Towards a Stochastic Neural Model for Combinatorial Optimization *

Luís Alfredo V. de Carvalho  
Valmir C. Barbosa

Programa de Engenharia de Sistemas e Computação, COPPE  
Universidade Federal do Rio de Janeiro  
Caixa Postal 68511  
21945 Rio de Janeiro - RJ, Brazil

Abstract. We propose a stochastic neural model for the solution of hard Combinatorial Optimization problems, inspired in the Hopfield-Tank model and on the stochastic search ability of Simulated Annealing. We start with a discrete-time algorithm for the simulation of a Hopfield-Tank network, and modify it through the incorporation of probabilistic decisions which allow energy increases at each time step. We have tested the algorithm on the Traveling Salesman Problem (TSP), and the results are encouraging. For these tests we employed a formulation of TSP which ensures that every stable state of the network corresponds to a feasible TSP tour. From our simulation algorithm we then extract some possible foundations for the formalization of our model.

1. Introduction

A problem in Combinatorial Optimization is said to be an \textit{NP-complete problem} if, when formulated as a decision problem, the following two properties are true of it. First, its objective function can be evaluated in a time which varies polynomially with the number of variables, for any given configuration of such variables. This first property characterizes membership in the class \textit{NP} of decision problems. The second property is that, once an algorithm is found that solves an \textit{NP}-complete problem in polynomial time, this same algorithm provides a solution, also in polynomial time, to all other problems in \textit{NP}. So the \textit{NP}-complete problems constitute a core of the hardest problems within \textit{NP} [6].

To this date, not one algorithm has been found that solves an \textit{NP}-complete problem in polynomial time, therefore suggesting that such problems are indeed intractable in terms of their computational complexity, and that it is a good idea to look for algorithms which allow for sub-optimal solutions that can be found efficiently.

Since 1982, with the publication of Hopfield’s article on the use of binary-neuron networks to find local minima of certain functions [9], many researchers have tried to approach hard Combinatorial Optimization problems on the basis of analogies with physical and biological systems. The cornerstone of such approaches has been to identify the objective function under study with some sort of “energy” in the physical or biological counterpart, and then proceed to its minimization as the natural system would do.

Hopfield’s 1982 proposal has in a way revived the interest in the so-called connectionist systems, which comprise a very large number of simple processing elements whose behavior is determined by the interconnections among them, or alternatively by how intensely they influence one another. His main achievement was to recognize that symmetrically interconnected binary neurons evolve in such a way as to minimize the system’s energy, provided each neuron has its state updated by a simple threshold rule, and that no two neurons have their states updated concurrently if they are interconnected by a synapse of nonzero value. This one-at-a-time condition imposed on “neighboring” neurons may at first seem discouraging from the standpoint of a parallel implementation, but certain distributed protocols, such as the one described in [2], can be employed to this end, and distributed parallel simulators have been designed following those protocols [3].

Concurrently with Hopfield’s proposal, but following a path apparently outside the connectionist realm, Kirkpatrick and his collaborators proposed in 1983 the use of Simulated Annealing as a centralized tool to stochastically seek global minima of functions [13]. They too relied on a powerful analogy with physical
systems, and proposed that the objective function to be minimized be identified with the energy of a gas undergoing a slow cooling process, since this is known to yield states of the gas in which the system's energy is globally minimum. Since for a fixed temperature the state of the gas is governed by the Boltzmann distribution, Simulated Annealing was made to perform a stochastic search based on that same distribution. Occasional "uphill" moves occur as a result, therefore explaining the ability of the method to escape from local minima. Later in 1986 an elegant study of Simulated Annealing was published by Mitra et al. [14], who proved the convergence of the method to a global minimum provided the temperature is reduced at a rate which requires an exponential number of steps in the simulation. This result is not surprising, in view of the \[NP\]-completeness of the problems under study, nor is it discouraging, since excellent results are many times obtained with a much faster temperature reduction.

The Boltzmann Machine [8] offers a combination of Hopfield's binary-neuron network with Kirkpatrick's Simulated Annealing. Essentially, in the Boltzmann Machine each neuron updates its state probabilistically in a Simulated Annealing fashion, seeking a global minimum of the same function Hopfield showed his network would (locally) minimize. It was later shown by S. Geman and D. Geman, in the context of image processing, that a class of networks which generalizes the Boltzmann Machine does indeed provide convergence to global optima, but again the exponentially growing time required for convergence appeared as necessary [7]. The Gemans' networks (and consequently the Boltzmann Machine) suffer, just as Hopfield's binary-neuron networks, from the constraint that neighboring neurons must be updated one at a time. Again the protocol described in [2] can be utilized, and in fact has been employed to compose a distributed parallel implementation of Simulated Annealing which is applicable to a number of hard problems [1,5].

Neurons with continuous response have an important role to play as well, since they were shown by Hopfield to have computational properties similar to those of binary neurons [10]. Specifically, it was shown that such neurons, when evolving according to differential equations which mimic the behavior of an electric circuit, can be used to find local minima of an energy function which in many cases looks exactly like the one minimized by their binary counterparts. The neurons' behavior is now parameterized by "real time," and all of them evolve concurrently with one another. In terms of a distributed parallel simulation, appropriate protocols are as before needed [3].

These continuous-neuron networks have been utilized to put forward a model for the local minimization of functions describing hard Combinatorial Optimization problems [11,12]. The resulting model is known as the Hopfield-Tank model, and was tested mainly with the notoriously intractable Traveling Salesman Problem (TSP) [6]. The results reported by Hopfield and Tank on their experiments with this problem were criticized [17], different objective functions were proposed for operation on their model [4,16], and the model itself was modified, as for example in [15]. Improvements seem to have been achieved, but the situation still looks to us far from settled.

Our primary goal in this paper is to propose a modification to the Hopfield-Tank model, aiming at a network which seeks global minima of the energy function. Just as the Boltzmann Machine can be regarded as an improvement of the binary-neuron Hopfield network through the use of Simulated Annealing, so can the new model we propose with respect to the Hopfield-Tank network.

Our departing point has been to recognize that what the Boltzmann Machine does to escape local minima is essentially to probabilistically "disobey" the threshold rule imposed by the original Hopfield model, thereby allowing for occasional updates of a neuron's state which lead to increases in the energy of the system. We elaborate on this point in Section 2. In Section 3 we start with a discrete-time simulation of the continuous-neuron model, and show how probabilistic decisions can be incorporated at each time step, meaning that as in the Boltzmann Machine the neurons' "response rule" is disobeyed occasionally. A new discrete-time simulation algorithm is then obtained which is parameterized by a slowly decreasing temperature-like quantity. In Section 4 we test the new algorithm on TSP. For such, we utilize a new formulation of the problem in which the neurons' thresholds too are used to code the particular instance of the problem being solved, instead of utilizing synaptic weights only. This new formulation is attractive because, as we prove in Section 4, it ensures that every stable configuration of the network is a valid solution to the problem, unlike previous formulations. After reporting some simulation experiments still in Section 4, we move to a more formal characterization of the new model in Section 5. This formalization is derived from the discrete-time simulation algorithm introduced in Section 3. We close with some concluding remarks in Section 6.

### 2. Motivation: the Boltzmann Machine

Each neuron \( N_i \) in the binary-neuron network introduced by Hopfield [9] is characterized by its state \( u_i \), which is either 0 or 1, a threshold \( \theta_i \), an external input \( c_i \), and weights \( w_{ij} = w_{ji} \) for the synaptic connections with each of the other neurons \( N_j \). The input potential \( u_i \) to \( N_i \) is given directly by the influence of all other neurons upon it, and by its external input:

\[
u_i = \sum_{j \neq i} w_{ij} v_j + c_i.\]
\( N_i \) updates its state \( v_i \) according to the threshold rule

\[
v_i = \begin{cases} 
0, & \text{if } u_i \leq \theta_i; \\
1, & \text{if } u_i > \theta_i.
\end{cases}
\] (2)

Hopfield showed that if no two neurons interconnected by a nonzero synaptic weight are updated concurrently by the rule in (2), then the network evolves to a local minimum of the energy function

\[
E = -\frac{1}{2} \sum_{i \neq j} w_{ij} v_i v_j - \sum_i c_i v_i + \sum_i \theta_i v_i.
\] (3)

In a Boltzmann Machine [8], the deterministic updating rule in (2) is replaced with the following probabilistic rule. Let \( \Delta E_i \) denote the energy gap between two states of the network which differ only in the state of neuron \( N_i \), i.e.

\[
\Delta E_i = -\sum_{j \neq i} w_{ij} v_j - c_i + \theta_i
\]

\[
= -(u_i - \theta_i).
\]

The Boltzmann Machine utilizes the Boltzmann distribution, conditioned on the states of all other neurons, in the updating of \( v_i \). So \( v_i \) is set to 0 with a probability which can be seen to be given by

\[
p_0 = \frac{1}{1 + e^{-\Delta E_i/T}},
\] (4)

where \( T \) is the temperature-like parameter of Simulated Annealing. With probability \( p_1 = 1 - p_0 \), \( v_i \) is set to 1.

Eq. (4) gives an interesting insight to the functioning of the Boltzmann Machine. For high values of \( T \), both \( p_0 \) and \( p_1 \) are approximately equal to 1/2, meaning that at such temperatures \( N_i \) may disobey the rule in Eq. (2), since there is a considerable probability that \( v_i \) be set to 0 (to 1) even if \( u_i > \theta_i \) (\( u_i \leq \theta_i \)).

As the network starts to freeze, i.e., at low values of \( T \), the value of \( p_0 \) starts to approach 0 if \( u_i > \theta_i \), and the value of \( p_1 \) approaches 0 if \( u_i \leq \theta_i \). In other words, at low values of \( T \) the network resumes its behavior as a binary-neuron Hopfield network, following the rules in (2).

The behavior of the Boltzmann Machine at high values of \( T \) accounts for the possibility that occasionally the value of \( E \) (Eq. (3)) increases during the computation. This happens whenever the rules in (2) are not followed, which can alternatively be seen as if in such situations the neuron chose to follow an input-output relation like that of Figure 1(b), instead of the standard relation depicted in Figure 1(a). In other words, what we get in the Boltzmann Machine is the following behavior of \( N_i \). Compute \( u_i \) and choose a value for \( v_i \) according to Eq. (2); let \( v \in (0, 1) \) be the chosen value. With probability \( p_v \), stick to this value \( v \); otherwise, set \( v_i \) to 1 - \( v \).

In the next section we employ this idea of a "flipped" input-output relation in trying to embed some "uphill climbing" behavior into the operation of the continuous-neuron Hopfield network.

\[\text{Figure 1. Direct (a) and reverse (b) binary-neuron response curves}\]

3. Simulated Annealing in the Hopfield-Tank Model

In the continuous-neuron Hopfield model [10], \( v_i \) may assume any value between the two extremes 0 and 1. The input potential to \( N_i \) is no longer given directly by Eq. (1), but rather follows the differential equation

\[
C_i \frac{du_i}{dt} = \sum_{j \neq i} w_{ij} v_j + c_i - \frac{u_i}{R_i},
\] (5)

where \( t \) is "real time," \( C_i \) is the neuron's "capacitance," and \( R_i \) its "resistance." Likewise, the rule determining \( v_i \) from \( u_i \) is no longer the threshold rule in Eq. (2), being replaced by the sigmoidal relationship

\[
v_i = f_i(u_i) = \left[ 1 + e^{-\gamma(u_i - \theta_i)} \right]^{-1},
\] (6)

where \( \gamma \) is the sigmoid's "gain," i.e., it regulates the steepness of the response curve around the point \( u_i = \theta_i \).

For symmetric weights \( w_{ij} = w_{ji} \), the time-derivative of an energy function very similar to that of Eq. (3) is strictly negative, indicating that the evolution of the network according to the differential equations in (5) promotes the search for a local minimum of that energy, which for high gain situations is
approximately given by

$$E = -\frac{1}{2} \sum_{i \neq j} w_{ij} v_i v_j - \sum_i c_i v_i + \sum_i \frac{\theta_i}{R_i} v_i. \quad (7)$$

Before we modify this model by the introduction of Simulated Annealing, it is worthwhile to take a look at how the differential equations in (5) can be simulated by the utilization of difference equations instead. The algorithm for a discrete-time centralized simulation of the network starting at a given set of neuron states is given in Figure 2.

for all \( N_i \) do
begin
\( v_i := \text{initial state of } N_i; \)
\( u_i := f_i^{-1}(v_i) \)
end;
while "unstable" do
begin
for all \( N_i \) do
begin
\( \Delta u_i := \left( \sum_{j \neq i} w_{ij} v_j + c_i - u_i/R_i \right) \Delta t/G_i; \)
end;
for all \( N_i \) do
begin
\( u_i := u_i + \Delta u_i; \)
\( v_i := f_i(u_i) \)
end;
end;

Figure 2. Deterministic simulation algorithm

In the algorithm of Figure 2, \( \Delta t \) is the time increment from iteration to iteration. The check to see whether the system is still "unstable" consists in evaluating the difference in the energy \( E \) of Eq. (7) between the last two iterations, and verifying whether it is confined below a certain pre-established value.

The algorithm of Figure 2 can be modified to allow probabilistic moves in which \( E \) grows, following our discussion about the Boltzmann Machine in the previous section. In order to allow occasional energy increases, we will sometimes "flip" the sigmoid response curve of a neuron \( N_i \). This will be done at the end of each iteration, so that instead of deterministically performing the assignment \( v_i := f_i(u_i) \), we will sometimes do \( v_i := 1 - f_i(u_i) \). This will correspond to the utilization of the sigmoid shown in Figure 3(b) in place of that of Figure 3(a) in such occasions. The intuition behind this proposal can be somewhat strengthened by the following observation. The time derivative of the energy \( E \) (see [10]),

$$\frac{dE}{dt} = -\sum_i C_i \left( \frac{dv_i}{dt} \right)^2 f_i^{-1}(v_i),$$

reveals that it can only be made positive if at least one of the sigmoids is "flipped" (it is certainly made positive if all sigmoids are "flipped").

The modified algorithm we propose appears in Figure 4. Notice first of all that now the value of \( t \) has to be recorded, since the algorithm makes use of the temperature-like parameter \( T(t) \) which is given by a slowly decreasing function of \( t \). At each iteration, the algorithm proceeds like the one of Figure 2 until each potential \( u_i \) is updated through the addition of the just-computed \( \Delta u_i \). At this point, two possible sets of neuron states are generated. One of them is precisely the one generated by the algorithm of Figure 2, and consists of values obtained directly from the sigmoid of Figure 3(a). The state of \( N_i \) in this first set is denoted by \( v_i \). The other set of states is obtained by independently "flipping" the sigmoid of each neuron with a probability \( p \). This second set of states consists of states obtained from the sigmoid of Figure 3(a), together with states obtained from the sigmoid of Figure 3(b). The state of \( v_i \) in this second set is denoted by \( \bar{v}_i \). Next the energy \( E \) is computed for the set of values \( v_i \), yielding \( E_1 \), and for the set of values \( \bar{v}_i \), yielding \( E_2 \). The gap \( \Delta E = E_1 - E_2 \) is then used to probabilistically choose between the two sets. The second set, the one in which some sigmoids were "flipped" to yield the neuron states, is chosen with probability

$$\frac{1}{1 + e^{-\Delta E/T(t)}}, \quad (8)$$

otherwise the first set is picked, and the iteration terminates.

Concerning the value of the probability \( p \), it is clear that \( p = 0 \) corresponds to the deterministic algorithm of Figure 2, while for \( p = 1 \) every sigmoid is "flipped" at the end of every iteration.
\( t := 0; \)
for all \( N_i \) do
begin
\( v_i := \text{initial state of } N_i; \)
\( u_i := f_i^{-1}(v_i) \)
end;
while "unstable" do
begin
for all \( N_i \) do
begin
\( \Delta u_i := \left( \sum_{j \neq i} w_{ij} v_j + c_i - u_i / R_i \right) \Delta t / C_i; \)
for all \( N_i \) do
begin
\( u_i := u_i + \Delta u_i; \)
\( v_i := f_i(u_i); \)
\( \tilde{v}_i := \begin{cases} 1 - v_i, & \text{with probability } p_i; \\ v_i, & \text{otherwise} \end{cases} \)
end;
\( E_1 := E \) using \( v_i \) for all \( N_i; \)
\( E_2 := E \) using \( \tilde{v}_i \) for all \( N_i; \)
\( \Delta E := E_1 - E_2; \)
replace every \( v_i \) with \( \tilde{v}_i \) with probability
\( 1 + e^{-\Delta E / T(t)} \)
\( t := t + \Delta t \)
end;

Figure 4. Stochastic simulation algorithm

From our previous discussion on the time derivative of \( E \), it is clear that the energy gap \( \Delta E \) of the algorithm in Figure 4 is always nonpositive, so along the cooling process the probability in Eq. (8) varies from its initial value of approximately 1/2 to a final value of approximately 0. This means that as the system freezes through the decrease of \( T(t) \), the set of values generated with some "flipped" sigmoids is no longer chosen, and the system stabilizes as in the deterministic case.

We proceed in the next section to an evaluation of the algorithm of Figure 4 on instances of TSP, following a novel formulation of the problem.

4. Performance on TSP

4.1. TSP Formulation

In TSP we are given \( n \) cities and the distances \( d_{AB} \) between any two cities \( A \) and \( B \). The problem then asks for a shortest tour that visits every city exactly once. In the Hopfield-Tank model [11,12], TSP is formulated with the aid of \( n^2 \) continuous-response neurons organized in a matrix. Each row in the matrix of neurons corresponds to a city, while each column corresponds to one of the possible positions a city may occupy in a tour. A stable state of such a network is a \textit{feasible} stable state if it comprises exactly \( n \) neurons in the \textit{on} state (close to 1), arranged in the matrix so that there is one of them per row and one of them per column. One such feasible stable state is illustrated in Figure 5, where \( n = 4 \).

Our formulation of TSP follows that of the Hopfield-Tank model, except that we utilize the neurons' thresholds, in addition to the synaptic weights, to program the particular instance of TSP at hand. In all of Section 4, we assume that, for all neurons \( N_i, R_i = R, C_i = C, \theta_i = \theta, \) and \( \gamma_i = \gamma \).

Let \( N_i \) and \( N_j \) be two neurons in the matrix. If they occupy the same row or the same column in the matrix, then they are interconnected by the inhibitory synapse \( w_{ij} = W < 0 \), in order to ensure that there will not be more than one neuron in the \textit{on} state per row or per column. If the neurons \( N_i \) and \( N_j \) are in rows corresponding to cities \( A \) and \( B \), respectively, and furthermore they occupy adjacent columns (notice that columns 1 and \( n \) are adjacent to each other), then they are interconnected by the inhibitory synapse of weight \( w_{ij} = -d_{AB} \). All other synapses in the network receive value zero, and so do the external inputs to all neurons. In order to make sure that no less than \( n \) neurons will be in the \textit{on} state in a stable state, we provide \( \theta \) with a negative value as well. In Proposition 1 below we derive sufficient conditions on the values of \( W \) and \( \theta \) in order to guarantee that every stable state is feasible.

**Proposition 1.** If
\[
W < \frac{\theta}{R} < \min_A \left\{ -2 \sum_{B \neq A} d_{AB} \right\},
\]
then every stable state is feasible.

**Proof.** We show that every infeasible state is unstable. For consider an infeasible state of the network; one of the following three cases must happen: (a) there are \( n \) neurons in the \textit{on} state, but not one per row and one per column; (b) there are more than \( n \) neurons in the \textit{on} state; (c) there are less than \( n \) neurons in the \textit{on} state.
If case (a) or case (b) holds, then at least one row (say \( k \)) and one column (say \( l \)) are such that row \( k \) has \( x \geq 1 \) neurons in the on state and column \( l \) has \( y \geq 1 \) neurons in the on state, in such a way that \( x + y > 2 \) and that the neuron (say \( N_i \)) occupying the position \((k,l)\) in the matrix is in the on state. An illustration is given in Figure 6(a). Because \( N_i \) is in the on state, we have \( u_i > \theta \), and then the time derivative of \( u_i \) becomes, by Eq. (5),

\[
C \frac{du_i}{dt} < (x + y - 2)W - \frac{\theta}{R},
\]

where we have taken into account the strict negativity of the synaptic weights that connect \( N_i \) outside its row or column. By the first inequality in the hypothesis, and considering that \( x + y > 2 \), it then follows that

\[
\frac{du_i}{dt} < 0.
\]

![Figure 6. Used in the proof of Proposition 1](image)

So either we have a neuron \( N_i \) for which \( u_i > \theta \) and \( du_i/dt < 0 \), or a neuron \( N_i \) for which \( u_i < \theta \) and \( du_i/dt > 0 \). As a consequence, every infeasible state is unstable.

By Proposition 1, it follows from Eq. (7) that the energy of the neural network proposed in this section is, at st.ble points, equal to the length of some tour plus the constant \( n\theta/R \). Consequently, the problem of finding a global minimum of the energy function \( E \) of Eq. (7) is equivalent to TSP.

4.2. Simulation Results

We have tested the algorithm of Figure 4 on TSP for \( n \) up to 15, following the formulation introduced in the previous section. For each value of \( n \), city positions were generated randomly inside a square of side length 5, and distances between each two cities calculated.

For our simulations we have employed the temperature decrease function

\[
T(t) = T_0 e^{-\alpha t},
\]

where \( 0 < \alpha < 1 \), \( t \geq 0 \), and \( T_0 \) is \( O(-n^3W) \) (in order that the probability in Eq. (8) be approximately equal to 1/2 at \( t = 0 \)).

All tests have been performed with \( R = C = 1, \gamma = 1 \), and \( \Delta t = 0.01 \). The probability \( p \) utilized for independently "flipping" each neuron's sigmoid at each time step has been set to \( p = 1/2 \).

We show in Figure 7 a tour found for the \( n = 10 \) case with \( W = -100, \theta = -55 \), and a temperature decrease rate of \( \alpha = 0.75 \). Initial neuron states were generated randomly. This tour represents a significant improvement over the purely deterministic simulation on the same problem (i.e., through the algorithm of Figure 2), starting at the same initial set of neuron states, as well as over probabilistic simulations with faster temperature decrease rates. This behavior has been typical among our other tests, and the results seem very encouraging to us, although due to limitations in the processing power of our simulation equipment we were precluded from trying, at this point, a more precise simulation with a significantly smaller value for \( \Delta t \).

5. The Stochastic Model

Just as the algorithm of Figure 2 simulates the deterministic model described by the differential equations in (5), it is also possible to generalize those equations in order to obtain a stochastic model corresponding to the simulation performed by the algorithm of Figure 4. In what follows, we explicitly indicate the dependence of the neuron states and neuron potentials upon the time parameter \( t \). It so is that, as in Eq. (6), we have

\[
u_i(t) = \left[ 1 + e^{-\gamma(t - t_0)} \right]^{-1}.
\]
Now, for each instant $t$, let
\[
\tilde{u}_i(t) = \begin{cases} 
1 - u_i(t), & \text{with probability } p; \\
u_i(t), & \text{otherwise,}
\end{cases}
\]
and
\[
\Delta E(t) = E_1(t) - E_2(t),
\]
where $E_1(t)$ is the value of the energy $E$ (Eq. (7)) when the state of every neuron $N_i$ is $u_i(t)$, and $E_2(t)$ is the value of $E$ when the state of every neuron $N_i$ is $\tilde{u}_i(t)$.

The state of neuron $N_i$ at time $t$ is in the new model best described by a random variable, which we denote by $\tilde{u}_i(t)$. $N_i$'s potential then follows the differential equation
\[
C_i \frac{d\tilde{u}_i}{dt} = \sum_{j \neq i} w_{ij} \tilde{v}_j + \epsilon_i - \frac{u_i}{K_i}.
\]

The stochastic, temperature-dependent behavior of the system, as simulated by the algorithm of Figure 4, is then described by indicating the joint distribution of the random variables $\tilde{u}_i(t)$ at an instant $t^+$. So the probability that at $t^+$ every neuron $N_i$ be in a state given by $\tilde{u}_i(t)$ is
\[
\frac{1}{1 + e^{-\Delta E(t)/T(t)}}.
\]
Otherwise, every neuron $N_i$ maintains at $t^+$ the state $u_i(t)$.

6. Concluding Remarks

In this paper we have provided, based on an analogy with the functioning of the Boltzmann Machine, a neural model for the solution of hard Combinatorial Optimization problems which tries to incorporate the "hill-climbing" features of Simulated Annealing into the deterministic Hopfield-Tank model. Our point of departure was the discrete-time simulation algorithm of Figure 2, used for the simulation of a Hopfield-Tank network. We proposed a variation to that algorithm by providing, at each time step, an opportunity for each neuron to "disobey" its input-output characteristic, thereby allowing for occasional increases in the system's energy. A time-dependent temperature-like parameter was employed to regulate these probabilistic energy increases, in such a way that at the end of the process the algorithm behaves as in the original Hopfield-Tank model and stabilizes at an energy minimum.

We performed simulations on various instances of TSP with encouraging results, despite severe limitations in our simulation power at the moment. For our TSP experiments we utilized a novel formulation which ensures that every stable state of the network is a feasible solution to the problem.

We finalized by proposing a model which complies with our modified algorithm. The model looks sound, and we expect to be able to analyze it further.

As a last comment, we briefly discuss possible distributed implementations of the simulation algorithm of Figure 4. Recall from Section 1 that the Boltzmann Machine, like its deterministic counterpart (Hopfield's binary-neuron network) can be simulated in a distributed parallel environment by the application of essentially the same techniques. The fact that in the Boltzmann Machine decisions are made as "locally" as in the binary-neuron Hopfield network (see [1]) implies that no parallelism is lost with respect to that network. Our modified Hopfield-Tank model, on the other hand, although still amenable to a distributed parallel simulation by practically the same techniques employed to simulate the deterministic Hopfield-Tank network (see [3]), has an inherently centralized component which is the computation of the total energy as required by the probabilistic decision at the end of each time step (see Figure 4). As a result, some global synchronization is required, whereas in the deterministic case neurons need only synchronize "locally."

References


