Fast combinatorial optimization with parallel digital computers

Fast Combinatorial Optimization with Parallel Digital Computers

Hideki Kakeya and Yoichi Okabe, *Member, IEEE*

***Abstract*—This paper presents an algorithm which realizes fast search for the solutions of combinatorial optimization problems with parallel digital computers. With the standard weight matrices designed for combinatorial optimization, many iterations are required before convergence to a quasioptimal solution even when many digital processors can be used in parallel. By removing the components of the eingenvectors with eminent negative eigenvalues of the weight matrix, the proposed algorithm avoids oscillation and realizes energy reduction under synchronous discrete dynamics, which enables parallel digital computers to obtain quasi-optimal solutions with much less time than the conventional algorithm.**

***Index Terms*—Combinatorial optimization, eigenspace, eigenvalue, eigenvector, geometry, Hopfield network, partition problem, traveling salesman problem (TSP).**

I. Introduction

NEURAL networks have been used as a method to obtain quasi-optimal solutions in various combinatorial optimization problems. In the neural optimization algorithms the weights of the network are made so that the optimal solutions may be located in the low energy area of the state space. As a result quasioptimal solutions are obtained as the state transition of the network proceeds.

The first neural algorithm for combinatorial optimization was the simulated annealing method [6]. In this algorithm one of the neurons is selected randomly and the state of the selected neuron is updated to reduce the energy of the network. Therefore the state transitions of the system must be operated serially though the network itself has parallel architecture.

Hopfield and Tank used analog neurons and continuous dynamics for energy reduction [2]. In their method it is possible to operate calculation in parallel. To simulate continuous dynamics with digital computers, however, many iterations are required before reaching low energy states.

Kindo and Kakeya have introduced geometrical approach to study associative memory [5]. This study suggests that the eigenspace analysis of the weight matrix gives major information to explain the global feature of the dynamics. In this paper the authors analyze the eigenspace of the weight matrices which are designed to solve the combinatorial optimization problems. Based on this analysis the authors propose a new algorithm which enables fast search for quasioptimal solutions with parallel digital computers, and show that the proposed algorithm works for the partition problem and the traveling salesman problem (TSP), each of which represents the combinatorial optimization problems expressed by $(-1, 1)$ neurons and $(0, 1)$ neurons, respectively.

In order to accomplish fast calculation with parallel and digital computers, the algorithm is required to operate in a synchronous and discrete way. When synchronous and discrete dynamics are applied to the standard network designed for combinatorial optimization problems, however, the system becomes oscillatory and the search fails completely. Eigenspace analysis of the weight matrix suggests that the oscillation is caused by the eminent negative eigenvalues. In the algorithm presented in this paper the weight matrix is modified to suppress oscillation and to reach quasioptimal solutions swiftly even when the system is updated in a parallel and synchronous manner.

This paper is organized as follows. In Section II the conventional neural algorithms to solve combinatorial optimization problems are reviewed. In Section III parallel and synchronous algorithm for partition problems is proposed based on the geometrical study. In Section IV parallel and synchronous algorithm for TSP is presented. In Section V detail of the proposed algorithm for practical use is discussed.

II. Conventional Neural Algorithms for Combinatorial Optimization

A. Design of Weight Matrix

Combinatorial optimization problems have two factors. One is the cost and the other is the constraint. The goal is to find the minimum cost solution which satisfies the constraints. The neural networks with the standard types of dynamics have the energy function given by

$$E(\mathbf{x}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}x_i x_j + \sum_{i=1}^{N}\theta_i x_i \tag{1}$$

where

- x_i state of the ith neuron;
- θ_i threshold of the ith neuron;
- w_{ij} weight of the connection from the jth neuron to the ith neuron $(w_{ij} = w_{ji})$.

Energy $E(\mathbf{x})$ decreases as the state transition of the network proceeds.

In the neural algorithms for combinatorial optimization, a candidate of the solutions is represented by one of the state vectors $\mathbf{x} = (x_1, x_2, \ldots, x_N)$. If the cost and the penalty for

Manuscript received April 30, 1998; revised November 25, 1998 and May 5, 2000.

H. Kakeya is with the Communications Research Laboratory, Ministry of Posts and Telecommunications, Tokyo 184-8795, Japan (e-mail: kake@crl.go.jp).

Y. Okabe is with the Research Center for Advanced Science and Technology, University of Tokyo, Tokyo 153-8904, Japan (e-mail: okabe@okabe.rcast.u-tokyo.ac.jp).

Publisher Item Identifier S 1045-9227(00)09948-3.

breaking the constraint can be expressed in a quadratic form of the state vector $\mathbf{x}$, the weight of the network can be designed so that good solutions may be located in the low energy states. For example, if the cost of the solution $\mathbf{x}$ is given by

$$E_c(\mathbf{x}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^{(c)} x_i x_j + \sum_{i=1}^{N} \theta_i^{(c)} x_i \tag{2}$$

and if the penalty of the solution $\mathbf{x}$ for breaking the constraint is written in the form

$$E_p(\mathbf{x}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}^{(p)} x_i x_j + \sum_{i=1}^{N} \theta_i^{(p)} x_i \tag{3}$$

the states $\mathbf{x}$ which have low values of the energy function

$$E(\mathbf{x}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \left(w_{ij}^{(c)} + w_{ij}^{(p)}\right) x_i x_j + \sum_{i=1}^{N} \left(\theta_i^{(c)} + \theta_i^{(p)}\right) x_i \tag{4}$$

represent good solutions of the combinatorial optimization problem. Among them the best solution is represented by the state with the lowest energy. Therefore the network which has the threshold $(\theta_i^{(c)} + \theta_i^{(p)})$ and the weight $(w_{ij}^{(c)} + w_{ij}^{(p)})$ approaches the state representing good solutions as the state transition proceeds.

B. Dynamics for Energy Reduction

Once the weight of the network is designed, the next problem is how to reduce the energy of the network. Here we review two well-known methods to reduce energy.

The first method is the simulated annealing [6]. In this algorithm, one of the neurons is selected randomly and the state of the selected neuron is updated according to the probability written in the form

$$P(x_i(t+1) = X_f) = f\left(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i\right) \tag{5}$$

$$P(x_i(t+1) = X_q) = 1 - P(x_i(t+1) = X_f) \tag{6}$$

$$f(u) = \frac{1}{1+\exp(-u/T)} \tag{7}$$

where $X_f = 1, X_q = 0$ in the $(0,1)$ neuron model and $X_f = 1, X_q = -1$ in the $(-1,1)$ neuron model. This is a very familiar method in the neural optimization. In this method, however, only one neuron can change its state at one time and the search cannot be operated in parallel.

The second method is the analog neural network [2]. This algorithm uses analog neurons and continuous dynamics given by

$$\tau \frac{du_i}{dt} = -u_i + \sum_{j=1}^{N} w_{ij} x_j - \theta_i \tag{8}$$

where

$$x_i = f_0(u_i) = \frac{1}{1+\exp(-u_i/T)} \tag{9}$$

in the $(0,1)$ neuron model and

$$x_i = f_1(u_i) = \frac{1-\exp(-u_i/T)}{1+\exp(-u_i/T)} \tag{10}$$

in the $(-1,1)$ neuron model.

These dynamics enable parallel state transitions of the system. Therefore fast search is realized if an analog computer is available. When a digital computer, which is more robust and general, is used, however, the calculation of the above differential equation must be converted to the difference equation written in the form

$$u_i(t+\tau\Delta) = (1-\Delta)u_i(t) + \Delta\left(\sum_{j=1}^{N} w_{ij} x_j(t) - \theta_i\right). \tag{11}$$

If time difference Δ is small, the search can be carried out successfully though it requires many iterations before reaching good solutions. It is expected that the number of iterations can be reduced when larger Δ is used. When Δ is large, however, the network designed to solve combinatorial optimization problems becomes oscillatory and the search fails completely.

The above discussion has shown that the use of parallel and synchronous digital computers cannot quicken convergence to a quasioptimal solution whether simulated annealing or analog neural network is adopted to the search. In the following sections we give two examples of combinatorial optimization and propose a new algorithm which overcomes the above dilemma. In Section III we take up partition problems and in Section IV we take up TSP. In each section the eigenspace of the weight matrix is analyzed in the beginning. Based on the eigenspace analysis the cause of oscillation is investigated. Then fast parallel and synchronous search algorithm for each problem is presented in the end of each section.

III. Fast Algorithm for Partition Problem

A. Formulation

Here we give a brief description of the partition problem [6] and review how it is solved by the neural networks. In the simple partition problem it is required to divide the units which are mutually connected with certain numbers of links into two boxes with limited capacity so that the interconnection between the two boxes may be minimized. Let the number of units be N and the number of links between the ith and the jth units be d_{ij}. Here we prepare N neurons, each of which corresponds to one of the units. Each neuron takes the values 1 or -1, which represents which box the unit is supposed to be put in. Here the ith neuron represents the state of the ith unit.

We denote the value of the ith neuron representing the state of the ith unit as x_i. Then the number of connections between the two boxes is written in the form

$$E_c = \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}(x_i - x_j)^2. \tag{12}$$

This equation is rewritten in the form

$$E_c = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}x_i x_j + \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{ij}. \tag{13}$$

Since the second term of the above equation is constant, the relative cost can be represented by the first term, which is expressed in a quadratic form of $\mathbf{x}$.

The penalty for breaking the constraint that each box has a limited capacity is expressed by

$$E_p = \sum_{i=1}^{N}\sum_{j=1}^{N} h(1-\delta_{ij})x_i x_j \quad (h>0) \tag{14}$$

for this value becomes minimum when the units are divided equally into the two boxes. This penalty function is also expressed in a quadratic form of $\mathbf{x}$. Hence the weight of the network to solve the partition problem

$$w_{ij} = d_{ij} - h(1-\delta_{ij}) \tag{15}$$

is obtained. Here $\theta_i = 0$ for all i.

B. Eigenspace Analysis of Weight Matrices

Eigenspace analysis of the weight matrix of Hopfield network was first introduced by Aiyer *et al.* [1]. In the framework of the autocorrelational associative memory, Kindo and Kakeya have extended the eigenspace analysis to explain the existence of capacity limit, the emergence of spurious memories, and the various phenomena caused by the nonmonotonic neurons [3]–[5].

We give a short review of the geometrical explanation on neural dynamics by Kindo and Kakeya. For simplicity $f_1(u) = \mathrm{sgn}(u)$ and $\theta_i = 0(\forall i)$ are assumed. Then $|\mathbf{x}| = \sqrt{N}$ holds, for the state vector $\mathbf{x}$ has $+1$ or -1 as its components. Therefore $\mathbf{x}$ is always on the surface of the hypersphere S^{N-1} with radius $\sqrt{N}$ as shown in Fig. 1. The neural dynamics given by (10) and (11) are divided into two phases. In the first phase the state vector $\mathbf{x}(t)$ is transfered to the vector $\mathbf{u}(t+\Delta) = (1-\Delta)\mathbf{u}(t)+\Delta W\mathbf{x}(t)$ linearly $(\tau = 1)$. In the second phase the vector is quantized to the nearest state vector which requires the least angle rotation. Therefore, from the hyperspherical viewpoint, linear transformation gives the major driving force of dynamics, while nonlinear transformation generates the terminal points of dynamics where the flow of linear transformation is slow. This suggests that the eigenspace analysis of the weight matrix gives major information to explain the global feature of the dynamics. Here we apply this approach and analyze the weight of the network which is designed to solve the combinatorial optimization problems.

As an example of partition problems, here we consider the case where d_{ij} takes an integer between 50 and 150 with the same probability. Below we use $m_{ij} = 0.01d_{ij}$ instead of d_{ij} to normalize the cost term so that $E[m_{ij}] = 1.0$ holds. As the constraint term we use $h = 1.5$, which is large enough to keep the number of exciting neurons around $N/2$. The eigenvalue distribution of the weight matrix in this case is shown in Fig. 2 $(N = 100)$. As shown in the figure, it has an eigenvector whose eigenvalue is by far the smallest. This eigenvalue derives from the penalty term $[w_{ij}^{(p)}] = [-h(1-\delta_{ij})]$. Therefore the corresponding eigenvector spans the space where the constraint is not satisfied.

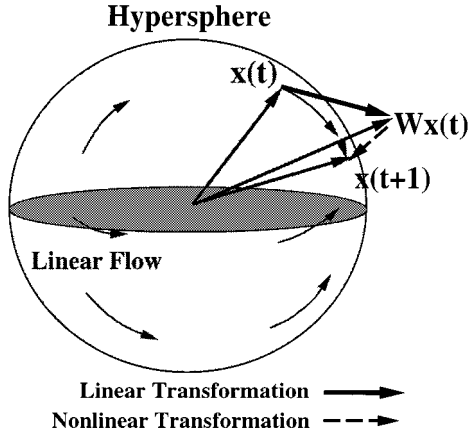

Fig. 1. Geometry of neural dynamics.

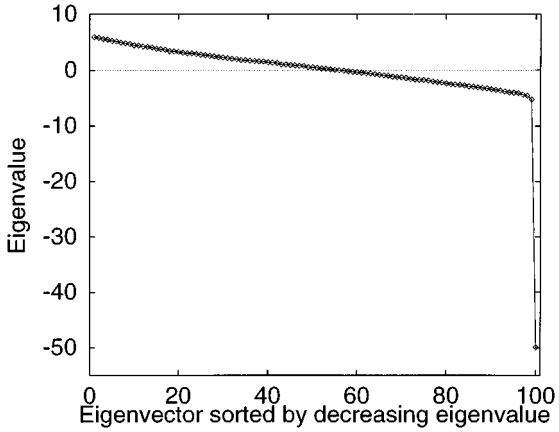

Fig. 2. Eigenvalue distribution of weight matrix designed to solve partition problem.

As stated above, good solutions are located in the low energy area of the state space. The low energy state of the network corresponds to the state which is composed mainly of the eigenvectors with large eigenvalues. Therefore good solutions have large components of eigenvectors with large eigenvalues and almost no components of eigenvectors with negative eigenvalues.

Though the asynchronous discrete dynamics and the continuous dynamics realize state transitions toward the low energy states, the synchronous discrete dynamics with large time difference Δ do not always work in the same way. We illustrate the simple mechanism of this difference in Fig. 3. Here the effect of nonlinear transformation is neglected for simplicity and the dynamics given by $\mathbf{u}(t+\Delta) = (1-\Delta)\mathbf{u}(t)+\Delta W\mathbf{u}(t)$ are illustrated. As shown in the figure, when Δ is small, the state vector converges to the eigenvector of W with the largest positive eigenvalue, which spans the low energy states. When Δ is

large, however, the state vector is attracted to the eigenvector of W whose eigenvalue has the largest absolute value. This means that the state vector stays in the high energy states when a negative eigenvalue has larger absolute value than the maximum positive eigenvalue. In this case Δ has to be kept small to ensure convergence to a low energy state though larger Δ leads to faster convergence when positive eigenvalues are dominant.

C. Fast Search with Parallel and Synchronous Computers

From the above discussion it is expected that the synchronous and discrete state transitions with large Δ proceed toward the low energy states if the effect of the minimum eigenvalue is canceled. In this case the component of the eigenvector with the minimum eigenvalue is eliminated easily, because its absolute value is by far the largest of all the eigenvalues. Therefore, by carrying out a few steps of state transitions given by

$$\mathbf{e}(t+1) = C_t W \mathbf{e}(t) \tag{16}$$

from a random vector $\mathbf{e}(0)$, (16) soon converges to the state where

$$\lambda \mathbf{e}(t+1) \simeq W \mathbf{e}(t) \tag{17}$$

holds and the minimum eigenvalue $\lambda_{\min}$ and its eigenvector $\mathbf{e}^{(\min)}$ are obtained. Here C_t is the coefficient which normalizes the length of the vector $|\mathbf{e}|$ to one.

The component of the eigenvector with the minimum eigenvalue is reduced from the weight matrix $[w_{ij}]$ by calculating

$$v_{ij} = w_{ij} - \kappa \lambda_{\min} e_i^{(\min)} e_j^{(\min)} \tag{18}$$

where κ is a positive constant. When $\kappa = 1$, the minimum eigenvalue component is eliminated completely.

If the weight matrix $V = [v_{ij}]$ and large time difference Δ are used, it is expected that the state vector converges to a quasioptimal solution quickly and stably. In the next section we confirm this prediction by numerical experiments.

D. Simulation and Result

Behavior of Hopfield network depends not only on the weight matrix and the time difference we have focused on so far, but also on the temperature parameter T and the initial conditions. In the numerical experiments we use various combinations of temperature and initial conditions to confirm that the algorithm we propose is effective in general.

To prove the superiority of the algorithm shown in the previous subsection, we compare the proposed algorithm using the matrix V and large time difference Δ with the conventional algorithm using the standard weight matrix W and small Δ under various parameter sets of temperature and initial conditions. In the experiments shown here, the behavior of a parallel computer is simulated by a serial computer. One step of iteration corresponds to one step of calculation in the parallel computer, though serial computers have to work N times for each iteration. As the initial conditions we give $x_i(0) = \pm 1$ randomly and $u_i(0) = Rx_i(0)$.

Fig. 4 shows the result of the numerical experiments given by the conventional algorithm and Fig. 5 shows the result given

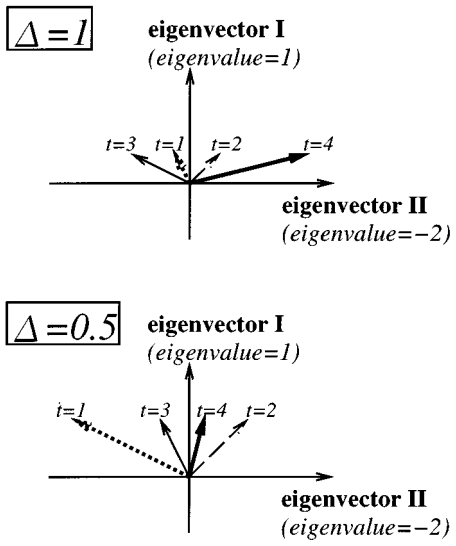

Fig. 3. Convergence of dynamics given by difference equations with large and small time differences Δ. Here effect of nonlinear transformation is neglected for simplicity.

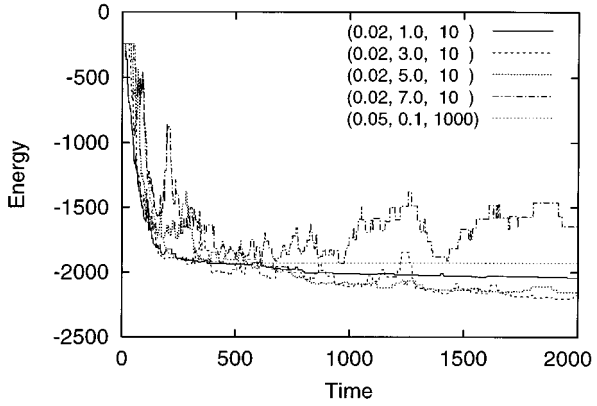

Fig. 4. Energy reduction process of conventional algorithm to solve partition problem. Each line (a,b,c) shows the process under $\Delta = a, T = b$, and $R = c$.

by the proposed algorithm. (Note that the scale of the x-axis differs in Figs. 4 and 5.) Dynamical behaviors under various temperature conditions are shown in both figures. As for the initial condition, the parameter R which gives the best result in each temperature is selected to be shown in both figures. (Note that R makes no difference to the results when Δ is near one.)

In Fig. 4, when T is large, the search dynamics become oscillatory and do not reach a state with lower energy. On the contrary, when T is small, the search dynamics freeze around $E = -2000$. In the other cases better solutions in the lower energy area are obtained. Nevertheless it takes about 500 or more iterations until solutions which satisfy $E < -2000$ are obtained. On the other hand, in Fig. 5 it takes only about ten iterations until E becomes less than $-2000(T = 3)$ and also it takes only about 50 iterations before the solutions near the optimum are obtained.

From these figures it is clear that the proposed algorithm realizes much faster convergence to a quasioptimal solution than the conventional algorithm in general, including under the best condition.

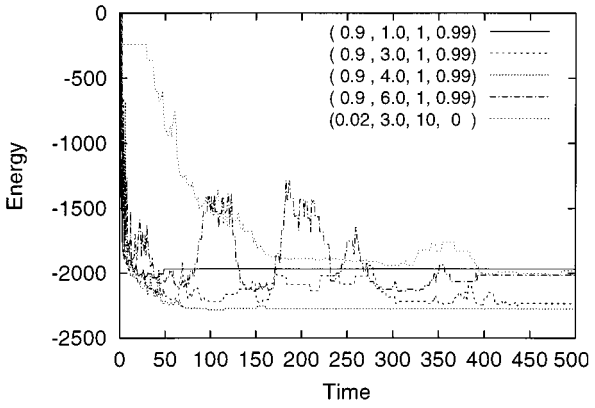

Fig. 5. Energy reduction process of conventional algorithm to solve partition problem. Each line (a,b,c,d) shows the process under $\Delta = a, T = b, R = c$ and $\kappa = d$. (the line with $\kappa = 0$ is from the conventional model for comparison. Note that the scale here is different from that in Fig. 4.

IV. Fast Algorithm for TSP

A. Formulation

Here we give a brief review of the TSP and the conventional algorithm to solve it by the neural networks [2]. In the simple TSP it is required to find the shortest round tour which covers all the cities the salesman is supposed to visit. Let the number of cities be N and the distance between city a and city b be d_{ab}. Here we prepare $N \times N$ neurons, each of which outputs the value x_{ai}. The output $x_{ai} = 1$ means that city a is visited in the ith trip, while the output $x_{ai} = 0$ means that city a is not visited in the ith trip. Then the whole distance of the travel path is given by

$$E_c(\mathbf{x}) = \frac{D}{2} \sum_a \sum_{b \neq a} \sum_i d_{ab} x_{ai} (x_{b,i+1} + x_{b,i-1}). \tag{19}$$

Here again the constraints which must be satisfied to be a solution of the problem exist. First only one city can be visited at the same time. Second each city is visited only once. Finally every city must be visited during the tour. The penalty for breaking these constraints can be written in the form

$$\begin{aligned} E_p(\mathbf{x}) = {} & \frac{A}{2} \sum_a \sum_i \sum_{j \neq i} x_{ai} x_{aj} \\ & + \frac{B}{2} \sum_i \sum_a \sum_{b \neq a} x_{ai} x_{bi} \\ & + \frac{C}{2} \left(\sum_a \sum_i x_{ai} - N \right)^2. \end{aligned} \tag{20}$$

By taking the above cost and penalty into account, the energy function of the network

$$\begin{aligned} E(\mathbf{x}) = {} & \frac{A}{2} \sum_a \sum_i \sum_{j \neq i} x_{ai} x_{aj} \\ & + \frac{B}{2} \sum_i \sum_a \sum_{b \neq a} x_{ai} x_{bi} \\ & + \frac{C}{2} \left(\sum_a \sum_i x_{ai} - N \right)^2 \\ & + \frac{D}{2} \sum_a \sum_{b \neq a} \sum_i d_{ab} x_{ai} (x_{b,i+1} + x_{b,i-1}) \end{aligned} \tag{21}$$

can be designed. From this energy function the weight matrix

$$\begin{aligned} w_{aibj} = {} & -A\delta_{ab}(1 - \delta_{ij}) - B\delta_{ij}(1 - \delta_{ab}) - C \\ & - Dd_{ab}(\delta_{j,i+1} + \delta_{j,i-1}) \end{aligned} \tag{22}$$

and the threshold

$$\theta_{ai} = -CN \tag{23}$$

for the network to solve TSP are derived.

The network with the above weight, however, has a tendency to give solutions which do not cover all the N cities. This tendency comes from the fact that the weights representing the cost term $w^{(c)} = -Dd_{ab}(\delta_{j,i+1} + \delta_{j,i-1})$ are all negative. This results in the decrease of firing rate of neurons. In the paper by Hopfield and Tank [2], the penalty term $(C/2)(\sum_a \sum_i x_{ai} - N)^2$ was replaced by $(C/2)(\sum_a \sum_i x_{ai} - \alpha N)^2 (\alpha > 1)$ to increase the firing rate. In this paper, the weight for cost $w^{(c)} = D(r - d_{ab})(\delta_{j,i+1} + \delta_{j,i-1})(r > 0)$ is introduced instead of changing the penalty term in order to keep the proper firing rate, which enables the network to converge to a solution which satisfies the constraint in a more natural way. Note that this change does not affect the convergence speed, on which the present paper focuses. It only facilitates the parameter selection, for the above discussion suggests that r near the average of d_{ij} cancels the influence of the cost term on the firing rate, while no hints are available to find proper α.

The dynamics of analog neural networks to solve TSP are given by

$$\tau \frac{du_{ai}}{dt} = -u_{ai} + \sum_{b=1}^{N} \sum_{j=1}^{N} w_{aibj} x_{bj} - \theta_{ai} \tag{24}$$

$$x_{ai} = f(u_{ai}) \tag{25}$$

$$f(u) = \frac{1}{1 + \exp(-u/T)}. \tag{26}$$

Thus the analog neural network to solve TSP is obtained.

B. Eigenspace Analysis and Fast Search Algorithm

Here we take up an example of TSP in the paper by Hopfield and Tank [2], analyze the eigenspace structure of the weight matrix and present an algorithm which enables fast search when parallel and synchronous computers are available.

As an example of TSP, we pick up ten cities randomly from the area $[0, 1] \times [0, 1]$ as Hopfield and Tank did, and try to find the shortest round tour. The parameters A, B, C, D are chosen as $A = 500, B = 500, C = 200, D = 500$, which are the same values that Hopfield and Tank adopted. As for the parameter $r, r = 0.9$ is used. The eigenvalue distribution of the weight matrix obtained in this way is shown in Fig. 6. As shown in the figure, the weight matrix of TSP also has an extremely small eigenvalue, which derives from the penalty term. Therefore search dynamics become oscillatory when Δ is large.

From the discussion on the partition problems, it is predicted that the weight matrix without the outstanding minimum eigenvalue

$$v_{aibj} = w_{aibj} - \kappa \lambda_{\min} e_{ai}^{(\min)} e_{bj}^{(\min)} \tag{27}$$

realizes search under large Δ. Simple adoption of this weight matrix, however, fails in this case because reduction of small

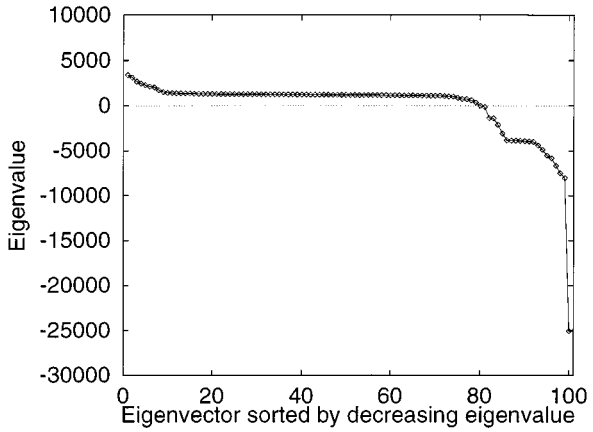

Fig. 6. Eigenvalue distribution of weight matrix designed to solve TSP.

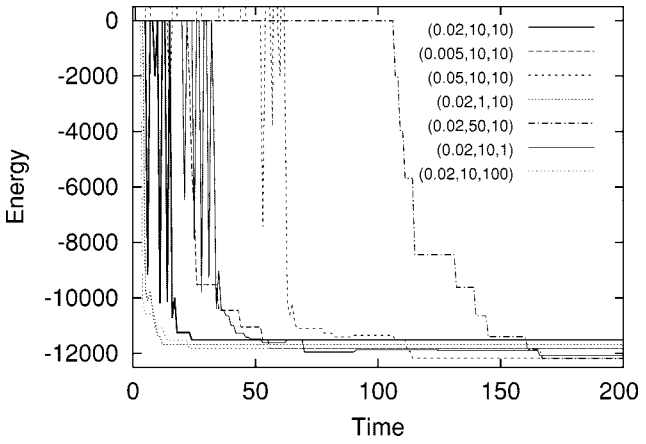

Fig. 7. Energy reduction process of conventional algorithm to solve TSP under various temperature and initial contidions. The line with (a,b,c) shows the search process under $\Delta = a, T = b$, and $R = c$. $(r = 0.9, A = B = D = 500, C = 200)$.

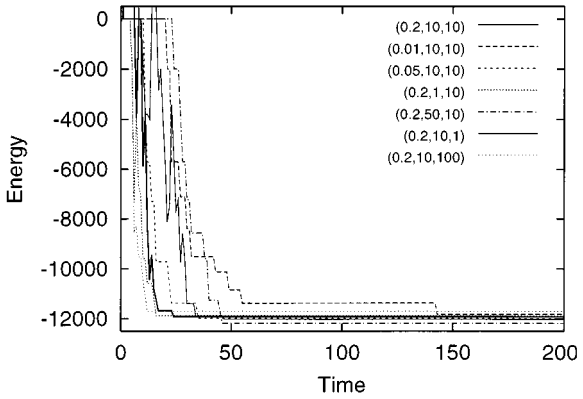

Fig. 8. Energy reduction process of proposed algorithm to solve TSP $(\kappa = 0.7)$ under various temperature and initial contidions. The line with (a,b,c) shows the search process under $\Delta = a, T = b$, and $R = c$. $(r = 0.9, A = B = D = 500, C = 200)$.

eigenvalues increases the firing rate of the network. As a result the network converges to a solution which does not satisfy the constraints.

To adjust the firing rate so that the network converges to a solution which satisfies the constraints, the threshold should be raised in accordance with the increase of the average weight. Since the threshold is always active while the firing rate of neurons in the feasible solutions is $1/N$, the effect of the threshold is N times larger than that of the neurons. Therefore it is expected that the threshold

$$\phi_{ai} = -CN + \frac{1}{N}\kappa\lambda_{\min}\sum_{b,j} e_{ai}^{(\min)} e_{bj}^{(\min)} \tag{28}$$

is suitable to keep the firing rate to the proper level.

In the next section the ability of the algorithm presented here is tested in the numerical experiments.

C. Simulation and Result

As written in the previous section, behavior of Hopfield network depends not only on the weight matrix and the time difference, but also on the temperature and the initial conditions. To prove the superiority of the algorithm shown in the previous section, we compare the proposed algorithm using matrix V, threshold ϕ, and large time difference Δ with the conventional algorithm using standard weight matrix W, threshold θ, and small Δ under various conditions. The initial conditions of the internal state of each neuron u_i is set so that it obeys the uniform distribution $[-R, R]$, and various sets of T and R are tried for each algorithm.

The results of the numerical experiments are shown in Figs. 7 and 8. Fig. 7 shows the dynamical behavior of the conventional algorithm and Fig. 8 shows that of the proposed algorithm. In both figures behaviors under various temperature and initial conditions are shown. These figures show that proposed algorithm realizes fast convergence in general, yet under low temperature and large R the conventical algorithm gives as fast convergence as the proposed algorithm.

Fig. 9 and 10 show the quality of the solutions obtained under the parameter sets used in the numerical experiments in Figs. 7 and 8. As shown in the figures the solutions obtained by the proposed algorithm are better than those obtained by the conventional algorithm in general. These figures also show that the dynamics with small T and large R, which realizes fast convergence in Figs. 7 and 8, tend to converge to a solution which is far from the optimal solution. This is because these conditions freeze the state vectors strongly and are apt to trap them at a high energy state. To avoid convergence to rather high energy states and realize convergence to low energy states, larger T and smaller R should be selected though they tend to delay the convergence.

Among the conditions in Figs. 7 and 8, only the conditions which lead to convergence to rather lower energy states are picked up and shown in Figs. 11 and 12. These figures show that the proposed algorithm with weight matrix V and threshold ϕ realizes faster convergence under the conditions which lead to relatively good solutions, while the conventional algorithm requires much more iterations before convergence.

V. Discussion

A. Emergence and Removal of Outstanding Negative Eigenvalues

In the examples discussed above, there exists only one outstanding negative eigenvalue, which facilitates removal of the

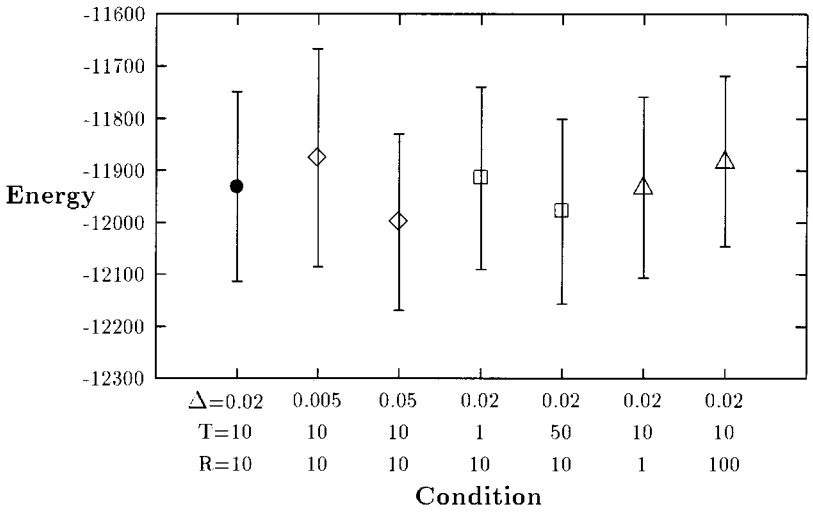

Fig. 9. Average snd variance of energy (solution quality) after convergence obtained by conventional algorithm under parameter sets used in Fig. 7. (Lower energy means better solution.)

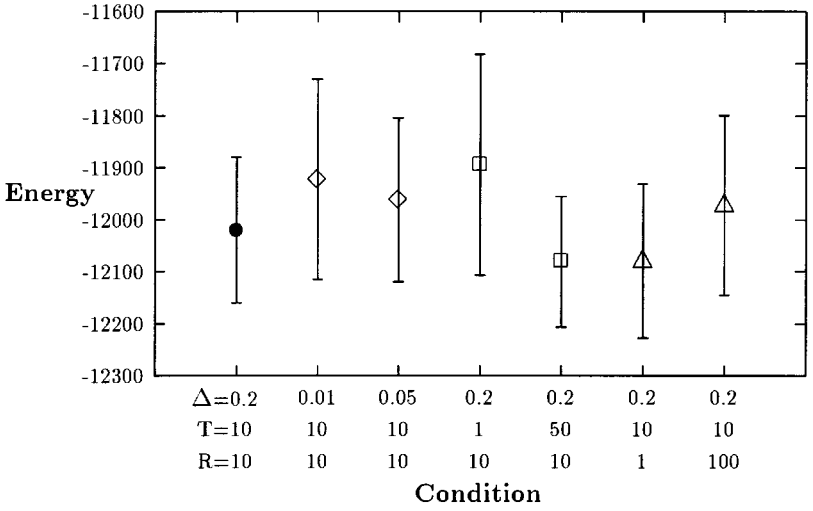

Fig. 10. Average and variance of energy (solution quality) after convergence obtained by proposed algorithm under parameter sets used in Fig. 8. (Lower energy means better solution.)

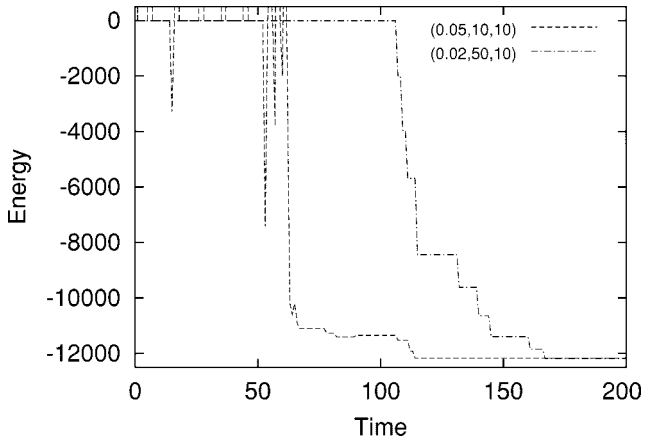

Fig. 11. Energy reduction process of conventional algorithm to solve TSP. Only the data which made good scores in Fig. 9 are extracted from Fig. 7 and shown here.

corresponding eigenvector. For the practical use of the presented algorithm, however, it must be confirmed whether this is true generally. If not, it is necessary to establish the algorithm which copes with the case where multiple outstanding eigenvalues appear.

In the partition problem only one negative eigenvalue emerges because the constraint space has only one dimension. In TSP many constraints exist, which generate many negative eigenvalues. Nevertheless one constraint is by far the strongest, which results in emergence of an eminent eigenvalue. These are not rare cases since the whole constraint or the main constraint are often given by a single equation in the combinatorial optimization.

Yet it is not assured that all the combinatorial optimization problems always have only one eminent negative eigenvalue. When a group of eminent negative eigenvalues exists, however,

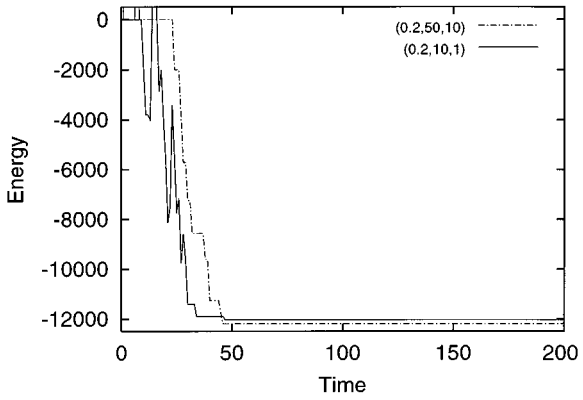

Fig. 12. Energy reduction process of proposed algorithm to solve TSP. Only the data which made good scores in Fig. 10 are extracted from Fig. 8 and shown here.

the corresponding space can be suppressed in the same way as discussed in the above examples.

For example, assume there exists M eminent negative eigenvalues $\lambda_{N-M+1}, \lambda_{N-M+2}, \ldots, \lambda_{N-1}, \lambda_N$, where $\lambda_{N-M+1} \simeq \lambda_{N-M+2} \simeq \cdots \simeq \lambda_{N-1} \simeq \lambda_N$ and $\lambda_1 > \lambda_2 > \cdots > \lambda_{N-M} \gg \lambda_{N-M+1} > \cdots > \lambda_N$. Let the eigenvectors corresponding λ_i be $\mathbf{e}^{(i)}$ and the weight matrix be W. To extract the minimum eigenvalue component $\mathbf{e}^{(N)}$, many iterations of (18) are required in this case. Nevertheless here the goal is to remove all the M eigenvector components $\mathbf{e}^{(N-M+1)}, \ldots, \mathbf{e}^{(N)}$. To achieve it, the following procedure is sufficient.

First repeat (18) a few times, then the state

$$\tilde{\lambda}\mathbf{e}(t) = W\mathbf{e}(t-1) \tag{29}$$

$$\mathbf{e}(t) \simeq \sum_{i=N-M+1}^{N} c_i \mathbf{e}^{(i)} \tag{30}$$

appears, where $|\mathbf{e}(t)| = 1$ for all t and $\tilde{\lambda} < 0$. Next remove the component of $\mathbf{e}(t)$ from W, then the matrix written in the form

$$W_{\text{new}} = W - \tilde{\lambda}\mathbf{e}(t)\mathbf{e}(t)^T \tag{31}$$

$$= W - \tilde{\lambda}\left(\sum_{i=N-M+1}^{N} c_i^2 \mathbf{e}^{(i)}\mathbf{e}^{(i)T}\right) \tag{32}$$

$$= \sum_{i=1}^{N-M} \lambda_i \mathbf{e}^{(i)}\mathbf{e}^{(i)T} + \sum_{i=N-M+1}^{N} \left(\lambda_i - \tilde{\lambda}c_i^2\right)\mathbf{e}^{(i)}\mathbf{e}^{(i)T} \tag{33}$$

is obtained. Here the components of the eigenvectors $\mathbf{e}^{(N-M+1)}, \ldots, \mathbf{e}^{(N)}$ are diminished. Therefore if this process is repeated until $|\tilde{\lambda}|$ becomes small (about M times), the components of all the eminent eigenvalues can be weakened and the stable search is enabled with the obtained weight matrix.

Including this procedure, the whole process of the proposed algorithm can be summarized as follows.

1) Analyze and grasp the feature of the eigenspace with a problem of a smaller size. Find suitable parameters T, R, Δ for this problem.
2) Repeat the process denoted in this section and remove the eminent negative eigenvalues.
3) Carry out the search dynamics using the weight matrix from which eminent negative eigenvalue components are removed.

B. Implementation on Parallel Computers

In the present paper parallel computation have been simulated on digital computers. When one carries outs the algorithm presented here with a parallel computer, one has to take the following two points into consideration.

One point is the size of the parallel computer. If the parallel computer used for the calculation has the same or more processors P than the size of the combinatorial optimization problems N (the number of neurons required for calculation), the present algorithm can be implemented directly on the computer. When $P < N$, one step of dynamics has to be divided into N/P pieces so that the computer can execute the required calculation. Simulation by a digital computer corresponds to the case where $P = 1$. Even when $P < N$, the solutions can be obtained P times faster than the serial computation.

The other point is the format of the parallel computers. So far we have assumed that the operation of the search dynamics is synchronous. Recently, however, the possibility of asynchronous parallel computers is attracting attention. In the present case, it may seem that asynchronous parallel computers can accomplish fast search without the modification of weight and threshold we have discussed so far.

If the interval between each processing occupies most of the calculation time, the system operates like a serial calculator eventually and modification of weight and threshold is not necessary. Nevertheless this can be regarded as the case where the system is not designed efficiently. If the calculation of each processor (neuron), communications among processors, and update of the output of each processor take most of the time, the system operates almost in the same manner as the synchronous dynamics we have simulated in this paper. Therefore asynchronous computing alone cannot avoid the problem of oscillation, and modification of weight and threshold we have discussed above is still effective and useful.

VI. Conclusion

In this paper an algorithm which enables parallel digital computers to realize fast search for quasioptimal solutions of the combinatorial optimization problems has been proposed. The proposed computational method avoids oscillation by removing the component of the eigenvector with the eminent negative eigenvalues of the weight matrix, which enables energy reduction under the synchronous discrete dynamics. In the simulation of the partition problem and the traveling salesman problem, it has been shown that the proposed algorithm requires much fewer iterations than the conventional algorithm before reaching quasioptimal solutions.

References

[1] S. V. B. Aiyer, M. Niranjan, and F. Fallside, "A theoretical investigation into the performance of the Hopfield model," *IEEE Trans. Neural Networks*, vol. 1, pp. 204–215, 1990.

[2] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–152, 1985.

[3] H. Kakeya and T. Kindo, "Hierarchical concept formation in associative memory composed of neuro-window elements," *Neural Networks*, vol. 9, pp. 1095–1098, 1996.
[4] ——, "Eigenspace separation of autocorrelation memory matrices for capacity expansion," *Neural Networks*, vol. 10, pp. 833–843, 1997.
[5] T. Kindo and H. Kakeya, "A geometrical analysis of associative memory," *Neural Networks*, vol. 11, pp. 39–51, 1998.
[6] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

Hideki Kakeya received the B.S. degree in biophysics and biochemistry, the M.E. degree in information physics, and the Dr. degree in advanced interdisciplinary studies from the University of Tokyo, Tokyo, Japan, in 1993, 1995, and 1998, respectively.

From 1993 to 1998, he was engaged in neural-network research. He currently works as a Researcher in Communications Research Laboratory.

Yoichi Okabe (S'67–M'72) received the B.E., M.E., and Ph.D. degrees in electric engineering from the University of Tokyo. Tokyo, Japan, in 1967, 1969, and 1972, respectively.

He became a Lecturer of the Department of Electrical Engineering in University of Tokyo in 1972 and was promoted to Associate Professor in 1983 and Professor in 1989. He was with IBM Watson Research Center as a Visiting Scientist from 1977 to 1978. He moved to the Research Center for Advanced Science and Techonology (RCAST), the University of Tokyo in 1990. Since 1999 he has been the director of RCAST. He has been engaged in the study of superconductive electrical device, biomagnetic measurement, and neural networks.