Genetic algorithms applied to formal neural networks: parallel genetic implementation of a Boltzmann machine and associated robotic experimentations

Pierre Bessiere

GENETIC ALGORITHMS APPLIED TO FORMAL NEURAL NETWORKS:

parrallel genetic implementation of a Boltzmann machine and associated robotic experimentations

P. Bessière
IMAG-LGI / LASCO3
Institut IMAG-Laboratoire de Génie Informatique / LAboratoire de SCiences COgnitives[1]

[1] *Adress: BP53X, F-38041 Grenoble, FRANCE; Phone: (33)76.51.45.72; Fax: (33)76.44.66.75; Email: bessiere@imag.imag.fr*

I. ABSTRACT

In this paper we describe a possible application of computing techniques inspired by natural life mechanisms (genetic algorithms and artificial neural networks) to an artificial life creature, namely a small mobile robot, called KitBorg.

We proposed in a previous work (Bessière 1990) **Probabilistic Inference** as a possible underlying theory or mathematical metaphor for numerous works in the field of formal neural networks.

Probabilistic Inference suggests that any cognitive problem may be split in two **optimization problems**. The first one called the "dynamic inference problem" is an abstraction of "learning", the second one, namely, the "static inference problem", being a mathematical metaphor of "pattern association".

In this previous paper, for instance, Boltzmann machines have been shown to be a special case of probabilistic inference, where the two optimization problems are dealt with using simulated annealing (Kirckpatrick 1983) for the pattern association part and using simple gradient descent for the learning one.

It was, then, suggested that other optimization technics should be considered in that context and especially **genetic algorithms. The purpose of this paper is to describe the state of the art of the investigations we are making about that question using a parallel genetic algorithm.**

We will first recall the principles of probabilistic inference, then, we will present briefly the parallel genetic algorithm and the ways it is used to deal with both optimization problems, to finally conclude about ongoing robotic experimentations and future planned extensions.

II. PROBABILISTIC INFERENCE

Probabilistic inference may be seen as the present state of the art resulting of numerous works aim to use probability theory as a model of inference and decision making in an incomplete and uncertain universe.

In probabilistic inference theory, knowledge is represented using the usual formalism of probability theory. Knowledge is encoded using a set $\Xi = \{X_1, ..., X_n\}$ of variables. The value space of Ξ is called Ω. The basic assumption is that **a knowledge state of a cognitive system is a probability distribution P over Ω.**

Probabilistic inference has to deal with two different problems:

1 - given a knowledge state (a probability distribution P) and some new information (a set of constraints Φ on Ξ), how to infer a new knowledge state (a probability distribution Q) taking into account the new information;

2 - given a knowledge state (a probability distribution P) and values of some of the variables of the set Ξ, how to infer the most probable values (according to P) of the other variables.

The first problem may be called the "dynamic inference problem" because it concerns how the knowledge state changes in order to take into account new information, while the second problem, the "static inference problem", concerns the consistency conditions of a knowledge state at a given time (see (Hunter 1986)).

Dynamic inference problem

Given a set of variables $\Xi = \{X_1, ..., X_n\}$, given Ω its value space, given a prior knowledge state (a probability distribution P) and given some new information (a set of constraints Φ on Ξ); the dynamic inference process has to find a posterior knowledge state (a probability distribution Q).

In the general case, there is an infinity of probability distributions which are potential solutions of this problem. However, all the probability distributions are not equivalent. Some appear to be more "coherent", more "probable", more "interesting" than some others. The function H(Q,P) (called Kulbach entropy, relative entropy or cross entropy) is a way of measuring the "interest"[2] of a given probability distribution Q relative to P and Φ, the smaller H the better Q. H is defined by:

$$H(Q,P) = + \int_{\Omega} Q(\omega) \log \frac{Q(\omega)}{P(\omega)} \, d\omega \qquad \text{[f.1].}$$

According to this, the dynamic inference problem **may** be restated as follow: given P and Φ, find the probability distribution Q which **minimizes** H. It can be shown that if Φ is a consistent set of constraints,

[2] *For a discussion of this crucial point see (Bessière 1990)*

there is one and only one solution Q* to this problem. Finding Q* is not a trivial mathematical problem.

However, for a very important class of problems, where Φ takes the form of a set of real functions ($\Phi = \{f_1, ..., f_m\}$) such that the mean value a_i ($A = \{a_1, a_2, ..., a_m\}$) of every function f_i is known, then it can be shown that the solution take the following form:

$$q^*(\omega) = \frac{1}{Z^*} e^{-\sum_{i=1}^{m} \lambda_i f_i(\omega)} \qquad [f.2]$$

where q* is the density of Q*, λi are the Lagrange multipliers and Z* is a normalizing constant.

Let us take an example: given two variables A (the Age of the captain) and L (the Length of the boat), the problem we want to solve is find A given L or find L given A.

We have: $\Xi = \{A,L\}$; $\Omega = [7,77] \times [4,444]$.

Starting from scratch (no prior information), P_0, the initial knowledge state, is a uniform distribution over Ω (figure 1). Let us suppose that we first learn the mean value of A ($E(A) = m_A$) and the variance of A ($E((A - m_A)^2) = \sigma_A^2$). An infinity of probability distributions over Ω have this mean value and this variance. However one and only one minimizes H: the normal distribution P_1 having this mean value and this variance as parameters (figure 2). If we then learn the mean value and variance of L, by the same process, we get the probability distribution P_2 (figure 3). Iterating this process for all the data we can get about our problem, and especially for information expressing correlations between A and L we will finally get a probability distribution Q which will sum up all the previously acquired information (figure 4). The surface correponding to Q* may be considered as a visualization of the "memory" of the system.*

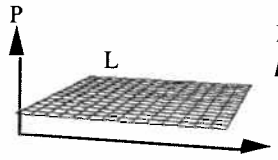

Figure 1

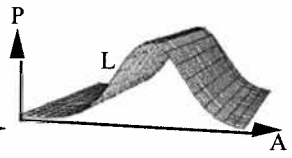

Figure 2

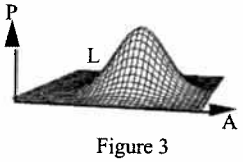

Figure 3

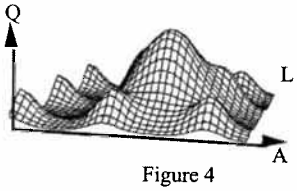

Figure 4

Static inference problem

Given a set of variables $\Xi = \{X_1, ..., X_n\}$, given Ω its value space, given P a probability distribution over Ω and given values of some of the variables of the set Ξ; the static inference process has to find the most probable values of the unspecified variables of Ξ.

According to the previous paragraph, the interesting cases to consider are those cases where P takes form [f.2]. Therefore, finding the most probable values of the non specified variables (i.e. maximizing P) corresponds to the **minimization** of the function:

$$U(\omega) = \sum_{i=1}^{m} \lambda_i f_i(\omega) \qquad [f.3]$$

over the sub-space of Ω defined by the given values on Ξ.

Getting back to our example, this means that for a given value a of the variable A we want to find the most probable value l of the variable L. This process may be visualized by looking for the maximum of the curve defined by the intersection of Q and the vertical plane corresponding to A = a (this curve is in bold on figure 5).*

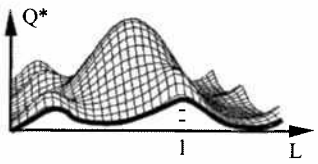

Figure 5

Boltzmann machine

In (Bessière 1990) we show in some details how Probabilistic Inference may be used as an underlying paradigm for numerous Artificial Neural Networks models. Let us just recalled how it is related to Boltzmann machines model.

A usual simplification of probabilistic inference consist in restricting the problem to cases where the X_i are binary variables and where the f_i depend on no more than two variables:

$$f_i(X_1, ..., X_n) = \alpha_i X_k X_l + \beta_i X_k + \gamma_i X_l + \delta_i \qquad [f.4]$$

In that case we get (from [f.2]):

$$Q^*(\omega) = \frac{1}{Z^*} e^{-\sum_{i<j} W_{ij} X_i X_j - \sum_i W_i X_i} \qquad [f.5]$$

where Q* is a 1-Gibbs distribution. Such distributions have well known dynamics with nice simple properties. This is the choice made, for instance, for Hopfield nets (Hopfield 1982) and for Boltzmann machines (Ackley 1988).

The dynamic inference problem has new parameters, the "weights" W_{ij}, in place of the old ones, the Lagrange multipliers λ_i. The search space, which was the space of all probability distributions over Ω, is approximated

by the space of weights' values. The dynamic inference process is replaced by a gradient descent dynamic in the space of the weights' values with H as objective function. One of the remarkable properties of the Gibbsian neural nets is that this gradient descent may be done, without any explicit computation of H, using simply classical local adaptation rules (for instance, Widrow-Hoff's rule for Boltzmann machines) on the system at "thermodynamic" equilibrium (see (Ackley 1988)). Reaching thermodynamic equilibrium itself is strictly equivalent to treating the static inference problem and is done using simulated annealing. U is the objective function but for Gibbsian neural nets, U takes the exact form of an energy function:

$$U = -\sum_{i<j} W_{ij} X_i X_j - \sum_{i} W_{ij} X_i \qquad \text{[f.6].}$$

III. PARALLEL GENETIC ALGORITHM

Genetic algorithms compose a very interesting family of optimization algorithms. Neither their principles, nor their application can be presented here but a very complete description may be found in (Goldberg 1989).

However, let us recall that the standard genetic algorithm has the following form:

Generate a population of random individuals.
While number_of_generations ≤ max_number_of_generations Do
Evaluation - assign a fitness value to each individual.
Selection - make a list of pairs of individuals likely to mate, with fittest individuals listed more frequently.
Reproduction - apply genetic operators to the selected pairs. New individuals produced constitute the new population.

Genetic algorithms are quite easy to implement on parallel machines. Two main approaches, to do so, have been considered so far:

- **standard parallel approach**: In this approach, the evaluation and the reproduction are done in parallel. However, the selection is still done sequentially, because parallel selection would require a fully connected graph of individuals as any two individuals in the population may be mated (Macfarlane 1990).

- **decomposition approach**: This approach consists in dividing the population into equal size sub-populations. Each processor runs the genetic algorithm on its own sub-population, periodically selecting good individuals to send to its neighbors and periodically receiving copies of its neighbors' good individuals to replace bad ones in its own sub-population (Pettey 1987)(Tanese 1987). The processor neighborhood, the frequency of exchange and the number of individuals exchanged are adjustable parameters.

Considering massively parallel architectures with numerous processors, namely, a SuperNode of Transputers, we chose a fine-grained model, where the population is mapped on a connected processor graph like a grid, one individual per processor. This may be considered as an extreme version of the decomposition approach, where the sub-populations are reduced to a single individual. We have a bijection between the individual set and the processor set. The selection is done locally in a neighborhood of each individual. The choice of the neighborhood is the adjustable parameter. To avoid overhead and complexity of routing algorithms in parallel distributed machines, we chose to restrict neighborhood to the only four directly connected individuals. The parallel genetic algorithm proposed is:

Generate *in parallel* a population of random individuals.
While number_of_generations ≤ max_number_of_generations Do
Evaluation - Evaluate *in parallel* each individual.
Reproduction - Each individual reproduces *in parallel* with the best of its' four neighbors.
Selection - Do *in parallel* a selection of best local offsprings.

Another version to this approach has been already proposed in (Mühlenbein 1989), where, furthermore, at each generation a hill-climbing algorithm is executed for each individual in the population.

Detailed description of this work and various benchmarks may be found in (Talbi 1991a), (Talbi 1991b) and (Talbi 1991c). This parallel genetic algorithm implemented on a SuperNode of Transputers shows a remarkable **"superlinear" speed-up**, in the sense that when multiplying both the number of processors and the size of the population by p, the execution time to reach a given quality of solution, is divided by kp (with k>1).

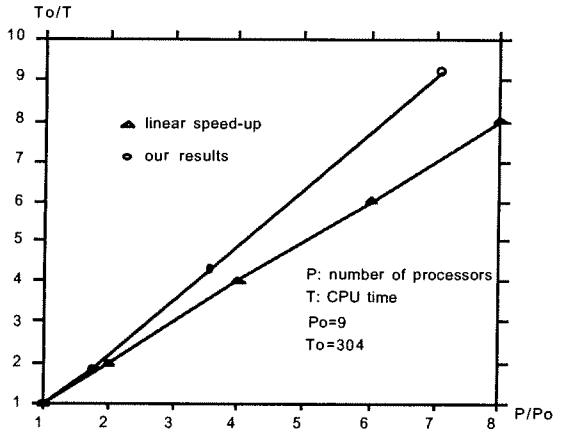

Figure 6

IV. A PARALLEL GENETIC BOLTZMANN MACHINE

Let us now describe how we use the parallel genetic algorithm presented in preceeding section to deal with

the two optimization problems a Boltzmann machine has to face:

Static inference problem

To deal with the static inference problem we replace the usual simulated annealing of Boltzmann machines by our parallel genetic algorithm. We work with a population of 64 individuals, each one being a vector of bits X_i representing the activities of the cells of the network. The search space is 2^N, where N is the number of cells in the network. The fitness function is the global energy function of the network

$$E = - \sum_{i<j} W_{ij}X_iX_j \qquad [f.7]$$

When the Boltzmann machine algorithm searches in parallel on the state of each cell, our algorithm searches in parallel on 64 global states of the network.

The very first comparisons between the two algorithms, made with single processor versions, show same run times' order of magnitude. It should however be noticed that the used version of the parallel genetic algorithm is a very simple and preliminary one, that may certainly be greatly improved. The same super-linear speed-up than for other applications was measured when running the algorithm on more processors. This means than using an implementation on a 128 Transputers MegaNode will leads to an acceleration of the algorithm by a factor of the order of one hundred.

A very remarkable result is that both algorithms (usual simulated annealing and parallel genetic algorithm) found the same attractors or, at least, different attractors but with the same fitness values. This may be considered as an indirect "clue" that both algorithms converge on global optima and do not get stuck in local optima because, in that last case, there would be no reasons for both of them to find two optima with the same fitness values.

Dynamic inference problem

To deal with the dynamic inference problem we replace the usual gradient descent of Boltzmann machines by our parallel genetic algorithm. We work with a population of 64 individuals, each one being a vector of bits representing the set of weights of the network. The search space's size is $Log_2(p)^M$, where M is the number of connections in the network and p the range of the weights. A huge search space, indeed! The fitness function depends on the specific application treated. It evaluates how good is a certain vector of weights, by averaging results obtain when treating the static inference problem on a set of different cases. When the Boltzmann machine algorithm searches sequentially the space of weights, our algorithm searches in parallel on 64 sets of possible weights.

The work is not advanced enough to draw any conclusions from our preliminary benchmarks on the dynamic inference problem. However, we may say that **the genetic parallel Boltzmann machine did learn** simple encoder problems (Ackley 1988).

V. ONGOING ROBOTIC EXPERIMENTATION

Let us finally describe the principle of some ongoing robotic experimentations planned to validate the parallel genetic Boltzmann machine approach. These experimentations are closely related to work describe in a paper of this conference by Dedieu and Mazer.

The prupose of these experimentations is to test a new robot programming paradigm on a small mobile robot named KitBorg developped by the company Aleph Technologies. This new paradigm is describe in details in (Dedieu 1991).

In usual robot programming, the programmer has a model of its' own for the environment and tries to both express the plan of robot's actions and the tasks of the robot's sensors and actuators relatively to his model. A main difficulty with this approach is that the programmer's model of the environment is usually very difficult to match with the sensors informations. For instance, a human programmer would describe the notion of "obstacle" in some geometrical terms, when, in contrast, the robot's sensors will get information about the presence of an "obstacle" through some variation in the intensity of lightenning or optical flow.

The principle of this new robot programming paradigm is based on the assumption that the robot is able to "learn" (using the adequate neural network algorithms) some classification of its' sensori-motor data that will be characteristic of some significant situations. The burden of interpreting these situations is let to the human programmer. The robot imposes its' "view" of the world to the programmer, in contrast with the usual robot programming approach where the human programmer imposes his conception of the environment.

In (Dedieu 1991), some results about the application of Kohonen's map to this task of learning sensori-motor categories are extensively described. We are trying to used the parallel genetic Boltzmann machine to the exact same task. After a learning phase where we classify sensori-motor data, our hope is that we will be able to "recall", in a second phase, adequate actuators command, given some sensors inputs, in order to reached a given objective sensors situation.

This work is not advanced enough to present some results, but we hope to be able to do so soon given that this task will be our main concern in the very next furture.

VI. CONCLUSION AND PERSPECTIVES

One aim of this work was to prove that formal neural networks could work using genetic algorithms to deal with the two optimization problems they have to face. This aim has been reached. **It suggests a totally new way to implement neural networks on**

parallel machines. This seems to be a very promising research track and will certainly be an important concern for us in the near future. We especially intend to improve our parallel genetic algorithm.

More work has to be done to really answer the question: "What optimization technics should be used for the different optimization problems of the various formal neural networks models?". We plan to do more benchmarking to compare the different parts of the two versions of the Boltzmann machine.

More work has also to be done to validate the parallel genetic Boltzmann machine approach on the described robotic experimentations. This task is one of our, present, main concern.

Finally, we hope that we will be able to propose a **cognitive algorithmic model inspired by probabilistic inference and using genetic optimization technics.**

BIBLIOGRAPHY

D. H. Ackley, G. E. Hinton & T. J. Sejnowsky. 1988. *A learning algorithm for Boltzmann Machines* in *Connectionist models and their implications* edited by D. Waltz and J.A. Feldman, Ablex Publishing Corporation.

P. Bessière. 1990. *Toward a synthetic cognitive paradigm: probabilistic inference*. Proc. of COGNITIVA90, Madrid, Spain.

David E. Goldberg. 1989.*Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley Publishing Company.

J. J. Hopfield. 1989. *Neural networks and physical systems with emergent collective computational abilities.* Proceedings of the National Academy of Sciences (U.S.A.).

Daniel Hunter. 1986. *Uncertain reasoning using Maximum entropy Inference*in *Uncertainty in Artificial Intelligence*; edited by L. N. Kanal & J. F. Lemmer. Elsevier Science Publishers.

S.Kirkpatrick, C.D.Gelatt & M.P.Vecchi. 1983. *Optimization by simulated annealing.* Science, Vol.220, No.4598, pp.671-680.

D.Macfarlane & I.East. 1990. *An investigation of several parallel genetic algorithm.* Proc. of the 12th Occam User Group, Exeter, UK, pp.60-67.

H.Mühlenbein & J.Kindermann. 1989. *The dynamics of evolution and learning: Towards genetic neural networks* Connectionism in Perspective, R.Pfeifer et al. eds., North-Holland, pp.173-197.

C.B.Pettey, M.R.Leuze & J.J.Grefenstette. 1987. *A parallel genetic algorithm.* Proc. of the Second Int. Conf. on Genetic Algorithms, MIT, Cambridge, pp.155-161.

E-G. Talbi & P. Bessière. 1991a. *A parallel genetic algorithm for the graph partitionning problem.* A.C.M. International Conference on SuperComputing, Cologne, Germany.

E-G. Talbi & P. Bessière. 1991b. *Superlinear performance of a genetic algorithm on the SuperNode parallel architecture.* S.I.A.M. News.

E-G. Talbi & P. Bessière. 1991c. *Genetic parallel algorithm : performances and applications.* Proceeding of "International Conference on Novel Optimization Technics", Copenhague, Danemark.

R.Tanese. 1987. *Parallel genetic algorithms for a hypercube.* Proc. of the Second Int. Conf. on Genetic Algorithms, MIT, Cambridge, pp.177-183.