Neural, Parallel & Scientific Computations 3 (1995) 451-466

A PARALLELIZABLE OPERATION SCHEME OF THE BOLTZMANN MACHINE OPTIMIZER BASED ON GROUP UPDATES

A. Likas, G. Papageorgiou and A. Stafylopatis

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

Abstract

An operation scheme for the Boltzmann Machine Optimizer is proposed that is suitable for parallelization and is based on the notion of group updates. It has the advantage of suggesting transitions between states that differ in more than one unit and exhibits greater flexibility in accepting such transitions compared to the pure sequential case. The performance of the method is evaluated on the Weighted Maximum Independent Set problem and comparisons with the pure Boltzmann Machine are presented concerning both solution quality and convergence speed. A parallel algorithm is formulated that ensures accurate cost calculations. Implementation on both shared memory and distributed memory architectures has yielded very good speedup.

1. INTRODUCTION

The Boltzmann Machine (Aarts & Korst, 1989; Ackley et al., 1985) constitutes an approach to combinatorial optimization that can be considered as a neural network implementation of the Simulated Annealing methodology (Kirkpatrick et al., 1983) and has been successfully employed for providing near-optimal solutions to many NP-complete problems (Aarts & Korst, 1989; Zissimopoulos et al., 1991). The function to be minimized (*energy function*) is determined by the connection weights w_{ij} and thresholds θ_i of the neural network and has the following quadratic form:

$$E(\vec{y}) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j w_{ij} - \sum_{i=1}^{n}\theta_i y_i \qquad (1)$$

where n is the number of units, $\vec{y} = (y_1, \ldots, y_n)$ is the binary state of the network ($y_i \in \{0, 1\}$) and the weight matrix is considered to be symmetric ($w_{ij} = w_{ji}$) with zero diagonal elements ($w_{ii} = 0$).

The basic idea is to encode the objective function and the constraints of the problem at hand in terms of an energy function of the above form and employ an annealing schedule from an initial high temperature down to a temperature near zero. The operation of the network is strictly sequential. At each instant a new state is generated by randomly selecting one unit i and changing its state. Then the energy difference between the current state $\vec{y}$ and the generated state is computed as follows:

$$\delta E_i(\vec{y}) = (2y_i - 1)(\sum_{j=1}^{n} w_{ji} y_j + \theta_i) \qquad (2)$$

Received June 1995

Based on this energy difference, a desicion is made about whether the generated state will be accepted or not. Two kinds of *acceptance criteria* that are commonly used are the *logistic* criterion and the *Metropolis* criterion. In the logistic case the change is accepted at temperature T with probability $p_i = 1/(1 + \exp(\delta E_i(\vec{y})/T))$, while in the Metropolis case the change is accepted with the above probability p_i only if $\delta E_i \geq 0$, otherwise it is accepted with probability 1. We shall use the term *trial* to denote the operations of next state generation and computation of δE_i and p_i. The acceptance of a suggested transition, i.e., the change of the network state will be referred to as *update*. Hence, depending on the acceptance probability, a trial may be eventually followed by an update. As the temperature value tends to zero, the probability of performing updates that increase the energy also tends to zero, and the network converges to an *one-update equilibrium state*, in the sense that there is no other network state differing in the state of one unit that is of lower energy value.

Since a Boltzmann Machine Optimizer constitutes in essence a special case of the Simulated Annealing method, the same results concerning asymptotic convergence to the global minimum point under certain assumptions that have been proved for Simulated Annealing (Hajek, 1988; Kirkpatrick et al., 1983) carry over to the Boltzmann Machine case too (Aarts & Korst, 1989). The same holds for the finite time implementations of the algorithm that attempt to approximate the global minimum. Simulated annealing is commonly described as a sequence of Markov chains, each corresponding to a temperature value. Every computational step of a chain starts only after the previous step has been completed. Thus, the operation of the Boltzmann Machine is strictly sequential and may require large computation time as the size of the problem grows. Moreover, in order for the annealing to be effective, the stationary distribution (or at least a quasi-equilibrium distribution) must be restored at each temperature, thus, sufficient state transitions must take place and consequently a large number of trials is required.

In the operation scheme presented in this paper the probability of accepting the state transition suggested at each trial is increased, thus, the expected number of trials performed at each temperature is reduced. The proposed scheme is based on the notion of *group update*; instead of selecting a single unit to update at each time step, we consider a group of units which are selected and updated simultaneously. Since there exists a significant amount of parallelism in the computation of the resulting energy difference, we have developed an operation scheme exploiting this parallelism that examines many group configurations at each step and finally performs a trial (and possibly an update) for the group that is characterized by the minimum energy difference. In this way, the state space is more adequately explored and it is much easier to escape from local minimum states than in the pure Boltzmann Machine case which is based on one-update transitions.

The next section concerns a brief discussion of parallel simulated annealing techniques. The general group update approach, which has been introduced in (Likas & Stafylopatis, 1996), is concisely described in Section 3. In Section 4 we present the operation of the Boltzmann Machine under the proposed mode which is based on group updates, whereas some relevant issues are discussed in Section 5. Experimental results obtained from application of our method to the Weighted Maximum Independent Set on both sequential and parallel hardware are discussed in Section 6. Finally, the main conclusions are contained in Section 7.

2. PARALLELISM IN SIMULATED ANNEALING

Although the Simulated Annealing (SA) approach is of inherently sequential nature, several attempts have been made to reduce the required computation time through

the use of parallel machines (Aarts & Korst, 1989; Azencott, 1992; Greening, 1990; Roussel-Ragot & Dreyfus, 1990). Parallel implementations can be obtained either on general purpose parallel machines (for example on Transputer-based machines (Barbosa & Lima, 1990; d'Acierno & Vaccaro, 1992)) or using special purpose VLSI (Alspector & Allen, 1987; Skubiszewski, 1992) or optical neuroprocessors (Ticknor & Barret, 1987).

Three kinds of parallelism can be considered with respect to the type of computation performed:

- *Functional decomposition*, namely parallelism in the evaluation of the cost function at each move. As can be observed from equation (2), the Boltzmann Machine is characterized by this kind of parallelism.
- *Spatial (or domain) decomposition*, where state variables are distributed among processors and variable updates are transmitted between processors to generate new states.
- *State-space sharing* techniques, where many trials are performed in parallel and one or many transitions can be made simultaneously.

According to the synchronization policy adopted and the resulting convergence properties, a comprehensive taxonomy of parallel simulated annealing techniques has been presented in (Greening, 1990):

- *Serial-like* algorithms identically preserve the convergence properties of sequential simulated annealing. Serial-like algorithms either involve functional decomposition or exploit sets of *serializable* transitions (Kravitz & Rutenbar, 1987), i.e. transitions that can be concluded in any order yielding the same result.
- *Altered generation* algorithms modify state generation, thus changing the pattern of state space exploration, but perform accurate cost function calculations. They are generally based on spatial decomposition or shared state-space techniques.
- *Asynchronous* algorithms do not retain the synchronization present in the above two categories and have the disadvantage of tolerating erroneously calculated state transitions. They are also based on spatial decomposition or shared state-space techniques.

Each category is characterized by some sort of trade-off between solution quality and speedup. As different schemes may be appropriate at different temperatures, several hybrid approaches have also been examined (Greening, 1990; Kravitz & Rutenbar, 1987).

The scheme proposed here, which is based on group updates, combines several features of the kinds of parallelism described above. Moreover, although the states of many units may be updated in parallel, the trial is performed using the *correct energy difference*, thus excluding the possibility of erroneous transitions.

3. GROUP UPDATES

The notion of group update constitutes a generalization of the notion of single update (Likas & Stafylopatis, 1996). The difference lies in the way in which the generation of the new state to be tested is performed. Instead of selecting a single unit to consider for update, we can select a group containing a number of units. In a manner analogous to the single unit case, we first perform a trial by calculating the difference in the

energy that would result if the state of *all* units in the group were altered. Then a decision must be made of whether a group update must take place, i.e., whether the states of *all* these units must change or not.

It is clear that group operation differs from the previously mentioned *synchronous* operation (Aarts & Korst, 1989), which consists of simultaneously performing individual trials and updates (unlimited parallelism), in that it performs only one trial using the correct energy difference.

Consider a Boltzmann Machine with n binary units as defined in the previous section. Suppose that while being in state $\vec{y} = (y_1, \ldots, y_n)$ during operation, we consider a group G of units and perform a trial on this group. If the trial is successful, a group update takes place and the new state $\vec{z} = (z_1, \ldots, z_n)$ of the network will be such that

$$z_i = \begin{cases} y_i + u_i & \text{if } i \in G \\ y_i & \text{if } i \notin G \end{cases} \tag{3}$$

where $u_i = 1 - 2y_i$, $i \in G$. Equation (3) simply expresses the fact that units in G change state.

The energy of the new state $\vec{z}$ can be obtained from (1) with z_i in place of y_i. Using (3) and based on the symmetry of the weights we find after performing some algebra:

$$\begin{aligned} E(\vec{z}) &= E(\vec{y}) - \sum_{i \in G} u_i \Big(\sum_{j=1}^{n} y_j w_{ij} + \theta_i\Big) \\ &\quad - \frac{1}{2} \sum_{i \in G} \sum_{j \in G} u_i u_j w_{ij} \end{aligned} \tag{4}$$

Thus, the change $\Delta E_G(\vec{y}) = E(\vec{z}) - E(\vec{y})$ in the network's energy due to the group update is given by:

$$\Delta E_G(\vec{y}) = \sum_{i \in G} \delta E_i(\vec{y}) - \frac{1}{2} \sum_{i \in G} \sum_{j \in G} u_i u_j w_{ij} \tag{5}$$

where, according to (2), $\delta E_i(\vec{y})$ is the change in the network's energy in case unit i is selected for a *single* update from state $\vec{y}$. If an ordering is imposed on the units of group G, equation (5) can be written in the following way:

$$\Delta E_G(\vec{y}) = \sum_{i \in G} \delta E_i(\vec{y}) - \sum_{i \in G} \sum_{j<i} u_i u_j w_{ij} \tag{6}$$

which is the formula that has been considered in our implementation.

It can be easily verified that the change in the network's energy due to a group update from state $\vec{y}$ as given by (5) is equal to the sum of the individual energy changes that would result if the units of the group sequentially changed state starting from $\vec{y}$ and following an arbitrary order. This provides a more intuitive interpretation of the above result and can be seen as a consequence of energy conservation.

It is obvious that in order for an operation style based on group updates to be successful, the identification and selection of appropriate groups of units must take place and this task is highly problem-dependent. The operation scheme described in the next section attempts to identify promising groups by examining many groups at each step and selecting the one with minimum energy difference.

4. PARALLEL GROUP-UPDATE OPERATION OF THE BOLTZMANN MACHINE

Group updates allow the Boltzmann Machine to perform a wide exploration of the state space, since transitions are enabled to states not belonging to the immediate neighbourhood of the current state. This is particularly desirable in the beginning of the search process to enhance mobility so that the system can rapidly attain regions of low energy. As the process advances, the search becomes more localized to perform a refinement of the obtained solution. This implies the idea of systematically adapting the breadth of the search area by adjusting the size of groups considered for update. The operation scheme described next is based on applying a group-size schedule simultaneously with the annealing schedule concerning the temperature parameter. Moreover, as the group-update technique provides an efficient way for computing the energy difference at each trial, the proposed scheme can take advantage of parallelism to achieve fast execution.

4.1 Group-Update Scheduling

Consider that, at each time step, λ network units are selected and the corresponding energy differences δE_i, that would result if the state of each unit were altered, are computed using equation (2). Then, using equation (6), several grouping possibilities among the λ selected units are considered and an appropriate group for update is identified. Obviously, among the possible groupings, the group G^* with the (algebraically) minimum ΔE_G would be the most desirable choice. A trial concerning this group is then performed using either the Metropolis or the logistic acceptance criterion. If the trial is successful, the state of the units participating in the group is appropriately updated. From the above description it is clear that, to specify next-state generation, two phases must be envisaged: unit selection and group selection. Both issues will be discussed in later subsections in connection with the implementation of the algorithm.

Comparing the above scheme with the pure Boltzmann Machine operation we can say that, although there is an increase in the number of computations performed at each step, there are two main advantages: i) we consider group updates instead of single updates, which provides wider search around each state, and ii) we incorporate sophistication in the next state generation scheme, i.e., we perform trials concerning the state with the lowest possible energy among the candidate states, thus, the probability of performing unsuccessful trials is decreased. These qualitative observations are made more clear through the following proposition.

Proposition 1. *Let $S = \{s_1, \ldots, s_\lambda\}$ be the set of randomly selected units and $G^* \subseteq S$ the group with minimum energy difference δE_{G^*}. Let also $H = (h_1, \ldots, h_m)$ ($h_i \in S$, $m \leq \lambda$) denote an arbitrary sequence of (serial) updates concerning units of S and $p_H(T)$ the probability that the entire sequence of updates be accepted at temperature T. Then for all H, it holds that*

$$p_{G^*}(T) \geq p_H(T) \tag{7}$$

where $p_{G^*}(T)$ is the probability that the group update concerning group G^* be accepted at temperature T.

Proof: Let δE_{h_i} denote the energy difference at the trial concerning unit h_i and $p_{h_i}(T)$ the probability of accepting the update, given that all preceding updates have been accepted following the order implied by the sequence H. Then the probability

that all updates of the sequence be accepted is given by

$$p_H(T) = \prod_{i=1}^{m} p_{h_i}(T)$$

Following the discussion of the previous section, the change in the network's energy due to a group update from state $\vec{y}$ is equal to the sum of the individual energy changes that would result if the units of the group sequentially changed state starting from $\vec{y}$ and following an arbitrary order. Considering the way of selecting group G^*, it holds that

$$\delta E_{G^*} \leq \sum_{i=1}^{m} \delta E_{h_i} \tag{8}$$

We consider now the following two acceptance criteria:

- *Logistic criterion.*
  In this case $p_{h_i} = 1/(1 + \exp(\delta E_{h_i}/T))$ and $p_{G^*} = 1/(1 + \exp(\delta E_{G^*}/T))$. From equation (8) we have

$$p_{G^*}(T) \geq \frac{1}{1 + \exp(\sum_{i=1}^{m} \delta E_{h_i}/T)} \tag{9}$$

  which yields

$$\frac{1 - p_{G^*}(T)}{p_{G^*}(T)} \leq \exp(\sum_{i=1}^{m} \delta E_{h_i}/T) = \prod_{i=1}^{m} \frac{1 - p_{h_i}(T)}{p_{h_i}(T)} \tag{10}$$

  Since $0 \leq p_{h_i}(T) \leq 1$, it holds that $\prod_{i=1}^{m}(1 - p_{h_i}(T)) \leq 1 - \prod_{i=1}^{m} p_{h_i}(T)$. Thus, from (10) we finally obtain

$$\frac{1 - p_{G^*}(T)}{p_{G^*}(T)} \leq \frac{1 - p_H(T)}{p_H(T)} \tag{11}$$

  which means that $p_{G^*}(T) \geq p_H(T)$.

- *Metropolis criterion.*
  If $\delta E_{G^*} < 0$ we have that $p_{G^*}(T) = 1$, thus (7) holds. If $\delta E_{G^*} \geq 0$ then each sequence H will contain some units $j_1, \ldots, j_k$ $(1 \leq k \leq m)$ for which $\delta E_{j_i} > 0$ and

$$\delta E_{G^*} \leq \sum_{i=1}^{k} \delta E_{j_i} \tag{12}$$

  Since for the units with $\delta E_{h_i} < 0$ it holds that $p_{h_i}(T) = 1$ we have that $p_H(T) = \prod_{i=1}^{k} p_{j_i}(T)$. In a manner similar to the logistic case it can be shown that $p_{G^*}(T) \geq p_H(T)$.

∎

The main implication of the above proposition is that, in the case of group-update operation, a faster annealing schedule can be employed, especially in what concerns the number of trials performed at each temperature. Since the probability p_{G^*} is higher than in the sequential case, less trials are required at each temperature in order for the corresponding Markov chain to reach a quasi-equilibrium state (Aarts & Korst, 1989). Also the transitions that lead to a decrease in energy are more steep and the network approaches near-optimal states faster. Moreover, the escape from local

minima is facilitated since the generated next state (with minimum positive energy difference) is the one that is more easy to approach. All these factors compensate for the increase in complexity due to the group update computations.

A local minimum state under group-update operation is one for which there is no group G having negative energy difference ΔE_G among all the $2^\lambda - 1$ possible groups. We will refer to a local minimum state of this type as a *group-update equilibrium.* It is clear that a group-update equilibrium is associated with a given set of selected units and, therefore, refers to a particular step of the process. Since the number of possible groups is large in general, it is not easy to check whether a given state constitutes an equilibrium point in the above sense. On the other hand, it is obvious that a group-update equilibrium is also an one-update equilibrium. When a problem is mapped onto the Boltzmann Machine, the one-update local minima are the states that are of interest, since in most cases they correspond to feasible problem solutions (Aarts & Korst, 1989; Zissimopoulos et al., 1991). Therefore, as a termination criterion for the group-update approach, we chose to check whether the current state constitutes an one-update equilibrium, and, in addition, we require that for a number of consecutive temperature values, no (group) transition has been accepted, i.e., the equilibrium point in our approach is 'more' than a local equilibrium of the pure Boltzmann Machine case.

4.2 Exploiting Parallelism

The key concept in group-update operation is the computation of ΔE_G according to equation (6). This computation is based on the individual energy differences δE_i that can be computed independently for each unit participating in group G. This fundamental operation, therefore, is characterized by functional parallelism as described in Section 2. Moreover, partitioning of this job to parallel processors may correspond to appropriate distribution of network units to processors, thus featuring a kind of spatial parallelism. Finally, we may observe that, since the available δE_i can be used for evaluating several candidate moves corresponding to different groupings, a possibility of state-space sharing is also offered. Clearly, at a given time instant during execution, the degree of parallelism provided by the parallel computing system can be greater than, less than or equal to the group size λ.

In what concerns the variation of the group size λ, and in particular its initial values, we have been guided by practical considerations. First, the group size cannot be very large, since the computational cost of searching for an appropriate group at each step would be prohibitively high. Second, a systematic experimentation has revealed that, even if a large value of λ is considered, the average effective size of the selected group does not exceed some observed range of values. We have thus determined an efficient suite of values for λ, that have been used in the group-size schedule, as explained in Section 6.

Due to parallelism, we can easily compute p individual energy differences δE_i at each step, where $p \geq \lambda$. Therefore, the set $S' = \{s'_1, \ldots, s'_p\}$ is generated at each step and the corresponding energy differences $\delta E_{s'_i}$ are computed. Then λ out of the p units of S' are selected and the set $S \subseteq S'$ is constructed. We can benefit from this redundancy and adopt the units corresponding to the λ lowest δE_i values, since, according to equation (6), these units are more probable to construct groups with low ΔE_G.

To select the p units, the network is partitioned into p regions containing equal number (n/p) of units. One unit from each region is then randomly selected and a set of p candidate for update units is formed. This type of partitioning can be readily implemented on both shared-memory and distributed-memory architectures.

- *Map problem on Hopfield Network (Boltzmann Machine).*
- *Initialize output vector $\vec{y}$ to random binary values.*
- *Repeat steps 1–7 until terminating criterion is satisfied:*
  1. *Periodically update group size λ.*
  2. *Periodically update temperature T.*
  3. *Select p out of the n units.*
  4. *Compute δE_i for every selected unit.*
  5. *Select the λ out of the p units having the lowest δE_i values.*
  6. *Find the most appropriate group G with energy difference ΔE_G.*
  7. *Perform trial according to the Metropolis criterion. If succesfull, then change state y_i for every unit $i \in G$.*

Figure 1: The Group-Update Algorithm

The proposed Group-Update algorithm is summarized in Figure 1.

5. COMPUTATION AND CONVERGENCE ISSUES

5.1 Implementation of Group Selection

As far as group selection is concerned, since in general $\lambda << n$, a first approach would be to compute the energy difference corresponding to *all* $2^\lambda - 1$ possible groupings concerning the λ selected units and find the group G^* with the (algebraically) minimum ΔE_G. This scheme has the advantage of considering all possible grouping decisions and facilitates escape from local minima, especially at low temperature values where it is more difficult to find transitions that lead to a decrease in the energy of the network. However, as this option risks to be computationally expensive, care has been given to its efficient implementation and the possibility of restricting the exhaustive search has been examined.

Given the set $S = \{s_1, \ldots, s_\lambda\}$ of the selected units, it can be readily obtained from equation (5) that the task of finding the group G^* with the minimum energy difference is equivalent to finding the global minimum point $\vec{r} = (r_1, \ldots, r_\lambda)$ ($r_i \in \{0, 1\}$, $\vec{r} \neq \vec{0}$) of the following function $Q(\vec{r})$:

$$Q(\vec{r}) = -\frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{\lambda} r_i r_j z_{ij} - \sum_{i=1}^{\lambda} \zeta_i r_i \tag{13}$$

where $r_i = 1(0)$ means that unit s_i belongs (does not belong) to group G^*, $z_{ij} = z_{ji} = u_i u_j w_{ij}$ and $\zeta_i = -\delta E_{s_i}$.

If the above global minimum is to be found through explicit enumeration of the $2^\lambda - 1$ possible values of $\vec{r}$, an efficient implementation can be obtained by considering

the vectors $\vec{r}$ in a sequence corresponding to Gray coding, according to which each vector differs from the previous one in just one component. Consequently, if $\vec{r}$ is the current vector and $\vec{r'}$ differring from $\vec{r}$ in component i is the next one in the Gray sequence, we have:

$$Q(\vec{r'}) = Q(\vec{r}) + (2r_i - 1)(\sum_{j=1}^{p} r_j z_{ij} + \zeta_i) \tag{14}$$

This possibility of incrementally computing the energy differences significantly reduces the required computational effort.

For large λ a near-optimal group selection $\vec{r}$ can be determined using some heuristic search technique. A genetic algorithm (Goldberg, 1989; Michalewicz, 1994) can be considered for this purpose. This idea has been tested experimentally yielding satisfactory results.

It is interesting to note that the function $Q(\vec{r})$ can be viewed as an energy function that can be minimized by a Boltzmann Machine with λ units, connection weights z_{ij} and threshold values ζ_i. Therefore, a lower level Boltzmann Machine could operate within the original one, providing at each step a group appropriate for selection. Although the idea seems appealing, we have not resorted to this approach, because it has the disadvantage of having in many cases the zero state as an equilibrium point, which is not acceptable in our scheme, since it does not suggest any transition.

5.2 Asymptotic Convergence

A theoretical implication of incorporating sophistication in the process of next state generation is that one of the conditions for asymptotic convergence of the Boltzmann Machine to the global minimum state is no longer valid. According to (Aarts & Korst, 1989), the first requirement for convergence is the ability to generate a path (finite number of transitions) from any state to an optimal solution. The random selection of units at the beginning of each step satisfies this requirement. The second condition is related to the symmetry in the probability G_{xy} of performing a trial concerning state y while being in state x, i.e. to the property $G_{xy} = G_{yx}$. This symmetry holds in the pure Boltzmann Machine case where the unit to be updated is randomly selected, but it is not valid in the proposed approach since trials are performed only between states with minimum energy difference. Therefore, if while being in a state x a trial is performed for moving to state y with $\Delta E_G < 0$, it is not likely that while being in state y a trial would be performed for moving to state x since the corresponding energy difference would be positive. In other words, the proposed scheme favors the generation of next states that are of low energy with respect to the current one.

Although the lack of proof of asymptotic convergence does not seem to affect the performance of the method, the problem can be overcome by introducing a probability q of applying the group update scheme at each step. This means that at each step we decide with probability q whether a group trial will be performed or not. In the latter case, a trial concerning a randomly selected single unit takes place as in the pure case. This modified scheme still lacks symmetry, which is a sufficient, but not necessary condition for convergence. Following the proof provided by Hajek in (Hajek, 1988), it can be shown that under this scheme the global convergence property is retained, provided a sufficiently slow logarithmic cooling schedule is employed. Although in practice the probability q can be set small enough so that normal group-update operation is not disrupted, we have not considered this option in our implementation.

6. EXPERIMENTAL RESULTS

The effectiveness of the proposed approach has been tested on instances of the Weighted Maximum Independent Set (MIS) problem. The Weighted Maximum Independent Set constitutes an imprortant discrete optimization problem and many other problems (for example Set Partitioning, Set Packing, Set Covering etc. (Zissimopoulos et al., 1991)) can be solved through its solution. Moreover the use of the Boltzmann Machine for the solution of this problem has proved very successful (Aarts & Korst, 1989; Zissimopoulos et al., 1991).

The formulation of the MIS problem (weighted case) is the following: Consider an undirected graph $G = (V, E)$ where V (with $|V| = n$) is the set of vertices and E denotes the set of edges. Let also A denote the adjacency matrix of graph G, i.e., $a_{ij} = 1$ if $(i,j) \in E$, otherwise $a_{ij} = 0$. An *independent set* V' of this graph is a subset of V that contains vertices not connected to each other. If $c : V \rightarrow \Re^+$ is a cost function assigning a cost to each vertex, the Maximum Independent Set problem is to find the independent set V' of maximum cost, where the cost of the set V' is defined as $f_c(V') = \sum_{k \in V'} c_k$.

A neural network architecture suitable for the MIS consists of n nodes with the following specification of weights w_{ij} and threshold values θ_i (Aarts & Korst, 1989; Zissimopoulos et al., 1991):

$$w_{ij} = \begin{cases} -\{\max\{\theta_i, \theta_j\} + \epsilon\} a_{ij} & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases} \tag{15}$$

$$\theta_i = c_i \tag{16}$$

where ϵ is a very small positive value (which is set equal to 0.5 in our experiments). This specification of weights and thresholds ensures that every one-change local minimum state corresponds to an independent set of the graph. Each such set is maximal in the sense that no other vertex can be added to it without violating the disjointness constraint. Moreover, the resulting energy function is *order preserving* (Aarts & Korst, 1989) in the sense that the lower the final energy value, the better the cost of the final solution.

6.1 Sequential Implementation

At first we examined the performance of a sequential implementation of the group-update approach on a Power Challenge (Silicon Graphics) machine. We considered three graphs with 480, 800 and 1600 vertices respectively that were constructed by deciding with probability 0.1 for each pair of vertices whether there would be an edge connecting the vertices of this pair. The cost of each vertex was an integer value specified through uniform selection in the range between 20 and 50.

Experiments were conducted for both the conventional and the group-update operation schemes of the Boltzmann Machine using $p = 16$ and initial $\lambda = 16$ in the latter case. The annealing schedule that was used in all tests has the following logarithmic form:

$$T_k = \frac{T_{k-1}}{1 + \log f(k)} \tag{17}$$

where $f(k) = f(k-1)(1+r)$ (with $f(0) = 1$) and $T_0 = 20$, $r = 0.001$ denote the initial temperature and the reduction rate respectively. For the conventional Boltzmann Machine we considered that n trials are performed at each temperature step and the annealing terminates if no update has been accepted for $3n$ consecutive time steps (trials).

	Boltzmann Machine			
n	*Avg. Cost*	*Best Cost*	*Avg. Time (sec)*	*Avg. Steps*
480	1889.5	1983	9.4	55027
800	2013.7	2087	25.8	90287
1600	2420.5	2547	183.1	189681

Table 1: Experimental results for Boltzmann Machine

	Group Update Scheme			
n	*Avg. Cost*	*Best Cost*	*Avg. Time (sec)*	*Avg. Steps*
480	1988.5	2112	72.8	30082
800	2158.9	2260	226.9	56165
1600	2618.2	2772	952.6	113604

Table 2: Experiments for Group Update Scheme (pure exhaustive search)

For the Group-Update scheme n/λ trials are performed at each temperature and the annealing terminates if no update has occured for $3n$ consecutive time steps. As far as the group-size schedule is concerned we considered the following fast dropping scheme: given an initial value (which was taken equal to 16) λ drops to half at time instants, such that the intervals between successive changes of λ constitute an increasing sequence. The interval length is initially equal to 100 time steps and is multiplied by 10 after each change of λ. If λ attains the value 1, it remains unchanged until termination of the algorithm. This group-size schedule proved to be the most efficient among the ones we have tested on this algorithm, both in terms of solution quality and execution time. From the above it is obvious that in the case of the Group-Update scheme we have decreased the number of steps per temperature value when the latter is high and the effective group size is relatively large. As the group size drops according to the schedule mentioned above, the number of trials per temperature increases in order for the algorithm to maintain its search ability.

Finally, as already mentioned in section 5.1, we experimented on two techniques for the task of finding the most appropriate group: (i) exhaustive search (using Gray coding) throughout the execution, and (ii) a combination of genetic search for the largest value of λ and exhaustive search for the remaining values of λ. In case (ii) we used the Simple Genetic Algorithm with string size 16, population size 50, number of generations 100, crossover probability 0.7 and mutation probability 0.06. The computation of the fitness function was based on the energy difference formula (equation (13)).

The results are summarized in Tables 1–3 and concern the quality of the provided solutions, the required computation time (in seconds) and the number of steps performed. The displayed results are average values obtained from 20 experiments for each problem instance. It is obvious that there is a significant improvement in the quality of the obtained solutions when the group-update approach is used, for all problem instances. It should be noted, that, in the case of the pure Boltzmann Machine, even when a much higher initial temperature was used, the obtained improvement of the solution was less than 2%. What is important in the new approach is the steep decrease of the energy to much lower values than in the conventional case. Figures

	Group Update Scheme			
n	*Avg. Cost*	*Best Cost*	*Avg. Time (sec)*	*Avg. Steps*
480	1999.5	2112	76.2	31002
800	2142.2	2204	216.3	54752
1600	2586.8	2702	910.6	109290

Table 3: Experiments for Group Update Scheme (genetic search for largest λ)

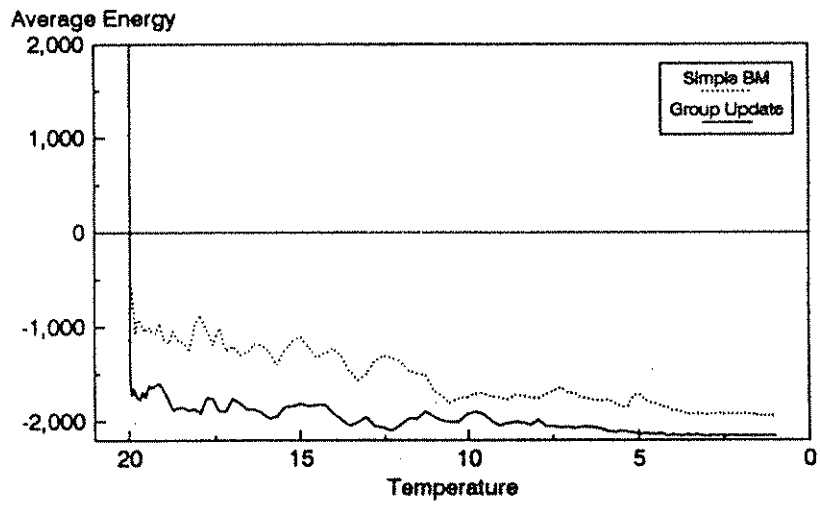

Figure 2: Comparison of energy evolution for $n = 800$

2–3 clearly depict this interesting feature. As far as genetic search is concerned, its contribution proved effective, since the corresponding scheme provided in slighly less time solutions that are very close to the ones obtained using pure exhaustive search.

In what concerns execution speed, the sequential execution of the group-update scheme requires much more time with respect to the conventional case, something not surprising, since each computational step in the new approach is much heavier than in the Boltzmann Machine. Nevertheless, a considerable benefit has been obtained through parallel implementation.

6.2 Parallel Implementation

From now on we will refer to the version of Group-Update scheme which uses only exhaustive search for determining the appropriate group. The proposed approach exhibits parallelism in the selection of the p units as well as in the computation of the p energy differences δE_i, since these quantities can be computed simultaneously in a straightforward manner. Another point, where parallelization can take place, is in the evaluation of the $2^\lambda - 1$ combinations of δE_i. Although this task looks sequential in principle, we can partition the set of combinations into subsets of consecutive elements in accordance with the Gray ordering, using some kind of preprocessing, and assign the task of finding the minimum of each subset of the sequence to one of the available processors. These partial minima can then be accumulated and the global minimum ΔE_G can be determined. However, we have not included this type of parallelism in our implementation, because for the MIS problem λ drops very fast and the computational cost of the appropriate group selection is very low, compared to

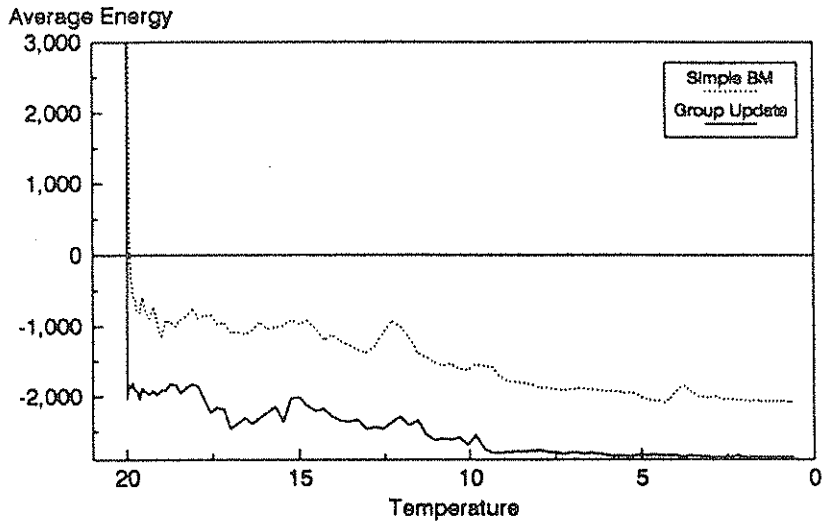

Figure 3: Comparison of energy evolution for $n = 1600$

	Avg. Speedup	
n	4 proc.	8 proc.
480	2.0	4.1
800	3.4	5.5
1600	3.9	6.6

Table 4: Speedup for shared memory implementation.

the cost of the computation of δE_i. In the parallel implementation we have considered the same number of experiments per problem instance, as well as the same parameter settings as in the sequential implementation.

The method was implemented on both shared memory and distributed memory parallel machines. The shared memory machine used was a Silicon Graphics Power Challenge with 14 R8000 processors, whereas the distributed memory machine was an Intel Paragon with 48 i860XP processors. Programming was performed using the C language with additional parallel calls for the coordination of the parallel processes.

For the evaluation of the shared memory implementation we considered execution on 4 and 8 processors. At each step, p units are selected in parallel from the p regions containing n/p units each, and the task of computing the corresponding energy differences δE_i is automatically partitioned into almost equivalent subtasks and automatically assigned to the available processors. Table 4 shows the average speedup obtained on the Power Challenge machine for the three instaces of the MIS problem.

On the distributed memory machine we followed a *manager-worker* parallelization scheme. The p regions, into which the network is partitioned, are uniformly assigned to the available *worker-processors*. At each step, every worker-processor randomly selects one unit from each region assigned to it and computes the corresponding δE_i. The manager-processor accumulates the δE_i, as well as the p selected nodes. Then the manager-processor selects the best ΔE_G, makes a trial and updates the output of the network. Then it informs the worker processors about the changes and continues until the stopping criterion is met. The results in Table 5 show good speedup obtained

	Avg. Speedup	
n	9 proc.	17 proc.
480	3.3	7.7
800	4.5	11.5
1600	6.8	12.1

Table 5: Speedup for distributed memory implementation.

on the Paragon machine, especially in large instances of our problem.

From Tables 4–5 we can see that in both machines we obtained very good speedup, especially for large problem instances. If we compare Tables 1, 2 and 4, we will see that for $n = 1600$ the execution time of parallel Group Update using 8 processors is even less than that of the Boltzmann Machine, while for the smaller sizes execution times are comparable.

7. CONLUSIONS

A new operation style for the Boltzmann Machine has been presented that is based on group updates. The method is general and can be applied to any problem for which the Boltzmann Machine is appropriate. The quality of the provided solutions is better compared to the conventional way of operation, despite the fact that the number of required trials and updates is significantly smaller. This quality improvement can be attributed to the fact that greater complexity is introduced in the process of generating the next state to be examined at each step. Therefore, the probability of successful trials is increased and it is easier to discover transitions that lead to a decrease in the network energy. Moreover, the method is characterized by a considerable degree of parallelism and a parallel implementation makes possible the reduction of the computation time required to deal with large scale optimization problems. Experimental results from both shared memory and distributed memory implementations indicate a very good performance of the group-update scheme in what concerns execution speedup.

REFERENCES

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

2. Aarts, E., & Korst, J. (1991). Boltzmann Machines as a Model for Parallel Annealing. Algorithmica, v. 6, pp. 437–465.

3. Ackley, D.H, Hinton, G. E., & Sejnowski, T. J. (1985). A Learning Algorithm for Boltzmann Machines. Cognitive Science, v. 9, pp. 147–165.

4. Alspector, J., & Allen, R. B. (1987). A Neuromorphic VLSI Learning System. In Advanced Research in VLSI, P. Losleben (ed.), pp. 313–346, MIT Press.

5. Azencott, R., (ed.), (1992) *Simulated Annealing: Parallelization Techniques.* John Wiley.

6. Barbosa, V.C., & Lima, P.M.V. (1990). On the Distributed Parallel Simulation of Hopfield's Neural Networks. Software - Practice and Experience, v. 20, pp. 967–983.

7. d'Acierno, A., & Vaccaro, R. (1992). Hopfield's Binary Networks: Simulations on Tree-Connected Transputer Networks. Proc. ICANN'92, v. 2, pp. 1409–1413, Brighton, UK.

8. Durand, M.D. (1989). Parallel Simulated Annealing: Accuracy vs. Speed in Placement. IEEE Design & Test of Computers, v. 6, pp. 8–34.

9. Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

10. Greening, D.R. (1990). Parallel Simulated Annealing Techniques. Physica D, v. 42, pp. 293–306.

11. Hajek, B. (1988). Cooling Schedules for Optimal Annealing. Mathematics of Operations Research, v. 13, pp. 311–329.

12. Hopfield, J.J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. Proc. of the National Academy of Sciences USA, v. 79, pp. 2554–2558.

13. Hopfield, J.J., & Tank, D.W. (1985). Neural Computation of Decisions in Optimization Problems. Biological Cybernetics, v. 52, pp. 141–152.

14. Kirkpatrick, S., Gellat, S., & Vecchi, M. (1983). Optimization by Simulated Annealing. Science, v. 220, pp. 671–680.

15. Kravitz, S., & Rutenbar, R. (1987). Placement by Simulated Annealing on a Multiprocessor. IEEE Trans. on Computer-Aided Design, v. 6, pp. 534–549.

16. Likas, A., & Stafylopatis, A. (1996). Group Updates and Multiscaling: An Efficient Neural Network Approach to Combinatorial Optimization. IEEE Transactions on Systems, Man, and Cybernetics, to appear.

17. Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag.

18. Romeo, F., & Sangiovanni-Vincentelli, A. (1991). A Theoretical Framework for Simulated Annealing. Algorithmica, v. 6, pp. 302–345.

19. Roussel-Ragot, P., & Dreyfus, G. (1990). A Problem Independent Parallel Implementation of Simulated Annealing: Models and Experiments. IEEE Trans. on Computer-Aided Design, v. 9, pp. 827–835.

20. Skubiszewski, M. (1992). An Exact Hardware Implementation of the Boltzmann Machine. Research Report, DEC Paris Research Laboratory.

21. Ticknor, A. J., & Barret, H. H. (1987). Optical implementations in Boltzmann Machines. Optical Engineering, v. 26, pp. 16–21.

22. Zissimopoulos, V., Paschos, V., & Pekergin, F. (1991). On the Approximation of NP-complete Problems by Using the Boltzmann Machine Method. The Cases of Some Covering and Packing Problems. IEEE Trans. on Computers, v. 40, pp. 1413–1419.