A New Relaxation Procedure in the Hopfield Network for Solving Optimization Problems

Xinchuan Zeng and Tony R. Martinez

Computer Science Department, Brigham Young University, Provo, Utah 84602

(*Email*: zengx@axon.cs.byu.edu, martinez@cs.byu.edu)

Abstract. When solving an optimization problem with a Hopfield network, a solution is obtained after the network is relaxed to an equilibrium state. The relaxation process is an important step in achieving a solution. In this paper, a new procedure for the relaxation process is proposed. In the new procedure, the amplified signal received by a neuron from other neurons is treated as the target value for its activation (output) value. The activation of a neuron is updated directly based on the difference between its current activation and the received target value, without using the updating of the input value as an intermediate step. A *relaxation rate* is applied to control the updating scale for a smooth relaxation process. The new procedure is evaluated and compared with the original procedure in the Hopfield network through simulations based on 200 randomly generated instances of the 10-city *traveling salesman problem*. The new procedure reduces the error rate by 34.6% and increases the percentage of valid tours by 194.6% as compared with the original procedure.

Keywords: constraint satisfaction, Hopfield network, neural networks, optimization, relaxation procedure

Introduction

Hopfield and Tank [1] proposed an approach of using a neural network to find a suboptimal solution of the traveling salesman problem (*TSP*). In their approach, a *TSP* instance is represented by an *energy function* including the cost and constraint terms that reflect the objective of a solution. The objective of the constraint term is to find a valid tour, which requires that each city must be visited once and only once. The objective of the cost term is to find the shortest valid tour. The energy function can be implemented by a neural network. For an N-city *TSP* problem, the network consists of $N \times N$ neurons and the links connecting the neurons. The weights of the links are encoded to represent the cost and the constraints for the given *TSP* instance to be solved. To reach a solution, the parameters in the energy function and the initial values of the neurons need to be properly chosen, and then the network is relaxed from the initial state. During the relaxation, each neuron updates its input value based on the information received from other neurons. A solution is obtained when the network reaches an equilibrium state. They showed that a neural network configured with symmetric weights always converges to a local minimum which may represent a valid solution. Their simulation on a 10-city *TSP* instance showed that the neural network was able to obtain a high percentage of valid tours with high quality (short tours). This network is commonly referred to as a *Hopfield network*.

Since Hopfield and Tank's work [1], there has been growing interest in the Hopfield network because of its advantages over other approaches for solving optimization problems. The advantages include massive parallelism, convenient hardware implementation of the neural network architecture, and a common approach for solving various optimization problems. Much research has focused on analyzing and improving the original model in order to obtain a higher percentage of valid solutions and solutions with better quality.

The work by Wilson and Pawley [2] showed that the Hopfield network often failed to converge to valid solutions. When it converged, the obtained solution was often far from the optimal solution. Since their report, various modifications have been proposed to improve the convergence of the Hopfield network. Brandt et al. [3] and Aiyer et al. [4] showed that the convergence of the Hopfield network could be improved by modifying the energy function. The modified energy functions have a higher degree of constraints for the network to relax to valid solutions. Protzel et al. [5] studied Brandt et al.'s formulation with different parameters. Despite the improved convergence of the modified versions of the Hopfield network, they may not converge to solutions with good quality [5]. Li combined the Hopfield network with the "augmented Lagrange multipliers" algorithm from optimization theory [6]. Although both the convergence and the quality of the solutions can be improved by this approach, much more iterations are required to converge to a solution. Catania et al. applied a fuzzy approach to automatically tune the parameters in the Hopfield network [7]. They reported that the automatic tuning of the parameters performed better than heuristic and manual tuning. Although the above approachs have demonstrated improvement, most of them were empirically tested only on a single or a small number of TSP instances with a narrow range of city distributions, and the general performances on a large number of data sets were not reported.

Despite the improvement of the performance of the Hopfield network over the past decade, this model still has some basic problems [8, 9]. One of the problems is that the performance of the Hopfield network is inconsistent. The performance is good for some instances of *TSP*, but poor for other instances with the same size. The performance is usually better for instances with simple topology, but poor for those with complex topology, in which case there are large number of similar competent fixed points and the solutions are often trapped in poor local minima. Another problem is that the performance is sensitive to the choice of the parameters in the energy function, and different parameter settings can lead to significant differences in the performance.

In the Hopfield model, a solution is achieved after the network is relaxed and has settled down to an equilibrium point. The process of relaxation is an important step in achieving a solution and can have an important impact on the quality of the solution. However, there has not been enough effort to study this process and the underlying mechanism during the process is not well understood.

This paper analyzes the effects of the relaxation process on the quality of the solutions and proposes a new relaxation procedure to improve the performance of the network. In the new procedure, each neuron updates the activation directly without updating the input value. The amplified signal received by each neuron from all other neurons is treated as the target value for updating its activation value. The activation is updated based on the difference between the received target value and the current activation value so that the activation value moves closer to the target value. For a smooth relaxation of the network, a relaxation rate is applied to control the pace of the updating process. The simulation results based on 200 randomly generated instances of the 10-city *TSP* show significant improvement in the performance of the new procedure over the original procedure. The new procedure reduces the error rate by 34.6% and increases the percentage of valid tours by 194.6% as compared with the original procedure. The two procedures are compared and the reasons for the differences in the performance are analyzed.

Background of the Hopfield Network

The Hopfield network [1] consists of a set of neurons and the links connecting the neurons. For an N-city *TSP*, there are $N \times N$ fully connected neurons in the network. The row index for a neuron represents the city. The column index represents the order of the city in the tour. The weights of the connecting links are determined according to the constraints and the cost function. *TSP* includes the following constraints: The salesman must visit each city exactly once in a tour (each row has exactly one "on" neuron) and must visit exactly one city at any time in a tour (each column has exactly one "on" neuron). The cost function is constructed to reflect the objective that the salesman chooses a valid tour with minimum tour length. Therefore, the Hopfield network is configured to have a better chance to find short tours.

The constraints and the cost function for *TSP* can be represented by an energy function. The energy function is then used to determine the values of all weights in the network. Hopfield's original energy function for an N-city *TSP* is given by [1]:

$$E = \frac{A}{2}\sum_{X=1}^{N}\sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} V_{Xi}V_{Xj} + \frac{B}{2}\sum_{i=1}^{N}\sum_{X=1}^{N}\sum_{Y=1, Y\neq X}^{N} V_{Xi}V_{Yi}$$
$$+\frac{C}{2}(\sum_{X=1}^{N}\sum_{i=1}^{N} V_{Xi} - N_0)^2 + \frac{D}{2}\sum_{X=1}^{N}\sum_{i=1}^{N}\sum_{Y=1, Y\neq X}^{N} d_{XY}V_{Xi}(V_{Y,i+1} + V_{Y,i-1}) \qquad (1)$$

where X and Y are row indices; i and j are column indices; V_{Xi} is the activation for each neuron; and d_{XY} is a measure of the distance between cities X and Y. The first two terms enforce the constraint that no city can be visited more than once. The third term reflects the constraint that each city should be visited. The last term represents the distance cost function. The value of each parameter (A, B, C, and D) is the measure of the importance of the corresponding term.

The weight of the connection matrix between neuron (X,i) and (Y,j), which is derived from the energy equation (1), is given by:

$$W_{Xi,Yj} = -A\delta_{XY}(1-\delta_{ij}) - B\delta_{ij}(1-\delta_{XY})$$
$$-C - Dd_{XY}(\delta_{j,i+1} + \delta_{j,i-1}) \tag{2}$$

Each neuron (X,i) is connected to an external input current:

$$I_{Xi} = CN_0 \tag{3}$$

There is an input value U_{Xi} and an activation (output) value V_{Xi} for each neuron (X,i). The initial values of U_{Xi} for all neurons are first set to be a small constant value which is determined by the condition: $\sum_{X=1}^{N}\sum_{i=1}^{N} V_{Xi} = N$. To break the symmetry of the network, a small fraction of random noise is added to the initial values of each U_{Xi}. The network is then relaxed until an equilibrium is reached.

At each iteration, each neuron (X,i) updates its input value U_{Xi} based on the weighted activations of all other neurons and its own value at the last iteration. Specifically, the value $U_{Xi}^{(n+1)}$ at iteration step *(n+1)* is given by:

$$U_{Xi}^{(n+1)} = U_{Xi}^{(n)} + \Delta U_{Xi} \tag{4}$$

where $U_{Xi}^{(n)}$ is the input value at iteration step *(n)*. The value of ΔU_{Xi} is given by the following equation:

$$\Delta U_{Xi} = (-\frac{U_{Xi}}{\tau} + \sum_{Y=1}^{N}\sum_{j=1}^{N} W_{Xi,Yj}V_{Yj} + I_{Xi})^{(n)}\Delta t \tag{5}$$

where $\tau(= RC)$ is the time constant of an RC circuit and was set to be 1.0 by Hopfield and Tank[1].

The activation $V_{Xi}^{(n+1)}$ at iteration step *(n+1)* is then determined by $U_{Xi}^{(n+1)}$

through an *activation (output) function.* In the Hopfield network, the activation function is the *sigmoid* function given by:

$$V_{Xi}^{(n+1)} = \frac{1}{2}(1 + tanh(\frac{U_{Xi}^{(n+1)}}{u_0})) \tag{6}$$

where u_0 is the amplification parameter that reflects the steepness of the activation function.

Hopfield and Tank [1] showed that the network is guaranteed to converge to a local minimum in the case of symmetric connecting weights. They tested this model for a 10-city *TSP*. For a total of 20 runs, they claimed 16 valid tours, and half of the valid tours were optimal.

New Relaxation Procedure

In this work, an improved relaxation procedure is proposed. Instead of using formula (2)-(6) to update the input and activation values of neurons, the new procedure uses different updating steps.

For the convenience of presenting the new procedure, Eq. (5) can be reformulated as:

$$\Delta U_{Xi} = (-\frac{U_{Xi}}{\tau} + Net_{Xi})^{(n)} \Delta t \tag{7}$$

where Net_{Xi} is the *net input value* received by the neuron (X, i) from other neurons, which can be written as:

$$\begin{aligned} Net_{Xi} &= \sum_{Y=1}^{N} \sum_{j=1}^{N} W_{Xi,Yj} V_{Yj} + I_{Xi} \\ &= -A \sum_{j=1, j \neq i}^{N} V_{Xj} - B \sum_{Y=1, Y \neq X}^{N} V_{Yi} \\ -C(\sum_{Y=1}^{N} \sum_{j=1}^{N} &V_{Yj} - N_0) - D \sum_{Y=1, Y \neq X}^{N} d_{XY}(V_{Y,i+1} + V_{Y,i-1}) \end{aligned} \tag{8}$$

In the new procedure, the activation value $V_{Xi}^{(n+1)}$ for neuron (X, i) at iteration $(n+1)$ is updated according to:

$$V_{Xi}^{(n+1)} = V_{Xi}^{(n)} + R(T_{Xi} - V_{Xi}^{(n)}) \tag{9}$$

where T_{Xi} is the *target value* for the activation V_{Xi} and $R(0.0 < R < 1.0)$ is a *relaxation rate*. T_{Xi} is derived from Net_{Xi} by using the *sigmoid* function:

$$T_{Xi} = \frac{1}{2}(1 + tanh(\frac{Net_{Xi}}{u_0'})) \tag{10}$$

where u_0' is the amplification parameter. The role of u_0' is similar to that of u_0 in Eq. (6), but its value has a different range.

According to the updating rule specified by Eq. (9), if $V_{Xi}^{(n)} > T_{Xi}$, then $V_{Xi}^{(n+1)} < V_{Xi}^{(n)}$; similarly if $V_{Xi}^{(n)} < T_{Xi}$, then $V_{Xi}^{(n+1)} > V_{Xi}^{(n)}$. Thus after updating, the activation value is getting closer to the target value. The relaxation rate R determines how much V_{Xi} should change given the difference between the target value and the current activation value. The role of the relaxation rate here is to control the step size of updating V_{Xi} for each iteration. The value of R should be set to be a small number (< 1.0); otherwise, V_{Xi} may oscillate and cause instability of the network.

There are several differences between the new and the original procedure. There is a parameter Δt in Eq. (5) but not in the new procedure. Δt and the amplification parameter u_0 in Eq. (6) are combined to the parameter u_0' in Eq. (10). The value of Δt is in the range about 10^{-5}. Thus the value of u_0' has a much larger scale than that of u_0. The value of u_0' should be chosen so that the *sigmoid* function (10) is in a sensitive range (from -10.0 to 10.0) for typical values of Net_{Xi}. The setting for the value of u_0' will be further discussed in the next section. The time constant τ in Eq. (5) is not included in the new procedure. The reason is that its effect is negligible compared with the other terms. Its value (=1.0) is much smaller than the coefficients A, B, C, and D (in the order of 10^2).

The original procedure first updates the input by Eq. (4), and then updates the activation through the *sigmoid* function given by Eq. (6). The input and the activation values are both important for specifying a neuron in the original procedure. The new procedure, however, directly updates the activation using Eq. (9) without updating of the input value as an intermediate step. The input value does not appear in any formula and thus is unnecessary in the new procedure. The relaxation rate in the new procedure has no direct analog in the original procedure. The objective of using a relaxation rate is to gain better control over the relaxation of the network.

Simulation Results

We evaluate the performance of the new relaxation procedure through simulations, and compare it with that of the original relaxation procedure. The evaluation in our experiment is based on 200 randomly generated 10-city *TSP* data sets, including wide varieties of city distributions.

Many previous studies used only one data set (the one used by Hopfield and Tank in their original paper [1]) or a small number of data sets (for example, 10 data sets in [2]) in their simulations. This may result in an unreliable conclusion when comparing two algorithms. The reason is that the performance of an algorithm often depends on the location distribution of the cities in a data set, and different algorithms may favor different types of distributions. Using a large number of data sets can reduce this effect and obtain a better evaluation of the algorithms.

In the simulation, 100 runs are conducted for each of the 200 data sets. For each of 100 runs, different random noise is added to the initial input values of the neurons.

For a fixed set of parameters (R, u_0', etc), the quantities to be evaluated (error rate, percentage of valid tours) is first averaged over 100 runs for each data set, and then averaged over the entire 200 data sets. Thus 20000 runs are needed to obtain each data point that is shown in the figures of the simulation results.

We experimented with different sets of 20000 runs for a fixed set of parameters. The result shows that the estimated quantities are fairly stable. Their values vary within a range of about 1% among different sets of 20000 runs. This demonstrates that the number of runs in our simulation is large enough to make a reasonable estimation of the evaluated quantities.

The original energy function of the Hopfield network is used in the simulation, and the parameters in the energy function are those used by Hopfield and Tank [1]:

$$A = B = D = 500, \quad C = 200, \quad N_0 = 15 \tag{11}$$

The fraction of random noise in the initial values of neurons is set to be 0.001 in the simulation. We tried several different values for the fraction in the range from 0.0001 to 0.01. The performance of the network is only slightly sensitive to this parameter, and the choice of 0.001 works relatively better than others.

The parameter u_0' in Eq. (10) reflects the steepness of the *sigmoid* function: smaller u_0' leads to a steeper slope around the center of the curve. The value of u_0' should be in such a range so that the *sigmoid* function is sensitive to the value of Net_{Xi}. For the parameters given by (11), the value of Net_{Xi} given by (8) is approximately in the range from 10 to 1000. We experimented with the following values of u_0': 50, 100, 150, 200, 300, 400 and 500. For each of them, the simulations with different values of relaxation rate R are conducted to evaluate the error rate and the percentage of valid tours. We found that the network has best overall performance when u_0' is 150. The performance drops slightly when u_0' is equal to 100 or 200. But when u_0' further decreases (< 100) or increases (> 200), the performance starts to drop rapidly. For the values of 400 and 500, no valid tour can be found by the network. For all the simulation results reported in the following, the value of u_0' is fixed at 150.

Fig. 1 shows the percentage of valid tours with different relaxation rates using the new procedure and compares them to that using the original procedure.

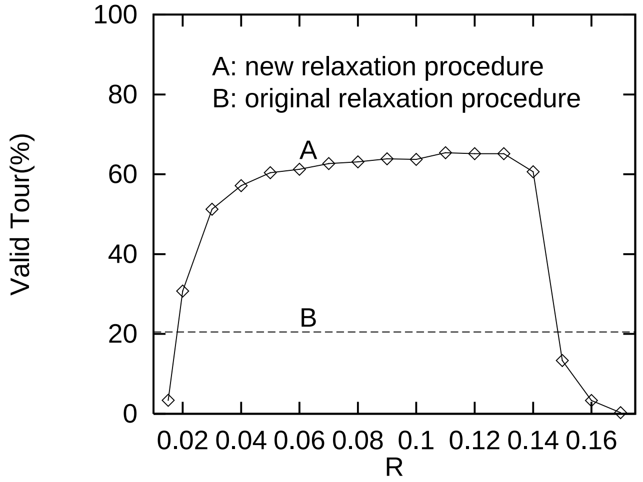

Figure 1: The percentage of valid tours of different relaxation rates using the new relaxation procedure compared to the original relaxation procedure.

Each data point for the percentage of valid tours shown in Fig. 1 is the weighted average over $N_{dataset}$ ($= 200$) data sets:

$$Valid = \frac{\sum_{i=1}^{N_{dataset}} Valid_i}{N_{dataset}} \tag{12}$$

where $Valid_i$ is the percentage of valid tours for data set i and is defined in the following.

For each data set i, there are total of $N_{total,i}$ ($= 100$) runs with different initial input values The maximum number of iterations allowed for each run is set to be 500. If a valid tour can not be reached within 500 iterations, the network will stop and the tour is counted as invalid. If $N_{valid,i}$ is the number of valid tours among the total of $N_{total,i}$ runs, then $Valid_i$ is defined by:

$$Valid_i = \frac{N_{valid,i}}{N_{total,i}} \tag{13}$$

The result in Fig. 1 shows that the new procedure significantly improves the percentage of valid tours over the original procedure when the relaxation rate R is in the range from 0.03 to 0.14.

Another important observation from Fig. 1 is that the percentage of valid tours is sensitive to the relaxation rate in this new procedure: it significantly drops (goes to zero beyond the range of the figure) when the relaxation rate is either too small (< 0.03) or too large (> 0.14), but its value is relatively stable in between.

Fig. 2 shows the errors of the solutions obtained by the network using the new relaxation procedure with different relaxation rates compared to that obtained by using the original relaxation procedure.

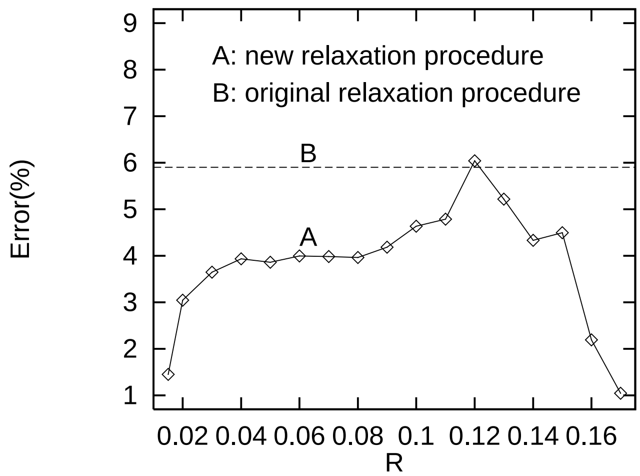

Figure 2: The errors (%) of different relaxation rates using the new relaxation procedure compared to the original relaxation procedure.

Each data point shown in Fig. 2 is averaged over different runs for each data set and then averaged over 200 different data sets as shown in the following.

For the data set i, the error of a valid tour j is defined by:

$$Err_{i,j} = \frac{d_{i,j} - d_{i,opt}}{d_{i,opt}} \tag{14}$$

where $d_{i,j}$ is the tour length of tour j and $d_{i,opt}$ is the optimal (shortest) tour length of data set i.

The error for data set i is the averaged error over all valid tours:

$$Err_i = \frac{\sum_{j=1}^{N_{valid,i}} Err_{i,j}}{N_{valid,i}} \tag{15}$$

The error shown in Fig. 1 is the average error of valid tours in all data sets and is weighted by the percentage of valid tours for each data set:

$$Err = \frac{\sum_{i=1}^{N_{dataset}} (Valid_i Err_i)}{\sum_{i=1}^{N_{dataset}} Valid_i} \tag{16}$$

Fig. 2 has a shape similar to Fig. 1: relatively flat in the middle and rapidly dropping at the ends. But it should be noted that we are seeking minimum points in Fig. 2 while seeking maximum points in Fig. 1.

One observation from Fig. 2 is that the error rate is very small when either $R < 0.2$ or $R > 0.16$. Looking at Fig. 1, however, we can see that the percentages of valid tours are extremely low for these values, and hence they are not good choices for R. In stead, the value of R should be chosen so that it can lead to a desired balance of performance.

The reason for the dropping error at the ends is as follows. Recall from Fig. 1 that there are very few valid tours discovered at these end points. When R is at these values, the network can only find valid tours for a very small partion of the 200 data sets, and those data sets typically have a very simple topology, with an obvious optimal tour. Since only valid tours are used in the error calculation, the error rates are therefore small at these points.

However, when R is in the middle range, the network can find valid tours for more data sets, and some of them have more complicated city topologies, and therefore it is more difficult to find optimal tours. As a consequence, the average error starts to grow when R is moving from two ends towards the middle, which explains the shape of the middle part in Fig. 2.

The new procedure also significantly improves the network over the original procedure in terms of error rates. As shown in Fig. 2, the error rates of the new procedure are all lower than that of the original procedure, except at one point $R = 0.12$.

To achieve a good overall performance, the value of R should be chosen properly with the consideration of balanced improvement in terms of both the percentage of valid tours and the error rate. For the parameter setting specified in the simulation, the optimal value of R is about 0.05. At this point, the percentage of valid tours (60.4%) is increased by 194.6% compared with that (20.5%) of the original procedure, and the error rate (3.86%) is dropped by 34.6% compared with that (5.90%) of the original procedure,

The results in Fig. 1 and Fig. 2 suggest that the network must be relaxed at an appropriate rate in order to have a good performance. Otherwise, if it is being relaxed too fast or too slow, the performance will degrade.

Why does the new procedure perform significantly better than the original procedure? What are the possible reasons for the differences in the performance between these two procedures ? We compare and analyze the details of the two procedures and propose the following explanation for the difference in performance.

The main reason attributed to this difference is that these two procedures have different variation ranges for the activation value V_{Xi} between two consecutive iterations. The magnitude of the variation for V_{Xi} between iteration (n)

and $(n+1)$ is:

$$|\Delta V_{Xi}| = |V_{Xi}^{(n+1)} - V_{Xi}^{(n)}| \tag{17}$$

In the original procedure, $V_{Xi}^{(n+1)}$ is the result of the *sigmoid* function of $U_{Xi}^{(n+1)}$ given by Eq.(6). The range of the *sigmoid* function is $(0.0, 1.0)$. $|\Delta V_{Xi}|$ is the difference of two *sigmoid* functions, and thus its possible range is also $(0.0, 1.0)$. When there is a significant difference between $U_{Xi}^{(n)}$ and $U_{Xi}^{(n+1)}$, this difference will be amplified by the *sigmoid* functions and causes a large value of $|\Delta V_{Xi}|$.

Such a large variation range for $|\Delta V_{Xi}|$ leads to a fast relaxation of the network. If the network is relaxed too fast, there will be less opportunity for exchange of information between neurons, and the solution formed under this condition is less likely to have high quality. Another potential problem with fast relaxation is possible instability and oscillation of the network.

In contrast, $|\Delta V_{Xi}|$ in the new relaxation procedure is given by $|R(T_{Xi} - V_{Xi}^{(n)})|$ according to Eq.(9). T_{Xi} is the *sigmoid* function shown in Eq.(10) and thus has a range of $(0.0, 1.0)$. V_{Xi} is initialized to be around 0.1, and its value will still be around the range from 0.0 and 1.0 after subsequent updates as long as the value of R is in the range from 0.0 to 1.0. Thus $|(T_{Xi} - V_{Xi}^{(n)})|$ is about in the range of $(0.0, 1.0)$, and the variation range for $|\Delta V_{Xi}|$ $(= |R(T_{Xi} - V_{Xi}^{(n)})|)$ is about $(0.0, R)$.

For a typical value of R chosen in our simulation, for instance, $R = 0.05$, $|\Delta V_{Xi}|$ will be in the range $(0.0, 0.05)$, which is much smaller than $(0.0, 1.0)$ in the original procedure. A smaller variation range for $|\Delta V_{Xi}|$ should cause a smoother relaxation process and result in better performance for the network. This is the main reason for the improved performance by the new relaxation relaxation.

Another issue is to explain the simulation result that the performance significantly decreases as the relaxation rate becomes very small. According to the above analysis, it seems that the performance should increase as the relaxation rate becomes smaller and the relaxation process becomes smoother.

One explanation for this effect is that the quality of a solution can be affected by the initial random noise. The initial value of each neuron must be set to be slightly different from each other. Otherwise, if they are all same, the input values of all neurons will always stay constant, making it impossible to reach a solution. So a small fraction of noise is added to the initial input of every neuron in order to break the symmetry of the network.

However, the noise fraction has a random nature and thus has negative effects on the formation of a solution. In the early phase, all neurons have similar input values, and noise thus has a relatively large influence on the formation of the early prototype of a solution. After a certain number of iterations, each neuron has received more information from others and the effects of the noise becomes relatively small.

If the network is relaxed too slowly, the effects of noise will continue in the neurons for a relatively longer period and compete with the signals of cost and constraint terms, resulting in a solution with poor quality.

Summary

The relaxation process in the Hopfield network is studied and a new relaxation procedure is proposed in this paper. The new procedure updates the activation of a neuron directly without using the updating of the input value as the intermediate step. The new procedure treats the amplified signal received by a neuron from all other neurons as the target value for the activation. The activation is updated based on the difference between the current activation and the received target value, with a relaxation rate controlling the updating scale. The performance of the new procedure is evaluated through simulations. The results of the simulations based on 200 randomly generated instances of the 10-city *TSP* show significant improvement in the performance of the network by using the new procedure. Compared with the original procedure, the new procedure reduces the error rate by 34.6% and increases the percentage of valid tours by 194.6%. The reasons for the differences in the performance of the two procedure are analyzed and two factors are proposed to explain the differences in the performance. One factor is that the new procedure has a smaller variation range than the original procedure for the activation value between two consecutive iterations. The other factor is that the new procedure only amplifies the relevant component representing the cost and constraint terms while the original procedure amplifies both the relevant and irrelevant components.

In future work we plan to extend the *constant* relaxation rate to several forms of *dynamic* relaxation rate which is allowed to vary at different phases of relaxation process. We also plan to evaluate the performance of the new relaxation procedure on applications other than *TSP*.

References

[1] Hopfield, J. J. and Tank, D. W. (1985), Neural Computations of Decisions in Optimization Problems. *Biological Cybernetics*, 52, pp. 141-152.

[2] Wilson, G. V. and Pawley, G. S. (1988), On the Stability of the Traveling Salesman Problem Algorithm of Hopfield and Tank. *Biological Cybernetics*, Vol. 58, pp. 63-70.

[3] Brandt, R. D., Wang, Y., Laub, A. J. and Mitra, S. K. (1988), Alternative Networks for Solving the Traveling Salesman Problem and the List-Matching Problem. *Proceedings of IEEE International Conference on Neu-*

ral Networks, San Diego, CA. II: 333-340.

[4] Aiyer, S. V. B., Niranjan, M. and Fallside, F. (1990), A Theoretical Investigation into the Performance of the Hopfield Model. *IEEE Transactions on Neural Networks*, Vol. 1, No. 2, pp. 204-215.

[5] Protzel, P. W., Palumbo, D. L. and Arras, M. K. (1993), Performance and Fault-Tolerance of Neural Networks for Optimization, *IEEE Transactions on Neural Networks*, Vol. 4, pp. 600-614.

[6] Li, S. Z. (1996), Improving Convergence and Solution Quality of Hopfield-Type Neural Networks with Augmented Lagrange Multipliers. *IEEE Transactions On Neural Networks*, Vol. 7, No. 6, pp. 1507-1516.

[7] Catania, V., Cavalieri, S. and Russo, M. (1996), Tuning Hopfield Neural Network by a Fuzzy Approach. *Proceedings of IEEE International Conference on Neural Networks*, pp. 1067-1072.

[8] Cooper, B. S. (1995), Higher Order Neural Networks-Can they help us Optimise? in *Proceedings of the Sixth Australian Conference on Neural Networks (ACNN'95)*, pp. 29-32.

[9] Van den Bout, D. E. and Miller, T. K. (1989), Improving the Performance of the Hopfield-Tank Neural Network Through Normalization and Annealing. *Biological Cybernetics*, Vol. 62, pp. 129-139.