Sādhanā, Vol. 15, Parts 4 & 5, December 1990, pp. 251–262. © Printed in India.

Stochastic networks for constraint satisfaction and optimization

P S SASTRY

Department of Electrical Engineering, Indian Institute of Science, Bangalore 560012, India

Abstract. Stochastic algorithms for solving constraint satisfaction problems with soft constraints that can be implemented on a parallel distributed network are discussed in a unified framework. The algorithms considered are: the Boltzmann machine, a Learning Automata network for Relaxation Labelling and a formulation of optimization problems based on Markov random field (MRF) models. It is shown that the automata network and the MRF formulation can be regarded as generalisations of the Boltzmann machine in different directions.

Keywords. Neural networks; Boltzmann machine; learning automata; consistent labelling problem.

1. Introduction

Many problems in computer vision and pattern recognition can be solved by a process that searches an appropriate space to find a point satisfying a given set of constraints. A special feature of such problems is that the constraints may be 'soft'; that is, it is not possible to categorically assert that a proposed solution satisfies or does not satisfy the constraints. The solution is more appropriately viewed as 'maximizing' the 'degree of satisfaction' of the constraints and hence such constraint satisfaction algorithms have much in common with optimization techniques. Constraint satisfaction has been used in computer vision for a long time (Waltz 1975, pp. 19–91; Davis & Rosenfeld 1981). The importance of soft constraints and the relationship with optimization have been pointed out by many people (Hummel & Zucker 1983; Blake & Zisserman 1987). Here our interest is in stochastic network-based solutions to such constraint satisfaction and optimization problems. Such networks can, by virtue of their massive parallelism, potentially deliver large processing rates needed for good performance in areas such as computer vision.

We begin our discussion with the Boltzmann machine (Hinton *et al* 1984; Rumelhart & McClelland 1986), a simple stochastic neural network that can solve some optimization problems in parallel. By incorporating simulated annealing we can reach global optima through a parallel algorithm. After this we show how this machine can be viewed as solving certain constraint satisfaction problems with soft constraints. Then we give a rigorous formulation of such problems and present a stochastic network, using learning automata, to solve them.

2. The Boltzmann machine

Boltzmann machine is a neural network whose units (or neurons) have stochastic input–output functions. Let the network have N units. The state of unit i, which is also its output, will be denoted by s_i, $1 \leqslant i \leqslant N$, and $s_i \in \{0, 1\}$, $\forall i$. The global state of the network is a vector of states of all units. Every unit in the network is connected to every other unit. Let w_{ij} denote the weight associated with the connection from the jth unit to the ith unit. The connections are symmetric; that is, $w_{ij} = w_{ji} \forall i, j$. At any instant k, the net input into the ith unit, denoted by net$_i(k)$, is calculated as

$$\mathrm{net}_i(k) = \sum_j w_{ij} s_j(k) - \tau_i, \tag{1}$$

where τ_i is called the threshold of the ith unit.

The state of a unit depends on its net input. Units change their state asynchronously in discrete time. At any given instant of time only one randomly chosen unit changes its state. This dynamics is governed by a probabilistic law. Suppose at instant k, unit i is chosen for changing state. Then,

$$p_i = \mathrm{Prob}\,[s_i(k+1) = 1] = 1/\{1 + \exp(-\beta \mathrm{net}_i(k))\} \tag{2}$$

where β is a positive constant.

It is easy to see that the probability of s_i being 1 is more than 0·5 if net input is positive and less than 0·5 if net input is negative. Thus, from (1) and (2), s_i being in state 1 is more probable when the input reaching the ith unit from other units exceeds its threshold. Except for the probabilistic state transition, as given by (2), this network is very similar to the Hopfield Net (Hopfield & Tank 1985). In the Hopfield model also the net input is calculated using (1); but the state is set to 1 if net input is positive and set to zero otherwise.

Let us denote the global state of the machine at k by $S(k) = [s_1(k), \ldots, s_N(k)]^t$. We denote specific global states by S_α, S_β, etc. and $S_\alpha = [s_1^\alpha \ldots s_N^\alpha]^t$, $s_i^\alpha \in \{0, 1\}$ $\forall i$. Thus the state space of the machine is the set of all N-bit binary numbers. By (2), it is clear that $S(k)$ is a Markov chain on this state space. It is easy to show that the steady state distribution of this Markov chain is given by

$$\pi_\alpha = \exp(-\beta E_\alpha)/Z, \tag{3}$$

where

$$E_\alpha = -\tfrac{1}{2} \sum_i \sum_j w_{ij} s_i^\alpha s_j^\alpha + \sum_i s_i^\alpha \tau_i. \tag{4}$$

$\pi_\alpha$is the probability, at steady state, of finding the machine in the global state S_α. Z is a normalising constant. E_α, given by (4), is called the energy of global state S_α. From (3), it is clear that, at equilibrium, global states with lower energy are more probable.

With the energy as defined by (4), it is now possible to reinterpret the dynamics of the machine, given by (2), in an optimization context. The net input to unit i, given by (1), is $E_\alpha - E_\beta$ where the global states S_α and S_β are such that $s_i^\alpha = 0$, $s_i^\beta = 1$ and $s_j^\alpha = s_j^\beta = s_j(k) \forall j \neq i$. By (2), the ith unit prefers state 1 (i.e. $p_i > 0{\cdot}5$) if net$_i$ is positive. Thus the ith unit prefers state 1 if its assuming state 1 will decrease the overall energy. It may be noted that (2) allows a state change by an ith unit that increases the overall energy also, though with a smaller probability. Contrasting this with the Hopfield model, we see that all state changes allowed there have to decrease the global energy. This is true of all gradient descent algorithms that continuously try to decrease the objective function. But in the Boltzmann machine we allow 'uphill moves' also; that is, we allow state changes that increase the energy (or equivalently the value of the objective function). Once again referring to (2), we see that the probability with which uphill moves are allowed depends on the value of the parameter β.

3. Boltzmann machine as an optimization technique

We are now in a position to appreciate the optimization capabilities of the Boltzmann machine. Suppose we have to determine values for N Boolean variables so as to minimize a quadratic objective function. Then we can view the given objective function as our energy and, by putting it in the form of (4), we can decide on the weights w_{ij}. If we now run a Boltzmann machine with these weights, then, at equilibrium, the machine will spend more time in states that result in lower values of the objective function. So, we can pick up minima by, for example, gathering statistics about state occupancy (see, for example, Marroquin *et al* 1987, for an application of this concept in computer vision).

As an example, let us consider the Travelling Salesman Problem (TSP). Given a set of cities and distances between pairs of them, the problem is to find a tour, i.e., a sequence of visiting all the cities without visiting any city twice, so that the tour length, i.e., total distance travelled, is minimum. We can think of any tour as an assignment of a position number to each city. If we have N cities and have one unit to represent each city then we need units that are capable of being in N distinct states so that the global states can be viewed as a possible tour. But since our units take only 0–1 values, we have to reformulate the problem involving only binary decision variables. We will choose our variables as s_{xi}, $1 \leqslant x, i \leqslant N$. The variable s_{xi} governs the decisions of putting city x in position i. We denote by $w_{xi,yj}$, the weight connecting the units xi and yj. It is easy to see that not every possible global state represents a tour. For example, if for a given x, $s_{xi} = 1$ for more than one i then it means we are assigning more than one position to a given city and that is not permitted. Hence in formulating our energy, in addition to incorporating the tour length, we should also incorporate constraints which make sure that energy is large for global states that do not represent a valid tour. With this motivation, a possible energy function is

$$E(S) = (A/2)\sum_x \sum_i \sum_{j \neq i} s_{xi}s_{xj} + (B/2)\sum_i \sum_x \sum_{y \neq x} s_{xi}s_{yi} + (C/2)\left(\sum_x \sum_i s_{xi} - N\right)^2 +$$
$$+ (D/2)\sum_x \sum_{y \neq x} \sum_i d_{xy}s_{xi}(s_{y,i+1} + s_{y,i-1}) \qquad (5)$$

The first term penalises (i.e. results in higher energy of) any global state where more than one position is assigned to the same city. The second term similarly penalises global states that assign two different cities to the same position. Now, if we take $s_{xi} = 0 \ \forall \ x, i$, then the first two terms are zero but we still have not got a tour. So, the third term enforces the condition that there should be exactly N cities on the tour. These three terms together incorporate the conditions for valid tours. The fourth term accounts for the actual length of the tour represented by a global state. Here

d_{xy} is the distance between cities x and y, and the subscripts $i+1$ and $i-1$ should be understood such that $N+1$ is 1 and $1-1$ is N.

If we rewrite (5) in the form of (4) (where w_{ij} now becomes $w_{xi,yj}$), we get

$$w_{xi,yj} = -A\delta_{xy}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{xy}) - C - D\, d_{xy}(\delta_{j,i+1} + \delta_{j,i-1}), \qquad (6)$$

and

$$\tau_{xi} = CN,$$

where

$$\delta_{ij} = 1, \text{ if } i=j,$$
$$= 0, \text{ otherwise.}$$

By proper choice of parameters A, B, C, D one can make sure that global states corresponding to a tour have lower energy than others. Further, due to the last term in (5), tours with smaller lengths have lower energy. If we run the Boltzmann machine with weights and thresholds as given by (6), then, at steady state, the machine will spend more time in global states that correspond to tours with lower length. If we gather statistics about state occupancy at steady state, we can find the solution to the TSP by picking the state with highest probability.

The procedure as outlined above, though offering a solution to an optimization problem in principle, is not very effective in many cases. In TSP, for example, there are 2^{N*N} possible global states and all of them have non-zero steady state probabilities [see (3)]. Hence the probability of finding the machine in the optimal tour might be numerically very small (though still greater than finding the machine in any other state). Thus one needs statistics about a larger number of states gathered over a long time to be able to obtain the solution.

A possible method to overcome at least the storage overheads of gathering statistics, is the use of simulated annealing (Kirkpatrick *et al* 1983).

As explained earlier, the dynamics of the Boltzmann machine allows state changes that increase the energy. The probability with which these uphill moves are made is controlled by the parameter β in (2). If $\beta = 0$ then all state changes (both 'uphill' and 'downhill') are equally probable. On the other hand, as β tends to infinity only those state changes that reduce the energy are possible. In the terminology of simulated annealing, $1/\beta$ is called temperature. To incorporate simulated annealing into the Boltzmann machine, we start the machine with a high temperature (low value of β) where state changes are essentially random. Then slowly, the temperature is decreased with time (i.e. β is increased with k) ultimately approaching zero temperature (β tending to infinity). This process is called cooling. If the cooling schedule, that is, the manner of decreasing temperature with time, is sufficiently slow then, at equilibrium, the machine will be in only those global states which represent global minima of the energy function. A cooling schedule that is sufficient is $T(k) \geqslant A/(B + \log k)$ where A and B are appropriate constants (Mitra *et al* 1986). Here $T(k)(=1/\beta(k))$ is the temperature at instant k. It is easy to see that the temperature needs to be reduced very slowly and hence a large number of iterations are needed to get the temperature sufficiently near zero.

Thus the Boltzmann machine with simulated annealing can find the global minima of quadratic objective functions over Boolean variables but will take enormous time. Unlike the case where the machine is run at constant temperature (and hence the need to gather statistics at the steady state), here the space complexity of the optimization algorithm is very small.

In conclusion, we can see that the Boltzmann machine is an interesting network-based optimization technique. It may be noted that by making the net input of a unit equal to the energy difference between appropriate global states (as explained in § 2), we can think of this as a general technique of optimization. But only in the case of quadratic functions can we write this energy difference as in (1). Viewing this computation as being performed on a network, where each unit decides its state based on the weighted sum of outputs of other units, is possible only when the energy function is quadratic. It may appear that there is not much parallelism here because at each instant only one unit is updated. But if we let each unit be run with its own clock, whose ticks are Poisson-distributed and which is independent of all other clocks, then we can implement the network on a set of processors without need for any explicit synchronism. To that extent it can be viewed as a parallel network.

4. Boltzmann machine and soft constraints

Consider a Boltzmann machine where the states s_i are such that $s_i \in \{-1, 1\}\ \forall i$. Now if we consider the global states S_α and S_β such that $s_i^\alpha = -1$, $s_i^\beta = 1$ and $s_j^\alpha = s_j^\beta\ \forall j \neq i$, then, forgetting the thresholds for the time being, we get $E_\alpha - E_\beta = 2\Sigma_j w_{ij} s_j$. Thus even now, the net input measures the change in global energy and hence by the dynamics given by (2), the machine still prefers downhill moves in trying to minimise the energy given by (4).

Now, the energy to be minimised is $-\Sigma w_{ij} s_i s_j$. Therefore, if w_{ij} is negative it is worthwhile making s_i ad s_j of opposite sign and vice-versa. Thus w_{ij} can be thought of as a constraint on the possible values for s_i and s_j. But this constraint is soft. Thus, just because w_{ij} is negative it may not always be desirable to make s_i and s_j of opposite sign. For example we can have weights w_{ki} and w_{kj} both of which are negative and much larger than w_{ij}. Hence, it is desirable to make the pairs s_i and s_k and s_j and s_k of opposite sign which forces us to make s_i and s_j of the same sign though w_{ij} is negative.

In the next section we give a formulation of such constraint satisfaction problems. If the objective is only to find global minimum of the energy, then the view of constraints satisfaction will not be very appealing. (A constraint satisfaction problem is appealing mostly in the context where there are multiple solutions that 'satisfy' the constraints.) So, we establish a connection between constraint satisfaction and local optima of the energy. Since the energy is defined on a finite set we have to clearly define what we mean by local optima. All these will be discussed in detail in the next section.

5. A mathematical formulation of constraint satisfaction

In formulating problems with soft constraints we follow the treatment of Hummel & Zucker (1983). This class of problems is termed the labelling problems. A general labelling problem is specified by giving

i) A set of objects, $\mathbf{O} = \{O_1, \ldots, O_N\}$
ii) A set of labels, $\Lambda = \{1, \ldots, M\}$
iii) A neighbour relation over $\mathbf{O}$ specifying which pair of objects constrain each other.
iv) A set of compatibility functions that specify the constraints.

The problem is to assign a label to each object such that the assignment is 'consistent' with respect to the compatibility functions. For every pair of objects O_i and O_j that constrain each other (as specified by the neighbour relation), there is a compatibility function, $r_{ij}: \Lambda \times \Lambda \to R$. $r_{ij}(\lambda, \lambda')$ is a measure of consistency of object-label pairs (O_i, λ) with (O_j, λ'). Thus this compatibility specifies, locally, how well the label λ on O_i fits the decision of putting the label λ' on O_j. We can assume that r_{ij} are defined for all pairs i and j and stipulate, as a notation, that $r_{ij} \equiv 0$ if O_i and O_j are not neighbours.

Once such local compatibility functions are given, we have to define what we mean by global consistency. Let us denote by $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)$, a label assignment that assigns label λ_i to object O_i, $1 \leqslant i \leqslant N$.

DEFINITION 1

A label assignment $(\lambda_1, \ldots, \lambda_N)$ is said to be consistent if

$$\sum_j r_{ij}(\lambda_i, \lambda_j) \geqslant \sum_j r_{ij}(\lambda, \lambda_j), \forall \lambda, i = 1, \ldots, N.$$

It is said to be strictly consistent if the above inequalities are strict for $\lambda \neq \lambda_i$.

We can think of the quantity $\Sigma r_{ij}(\lambda_i, \lambda_j)$ as the amount of evidence in support of label λ_i on O_i given the labels on all other objects. Thus at a consistent labelling if we change the label on any one object, the 'amount of consistency' decreases. That is, changing the label on O_i will decrease net supporting evidence at O_i and this is true for all objects. This can also be viewed as a local maximum of an appropriate objective function.

Let L denote the space of all labellings. L will be an N-fold Cartesian product of the label set, Λ. We define a neighbourhood structure on L as follows.

DEFINITION 2

Let $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N) \in L$ and let us denote by $N(\boldsymbol{\lambda})$ all neighbours of $\boldsymbol{\lambda}$. Then for any $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_N) \in L$, $\boldsymbol{\mu} \in N(\boldsymbol{\lambda})$ if and only if there exists i, $1 \leqslant i \leqslant N$, such that $\lambda_i \neq \mu_i$ and $\lambda_j = \mu_j$ $\forall j \neq i$.

Thus given a labelling $\boldsymbol{\lambda}$, any other labelling differing from it in the assignment of a label to only one object, is a neighbour of $\boldsymbol{\lambda}$. Now we define a functional, F, on L by

$$F(\boldsymbol{\lambda}) = \sum_{i,j} r_{ij}(\lambda_i, \lambda_j). \tag{7}$$

DEFINITION 3

A labelling $\boldsymbol{\lambda} \in L$ is said to be a local maximum of F if

$$F(\boldsymbol{\lambda}) \geqslant F(\boldsymbol{\mu}), \text{ for all } \boldsymbol{\mu} \in N(\boldsymbol{\lambda}).$$

Now we are ready to establish the connection between consistency as defined by definition 1 and the local maxima of F.

Theorem 1. *Let the compatibility functions be symmetric. That is,* $r_{ij}(\lambda, \lambda') = r_{ji}(\lambda', \lambda)$, *for all* i, j, λ, λ'. *Then, a labelling* $\boldsymbol{\lambda} \in L$ *is consistent* (*definition* 1) *if and only if it is a local maximum of the functional* F (*definition* 3).

The proof of this theorem is very straightforward and directly follows from the definitions. Now let us take another look at the Boltzmann machine to understand it in the labelling framework.

Consider a labelling problem with N objects and the label set $\{+1, -1\}$. Let the compatibility functions be $r_{ij}(s_i, s_j) = w_{ij}s_is_j$. We now apply definition 1 to determine when a labelling (global state), S, is consistent. We must have

$$\sum_j r_{ij}(s_i, s_j) \geqslant \sum_j r_{ij}(s, s_j).$$

Since each s is ± 1, substituting the values for r_{ij} we get, for the case when $s_i = +1$,

$$\sum w_{ij}s_j > -\sum w_{ij}s_j (s = -1 \neq s_i),$$

i.e.

$$\sum w_{ij}s_j > 0.$$

Similarly we get, if $s_i = -1$,

$$-\sum w_{ij}s_i > \sum w_{ij}s_j,$$

i.e.

$$\sum w_{ij}s_j < 0.$$

Hence, for a global state S to be consistent, s_i should be of the same sign as $\Sigma\, w_{ij}s_j$. That is, we want to make $s_i > 0$ if the net input is positive and vice-versa. That is exactly the same as the Hopfield net. In the Boltzmann machine we said that we do not make $s_i > 0$ with probability 1 every time the net input is greater than zero because we want to include uphill moves also to avoid local minima. To understand this let us look at the F function for this case,

$$F(S) = \sum_{i,j} w_{ij}s_is_j.$$

Thus $F(.) = -E$. We know that the consistent labellings are local maxima of F (in the sense of definition 3). Thus they are local minima of E. Hence the consistency as given by definition 1 makes sure that we reach a local minimum of the energy function E. However, by allowing uphill moves and using simulated annealing, the Boltzmann machine can, in principle, find the global minima of energy E.

The labelling problem as formulated is an extension to the set of problems that can be tackled by the Boltzmann machine. Here we allow an arbitrary number of labels rather than restrict each object to a binary state. Therefore, even though consistency guarantees only a local maximum of F, due to the extra freedom allowed in the formulation, it might often be a more preferable alternative to formulating the problem in the framework of a Boltzmann machine. For example, in the case of TSP we have seen that it is essentially the need to work with Boolean variables that force us to add many constraints as penalty terms in the energy. By choosing the set of objects and labels intelligently we may avoid many such penalty terms in our energy function and thus even a local maximum of F might be an acceptable solution. The second reason for investigating labelling problems is that we can get parallel network-based algorithms to solve them so that the time complexity is much smaller than that of the Boltzmann machine.

6. A network of learning automata for solving the labelling problem

In this section we present a stochastic network based on learning automata for solving the consistent labelling problem. A learning automaton is an adaptive decision-making device that can identify the optimal action out of a finite set of actions through interactions with a random environment. We will not be giving any details regarding general learning automata models and the reader is referred to Narendra & Thathachar (1989) for such details. The automata algorithm for the labelling problem is presented in Thathachar & Sastry (1986). Here our interest is in viewing it as a stochastic network and contrasting it with other similar stochastic networks.

Consider a labelling problem with N objects and M labels. We use a network of N automata to solve the problem. The state of each automaton is a probability distribution over the set of labels. The state of the automaton i (associated with object O_i) at instant k is $\mathbf{p}_i(k) = [p_{i1}(k) \ldots p_{iM}(k)]^t$ where p_{is} is the probability with which automaton i chooses (i.e. associates object O_i with) label s. The output of the automaton is a random realisation of this probability distribution. That is, the output of automaton i at instant k is a choice of label for object O_i, at random, based on $\mathbf{p}_i(k)$. Each automaton receives at its input the outputs of other automata (i.e. labels chosen for other objects) and uses this information to update its state, i.e., its label probability distribution. The network functions synchronously. That is, at each instant all the automata, simultaneously and independently, select labels for their objects at random based on their label probability distributions. Then each of them receives at its input the outputs of other automata, computes their net input and uses the net input to update the states. The state updating is also done simultaneously by all the automata. We now specify the dynamics of the network, that is, rules for computing net input and updating the state.

Suppose at instant k, the ith automaton has chosen label λ_i. Then the net input to automaton i at k, $\mathrm{net}_i(k)$, is given by

$$\mathrm{net}_i(k) = (1/N) \sum_j r_{ij}(\lambda_i, \lambda_j). \tag{8}$$

The state of the automaton at $(k+1)$ is computed as

$$\begin{aligned} p_{i\lambda_i}(k+1) &= p_{i\lambda_i}(k) + a\,\mathrm{net}_i(k)(1 - p_{i\lambda_i}(k)), \\ p_{ij}(k+1) &= p_{ij}(k) - a\,\mathrm{net}_i(k) p_{ij}(k), \quad j \neq \lambda_i, \end{aligned} \tag{9}$$

where $0 < a < 1$ is a constant.

The net input into an automaton is still given by a summation of the effects of outputs of other automata. But instead of a simple weighted sum, we use a 'nonlinear' summation using the functions r_{ij}.

In the terminology of Learning Automata theory, net_i is called the reaction received by the ith automaton from its environment and the state updating given by (9) is called Linear Reward Inaction algorithm. Since $\mathbf{p}_i(k+1)$ should be a probability vector, it is necessary that $0 \leqslant \mathrm{net}_i(k) \leqslant 1$ for all i and k. To ensure this we assume, without loss of generality (Thathachar & Sastry 1986) that $r_{ij}(\lambda, \lambda') \in [0, 1]$ for all i, j, λ, λ'.

Let us denote by $P(k) \in R^{MN}$, the collection of all label probability vectors $\mathbf{p}_i(k)$,

$i = 1, \ldots, N$. Since each $\mathbf{p}_i$ is a probability vector, $P(k)$ belongs to $K \subset R^{MN}$, defined by

$$K = \{P \in R^{MN}: P = (p_1^t \ldots p_N^t)^t, p_i = (p_{i1} \ldots p_{iN})^t \in R^M,$$
$$p_{ij} \geqslant 0, \forall i,j, \sum_j p_{ij} = 1, i = 1, \ldots, N\}.$$

We can think of K as the N-fold cartesian product of M-dimensional simplices. Let $\mathbf{e}_i$ denote an M-dimensional unit (row) vector with its ith component unity. $(\mathbf{e}_{\lambda_1} \ldots \mathbf{e}_{\lambda_N})^t \in K$ is, for our algorithm, a labelling $(\lambda_1, \ldots, \lambda_N)$ that assigns the label λ_i to object i, $i = 1, \ldots, N$. All such points are termed corners of the space K.

$P(k)$ is the global state of our network at instant k. By the dynamics specified by (8) and (9), $P(k)$ is a Markov process with state space K. To understand the performance of our network we need to obtain the asymptotic behaviour of $P(k)$. For this, consider a piecewise-constant continuous time interpolation of $P(k)$, $\tilde{P}^a(.)$, defined by

$$\tilde{P}^a(t) = P(k) \text{ for } t \in [ka, (k+1)a),$$

where a is the parameter used in (9).

$\tilde{P}^a(\cdot) \in D^{MN}$, the space of all functions from R into R^{MN} which are left-continuous and have right-hand limits. Under the Skorohod metric (Billingsley 1968) this is a complete metric space. Now consider the sequence $\{\tilde{P}^a(\cdot)\}$ indexed by a. Using weak convergence results, it can be proved (Kushner 1984; Thathachar & Sastry 1986) that $\tilde{P}^a$ converges weakly to $\tilde{P}^0$, where $\tilde{P}^0$ is a solution of the ordinary differential equation (ODE)

$$z = g(z), \quad z(0) = P(0), \tag{10}$$

where

$$ag(z) = E[P(k+1) - P(k) | P(k) = z].$$

By analysing the asymptotic behaviour of the ODE (10), and by the properties of weak convergence, one can obtain the asymptotic properties of $P(k)$. The is stated in the following theorem.

Theorem 2. *If the value of the parameter a in (9) is sufficiently small then the following is true about the asymptotic behaviour of the dynamical process $\{P(k), k \geqslant 1\}$.*

(i) *Every strictly consistent labelling is an asymptotically stable stationary point of the process.*
(ii) *Every stable corner of the state space K is a consistent labelling.*
(iii) *All stationary points in the interior of K are unstable.*

The implication of this theorem is that the network mostly converges to a corner in K that is a consistent labelling. The convergence result is incomplete because we do not know about the stability or otherwise of stationary points on the faces of K (i.e., points that are neither corners nor are in the interior). Also, the ODE being nonlinear, there is a possibility of other limiting behaviour such as limit cycles. In practice, these cases never seem to arise.

6.1 *Comparison with the Boltzmann machine*

In the Boltzmann machine the global state is an ergodic Markov chain. Thus, at any finite temperature, the machine converges to a probability distribution over the set of all labellings. The global state of the automata network is a Markov process with absorbing states. [It can easily be verified that all corners of K are absorbing for $P(k)$.] Thus the automata network converges to a specific labelling which is one of the possibly many consistent labellings.

The steady state probability distribution of the Boltzmann machine prefers low energy states. Though the machine keeps visiting all labellings with a non-zero probability, the probability of visiting states with lower energy is higher. By incorporating simulated annealing we can reach the global minimum; but it will take a very long time. In constrast, the automata network converges to a specific labelling that is a local maximum of $F(.)$ This may or may not a global maximum depending on $P(0)$. But there is no scope for incorporating a mechanism like simulated annealing into the automata network.

For the Boltzmann machine, viewed as an optimizer, the objective function has to be quadratic involving only binary variables. In the more general labelling problem, the labels need not be binary, and also, by clever choice of r_{ij}'s one can tackle nonquadratic functions as well. By utilising the extra freedom given by the labelling framework one can reduce the possible unwanted local minima as compared to the energy function formulation of the Boltzmann machine. For example, in the energy function given by (5), we had three terms for incorporating the constraints which are necessary when formulating TSP using only Boolean variables. In a labelling framework, suppose we let each automaton represent a city and the actions represent position numbers. Then we do not need constraints such as no city be placed in more than one position (the 'A-term' in (5)) and every city be placed in some position (the 'C-term' in (5)). This can effectively be used in reducing the unwanted local minima (Muralidharan 1989).

As mentioned earlier, the automata network is primarily for constraint satisfaction problems with soft constraints. In such problems there will be, in general, many consistent solutions and what is desired is a consistent labelling 'close' to the initial problem data provided. Many problems in computer vision are of this type. In the automata algorithm the initial data can be incorporated through $\mathbf{p}_i(0)$, the initial values for the action probabilities thereby biasing the network to converge to a consistent labelling closeby. Since the Boltzmann machine results in an ergodic chain, the initial state has no effect on the asymptotic behaviour and hence we have to code the initial data also as part of the energy function.

This algorithm was successfully employed in computer vision problems such as stereopsis and object recognition (Sastri *et al* 1988). Also unlike the Boltzmann machine, where only one unit changes state at any given instant, here all automata function synchronously and hence the network can be implemented in parallel, for example, on an single instruction multiple data (SIMD) machine (Banerjee *et al* 1987).

6.2 *Another stochastic network for the labelling problem*

An important and widely studied algorithm for the constraint satisfaction and optimization problems of the kind studied in this paper is the algorithm of Geman & Geman (1984).

Consider a labelling problem as specified at the beginning of § 5. The neighbour relationship can be viewed as a graph with the set of objects as vertices. A clique of this graph is a set of objects, where all pairs of objects are neighbours. Instead of specifying a set of compatibility functions r_{ij}, let us assume that the prior knowledge about the constraints to be satisfied by the solution can be encoded through an energy function defined on the set of labellings by

$$E(\lambda) = \sum_{c \in \mathscr{C}} V_c(\lambda), \tag{11}$$

where $\mathscr{C}$ is the set of all cliques. $V_c(\boldsymbol{\lambda})$ is called a potential function and its value depends only on the labels (components of $\boldsymbol{\lambda}$) of those objects which are contained in the clique c. If we allow only cliques containing pairs of objects then V_c is similar to $r_{ij}(\lambda_i, \lambda_j)$ and $E(.)$ is similar to $F(.)$ defined by (7). So the local minima of $E(.)$ can be thought of as the 'consistent' labellings. We define a prior probability for each labelling by

$$\pi(\boldsymbol{\lambda}) = \exp(-E(\boldsymbol{\lambda})/Z, \tag{12}$$

where Z is a normalising constant. It may be noted that this is the same as the steady state distribution of a Boltzmann machine though here the labels are not binary. The prior probability distribution given by (12) prefers labellings with low energy. As earlier, the objective is to find a consistent labelling that is close to the initial data given. In this approach we define the solution to be the global maximum of posterior probability $P(\boldsymbol{\lambda}|\mathbf{d})$, which is the probability that the 'correct' labelling is $\boldsymbol{\lambda}$ given the data $\mathbf{d}$. We can use Bayes rule to calculate $P(\boldsymbol{\lambda}|\mathbf{d})$ using (12) if we have the conditional probabilities, $P(\mathbf{d}|\boldsymbol{\lambda})$. These can be specified by having a model for the process which generates the data. Under some mild conditions on the conditional probabilities (which are generally true for image processing problems), $P(\boldsymbol{\lambda}|\mathbf{d})$ will have the same form as (12) with the posterior energy containing one term for the data in addition to the summation of clique potentials as in (11). So now the problem is to find a parallel algorithm for finding the global maximum of a probability distribution as in the case of the Boltzmann machine. In the algorithm we have a unit for each object which chooses a label at random using the conditional probabilities given the current labels of all the neighbouring objects. This so-called Gibbs sampler algorithm can be shown to generate an ergodic Markov chain whose steady state distribution is the needed posterior probability distribution (same form as (12) but with E replaced by posterior energy). So as in the case of the Boltzmann machine we can reach the global maximum of the posterior probability by incorporating simulated annealing.

Thus the Geman & Geman (1984) algorithm can be viewed as an extension of the Boltzmann machine to the more general labelling framework. It generates an ergodic Markov chain (and hence we need to incorporate the data into the energy function) and we aim to reach the global maximum by incorporating simulated annealing. The main problem with this algorithm is its extremely slow rate of convergence.

This algorithm is presented rather briefly because it is best explained in the context of image processing problems and such a discussion is beyond the scope of this paper. The details can be found in Geman & Geman (1984), Geman & Graffigne (1987) and Geman *et al* (1990). An extensive comparison of the automata model with the Geman & Geman algorithm is found in Banerjee (1989).

Though in this paper we have discussed the automata model and the Geman & Geman algorithm as if they are extensions of the Boltzmann machine, all the algorithms have been originally proposed more or less independently from different perspectives.

7. Conclusions

In this article we have presented a brief account of some stochastic network models for solving special types of constraint satisfaction and optimization problems. The use of such soft constraints is important in areas such as computer vision, image processing etc. Also these models give a flavour of the type of problems that can be tackled by artificial neural networks. There are deterministic schemes also for tackling soft constraints, for example, the graduated non-convexity algorithm of Blake & Zisserman (1987). See Blake (1989) for a comparison of the deterministic scheme against those that use simulated annealing.

The Boltzmann machine discussed here also has applications in pattern recognition and learning. These are not discussed because it would be outside the scope of the topic of this paper. The reader can refer to Rumelhart & McClelland (1986) for details.

References

Banerjee S 1989 *On stochastic relaxation paradigms for computational vision*, Ph D thesis, Dept. Elec. Eng., Indian Institute of Science, Bangalore

Banerjee K, Sastry P S, Ramakrishnan K R, Venkatesh Y V 1988 An SIMD machine for low-level vision. *Inf. Sci.* 44: 19–50

Billingsley P 1968 *Convergence of probability measures* (New York: John Wiley)

Blake A 1989 Comparison of the efficiency of deterministic and stochastic algorithms for visual reconstruction. *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-11: 2–12

Blake A, Zisserman A 1987 *Visual reconstruction* (Cambridge: MIT Press)

Davis L, Rosenfeld A 1981 Cooperating processes in low level vision. *Artif. Intell.* 17: 245–263

Geman S, Geman D 1984 Stochastic relaxation, Gibbs distribution and Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-6: 721–741

Geman D, Geman S, Graffigne C, Dong P 1990 Boundary detection by constrained optimization. *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-12: 609–628

Geman S, Graffigne C 1987 Markov random field image models and their applications to computer vision. *Proc. Intl. Congress of mathematicians* (ed.) A M Gleason (Providence: Am. Math. Soc.)

Hinton G E, Sejnowsky T J, Ackley D H 1984 Boltzmann machines: Constraint satisfaction networks that learn, Tech. Rep. CMU-CS- 84–119, Carnegie Mellon Univ. Pittsburgh

Hopfield J J, Tank D W 1985 Neural computation of decisions in optimization problems. *Biol. Cybernet.* 52: 141–152

Hummel R A, Zucker S W 1983 On the foundations of relaxation labelling processes. *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-5: 267–286

Kirkpatrick S, Gelatt C D, Vecchi M P 1983 Optimization by simulated annealing. *Science* 220: 671–680

Kushner H 1984 *Approximation and weak convergence methods for random processes* (Cambridge: MIT Press)

Marroquin J, Mitter S, Poggio T 1987 Probabilistic solution of ill-posed problems in computational vision. *J. Am. Stat. Assoc.* 82 (397): 76–89

Mitra D, Fabio R, Vincentelli A S 1986 Convergence and finite-time behaviour of simulated annealing. *Adv. Appl. Probab.* 18: 747–771

Muralidharan S 1989 *Parallel distributed processing schemes for combinatorial optimisation*, M Sc (Eng.) thesis, Dept. Elect. Eng., Indian Institute of Science, Bangalore

Narendra K S, Thathachar M A L 1989 *Learning automata: An introduction* (Englewood Cliffs, NJ: Prentice-Hall)

Rumelhart D E, McClelland J L 1986 *Parellel distributed processing* (Cambridge: MIT Press) Vols. 1 & 2

Sastry P S, Ramakrishnan K R, Banerjee S 1988 A local cooperative processing model for low level vision. *Indo-US Workshop on Systems and Signal Processing* (New Delhi: Oxford and IBH)

Thathachar M A L, Sastry P S 1986 Relaxation labelling with Learning Automata. *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-8: 256–268

Waltz L D 1975 Understanding of line drawings of scenes with shadows. In: *Psychology of computer vision* (ed.) P H Winston (New York: McGraw-Hill)