[12] L.-X. Wang, "Fuzzy systems are universal approximators," in *Proc. IEEE Int. Conf., Fuzzy Systems*, San Diego, CA, Mar. 1992.
[13] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. Syst. Man, Cybern.*, vol. 3, pp. 28–44, Jan. 1973.

Efficient Implementation of the Boltzmann Machine Algorithm

A. DeGloria, P. Faraboschi, and M. Olivieri

Abstract— **The availability of efficient software implementations of neural network algorithms is a key task in the development phase to evaluate the network results in real cases. In this letter we address the problem of optimizing the sequential algorithm for the Boltzmann Machine (BM). We present a solution which is based on the locality properties of the algorithm and enables a very efficient computation of the cost difference between two configurations. Since the algorithm performance depends on the number of accepted state transitions in the annealing process, we formulate a theoretical procedure to estimate the acceptance probability of a state transition. In addition, we provide experimental data on a well-known optimization problem (TSP) to have a numerical verification of the theory, and to show that the proposed solution obtains a speedup between 3 and 4 in comparison with the traditional algorithm.**

I. Introduction

The Boltzmann Machine (BM) is an interesting neural network which can be used in the fields of combinatorial optimization problems [1], [2], [10], knowledge representation and learning [8]. Rigorous mathematical foundations show convergence properties that can be analyzed by using techniques derived from physics [3].

The huge runtime requirements of the BM have limited its applications in real cases. Long simulation runtime is mainly due to the annealing task performed by the network, whose basic operation is the computation of the cost difference between two configurations.

Although parallel hardware approaches [4], [7] are the best solution for practical applications, efficient implementations of the BM algorithm on conventional workstations are undoubtedly very useful. When approaching a new problem, there is always an algorithm development stage to verify the validity of the BM technique and to tune the methods for weight computation. In this context, a fast sequential BM implementation becomes a primary subject of investigation, and solutions which introduce improvements in the BM algorithm without violating the BM theory can be significantly interesting.

In this letter, we analyze the problem of accelerating the BM algorithm on conventional hardware, and we show a technique to restructure the algorithm which reaches a speedup around three with respect to standard implementations. This is possible by means of the introduction of a different method to compute the cost difference between two configurations based on locality properties of the neurons.

Manuscript received April 29, 1992.

The authors are with the Department of Biophysics and Electrical Engineering (DIBE), University of Genoa, 16145, Genoa, Italy.

IEEE Log Number 9204539.

After a brief presentation of the BM model and the standard algorithm, we show the proposed implementation of the BM, an analytical estimation of the expected performance, and some experimental results on the Traveling Salesman Problem (TSP).

II. The Formal Model of the BM

For reader's convenience, we present an outline of the formal model of the BM, following [3] with minor modifications.

A BM consists of a number N of logical neurons, that can be represented by an undirected graph composed of vertices connected by edges. A number is associated with each vertex, denoting the state of the corresponding logical neuron, i.e., 0 or 1, corresponding to *"off"* or *"on"*, respectively. A configuration k of the BM is uniquely defined by the states of all individual vertices. The state of the ith vertex in configuration k is denoted by $S_i^{(k)}$.

- An edge between vertices i and j defined to be activated in a given configuration k if $S_i^{(k)}S_j^{(k)} = 1$.
- A weight $(W_{ij},)$ is associated with each edge, determining the strength of the connection between the vertices.
- A consensus function (C_k) of a configuration k measures the desirability of all the activated edges in the configuration, and can be defined as:

$$C_k = \sum_{i,j,j\geq i} W_{ij}S_i^{(k)}S_j^{(k)}. \tag{1}$$

From a given configuration k, a neighboring configuration k_i can be obtained by changing the state of the vertex i, so that:

$$S_j^{(k_i)} = \begin{cases} S_j^{(k)} & j \neq i \\ 1 - S_j^{(k)} & j = i \end{cases}. \tag{2}$$

The corresponding difference in the consensus $\Delta C_{kk_i} = C_{k_i} - C_k$ is given by:

$$\Delta C_{kk_i} = \left(1 - 2S_i^{(k)}\right)h_i \tag{3}$$

where

$$h_i = \sum_{i,j,j\neq i} W_{ij}S_j^{(k)} + W_{ii} \tag{4}$$

is the so-called "local field" of neuron i and W_{ii} is the bias of neuron i.

The difference of consensus $\Delta C_{k,k_i}$ is completely determined by the states of the neurons j connected to i and by their corresponding weights, so that it can be computed locally, thus allowing a parallel execution [7].

The probability to accept a transition (k, k_i) with cost $\Delta C_{k,k_i}$ is given by:

$$B_{kk_i} = \frac{1}{1 + e^{-\frac{\Delta C_{kk_i}}{T}}} \tag{5}$$

where T is a cooling parameter, real and positive, whose initial value T_0 is initialized on the basis of the $\sum_{ij}|W_{ij}|$ [1], and $T_{j+1} = \beta \cdot T_j$, $\beta < 1$ and β "close" to 1. The decrement rule for calculating the next value of the cooling parameter T_{j+1} is applied each time the unit has completed a number K of trials.

The annealing process begins with $T = T_0$ and several (K) consecutive trials of transitions (k, k_i) are randomly tossed according to the probability B_{kk_i}. As T approaches 0, the accepted transitions are less and less frequent and finally the BM stabilizes in a steady configuration. The whole process can be thought as a sequence of Markov chains and it can be demonstrated that for long enough chains the system converges to a unique equilibrium state [3].

The process stops when the number of state transitions after K trials at the same temperature becomes smaller than a user-provided value.

III. The Proposed Algorithm for the BM

After the selection of neuron i, the traditional algorithm for the BM annealing process operates as follows:

- compute the consensus difference $\Delta C_{k,k_i}$ that would be generated if the state transition were accepted according to (3) and (4);
- decide if the transition is to be accepted on the basis of (5);
- if the transition is accepted, update $S_i^{(k_i)} = 1 - S_i^{(k)}$.

In order to accelerate the execution, we can store the value of $h_i^{(k)}$ for each neuron i in configuration k. The initialization value is $h_i^{(0)} = W_{ii}$, assuming to begin the annealing in a configuration where all neurons are in the "off" state. In this way, the algorithm becomes:

- compute the consensus difference $\Delta C_{k,k_i}$ that would be generated if the state transition were accepted substituting $h_i^{(k)}$ in (4);
- decide if the transition is to be accepted on the basis of (5);
- if the transition is accepted, update $S_i^{(k_i)} = 1 - S_i^{(k)}$ and the local field values of all neurons but i, according to:

$$h_j^{(k_i)} = h_j^{(k)} + (1 - 2S_i^{(k)})W_{ij}. \tag{6}$$

A C-like code of the two algorithms is presented in Fig. 1. Some consideration can be drawn.

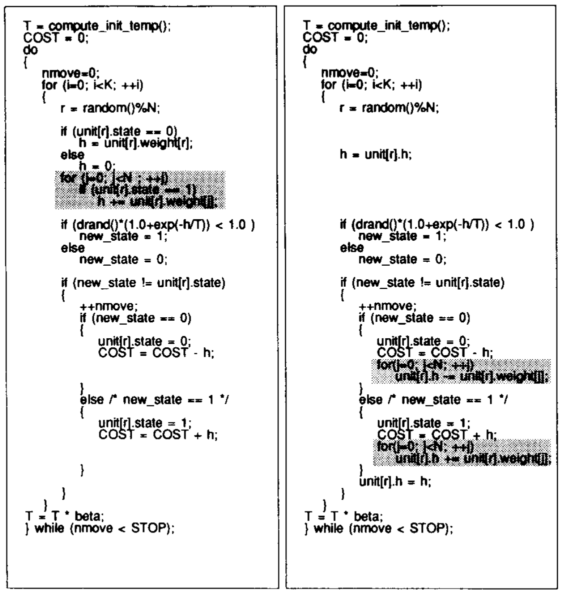
- If the number of units is large, the most time consuming part for both algorithms is represented by the innermost loop (represented by shaded area in Fig. 1). The contribution of the other instructions can be estimated approximately with a fraction of 10% in both approaches.
- The innermost loop for the proposed algorithm requires the same number of instructions than the traditional one.
- The advantage of the proposed approach (Fig. 1(b)) is that the innermost loop is executed only when a state transition is accepted. On the contrary, the traditional approach (Fig. 1(a)) requires to recompute the loop every time a tentative transition is evaluated. Since the probability to accept a state transition is always less than 0.5 (asymptotic value of (5) when T goes to infinitum), the lower bound of the speedup is 2, but we can expect a much higher gain (around 4). A precise estimation of this value is detailed in the following sections.

IV. Performance Comparison

In order to compare the two algorithms, we need to take into account both memory requirements and execution time.

Memory requirement: The proposed approach requires N more locations to store the local field for the neurons. However, the total amount of memory is $N^2 + 2N$ for the proposed algorithm, in comparison with $N^2 + N$ for the traditional one. Then both techniques require $O(N^2)$ locations, and the ratio $(N+2)/(N+1)$ becomes negligible as N becomes large.

Execution time: We can compute the total execution time of an annealing schedule for both approaches. If N_T is the number of



(a)

```
T = compute_init_temp();
COST = 0;
do
{
    nmove=0;
    for (i=0; i<K; ++i)
    {
        r = random()%N;

        if (unit[r].state == 0)
            h = unit[r].weight[r];
        else
            h = 0;
        for (j=0; j<N ; ++j)
            if (unit[r].state == 1)
                h += unit[r].weight[j];

        if (drand()*(1.0+exp(-h/T)) < 1.0 )
            new_state = 1;
        else
            new_state = 0;

        if (new_state != unit[r].state)
        {
            ++nmove;
            if (new_state == 0)
            {
                unit[r].state = 0;
                COST = COST - h;


            }
            else /* new_state == 1 */
            {
                unit[r].state = 1;
                COST = COST + h;


            }
        }
    }
T = T * beta;
} while (nmove < STOP);
```

(b)

```
T = compute_init_temp();
COST = 0;
do
{
    nmove=0;
    for (i=0; i<K; ++i)
    {
        r = random()%N;


        h = unit[r].h;




        if (drand()*(1.0+exp(-h/T)) < 1.0 )
            new_state = 1;
        else
            new_state = 0;

        if (new_state != unit[r].state)
        {
            ++nmove;
            if (new_state == 0)
            {
                unit[r].state = 0;
                COST = COST - h;
                for(j=0; j<N; ++j)
                    unit[r].h -= unit[r].weight[j];
            }
            else /* new_state == 1 */
            {
                unit[r].state = 1;
                COST = COST + h;
                for(j=0; j<N; ++j)
                    unit[r].h += unit[r].weight[j];
            }
            unit[r].h = h;
        }
    }
T = T * beta;
} while (nmove < STOP);
```

Fig. 1. A C-like description of the two algorithms: (a) traditional, (b) new. The shaded boxes highlight the difference in the two approaches.

temperature steps, and n_{loop} is the number of instruction for the innermost loop, we have:

$$\text{Time}_1 = K \cdot n_{\text{loop}} \cdot N_T \tag{7}$$

$$\text{Time}_2 = K \cdot n_{\text{loop}} \cdot \sum_{j=0}^{N_T - 1} q(T_j) \tag{8}$$

where $q(T_j)$ is the probability to accept a state transition, function of the cooling parameter $T_j = T_0 \cdot \beta^j$ according to the cooling schedule. Then we can compute the speed-up for the innermost loop as:

$$SU_{\text{loop}} = \frac{1}{\bar{q}} \tag{9}$$

$$\bar{q} = \frac{1}{N} \cdot \sum_{j=0}^{N_t - 1} q(T_0 \cdot \beta^j) \tag{10}$$

The term $\bar{q}$ represents the average probability of a state transition in the annealing schedule. To compute the overall speed-up SU_{tot} obtained by the proposed algorithm, we can apply the well-known Amdahl Law [6]:

$$SU_{\text{tot}} = \frac{1}{f_e \cdot \bar{q} + (1 - f_e)} \tag{11}$$

where f_e is the fraction enhanced (in our case approximately 0.9) of the whole computation.

A. Evaluation of State Transition Probability

We define M the stochastic event corresponding to a state transition. Applying Bayes theorem, we obtain:

$$p_T(M) = q(T) = \int_{-\infty}^{+\infty} p_T(M|\Delta C) \cdot p_T(\Delta C)\, d\Delta C \tag{12}$$

where $p_T(M|\Delta C)$ is expressed by (6), and depends on the given temperature T. As a consequence, we can write:

$$q(T) = \int_{-\infty}^{+\infty} \frac{1}{1 + e^{-\Delta C/T}} \cdot p_T(\Delta C)\, d\Delta C \tag{13}$$

where $p_T(\Delta C)$ represents the probability of having a consensus difference equal to ΔC, given a temperature T. We can estimate $p_T(\Delta C)$ in some special cases, namely in the very high and in the very low temperature ranges.

- With *high temperature* (when $T \rightarrow +\infty$), the transitions of all neurons have the same probability 0.5. Then, we can approximate $p_\infty(\Delta C)$ with a bimodal probability function with two symmetrical peaks.
- With *medium temperature*, the two peaks of the probability functions are modified in such a way that the one within the positive range increases, and the other one in the negative range decreases, leading to an asymmetrical probability density function
- With *low temperature* (when $T \rightarrow 0$), the machine converges to a (local) minimum of the problem and transitions which improve the solution becomes less and less probable. This implies that the peak within the negative range in the density function disappears.

Starting from these considerations, we can approximate the behavior of the probability density function with a linear combination of two Gaussian distributions, whose parameters and coefficient vary with the temperature. In Fig. 2(a), we show the behavior of the theoretical probability function for three different temperature regimes (high, medium, low). If we indicate with $G(x, \bar{x}, \sigma_x)$ a Gaussian distribution of the form:

$$G(x, \bar{x}, \sigma_x) = \frac{1}{\sigma_x\sqrt{2\pi}} \exp\left[(x - \bar{x})^2/2\sigma_x^2\right] \tag{14}$$

we can approximate $p_T(\Delta C)$ with:

$$\begin{aligned} p_T(\Delta C) \approx\ & \alpha(T) \cdot G\big(\Delta C, \overline{\Delta C}(T), \sigma_{\Delta C}(T)\big) \\ & + (1 - \alpha(T)) \cdot G\big(\Delta C, -\overline{\Delta C}(T), \sigma_{\Delta C}(T)\big). \end{aligned} \tag{15}$$

To compute the functions $\overline{\Delta C}(T)$, $\sigma_{\Delta C}(T)$, $\alpha(T)$, we can assume a linear dependency with the temperature. According to the exponential cooling schedule, we can express the parameters as a function of the ith iteration temperature T_i, and their asymptotic values for $T- > 0$ $(\overline{\Delta C}_0, \sigma_{\Delta C,0}, \alpha_0)$ and $T \rightarrow \infty$ $(\overline{\Delta C}_\infty, \sigma_{\Delta C,\infty}, \alpha_\infty)$:

- The symmetry of the density function for $T \rightarrow \infty$ leads to:

$$\begin{aligned} \lim_{T \rightarrow \infty} \alpha(T) = \alpha_\infty = \frac{1}{2} \\ \alpha(T_i) = \beta^i \cdot \alpha_\infty = \frac{1}{2}\beta^i. \end{aligned} \tag{16}$$

- The parameter $\overline{\Delta C}(T)$ represents the expected value of the positive (or negative) cost differences at a given temperature, that is:

$$\overline{\Delta C}(T) = \int_0^{+\infty} \Delta C \cdot p_T(\Delta C)\, d\Delta C \tag{17}$$

$$\overline{\Delta C}(T_i) = \overline{\Delta C}_0 + \beta^i \cdot \big(\overline{\Delta C}_\infty - \overline{\Delta C}_0\big). \tag{18}$$

- Similarly, the parameter $\sigma(T)$ represents the standard deviation:

$$\sigma_{\Delta C}(T_i) = \sigma_{\Delta C,0} + \beta^i \cdot (\sigma_{\Delta C,\infty} - \sigma_{\Delta C,0}). \tag{19}$$

1) Probability of ΔC with very high temperature:

When $T \rightarrow \infty$, all the state transitions have the same probability. Then we can say that:

$$\begin{aligned} \overline{\Delta C}_\infty =\ & f(0) \cdot E_i[|W_{ii}|] + f(1) \cdot (E_i[|W_{ii}|] + E_{ij}[|W_{ij}|]) \\ & + \cdots \cdots + f(k) \cdot (E_i[|W_{ii}|] + k \cdot E_{ij}[|W_{ij}|]) \\ & + \cdots \end{aligned} \tag{20}$$

where $f(k)$ represents the fraction of configurations with k units in the "on" state, and can be written as:

$$f(k) = \frac{1}{2^N} \cdot \binom{N}{k}. \tag{21}$$

Combining (20) and (21), we obtain:

$$\begin{aligned} \overline{\Delta C}_\infty &= \sum_{k=0}^{N} \frac{1}{2^N} \binom{N}{k} E_i[|W_{ii}|] + \sum_{k=0}^{N} k \cdot \frac{1}{2^N} \binom{N}{k} E_{ij}[|W_{ij}|] \\ &= E_i[|W_{ii}|] + E_{ij}[|W_{ij}|] \cdot \frac{1}{2^N} \cdot \sum_{0k}^{N} k \cdot \binom{N}{k} \\ &= E_i[|W_{ii}|] + \frac{N}{2} E_{ij}[|W_{ij}|]. \end{aligned} \tag{22}$$

The same considerations can be applied for $\sigma_{\Delta C,\infty}$:

$$\sigma_{\Delta C,\infty} = \sigma_i[|W_{ii}|] + \frac{N}{2} E_j[\sigma_i[|W_{ij}|]] \tag{23}$$

2) Probability of ΔC with very low temperature:

Following the same considerations of the paragraph above, we can compute an estimate of $\overline{\Delta C}_0$ and $\sigma_{\Delta C,0}$. In this case, as $T \rightarrow 0$, the values of the parameter depend on the number r of neurons which have to be in the "on" state in the best solution. The parameter r depends on the applications to which the algorithm is applied. We can say that:

$$\overline{\Delta C}_0 = E_i[|W_{ii}|] + r \cdot E_{ij}[|W_{ii}|] \tag{24}$$

$$\sigma_{\Delta C,0} = \sigma_i[|W_{ii}|] + r \cdot E_j[\sigma_i[|W_{ij}|]]. \tag{25}$$

B. Speedup Computation

Combining (13) and (15), we can write an approximation of the state transition probability (see (26) at the bottom of the page). Then, from (9), (10), and (11) we can also compute an approximation of the speedup of the proposed algorithm in comparison with the traditional one.

Unfortunately, integral (26) is not solvable in a closed form, but we can evaluate it through numerical methods. As an example, in the following section we apply the described procedure to a well-known

$$q(T) = \int_{-\infty}^{+\infty} \frac{\frac{1}{\sigma_{\Delta C}(T)\sqrt{2\pi}} \left(\alpha(T) \cdot \exp -(\Delta C - \overline{\Delta C}(T)) \,/\, \sigma_{\Delta C}(T) + (1 - \alpha(T)) \cdot \exp -(\Delta C + \overline{\Delta C}(T))/\sigma_{\Delta C}(T)\right)}{1 + \exp -\left(\Delta C/_T\right)}\, d\Delta C. \tag{26}$$

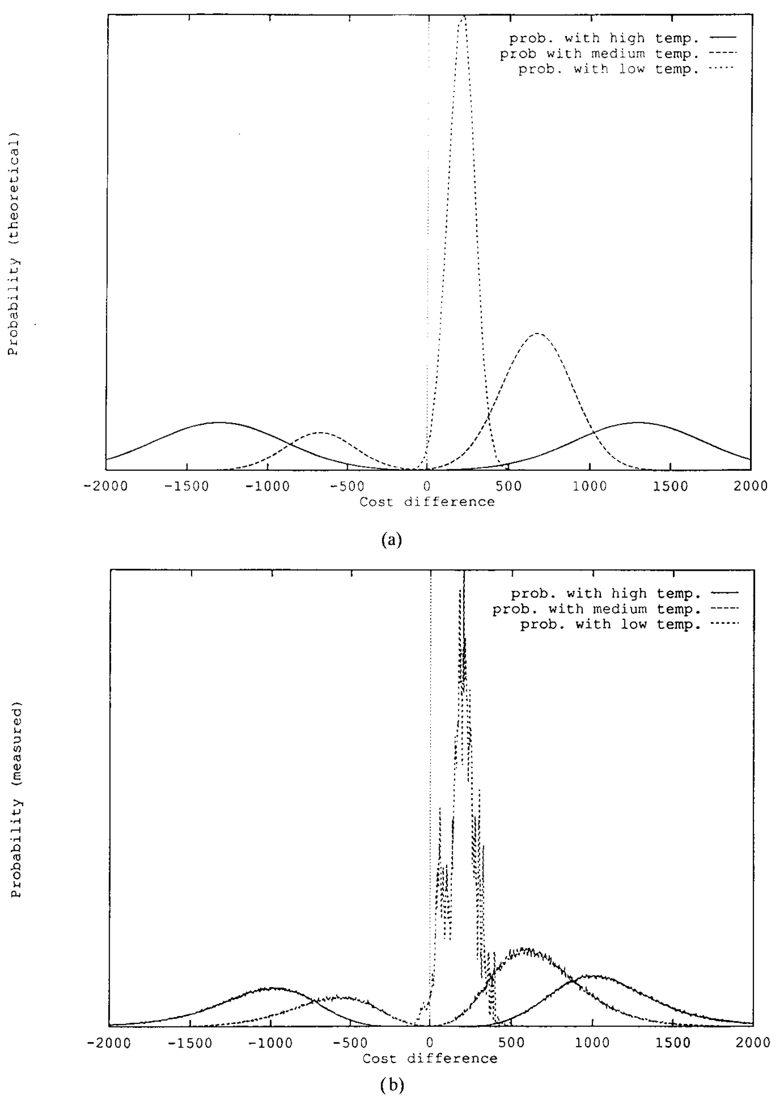


Fig. 2. Behavior of the probability density function of the cost difference in different temperature regimes: (a) theoretical; (b) experimental (10-city TSP).

np-complete optimization problem (the TSP). Similar results can be obtained also for other kinds of problems.

V. Algorithm Performance for the TSP Problem

A significant example where we can apply the detailed analysis is the TSP [9]. The TSP is a classical optimization problem that consists in finding a minimum cost tour in an n-vertex directed graph, that visits each vertex exactly once.

The approach to solve the TSP with a BM rewrites it as an instance of a 0-1 programming problem, in terms of a Quadratic Assignment Problem and can be found extensively described in [5].

For a 10-city TSP problem, we have computed the values of the parameters, which are expressed in Table I.

Fig. 2(b) shows $p_T(\Delta C)$ measured for three different regimes. If we compare Fig. 2(a) (b), the qualitative accuracy of the developed theoretical model is confirmed by the similarity of the behaviors.

We have numerically computed integral (26) with the parameters of Table I, obtaining the values of $q(T)$ for the different temperature values of an annealing schedule. The results of the computation are shown in Fig. 3, together with the experimental measurements of the same values in a sample annealing schedule of the BM. Also in this case, the model demonstrates to be rather precise, leading to an error less than 3% for the computation of the average probability of a state transition $(\bar{q})$, as it is reported in Table II.

TABLE I
Parameters for the 10-City TSP

$E_i[W_{ii}]$	71.7
$E_{ij}[W_{ij}]$	27.3
$\sigma_i[\|W_{ii}\|]$	18.7
$E_j[\sigma_i[\|W_{ij}\|]]$	6.1
r	10
$\overline{\Delta C}_\infty$	1436.7
$\overline{\Delta C}_0$	344.7
$\sigma_{\Delta C,\infty}$	323.7
$\sigma_{\Delta C,0}$	79.7

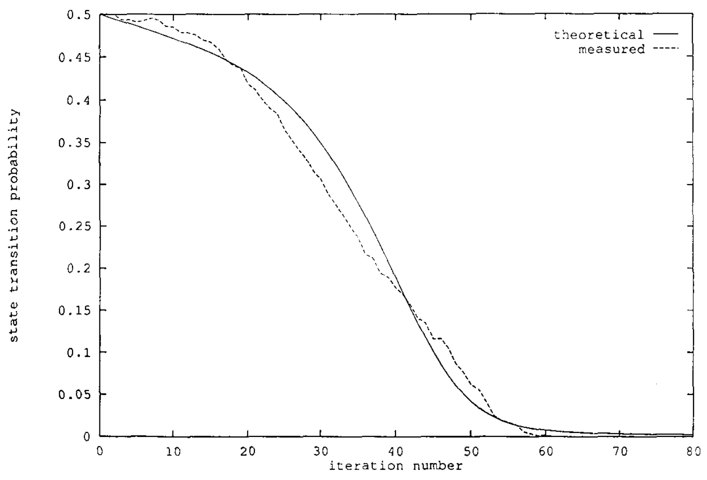


Fig. 3. Behavior of the state transition acceptance probability in a sample annealing run for the TSP example.

TABLE II
AVERAGE STATE TRANSITION ACCEPTANCE PROBABILITY ($\bar{q}$) (10-CITY TSP, 78 TEMPERATURE STEPS)

Theoretical	0.2420
Measured	0.2485
Error (%)	2.61

TABLE III
AVERAGE CPU TIME PER TEMPERATURE STEP (10-CITY TSP, 1000 ITERATIONS PER TEMPERATURE STEPS)

Theoretical Algorithm	486 msec.
Measured Algorithm	152 msec.
Speedup	3.2

Applying (9) and (11), we obtain:

$$SU_{\text{tot}} = \frac{1}{0.9 \cdot 0.24 + 0.1} \approx 3.1.$$

This value is coherent with the measured execution time ratio between the traditional and the proposed BM algorithm, as it is reported in Table III. Data are obtained as an average on 10 annealing runs on the 10-city TSP problem. The program was written in C and executed on a DEC 5000 workstation.

VI. CONCLUSIONS

In this letter we have presented an enhancement of the algorithm of the BM for a sequential implementation. We have also shown a mathematical analysis for the evaluation of the state transition probability and tested its accuracy with the TSP problem.

The proposed approach allows to reach a speedup around three with respect to the traditional algorithm, and does not affect the convergency properties of the network. This can be advantageous to accelerate the sequential algorithm, in particular in the network development and tuning phases.

Obviously, future applications of the BM will require massively parallel processing to reach order of magnitudes of improvement. In this context, the proposed method still retains its validity, as the introduced modifications increase the locality of the algorithm and make it particularly suited for a parallel implementation [7].

REFERENCES

[1] E. Aarts and J. Korst, "Boltzmann machines and their applications," in *Lecture Notes in Computer Science.* New York: Springer Verlag, 1987, no. 258.

[2] E. H. L. Aarts and J. H. M. Korst, "Combinatorial optimization on a Boltzmann machine," in *Proc. European Seminar on Neural Computing,* London, UK, 1988.

[3] ——, *Simulated Annealing and Boltzmann Machines.* New York: Wiley, 1988.

[4] ——, "Computations in massively parallel networks based on the Boltzmann machine: A review," in *Parallel Computing.* Amsterdam, The Netherlands: North-Holland, 1989, no. 9, pp. 129–145.

[5] ——, "Boltzmann machines for traveling salesman problems," in *European Journal of Operational Research.* Amsterdam, The Netherlands: North Holland, 1989.

[6] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing," in *Proc. AFIPS 1967 Spring Joint Conf.,* 1967.

[7] A. De Gloria, P. Faraboschi, and S. Ridella, "A dedicated massively parallel architecture for the Boltzmann Machine," in *Parallel Computing.* Amsterdam, The Netherlands: North Holland, 1992, no. 18.

[8] G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Eds. Cambridge, MA: Bradford Books, 1986.

[9] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan, and D. B. Shmoys, *The Traveling Salesman Problem.* New York: Wiley, 1990.

[10] V. Zissimopoulos, V. Th. Paschos, and F. Pekergin, "On the approximation of NP-Complete Problems by Using the Boltzmann Machine Method: The Case of Some Covering and Packing Problems," *IEEE Trans. Computers,* vol. 40, Dec. 1991.