TNN 707

Cooperative Updating in the Hopfield Model

Tomo Munehisa, Masaki Kobayashi and Haruaki Yamazaki

Faculty of Engineering, Yamanashi University
Kofu, Yamanashi, 400-8511 Japan

Abstract

We propose a new method for updating units in the Hopfield model. With this method two or more units change at the same time, so as to become the lowest energy state among all possible states. Since this updating algorithm is based on the detailed balance equation, convergence to the Boltzmann distribution is guaranteed.

If our algorithm is applied to finding the minimum energy in constraint satisfaction and combinatorial optimization problems, then there is faster convergence than with the usual algorithm in the neural network. This is shown by experiments with the travelling salesman problem, the four color problem, the N-queen problem and the graph bipartitioning problem.

In constraint satisfaction problems, for which earlier neural networks are effective in some cases, our updating scheme works fine. Even though we still encounter the problem of ending up in local minima, our updating scheme has a great advantage compared with the usual updating scheme used in combinatorial optimization problems.

Also we discuss parallel computing using our updating algorithm.

Key words: Hopfield model, updating, detailed balance equation, constraint satisfaction problems, combinatorial optimization problems

1 Introduction

Since Hopfield and Tank [1],[2],[3] pointed out that neural networks can be applied to optimization, many researchers have studied this application, especially as applied to constraint satisfaction and combinatorial optimization problems. See [4],[5],[6],[7] for some reviews of these problems. In applications of the Hopfield model to problems with constraints, searches for the optimum solution do not work well since the state stays at a local energy minimum. The approaches of keeping some constraints satisfied in the neural network are the k-out-of-n rule, which defines the constraint that for a set of n neurons k of them are active and the others inactive, and its extensions [8],[9],[10],[11],[12]. But in these schemes units of the set are fixed, so that they can not keep more complicated constraints satisfied, such as those that arise in travelling salesman problems. Also annealing has not been addressed in these approaches. In this paper we would like to extend the updating algorithm of the k-out-of-n rule and formulate it in the annealing algorithm.

One of the most important properties of the Hopfield model is asynchronous updating in which only one unit updates while other units remain unchanged during the updating[2],[3]. Our proposal is that more than one unit is updated simultaneously in order to reduce the energy. We call this algorithm cooperative updating, since these units try to reduce the energy cooperatively. The cooperative updating algorithm gives us faster convergence than the usual algorithm in the neural network. This is shown (below) experimentally by applying cooperative updating in constraint satisfaction and combinatorial optimization problems.

The updating algorithm on multiple units has been suggested by Goles[13], but it does not use the annealing algorithm thus resulting in too severe a convergence condition to apply to many optimization problems. Also the cluster algorithm has been well developed in statistical physics[14], but this makes use of the fact that numbers of weights per unit are few so that it can not apply to optimization problems.

In the next section we make a detailed description of our algorithm with theoretical justifications using the annealing algorithm. Here we would like to stress that the detailed balance equation plays an essential role in our algorithm. In sec-

tion 3-6 we apply the cooperative updating algorithm to the travelling salesman problem (TSP), the four color problem (4CP), the N-queen problem (NQP) and the graph bipartitioning problem (GBP). The TSP and the GBP are typical examples of combinatorial optimization problems, while the 4CP and the NQP are examples of constraint satisfaction problems. In section 7 we discuss parallel computation using our algorithm. A final section is devoted to a summary.

2 Cooperative Updating

We use the terminology of statistical physics to discuss the stochastic Hopfield model, or the Boltzmann machine. Discussions of the (deterministic) Hopfield model are not made because it is the limiting model as the temperature T goes to zero.

One unit $i(i = 1, 2, .., N)$ has its value S_i, that is 0 or 1. The energy H is given by symmetric weights $W_{ij} = W_{ji}$ between units i,j and the threshold J_i of unit i.

$$H(S_1, \cdots, S_N) = -\frac{1}{2} \sum_{i,j=1}^{N} W_{ij} S_i S_j + \sum_{i=1}^{N} J_i S_i. \tag{1}$$

At T the probability of finding a state $\{S_1, \cdots, S_N\}$ is given by the Boltzmann distribution.

$$P(S_1, \cdots, S_N) = \exp(-\frac{1}{T} H(S_1, \cdots, S_N))/Z, \tag{2}$$

$$Z = \sum_{\{S_1, \cdots, S_N\}} \exp(-\frac{1}{T} H(S_1, \cdots, S_N)). \tag{3}$$

In simulated annealing many updating steps are repeated in order that the number of produced states is proportional to the Boltzmann distribution. An updating process from $\{S_1, \cdots, S_N\}$ to $\{S'_1, \cdots, S'_N\}$ is defined as trying to change values of units, according to the transition probability $P_t(S_1, \cdots, S_N; S'_1, \cdots, S'_N)$. If the transition probability satisfies the detailed balance equation,

$$P(S_1, \cdots, S_N) P_t(S_1, \cdots, S_N; S'_1, \cdots, S'_N) = P(S'_1, \cdots, S'_N) P_t(S'_1, \cdots, S'_N; S_1, \cdots, S_N), \tag{4}$$

and an infinite number of updatings is performed, then one can prove that the number of produced states is proportional to the Boltzmann distribution[5],[15],

assuming that the probability of a transition from a given state to any other state is not zero, i.e. one state can change into any state by repeated updating. The updating process having this non-zero probability is called irreducible or ergodic. For a detailed overview see discussions in section 3 of [5] and [15]. Many applications are ergodic, if the updating rule does not have some symmetry. This proof is not difficult since one can show an explicit procedure for a transition to a specific state from any other state. In the next section we present the proof for the TSP.

The detailed balance equation is quite general, so various kinds of transition probabilities are possible. In the case of updating one unit only, the detailed balance equation is written as

$$P(S_i)P_t(S_i;S_i') = P(S_i')P_t(S_i';S_i). \tag{5}$$

Here all variables $S_j(j \neq i)$ can be omitted in the above expression, since the remaining units are unchanged. The Boltzmann machine uses the sigmoid function, corresponding to the heat bath method in statistical physics[15], where the transition probability into $S_i' = 1$ is given by

$$\begin{aligned} P_t(S_i;S_i'=1) &= \frac{P(S_i=1)}{P(S_i=0)+P(S_i=1)} \\ &= \frac{\exp(-H(S_i=1)/T)}{\exp(-H(S_i=0)/T)+exp(-H(S_i=1)/T)} \\ &= \frac{1}{1+\exp\{(H(S_i=1)-H(S_i=0))/T\}}. \end{aligned} \tag{6}$$

Here the transition probability is given by the ratio of the exponential of the energy for the $S=1$ state, to the sum of the exponentials for the $S=0$ and $S=1$ states.

Also the Metropolis method is often used in simulated annealing[5], and in statistical physics[15]. Its transition probability is given by

$$P_t(S_i;S_i') = min(1,\exp\{-(H(S_i')-H(S_i))/T\}). \tag{7}$$

This implies that if $H(S_i') \leq H(S_i)$,

$$P_t(S_i;S_i') = 1,$$

while if $H(S_i') > H(S_i)$,

$$P_t(S_i;S_i') = \exp\{-(H(S_i')-H(S_i))/T\}.$$

One can justify that two or more units can be updated simultaneously, if the transition probability for their updating satisfies the detailed balance equation. Hereafter we call this the cooperative updating algorithm, as stated in the Introduction. For updating two units from S_i, S_j into S_i', S_j', the detailed balance equation becomes

$$P(S_i, S_j)P_t(S_i, S_j; S_i', S_j') = P(S_i', S_j')P_t(S_i', S_j'; S_i, S_j).$$

Applying the heat bath method, the transition probability is

$$P_t(S_i, S_j; S_i', S_j') = \frac{P(S_i', S_j')}{P(S_i = 1, S_j = 1) + P(S_i = 1, S_j = 0) + P(S_i = 0, S_j = 1) + P(S_i = 0, S_j = 0)},$$

which clearly satisfies the equation. If one relies on the Metropolis method, the transition probability is given by

$$P_t(S_i, S_j; S_i', S_j') = min(1, P(S_i', S_j')/P(S_i, S_j)).$$

In this method we need to calculate energies for two states, while the heat bath method requires the energies for four states.

Next we discuss what pair or group can be updated in the cooperative updating algorithm. In principle, any pair or group is possible. One restriction on the selection of pairs or groups is that one must not use energy to select them. The reason for this restriction is that if the selection probability depends on the state, this dependence invalidates the detailed balance equation. For faster convergence of the neural network, however, there are good pairs or groups, and bad ones. In the case where the weights among some units have large absolute values, the selected group should consist of these units. In order to understand this selection, we separate the energy of two units i, j from others.

$$H = -\frac{1}{2}\sum_{k,l\neq i,j} W_{k,l}S_kS_l + \sum_{k,\neq i,j} J_kS_k - \sum_{k\neq i,j} W_{k,i}S_kS_i - \sum_{k\neq i,j} W_{k,j}S_kS_j \\ -W_{ij}S_iS_j + J_iS_i + J_jS_j. \qquad (8)$$

If the absolute values of J_i, J_j and W_{ij} are much larger than other weights, i.e.

$$\mid W_{ij} \mid, \mid J_i \mid, \mid J_j \mid \gg \mid W_{ik} \mid, \mid W_{jk} \mid, \quad k \neq i, j,$$

only specific values of units i, j are apt to be selected, depending on W_{ij}, J_i and J_j. For example, values of (S_i, S_j) are more likely to become $(1,1)$ or $(0,0)$ than $(0,1)$ and $(1,0)$ for $J_i = J_j = W_{ij}/2 \gg 1$.

Based on the above discussion, we suggest the following updating scheme for pairs. One unit is selected randomly or sequentially, and we choose another unit whose weight with the former has the largest absolute value. If there are several units with equally large weights, one can choose one among them randomly. Then we apply the cooperative updating algorithm to these units. Also we can extend this scheme to updating more than two units.

The above discussion enables us to understand that the cooperative updating algorithm is effective when one applies the Hopfield model to constraint satisfaction and combinatorial optimization problems. In these problems the constraint is expressed by the cost function. In the usual algorithm this cost function has a finite energy, even if the constraint is not satisfied. In our algorithm it is possible to update units keeping the constraint satisfied by a suitable choice of units. In other words, the cost function becomes infinitely large if the constraint is not satisfied. As a result, the search space is reduced drastically. For example, in the N_c-*city* TSP $2^{N_c^2}$ state space reduces to $N_c!$ one. In the following sections our experiments show that the cooperative updating algorithm is effective in applications to the TSP, the 4CP, the NQP and the BGP.

3 Travelling Salesman Problem

In this section we apply the Hopfield model to the TSP[1], [16], [17] using our algorithm. In the neural network, the problem is mapped to searching for the minimum energy state that corresponds to the minimum-length closed tour. In the original approach, a value S_{ai} of unit ai is 1 if and only if city a is the ith visit on the tour, otherwise it is zero. So this problem has constraints.

$$\sum_{a=1}^{N_c} S_{ai} = 1, \quad \sum_{i=1}^{N_c} S_{ai} = 1. \tag{9}$$

Here N_c is the number of cities. It is worthwhile to note that by updating one unit, we have to break these constraints if the old state has already satisfied them. The cooperative updating algorithm enables us to keep the constraints satisfied in the TSP, if four units are updated at the same time. Because when $S_{ai} = 1$ is changed by $S'_{ai} = 0$, the constraints require units $bi(b \neq a)$ with $S'_{bi} = 1$ and $aj(j \neq i)$ with $S'_{aj} = 1$ and therefore a unit bj has to change its value into $S'_{bj} = 0$(See Fig.1). As a result, S_{ai}, S_{bi}, S_{aj} and S_{bj} satisfy

$$S_{ai} + S_{bi} = 1, \quad S_{ai} + S_{aj} = 1, \quad S_{aj} + S_{bj} = 1, \quad S_{bi} + S_{bj} = 1.$$

Possible configurations of $\{S_{ai}, S_{bi}, S_{aj}, S_{bj}\}$ are $(1,0,0,1)$ and $(0,1,1,0)$. Calculating the energy difference ΔH between these configurations, updating can be performed easily in the stochastic approach.

Here we prove the ergodicity in this updating rule by giving an explicit procedure for the transition from any state to the specific state that has $S_{ii} = 1$ and $S_{ij} = 0$ for $i \neq j$, as in the example shown in Fig.1. First we change $S_{11} = 0, S_{12} = 1, S_{21} = 1$ and $S_{22} = 0$ to $S_{11} = 1, S_{12} = 0, S_{21} = 0$ and $S_{22} = 1$. Second we change $S_{33} = 0, S_{34} = 1, S_{53} = 1$ and $S_{54} = 0$ to $S_{33} = 1, S_{34} = 0, S_{53} = 0$ and $S_{54} = 1$. Next we change $S_{44} = 0, S_{45} = 1, S_{54} = 1$ and $S_{55} = 0$ to $S_{44} = 1, S_{45} = 0, S_{54} = 0$ and $S_{55} = 1$, which is just the specific state. These changes have finite transition probabilities so that the ergodicity is proved. This procedure can be extended to any size problem.

First we confirm that the number of states generated by this updating algorithm agrees with the Boltzmann distribution, i.e.

$$P_B(S_{a,1}, S_{a,2}, \cdots, S_{N_c,N_c}) = \frac{\exp(-H(S_{a,1}, S_{a,2}, \cdots, S_{N_c,N_c})/T)}{Z},$$

$$Z = \sum_{\{S_{a,1}, S_{a,2}, \cdots, S_{N_c,N_c}\}} \exp(-H(S_{a,1}, S_{a,2}, \cdots, S_{N_c,N_c})/T).$$

For this purpose we have to be content with a quite small number of units. Here five cities, which corresponds to 25 units, are employed.

We adopt the Metropolis method so that the transition probability for changing $(1,0,0,1)$ into $(0,1,1,0)$ of $\{S_{ai}, S_{bi}, S_{aj}, S_{bj}\}$ is

$$P_t((S_{ai} = 1, S_{bi} = 0, S_{aj} = 0, S_{bj} = 1); (S_{ai} = 0, S_{bi} = 1, S_{aj} = 1, S_{bj} = 0))$$

$$= min(1, e^{-\Delta H/T}),$$

$$\Delta H = H(S_{ai}=0, S_{bi}=1, S_{aj}=1, S_{bj}=0) - H(S_{ai}=1, S_{bi}=0, S_{aj}=0, S_{bj}=1).$$

Figure 2 shows numbers of generated states and the Boltzmann distribution, which are multiplied by the total number 10^5. The solid line in the figure shows analytical calculations of the Boltzmann distribution. Integer numbers on the horizontal line denote labels of the state. Here the label i represents the $i-$th state if every state is ordered according to the energy. In the five city TSP, ten states have the same energy due to the cyclic and inverse symmetry, which can be seen by noting plateaus of the solid line. States that do not satisfy the constraint are excluded from the figure. At low T the Boltzmann distribution becomes sharp, as seen in Fig.2(b). Numbers of the generated states in the experiments, which are represented by histograms, agree with the solid line though there are some stochastic fluctuations. Agreement between them justifies our discussion of the cooperative updating algorithm.

Next we study problem instances given by the TSP database[18], TSP225 and RAT575. The problem that we encounter first is that there are many candidates (of states) for updating, some of which are difficult to change. Here we adopt a method of restricting the number of candidate states. The method of restriction is that a distance d_{ab}^{nn} used in the search between city a and its neighbor city b, where a number of these neighbor cities is fixed by N_{near}, equal to a distance d_{ab} given in the problem, while those between other cities and city a are set to be quite large, i.e., using large D,

$$d_{ab}^{nn} = d_{ab} \quad for \quad b \quad \in \quad neighborhood \quad of \quad the \quad city \; a,$$

$$d_{ab}^{nn} = D \quad for \quad b \quad \notin \quad neighborhood \quad of \quad the \quad city \; a.$$

If $d_{ab}^{nn} \neq d_{ba}^{nn}$, we use $d_{ab}^{nn} = d_{ba}^{nn} = d_{ab}$, so that city a is connected to one of the near N_{near} cities. If N_{near} is large, the frequency of no change increases, while if it is small, there is the danger that the optimized connection is out of our search range. Although the best value for N_{near} is difficult to find, we estimate that suitable values are several times the lowest value. This lowest value can be found by decreasing N_{near} until we can not obtain a close path for the case where $d_{ab}^{nn} = 0$. The specific

value of N_{near} is not particularly important. Here we assume N_{near} is in the range of $10-40$ and $D=10^4$. Note that the calculation cost does not depend on N_{near} because we randomly select one city b among the N_{near} cities.

The updating scheme suggested in the previous paragraph corresponds to exchanges of two cities in the order. Experiments using this method were discouraging because the observed length was larger than the optimal length by more than 10%, which is not good compared with the few % obtained by the standard heuristic method such as the $k-opt$ method.

In order to improve the quality, we changed assignment of neurons so that if a city b is visited next to a city a, the neuron value $S_{ab}=S_{ba}$ is 1 , and 0 otherwise, i.e. if a path between cities a and b is used, $S_{ab}=1$, while if not, $S_{ab}=0$. A constraint that a city connects with two cities is expressed by

$$\sum_{b=1}^{N_c} S_{ab}=2. \tag{10}$$

Here we assume that $S_{aa}=0$. Further we need a constraint that a tour has to be closed, which is guaranteed by

$$\sum_{a_2\neq 1}\sum_{a_3\neq 1}\cdots\sum_{a_{N_c-1}\neq a_2,1} S_{1a_2}S_{a_2a_3}S_{a_2a_3}\cdots S_{a_{N_c-1}1}=1. \tag{11}$$

In the present algorithm the constraint does not need to be quadratic, since the constraint is not included as part of the energy.

Before annealing we search a state in which every city connects to one of its neighboring cities. This search is performed by setting $d_{ab}^{nn}=0$. This is an initial state for annealing.

We describe the experiments in detail. One cycle is defined as updating every unit sequentially at a time t. The temperature T decreases as

$$T=\frac{1}{\beta_0\ln(t+1)}.$$

Annealing ends at 10^4 cycles and we denote this annealing as one trial[19]. Then we record the smallest length, l_{sm}, in a trial, not the tour length at the end. Further, we make 100 trials to get the statistical average. For each trial we randomized the state by 100 cycles with $T=10^4$. The reason for randomizing states by the non-zero

value of $1/T$ is that we want to maintain the condition that every city connects to its near cities. In Tables 1 and 2 we collect experimental results on errors for problem instances, TSP225 and RAT575. Here errors are defined as $(l_{sm} - l_{opt})/l_{opt}$, where l_{opt} is the length of the optimized tour. The best results among the experiments for various N_{near} and β_0 are 1.83% (0.13%) for the average(smallest) value for $N_{near} = 30(30)$ and $\beta_0 = 0.04(0.04)$ in TSP225 while for RAT575, they are 2.10% (1.14%) for the average(smallest) value for $N_{near} = 20(25)$ and $\beta_0 = 0.06(0.06)$. These results are encouraging because for RAT575 the study of the $2-opt$ method produced an average value of 6.19% and smallest value of 3.23%[20],[21]. However the values are not satisfying because these researchers report that the average value(the smallest) is 0.69% (0.04%) for the $9-opt$ method and that the genetic algorithm can find the optimized tour.

Next we discuss the effect of restricting the number of neighborhood cities. Tables 1 and 2 show that suitable choices of N_{near} improve the error, but the dependence on N_{near} is weak. Also comparing the results of TSP225 and RAT575, the suitable N_{near} may be irrelevant to the size of the instance. Also we can say that the best value of the initial temperature does not depend on the size. It may be determined by the average distance between connected cities in the tour, which is 17.3 (11.7) for TSP225 (RAT575), because we obtain the same value 6.4 for both instances, after dividing them by the temperature of the final cycle.

Finally we report the experiments without annealing, i.e. those for a fixed temperature. For $1/T = 0.368$, which was the temperature of the final cycle in annealing, we have errors of 2.78%(0.95%) for the average(smallest) values with $N_{near} = 30$ in TSP225. Comparing these values to 1.83%(0.13%) with annealing, we can find some effect due to annealing[23]. While for quite low temperature, $T = 0.2(1/T = 5)$, these value are 12.5%(8.04%) for $N_{near} = 30$, which implies that the deterministic Hopfield model does not work well in the TSP.

Since our method in the TSP is based on the $2-opt$ method, it is possible to update our algorithm and extend it to the higher $k-opt$ methods and the LK method[22]. This extension is an important subject for further study.

4 Four Coloring Problem

The four coloring problem (4CP) is the problem of assigning four colors to a set of areas so that adjacent areas do not have the same color. In this problem one can easily see whether the neural network finds a solution or not, because the energy of a solution is clearly defined. Here we make comparisons of results by cooperative updating with those by the usual updating. In order for the neural network to solve the 4CP, we define that a value S_{ia} of unit $ia(i = 1, .., N_A, a = 1, 2, 3, 4)$ is 1 for the case when an area i is assigned to a color a and is 0 for other cases. Here N_A is the number of areas. For two adjacent areas i, j a weight v_{ij} is -1, while v_{ij} is 0 for other cases. The energy H_0 is given by

$$H_0 = -\sum_{i<j,a,b} v_{ij}\delta_{a,b}S_{ia}S_{jb}. \tag{12}$$

To assign one color to one area, we define a constraint energy.

$$H_\lambda = \lambda\sum_i(\sum_a S_{ia} - 1)^2. \tag{13}$$

The total energy $H = H_0 + H_\lambda$ is zero when the correct assignment is made. For the usual updating algorithm λ is 1, while for cooperative updating it is infinitely large so that the constraint is always satisfied. This constraint imposes only one unit ia to be 1 among units $i1, i2, i3, i4$ and the others, S_{ib}, to be zero. The updating candidates are ia with its value $S_{ia} = 1$ and a randomly selected unit ib whose values $S_{ib} = 0$. These units ia and ib are a pair for cooperative updating.

Experiments were made for $N_A = 50$. For 100 trials we plot the number of trials as a function of the cycle number where the trial found a solution (Fig.3). In the cooperative updating algorithm, all trials reach a solution by cycle 250 and the peak in the distribution shows 18 trials at 60 cycles. In contrast, in the usual updating algorithm, 67 trials are needed for over 1000 cycles. Also 7 trails required over 2000 cycles. These are dropped from the figure. Noting that faster convergence corresponds to fewer cycles, the experimental result shows that the cooperative updating algorithm enables us to obtain faster convergence, as expected. Finally one should notice that the CPU time for calculations by our updating is less than

one one-hundredth that for the usual one. This is because the number of updating units and sizes of the array decrease.

5 N-Queen Problem

Next we apply the cooperative updating algorithm to the N-queen problem(NQP), which is frequently used as a bench mark test, in order to compare our method to other neural network techniques. The NQP is to place N queen figures on a $N \times N$ chess board so that there is only one queen on each horizontal, vertical and diagonal line. Applying the Hopfield model to this problem, a value S_{ij} of unit i, j is 1 if a queen is at a point (i,j) on the board and 0 otherwise. We impose the following constraints to satisfy the horizontal and vertical conditions.

$$\sum_{i=1}^{N} S_{ij} = 1, \quad \sum_{j=1}^{N} S_{ij} = 1. \tag{14}$$

We define weights so that the energy increases with the number of queens on oblique lines, that is, if there are two or more.

$$H = -\frac{1}{2} \sum_{i=1} \sum_{j=1} \sum_{i'=1} \sum_{j'=1} J_{ij,i'j'} S_{ij} S_{i'j'}. \tag{15}$$

Here

$$J_{ij,i'j'} = -1 \quad if \quad i - j = i' - j' or \quad i + j = i' + j', other cases \quad 0.$$

The constraints for the NQP are the same as those for the TSP, so that we can maintain satisfaction of the constraints by updating four units.

For $N = 1000$ we try to search solutions by various cooling schedules. First we make a trial for the case where the temperature decreases as $T_0/\ln(t+1)$. For the initial temperature $T_0 = 1$ we get a solution in 1095 cycles. Second we study the case where the temperature decreases as T_0/t. In this case we get a solution in 149 cycles for the initial temperature $T_0 = 10$, which suggests that one can obtain solutions by a rapid decrease of the temperature. In other words, the updated states are not trapped local minima. Based on this result we made a trial with a very low temperature, which is actually a search by the deterministic Hopfield model. For a constant temperature $T = 0.1$ the energy decreases monotonically as seen Fig.4.

To confirm that this result is not accidental, we made 100 trials choosing random numbers. For every trial the energy decreased monotonically and solutions were obtained. The computational time was reduced drastically due to the constant and low temperature, so that calculations for larger size cases are possible. We have succeeded in finding solutions for $N = 2 \times 10^6$, where the CPU time is 1555 seconds in a Pentium III 500 MHz machine. In fact, the size N is limited by the available computer memory.

We can say that the NQP has become an easy problem using the cooperative updating algorithm, since any local minimum is always the global one. This fact may be accidental, but it suggests that we should study other satisfaction problems.

On the NQP we make comparisons of our method with others. Schaller gives an overview of constraint satisfaction problems when using the neural network approach [4]. In this article he reports the maximum sizes in the experiments by various methods. They are 3000 for the strictly digital neural network[11], 1000 for the Boltzmann machine[24], 200 for the dynamic barrier neural network[10], 100 for the neural computing network[6] and 6001 for the guarded discrete stochastic net [12],[25]. Our maximum size of 2×10^6 is much larger than these. In the same review[4], he reported that the non-neural network approaches succeeded in experiments with large maximum sizes of 3×10^5[26],[27],[28], 10^6[29]. This fact may indicate that the neural network approach has the ability comparable with other heuristic approaches[30].

6 Graph Bipartitioning Problem

In this section, we apply our algorithm to the graph bipartitioning problem[2],[32],[33], which is another example of a combinatorial optimization problem, and NP-complete [33]. The problem is specified by a graph G which consists of a set $V = \{v_1, v_2, \cdots, v_N\}$ of N vertices, with N even, and a set $E = \{(v_i, v_j)\}$ of edges which connect pairs of the vertices. The task is to divide the vertices into two sets V_1 and V_2 of equal size in such a way as to minimize the number of edges between the sets.

In the neural network we define a unit variable S_i as 1 if v_i belongs to V_1, and as

-1 if it belongs to V_2. Also we assign a weight W_{ij} as 1 if (v_i, v_j) belongs to E, and 0 otherwise. Then the energy function is given by

$$H = -\sum_{i<j} W_{ij} S_i S_j + \lambda (\sum S_i)^2,$$

where λ is a large positive number. If the size of V_1 is equal to V_2, the second term vanishes.

In the cooperative updating algorithm, the penalty constant λ is infinitely large. Therefore the initial state must satisfy the constraint of equal size sets, which is achieved by setting $S_i = 1$ for $i = 1, \cdots, N/2$, and $S_i = -1$ for $i = N/2 + 1 \cdots, N$. For updating units, we randomly select two units i and j so that $S_i \neq S_j$, and update these units by using the algorithm.

In the experiments for this problem we use the graphic of Fig.5. This graphic consists of two groups. Each group has 500 vertices and 5000 edges given randomly. And no edges connect any vertex of one group with any one of the other. Therefore in this problem instance the minimum number of edges between V_1 and V_2 is zero so that the minimum energy is -10000. The experiments are made at constant temperature $T = 0.1$. Also one cycle is defined as 1000 updatings.

Figure 6 shows the average and the lowest energy at each cycle, which are obtained by running 100 trials. The energy decreases rapidly to -9800 in 50 cycles and then decreases slowly to the minimum energy. By cycle number 300, we find the optimized solution for 47 of 100 trials. The shortest time to find the solution was 120 cycles among the 100 trials, which is seen from the fact that the lowest energy is -10000 at this cycle.

The experiments using the usual algorithm, where a parameter A is a finite value 100, were made with several values of constant temperature from 0.01 to 100. In these experiments we find that the network quickly goes to the local minimum, whose energy is much larger than the minimum of -10000, for any temperature. These results in the cooperative and usual updating algorithms show us clearly that if one uses our algorithm the neural networks can avoid getting stuck in local minima induced by constraint conditions.

7 Parallel Computing

The important feature of the neural network is the massive parallel computation. Here we discuss parallel computing by the cooperative updating algorithm in constraint satisfaction problems. This algorithm enables the state to satisfy certain constraints, called the strict constraints. The problems are divided into two categories. In some problems member of units that satisfy the strict constraints do not change during updating. The 4CP is an example, because only one of four units corresponding to four colors and the same area has the value 1 and the others have zero value. For these problems some network models have already been proposed. See [6], [8], [10], [11]. Our algorithm can be implemented easily into parallel computing.

In another type of problems, we have to change the set of units that satisfy the strict constraints during updating. Examples are the TSP and the NQP. On the NQP we show methods of parallel computing. In the NQP only the i-th unit in one column i_c has value 1. In another column j_c only the j-th unit has value 1. In the cooperative updating algorithm, they change to 0 while values of the j-th unit in column i_c and the i-th unit in column j_c change from 0 to 1 if the energy decreases. Therefore units in these columns, i_c and j_c, have to be changed. The scheme we suggest here is the following. First we define the group that consists of two units with their values 1, which are the $i-$th in the i_c column and $j-$th in the j_c column, and two units with their value 0, which are the $j-$th in the i_c column and the $i-$th in the j_c column. Then we select these groups randomly. One could not make a completely simultaneous and parallel selection of the groups because two or more groups can contain the same unit. In order to avoid this problem the selection is made sequentially over a short time period. As a result earlier selected units can be marked and only unmarked units are candidates for subsequent selections. After the selections, units in each group are updated independently so that the parallel computation can be implemented easily.

To simulate the parallel computation, we repeat the following two steps of the selection and updating. Here the maximum number of selected groups is fixed.

- step(1) Randomly select two units with values 1 and two units with zero values

that correspond to the former. Then mark the selected units. If there are previously marked units, abandon this selection.

- step(2) Apply the cooperative updating algorithm to units in each group.

Here we show the experiments on the parallel computation for $N = 1000$[34]. Since we fix the the maximum number for the selected pairs, the numbers of pairs actually selected varies during the simulations. First we measure the average number of these pairs as a function of the maximum number. This is plotted in Fig.7. The average number decreases from 250[35], as the the maximum number decreases, while the efficiency, which is the ratio of the average number to the the maximum number, rises from 0.5 to 0.9 at the maximum number 50.

Figure 8 shows the energy in a trial during updating. Notice that the energy does not decrease monotonically, but oscillates for a long time. The reason for this oscillation is that each unit is updated based on the old knowledge of the states so that decreases of the energy in each unit do not imply a decrease in the state of the neural network. As a result, the number of cycles until finding solutions is greater than in sequential computing.

We examine the average number of cycles until finding solutions by making 100 trials for various cases of the maximum number of the section. Figure 9 show that there exists the optimal maximum number 80. In parallel computing, the number of cycles is proportional to the computing time, so that 80 pairs corresponds to the best calculation time for parallel computing.

In parallel computing the oscillation is theoretically an important problem. Also communication among the units is a subject for further study in implementing the cooperative updating algorithm in hardware.

8 Summary

Summarizing this paper, we have suggested the cooperative updating algorithm for fast convergence. We presented theoretical arguments based on the detailed balance equation to justify our updating algorithm in terms of statistical physics. The

detailed balance equation allows us to use various kinds of transition probability. Therefore, updating two or more units at the same time becomes possible if they satisfy this equation.

Also we demonstrated that the cooperative updating algorithm is effective for constraints satisfaction and combinatorial optimization problems by applying our algorithm to the TSP, the 4CP, the NQP and the BGP. In the TSP we showed that our updating algorithm can give the Boltzmann distribution for the small size case. Using instances of TSP225 and RAT575, we demonstrated that the neural network can give us adequate results in combinational optimization problems such as TSP. In the 4CP we made comparisons between the results of the usual updating methods and ours to show that the cooperative updating algorithm is powerful on fast convergence. The NQP was used for a bench mark test for comparisons with other neuron models. Finding solutions for $N = 2 \times 10^6$ has greatly encouraged us, though other heuristic approaches are more effective[26],[27],[28], [29],[30]. Also we succeeded in finding the optimum solution in the BGP, which is another example of a combinatorial optimization problem. In addition, we suggested a scheme for parallel computing using our updating algorithm.

The cooperative updating scheme has a great advantage compared with the usual updating scheme, but using this scheme we can not solve the problem of ending up in local minima for combinatorial optimization problems. Concerning this difficulty the approach by the chaotic dynamics[36],[37] is worth noticing, so that further study is required to consider the possibility of combining the cooperative updating algorithm with this approach.

We extended the algorithm of quite local updating to a slightly global one, i.e. only one variable updating to many variable updating. But we were not able to include the global updating scheme, which is seen in the genetic algorithm[20],[21]. We understand that further study of global updating is required to find the global minimum in constrained optimization problems.

Finally we expect that the cooperative updating algorithm can reduce the learning time, if it is applied to learning in the Boltzmann machine[2], where time is impractically large. This is because the sharp probability distributions required in

many applications are similar to those in constraint satisfaction problems. This is another subject that we will pursue in the future.

Acknowledgments

We would like to thank the referees who made suggestions and comments on the original version.

References

[1] J.J.Hopfield and D.W.Tank,"Neural computation of decisions in optimization problems," *Biological Cybernetics,* vol.52, pp.141-152,1985.

[2] J.Hertz, A.Krogh and R.Palmer, *Introduction to the Theory of Neural Computation*, Reading:Addison Wesley, 1991.

[3] P. Peretto, *An Introduction to the Modeling of Neural Networks,* Cambridge: Cambridge University Press 1992.

[4] H.N.Schaller, "Constraint satisfaction problems," *Optimization Techniques*, edited by C.T.Leondes, San Diego:Academic Press, pp.209-248, 1998.

[5] E.Aarts and J.Korst, *Simulated Annealing and Boltzmann Machines*, Chichester:John Wiley & Sons, 1989.

[6] Y.Takefuji, *Neural Network Parallel Computing,* Boston:Kluwer Academic, 1992.

[7] K.Smith, M.Palaniswami and M.Krishnamoorthy, " Neural techniques for combinatorial optimization with Application," *IEEE Trans. Neural Networks*, vol.9, pp.1301-1318, 1998.

[8] Y.Takefuji and C.K.Lee, "A parallel algorithm for tiling problems," *Neural Networks,* vol.1, pp.143-145,1990.

[9] E.W.Page and G.A.Tagliarini, " Algorithm development for neural networks," In *Proc. IEEE SPIE:High Speed Computing,* vol.880,pp.11-19,1988.

[10] H.N.Schaller, " Design of neurocomputer architectures for large-scale constraint satisfaction problems," *Neurocomputing*, vol.8, pp.315-339, 1995.

[11] T.Nakagawa and H.Kitagawa, "SDNN:an $O(1)$ parallel processing with strictly digital neural networks for combinatorial optimization," *Artificial Neural Networks*, edited by T.Kohohen, K.Makisara, O.Simula and J.Kangas, Amsterdam:Elsevier, pp.1181-1184, 1991.

[12] M.D.Johnston and H.-M.Adorf, "Learning in stochastic neural networks for constraint satisfaction problems," In *Proc. of the NASA Conference on Space Telerobotics,*,1988.

[13] E.Goles, " Fixed point behavior of threshold functions on a finite set," *SIAM J. Alg. Disc. Meth.*, vol.3, pp.529-531, 1982.

[14] R.H.Swendsen, J.-S.Wang and A.M. Ferrernberg " New Monte Carlo methods for improved efficiency of computer simulations in statistical mechanics," *The Monte Carlo Method in Condensed Matter Physics*, edited by K.Binder, Berlin:Springer, pp.75-91, 1991.

[15] W.Krauth, "Introduction to Monte Carlo algorithms," *Advances in Computer Simulation* , edited by J.Kertesz and I.Kondor, Berlin:Springer, pp.1-35, 1998.

[16] E.B.Baum, " Towards practical neural computation for combinatorial optimization problems," *Neural Networks for Computing*, edited by J.S.Denker, New York:American Institute of Physics, pp.53-58, 1986.

[17] R.Lister, " Segment reversal and the TSP," in *Proc. Intl. Joint Conference on Neural Networks*, Hillsdale:Erlbaum, vol.1, pp.424-427, 1990.

[18] TSPLIB95,
http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[19] The CPU time for one trial is 71(537) seconds for TSP225(RAT575) in machines with a Pentium III 500MHz.

[20] Y.Nagata and S.Kobayashi, " Edge Assembly Crossover: A High-power Genetic Algorithm for the Travelling Salesman Problem," *Proc. of the 7th International Conference on Genetic Algorithms*, pp.450-457, 1997.

[21] Y.Nagata and S.Kobayashi, " The proposal and evaluation of a crossover for travelling salesman problems," *J. of Japanese Society for Artificial Intelligence*, pp.848-859, 1997(in Japanese).

[22] S.Lin and B.W.Kernighan, " An efficient heuristic algorithm for the travelling salesman problem," *Operations Research* , vol.21, pp.498-516, 1973.

[23] More sophisticated scheme on the cooling schedule may improve results [5].

[24] M.Kajiura et al, "Solving large scale puzzles with neural networks," In *Proc. IEEE TAI'89*, pp.562-569, 1989.

[25] J.Lorenz, "Vergleich verschiedener Neuronenmodelle zur Lösung kombinatorischer Suchproblemes," Diploma Thesis, Technical University of Munich, 1993.

[26] R.Sosic and J.Gu, "Fast search algorithms for the N-queens problem," *IEEE Trans. Systems Man Cybernet*, vol.21, pp.1572-1576, 1991.

[27] ____ , "A polynomial time algorithm for the n-queens problems," *IEEE SIGART Bull.*, vol.1, pp.7-11, 1990.

[28] ____ , "3,000,000 Queens in less than one minute," *SIGART Bull.*, vol.2, pp.22-24, 1991.

[29] S.Minton et al, "Minimizing conflicts: a heuristic repair method for constraint satisfaction problems," *Artificial Intelligence*, vol.58, pp.161-205, 1992.

[30] Some people cast doubt on tests by the NQP since other heuristic approaches have given better results[26],[27],[28],[29] as well as the closed form solution[31] is found.

[31] B.Falkowski and L.Schmitz, " A note on the queens' problem," *Inform. Processing Lett.* , vol.23, pp.39-46, 1986.

[32] B. W. Kernighan and S. Lin, " An efficient heuristic procedure for partitioning graph," *the Bell system technical journal* , pp.291-307, 1970.

[33] Yaotian Fu and P. W. Anderson, " Application of statistical mechanics to NP-complete problems in combinatorial optimization," *J. Phys A* , vol.19,pp.1605-1620, 1986.

[34] Actually we simulate experiments for parallel computing by the single CPU machine.

[35] If the maximum number of the selection is half of the size N, the average number is approximately $0.25 \times N$. That is, in parallel computing the half of units with their values 1 can change their values.

[36] M.Hasegawa, T.Ikeguchi and K.Aihara, "Combination of Chaotic Neurodyanmics with the 2-opt Algorithm to Solve Traveling Salesman Problems," *Phys.Rev.Lett.*, vol.79, pp.2344-2347, 1997.

[37] L.Wang and K.Smith, " On Chaotic Simulated Annealing," *IEEE Trans. Neural Networks*, vol.9, pp.716-718, 1998.

$1/T_0$	$N_{near}=20$	$N_{near}=25$	$N_{near}=30$	$N_{near}=35$	$N_{near}=40$
0.03(0.276)	2.57%(0.82%)	2.48%(0.66%)	2.53%(0.79%)	2.59%(0.45%)	2.58%(0.89%)
0.04(0.368)	2.43%(0.74%)	2.02%(0.25%)	1.83%(0.13%)	1.89%(0.16%)	1.99%(0.22%)
0.05(0.460)	2.61%(0.98%)	2.39%(0.95%)	2.29%(0.61%)	2.38%(0.42%)	2.50%(1.08%)
0.06(0.552)	3.69%(1.09%)	3.09%(0.39%)	3.21%(0.70%)	2.97%(0.38%)	3.12%(0.82%)
0.07(0.644)	4.46%(1.10%)	3.83%(1.61%)	3.64%(1.07%)	3.79%(1.09%)	4.21%(1.34%)

Table 1: Errors of obtained tours for various N_{near} and initial temperatures T_0 in the 225-*city* instance, TSP225. The table shows the average (smallest) values obtained by making 100 trials of 10^4 cycles. Also values in the first column are the inverse of the initial (final) temperature in a trial.

$1/T$	$N_{near}=15$	$N_{near}=20$	$N_{near}=25$	$N_{near}=30$
0.04(0.368)	3.41%(2.36%)	3.49%(2.31%)	3.58%(2.71%)	3.63%(2.57%)
0.05(0.460)	2.24%(1.51%)	2.29%(1.50%)	2.33%(1.61%)	2.37%(1.40%)
0.06(0.552)	2.15%(1.49%)	2.10%(1.27%)	2.13%(1.14%)	2.17%(1.43%)
0.07(0.644)	2.21%(1.36%)	2.25%(1.44%)	2.28%(1.49%)	2.39%(1.49%)
0.08(0.736)	2.65%(1.74%)	2.54%(1.72%)	2.79%(1.90%)	2.83%(1.61%)

Table 2: Errors of obtained tours for various N_{near} and initial temperatures in the 575-*city* instance, RAT575. Notations are the same as those in Table 1.

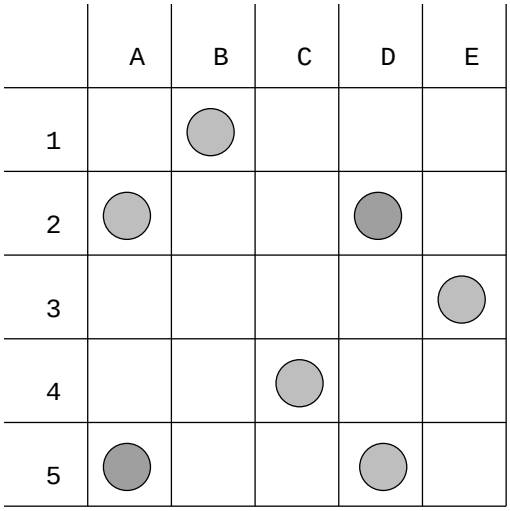
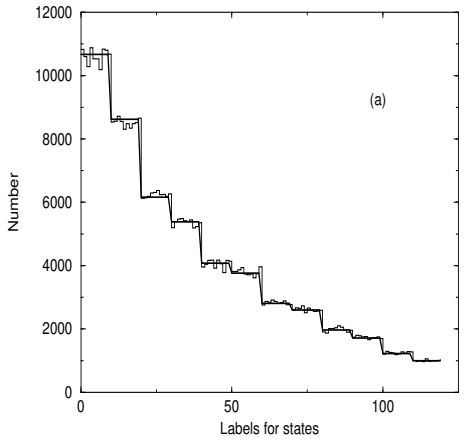


Figure 1: Units in the TSP. A-E denotes names of cities (1-5), while numbers in the column express the order. The existence of circles shows unit values are 1, otherwise they are zero. Gray circles show states before updating. In this figure units (A2) and (D5) have been updated to be zero, while units (A5) and (D2), denoted by circles with slant lines, are 1.



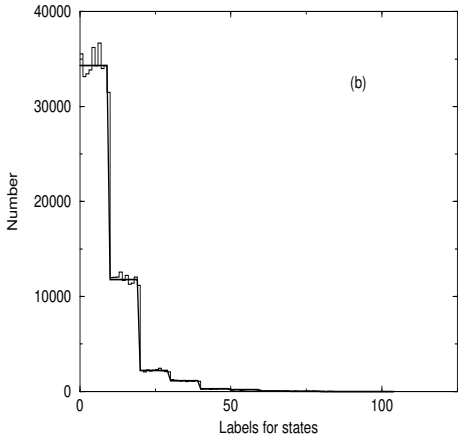


Figure 2: State distributions in the five city TSP. Temperatures are $T = 0.5$ for (a) and $T = 0.1$ for (b). Bold lines express analytical calculations for the Boltzmann distribution, while results obtained in experiments are denoted by the histogram, where the total number of generated states is 10^5. The horizontal lines indicate states, which are ordered by their energy. Numbers of states are shown by the vertical lines.

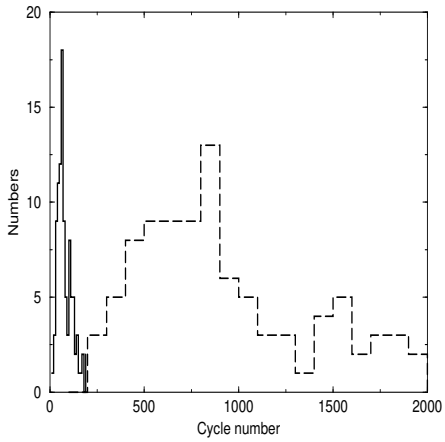


Figure 3: Numbers of trials that the neural network found a solution of the 4CP at the cycle number given on the horizontal axis. 100 trials were carried out for $N_A = 50$. Results using the cooperative updating algorithm are shown by the solid histogram, while those using the usual updating algorithm are shown by the dashed histogram. Temperature T decreases according to $1/\ln(t+1)$. In order to obtain similar heights for both distributions, the width of the solid line is one-tenth of the dashed one. In the usual updating method, 7 trials exceeded 2000 cycles and they are dropped from the figure.

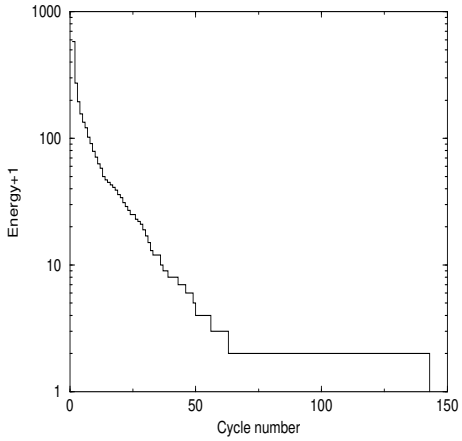


Figure 4: Energy versus number of cycles in experiment for the NQP with a constant temperature 0.1. Here $N = 1000$.

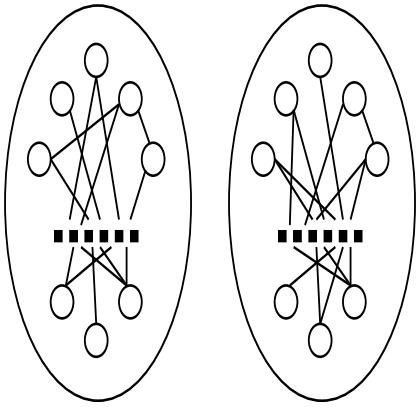


Figure 5: A graph of a problem instance used for experiments for the GBP, which consists of two groups. Each groups has 500 vertices and 5000 edges given randomly. And there are no edges that connect any vertex of one group with any one of the other.

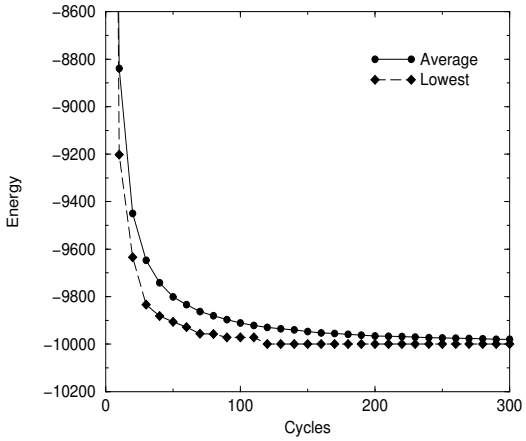


Figure 6: The average and the lowest energy at each cycle in the BGP. Data is from running 100 trials.

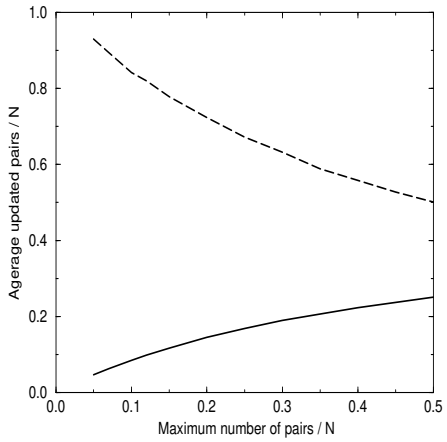


Figure 7: Average numbers of selected groups as a function of the maximum number for them. They are normalized to N. The solid line gives the ratio, while the dashed line denotes the efficiency, obtained by dividing the average number by the maximum number.

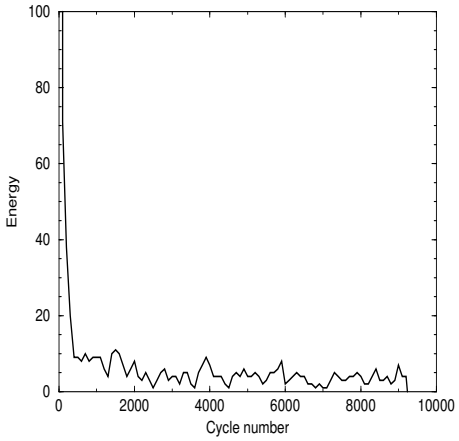


Figure 8: Energy versus numbers of cycles in experiments for the NQP using parallel computation. with a constant temperature 0.1. Here $N = 1000$.

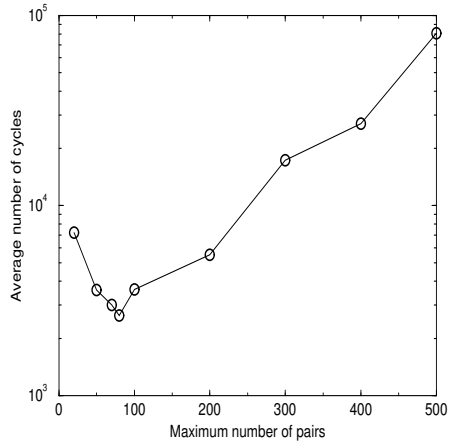


Figure 9: Average numbers of cycles until finding solutions in the NQP, which were obtained by experiments over 10 trials for several values of the maximum number of selected groups. Here $N = 1000$ and $T = 0.1$