solution of Some Hard Problems Using Genetic Algrithm (GAs). PhD thesis, Department of Computer Science University of Pune .

Mak, K. Ku, M. and W. Siu. (1999). A Study of the Lamarckian Evolution of Recurrent Neural Networks. IEEE 'Transactions on Evolutionary Computation, 4(1):31-42, April.

Rumelhart, G. Hinton, and R. Williams. (1986) Learning Internal Representations by Error Propagation. In D. Rulmelhart and J. McClelland, editors, Parallel Distributed Processing: Explorations in the Micro structures of Cognition, volume 1, pages 318-362, MIT Press, Cambridge MA.

Saffa.A.Salehea.M.(1998) " Using Genetic Algorithm in Design Neural Networks " Basra University.

Shang Y. and B. W. Wah. ,(1996) Global optimization for neural network training. IEEE Computer, 29(3):45-54.

Whitley, D. V. S. Gordon and K. Mathias (1994) ."Lamarckian Evolution, The Baldwin Effect and Function Optimization." Parallel Problem Solving from Nature III, pp. 6–15.

Whitley.D.(1995) Genetic Algorithms and Neural Networks. In J. Perioux, G. Winter, M.Galan, and P.Cuesta, editors, Genetic Algorithms in Engineering and Computer Science. John Willey & Sons Ltd.,.

Yao. .X. (1993) Evolutionary artificial neural networks. International Journal of Neural Systems, 4(3):203-222.