ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 21-26

Embedding Knowledge into Stochastic Learning Automata for Fast Solution of Binary Constraint Satisfaction Problems

D. Kontoravdis, A. Likas and A. Stafylopatis

Computer Science Division
Department of Electrical and Computer Engineering
National Technical University of Athens
157 73 Zographou, Athens, Greece

Abstract

We present a model for solving binary constraint satisfaction problems (CSPs) based on stochastic learning automata and an underlying network whose connection pattern represents the CSP. The operation of the learning automata yields fast convergence due to the incorporation of local and global knowledge about the state of the network. The model, referred to as Enhanced Stochastic Automata (ESA) model converges with probability 1 to a solution of the CSP and proves remarkably fast in dealing with large problems.

1 Introduction

A stochastic learning automaton is a finite state machine that interacts with a stochastic environment. The purpose of the automaton is to learn select more frequently the action having the highest reward probability, based on an evaluative signal (reinforcement) received from the environment. Stochastic learning automata have been widely studied in the literature both as single units as well as multiple units interacting with a random environment [5, 6]. In the latter case, either there are no direct interactions between the automata (team approach) or they can serve as interacting connectionist network components.

Constraint satisfaction problems (CSPs) represent typical combinatorial problems that are usually solved with search techniques based on backtracking algorithms [4]. A main characteristic of such techniques is that they require a centralized control by the problem solver making difficult a parallel implementation. On the contrary, the learning automata approach constitutes a high parallelizable process that adapts with time based solely on feedback received in the past. Neural network approaches to solving CSPs share with the automata approach the property of not advocating a hierarchical control. They are typically based on the Hopfield model [2] which suffers from the existence of local

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 21-26

energy minima that do not represent a solution. The above problem can be overcome by the use of a Boltzmann optimizer [1, 3], although this is achieved at the cost of increased execution time, since a slow annealing schedule is required to guarantee convergence to a valid solution.

In [7] the application of a team of stochastic learning automata to the solution of constraint satisfaction problems has been proposed. However, the proposed algorithm suffers from slow convergence especially when the size of the problem was increased. The work presented in this paper concerns a solution technique with fast convergence properties based on the incorporation of knowledge in the operation of learning automata.

2 Stochastic Automata and CSPs

A stochastic learning automaton selects an action from a finite set and receives feedback from the environment evaluating this action. The selection of actions is based on an internally stored distribution over the set of possible actions. This distribution is updated based on the feedback received from the environment and, in turn, is used to generate the next action. Our work is concerned with the case where the evaluative feedback (reinforcement) takes on two values 1 (reward) or 0 (penalty); thus, each action is characterized by a number representing the probability that the environment responds with a reward when that action has been selected.

A constraint satisfaction problem consists in finding an assignment to a collection of variables that satisfies a set of constraints among them. In the following we shall always assume that constraints are binary, i.e., each constraint specifies the values that cannot be taken on simultaneously by a given pair of variables. However, the extension to the general case is not difficult. In solving a CSP with learning automata, every variable is associated with a learning automaton that controls the assignment to that variable. At every instant, each automaton chooses an action which corresponds to the assignment of a value to the associated variable. Following this, the automaton receives reward or penalty according to whether the set of constraints associated to the variable are satisfied or not. Based on the received feedback the automata modify their preferences among the variable values, and the above process is iterated until an assignment of values to all of the variables is found so that no constraints are violated. The feedback received by each automaton is stochastic and this derives from the lack of knowledge about the choices made by other automata. Moreover, every automaton operates in a non-stationary environment meaning that the reward probabilities of the possible actions change with time. This arises from the fact that each automaton adapts its state in parallel with the other automata.

Suppose that we have a team of N automata $V_1, \ldots, V_N$, representing N variables $X_1, \ldots, X_N$ with domains $A_1, \ldots, A_N$ respectively. Without loss of generality we may assume that $|A_1| = \ldots = |A_N| = M$. Let $\alpha_i(n)$ denote the output of automaton V_i at time instant n. This output is drawn from a set of possible actions $A_i = \{\alpha_{i1}, \ldots, \alpha_{iM}\}$ according to an action probability

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 21-26

distribution $p_i(n) = (p_{i1}(n), \ldots, p_{iM}(n))$. In other words the probability that $\alpha_i(n) = \alpha_{ij}$ is equal to $p_{ij}(n)$. The action probability vector $p_i(n)$, called the *state* of automaton V_i at time n, is updated based on the feedback $x_i(n)$ sent by the environment to V_i.

In [7], the *linear reward-penalty* (L_{R-P}) reinforcement scheme [5, 6] is used for updating the action probabilities. Assuming that the output of automaton V_i at instant n is $\alpha_i(n) = \alpha_{ik_i}$ the following changes are specified:

$$p_{ij}(n+1) = \begin{cases} p_{ij}(n) + a(1 - p_{ij}(n)) & \text{if } j = k_i \text{ and } x_i(n) = 1 \\ (1-a)p_{ij}(n) & \text{if } j \neq k_i \text{ and } x_i(n) = 1 \\ (1-b)p_{ij}(n) & \text{if } j = k_i \text{ and } x_i(n) = 0 \\ b/(M-1) + (1-b)p_{ij}(n) & \text{if } j \neq k_i \text{ and } x_i(n) = 0 \end{cases} \quad (1)$$

where $0 < \alpha, b \leq 1$. According to the above scheme if the automaton V_i selects an action and a favorable input results, the action probability of that action is increased while the action probabilities of the remaining actions are decreased. Another interesting observation is that upon failure, the actions not selected tend to become equally probable. In contrast, the change due to success preserves the relative probabilities of all the actions not selected. Let us denote by $p(n) = (p_1(n), \ldots, p_N(n))$ the state of the team of stochastic automata at instant n. It is shown in [7] that under the reinforcement scheme given by equation (1) the discrete time continuous space Markov process $\{p(n)\}_{n \geq 0}$ converges with probability 1 to a solution of the CSP. In practice, however, the average number of steps required by the above technique to converge is particularly large, thus limiting the size of the problems that can be solved.

3 Proposed Learning Model

Our aim is to construct a learning model that achieves a high level of performance in terms of speed of convergence, while at the same time has the property of being free from local minima. To this end, we introduce two concepts, both applying whenever we have an inconsistent assignment of values to the variables.

The first concept is based on the *potentiality* of a selection. Suppose that a variable X_i involved in a CSP is assigned a value that violates some constraints. At the next step it is reasonable to select among the remaining values that variable X_i can take on, the one which, according to the current assignment, violates the smaller number of constraints. In other words we must select the value that appears to have the greatest potential. In terms of the stochastic automaton, upon a failure signal, the actions not selected must not become equiprobable. Instead, the automaton based on the current outputs of all the automata should favor the selection of the action which comparatively violates the smaller number of constraints and, hence, has the possibility of becoming the 'correct' output.

The second concept employed in our model is based on *graded punishment*. The rationale behind this concept is that whenever a set of automata receive a failure signal from the environment, they should be penalized in proportion

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 21-26

to the number of constraints each one violates. The automaton violating the largest number of constraints deserves the largest part of the blame.

In order to incorporate these concepts into the operation of stochastic automata we have developed an approach based on mapping a binary CSP onto the connection pattern of an $N \times M$ unit network. The network offers the possibility of quantifying the above ideas and is defined as follows. Let y_{ij} denote the output of the ij unit, where i refers to the i^{th} automaton V_i and j refers to a_{ij}, the j^{th} action of V_i. The selection of action a_{ij} from automaton V_i at time instant n, is represented by $y_{ij}(n) = 1$. The thresholds θ_{ij} and the connection weights $w_{ij,lm}$ are chosen so as to enforce the constraints of the CSP. In particular, we have $w_{ij,lm} = -\omega$ if $i \neq l$ and actions a_{ij} and a_{lm} belong to the constraint set, while $w_{ij,lm} = 0$ in all other cases. Moreover, $\theta_{ij} = \eta$ where η is a positive constant less than ω.

The notion of potentiality as explained above can be exploited in the following manner through the structure of the underlying network. Suppose that at instant n an automaton V_i selects the action a_{ik_i} resulting in a negative response from the environment. Then for each action a_{ij}, $j \neq k_i$, not selected by V_i, we consider the quantity:

$$\delta_{ij}(n) = [-\sum_{lm} w_{ij,lm} y_{lm}(n) - \theta_{ij}]^{-1} \tag{2}$$

If $\delta_{ij}(n) < 0$ the action a_{ij} could be selected without any constraints being violated. The information contained in the quantities $\delta_{ij}(n)$ can be properly used for each V_i to quantify the relative potential of the actions a_{ij}, in terms of the *potentiality coefficients* $f_{ij}(n)$ lying in the closed interval [0,1] and summing up to unity. If k among the $\delta_{ij}(n)$ are negative then the corresponding coefficients $f_{ij}(n)$ are set to $1/k$, and the remaining coefficients are set to 0. If all $\delta_{ij}(n)$ are positive, the coefficients are obtained through normalization.

The graded punishment concept is also applied in a straightforward manner by means of the network. Suppose that at some instant n there is a set V of automata receiving penalty as an evaluation of their action. For each such automaton V_i let a_{ik_i} be the selected action at that time instant. The following quantities can be computed for each V_i in V:

$$\zeta_i(n) = -\sum_{lm} w_{ik_i,lm} y_{lm}(n) - \theta_{ik_i} \tag{3}$$

It is easy to see that $\zeta_i(n) > 0$ for all V_i. Moreover, the larger is the value of $\zeta_i(n)$ the greater is the number of constraints violated by V_i. Thus, the punishment of each V_i can be performed proportionaly to $\zeta_i(n)$. We define the coefficient $b_i(n)$ as the value of $\zeta_i(n)$ normalized over all V_i in V and multiplied by an experimentally determined factor so that a meaningful punishment is received by each automaton.

We can now modify the penalty portion of the (L_{R-P}) algorithm in (1) to incorporate the ideas introduced in this section. The update of the action

ESANN'1993 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 7-8-9 April 1993, D-Facto public., ISBN 2-9600049-0-6, pp. 21-26

probabilities is then performed using the following rule:

$$p_{ij}(n+1) = \begin{cases} p_{ij}(n) + a(1 - p_{ij}(n)) & \text{if } j = k_i \text{ and } x_i(n) = 1 \\ (1-a)p_{ij}(n) & \text{if } j \neq k_i \text{ and } x_i(n) = 1 \\ (1 - b_i(n))p_{ij}(n) & \text{if } j = k_i \text{ and } x_i(n) = 0 \\ b_i(n)f_{ij}(n) + (1 - b_i(n))p_{ij}(n) & \text{if } j \neq k_i \text{ and } x_i(n) = 0 \end{cases} \quad (4)$$

where $f_{ij}(n)$ are the potentiality coefficients defined above and $0 < a, b_i(n) \leq 1$. In practice it was found that it is better to have $b_i(n) < a$ so that the effect of a penalty response from the environment is much less than that of a reward on the action probabilities. The reinforcement scheme in (4) is a valid update rule by the definition of the quantities $f_{ij}(n)$. Furthermore, it can be proved that under this scheme the Markov process $\{p(n)\}_{n \geq 0}$ converges with probability 1 to a solution of the CSP.

We will refer to this learning model as the Enhanced Stochastic Automata (ESA) model. The ESA model combines the set of learning automata with the underlying network and provides a distributed approach to CSPs. The operation of each automaton is based on local information received from adjacent automata (linked by constraints), as well as on global information characterizing the state of the whole set. The fast convergence of the model to a solution of a CSP, as will be shown in the next section, compensates for the slightly increased complexity and makes it a very promising approach.

4 Results

Simulation studies of the proposed model have been carried out by considering the well known N-queens problem[8]. This problem relates to the placement of N queens on a chessboard so that no queen threatens another. We can represent this problem as a binary CSP with N variables representing the chessboard rows and N values representing the columns in which the queens can be placed. The connections $w_{ij,lm}$ of the network encode the constraints that two queens cannot threaten each other along columns or diagonals. Row threats are automatically disallowed by the learning automata scheme.

All the simulations were performed starting with a uniform distribution of the action probabilities, that is $p_{ij}(0) = 1/N$ for all $i = 1, \ldots, N$ and $j = 1, \ldots, N$. Moreover, tests were carried out for a range of N between 8 and 30. The network parameters were fixed at $\eta = 4$ and $\omega = 1000$ for all experiments, while the best value for the parameter a in equation (4) was found to be 1.0. In order to estimate the search complexity of the ESA model we performed for each N a series of 50 runs and for each run we computed the number of iterations until a solution was found. The results for each value of N averaged over the 50 runs are presented in Table 1, along with a comparison between the ESA and the model presented in [7] which is denoted by PSA (Pure Stochastic Automata).

The results reveal that the convergence of the model is remarkably fast. It should be pointed out that each step of the ESA model is computationally more expensive than that of the PSA, due to the estimation of the quantities $f_{ij}(n)$

N	*Number of steps*		*Speedup*
	PSA	*ESA*	
8	636	74	1.53
10	1665	169	1.70
15	3073	190	1.90
20	4045	230	2.08
25	6532	275	2.20
30	7717	320	4.18

Table 1: Comparative results.

and $b_i(n)$. However, the average time required by PSA to find a solution on a serial machine (Sun Sparc-station) was larger than that of ESA by a factor which increased with N (last column of Table 1). It is expected that the efficiency of the proposed approach will be even more apparent in the case of a parallel implementation. Another interesting observation is that our model compared to the Boltzmann optimizer appeared to be at least 20 times faster on the average.

References

[1] Aarts, E. and Korst, J., *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, John Wiley Sons Ltd., (1989).

[2] Hopfield, J.J and Tank, D.W., *Neural Computation of Decisions in Optimization Problems*, Biological Cybernetics, Vol. 52, pp. 141-152, (1985).

[3] Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P., *Optimization by Simulated Annealing*, Science, Vol. 22, pp. 671-680, (1983).

[4] Kumar, V., *Algorithms for Constraint Satisfaction Problems: A Survey*, The AI Magazine, 13(1), pp.32-44, (1992).

[5] Lakshmivarahan, S., *Learning Algorithms: Theory and Applications*, Springer-Verlag, New York, (1981).

[6] Narendra, K.S. and Thathachar, M.A., *Learning Automata*, Prentice-Hall, (1989).

[7] Ricci, F., *Constraint Reasoning with Learning Automata*, Proceedings 3rd Workshop of AI*IA Interest Group in Automatic Learning, Rome, May 6-8 (1992).

[8] Stone, H.A. and Stone, J.M., *Efficient Search Techniques - An Empirical Study of the N-Queens Problem*, IBM Journal on Research and Development, Vol. 31, pp. 464-474, (1987).