A Parallel Algorithm of Boltzmann machine with Rejectionless Method

H. ZHU[*a], K. EGUCHI[*b], H. KAI[*a], N. SUN[*b], T. TABATA[*b]
[*a] Department of Computer Science, Faculty of Informatics,
Hiroshima Kokusai Gakuin University,
6-20-1 Nakano Aki-ku, Hiroshima 739-0321 JAPAN
Phone : +81-82-820-2649, Fax : +81-82-820-2640
E-mail : kohe@wuchang.cs.hkg.ac.jp
[*b] Kumamoto National College of Technology
2659-2, Suya, Nishigoshi, Kikuchi, Kumamoto 861-1102, JAPAN

Abstract: - Parallelization is of great advantage to speed-up of calculation. In this paper, the proposition of parallelizing the rejectionless method [1] is presented. While for this parallelization the efficiency was confirmed with an experiment of TSP and the relationship with the number of cities was analyzed.

Key-Word: - Boltzmann machine, Hopfied network, simulated annealing, rejectionless method

1 Introduction

The Boltzmann machine – introduced by Geoffrey Hinton, David Ackley and Terrence Sejnowski (1984) - is a kind of neural network which is called as a discrete time Hopfied network or as a stochastic network in which the neuron state is binary and decided by a propabability. It can rigorously established that the Boltzmann machine will probabilitically converge to a global optimum point. The Boltzmann machine can be quite useful for solving certain type of combinatorial problems [2, 3, 4]. And the Boltzmann machine can be used in the another fields of knowledge representation and learing [5], etc.

The huge runtime requirements of the Boltzmann machine have limited its applications in real cases. Long runtime is mainly due to the annealing task performed by the network. Annealing is the physical process of heating up a solid until it melts, followed by cooling it down until it crystallizes into a state with a perfect lattice. During this process, the free energy of the solid is minimized. Practice shows that the cooling must be done carefully in order not to get trapped in locally optimal lattice structures with crystal imperfections.

With the temperature of the simulated annealing lowered, the network selects randomly neuron and makes if it change its state or not in the light of a propabability. As seen above-mentioned, this processing must go on slowly to make the final stable configuration of the Boltzmann machine on the global configuration or nearest to it. Therefore, it is the important investigating topic for enhancing the processing speed. Some papers about it have been reported up to now [6, 7, 8, 9, 1]. In the Reference [1], we proposed a novel method – the rejectionless method, to resolve the problem of the long runtime requirement.

The rejectionless method is to keep a list of the effects of each possible state movement on the energy function and use this information to bias the selection of moves. A important characteristic of the rejectionless is that its speed does not depend on the acceptance ratio, which result the efficiency of speed-up of the Boltzmann machine. But, the rejectionless method required a great of calculational volume. Here, we presents a algorithm to parallelize the rejectionless method. With the parallel algorithm, the calculational volume was decreased greatly, and the efficiency of speed-up and the much nearest to the global optimum point than the standard Boltzmann machine have been confirmed with an experiment of well-known combinatorial optimization problem–TSP (Traveling Salesman Problem). As the analysis of the parallelization and the number of cities, we have presented a limit of efficiency of the parallelization.

This paper is organized as follows. In Section 2, the problem of long running time at the low temperature will be shown while an review of the Boltzmann machine will be described. Section 3 will give a description of the rejectionless method and Section 4 gives an analysis of calculational volume for the unparallelized and parallelized algorithm to explain the reason of parallelizing the rejectionless algorithm. Section 5 shows the parameters, conditions and results of some TSP's experiments. On Section 6, one analysis of efficiency of the parallelization for the number of cities is presented. In Section 7 we will conclude with a discussion and some recommendations.

2 Boltzmann machine

We assume that the Boltzmann machine consists of a number N of neurons. The neurons' state is associated with binary value "`0`" or "`1`", corresponding to "`off`" or "`on`", respectively. The i-th neuron's state x_i is depend on its activity u_i and is decided as follows:

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

$$P(x_i = 0) = 1 - P(x_i = 1) = \frac{\exp(-u_i/T)}{1 + \exp(-u_i/T)} \tag{2}$$

where, $P(x_i = 1)$ and $P(x_i = 0)$ are the probability that the i-th neuron's state x_i take "`1`" and "`0`", respectively. T is a parameter called as the temperature, is from high to low to simulate the annealing process and affects the quantity of probability $P(x_i = 1)$ and $P(x_i = 0)$. At arbitrary T, the procedure of the Boltzmann machine is as follows:

Procedure-1

Boltzmann_machine
begin
 t:=0; { t:iteration index }
 for i:=1 to N do $x_i(t)$:=initial; { initialize }
 while ($t <= L$) do
  begin
   Select k-th neuron from N neurons randomly;
   Evaluate k-th neuron on Eq.(1);
   if fi rethen $x_k(t)$:=1 else $x_k(t)$:=0; {update}
   for i:=1 to N do $u_i(t) := \sum_{j=1}^{N} W_{ij} x_j(t)$;
   t:=t+1
  end
end

W_{ij} is the weight connected from the j-th neuron to the i-th neuron, and determines the connective strength of them. L is the iterative times at arbitrary temperature T.

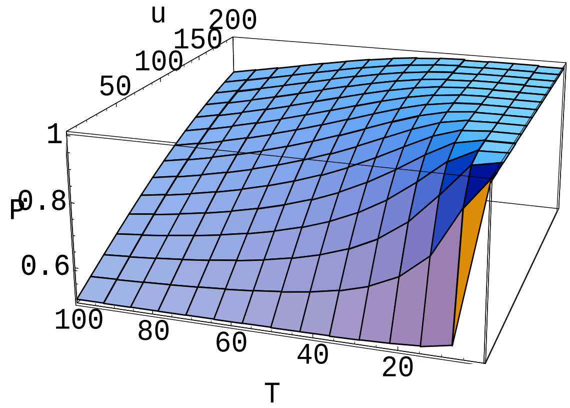

Fig.1 Diagram of the Eq.(1)

With the *Mathematica*, the Eq.(1) was analyzed, which the result of is shown at Fig.1. From Fig.1, it is clear that the probability of neuron's state taking "`1`" is decreasing with the temperature falling. The decrease is very hard at the low temperature. That is the basilical cause why the requirement of huge running time of the Boltzmann machine. So, a efficient way of process speed-up of the Boltzmann machine is to increase the probability of neuron's state taking "`1`" at the low temperature.

3 Rejectionless method

For improving the speed of the Boltzmann machine at low temperatures, a novel method, called the rejectionless method, was introduced [1]. The method is to keep a list of the probabilities of each neuron's state movement on the energy function and use this information to bias the selection of neurons' moves. The procedure of the rejectionless method is described as follows.

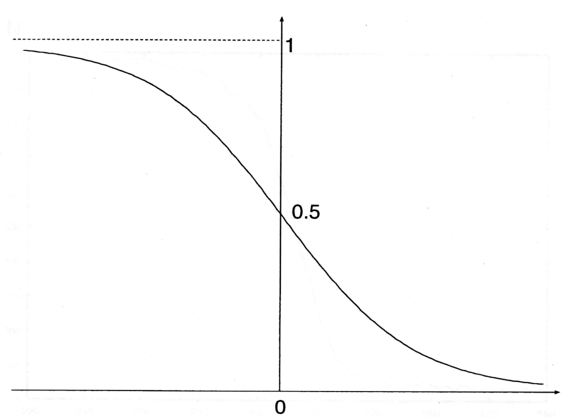

Fig.2 Diagram of the function $\frac{1}{1+exp(\frac{\Delta_i}{T})}$

For each possible move i, $N \geq i \geq 1$, store a value d_i

$$d_i = -\frac{1}{1 + exp(\frac{\Delta_i}{T})} \log(\zeta) \quad (3)$$

where Δ_i is the change in the cost function that would result from the move $x_i \Rightarrow \acute{x_i}$. The general energy function of Boltzmann machine is[10]

$$E(x_i) = \sum_{i,j=1}^{N} W_{ij} x_i x_j = \sum_{i=1}^{N} u_i x_i \quad (4)$$

where, the activity u_i is

$$u_i = \sum_{j=1}^{N} W_{ij} x_j \quad (5)$$

So,

$$\Delta_i = E(\acute{x_i}) - E(x_i) = (2x_i - 1)u_i = \\ = \begin{cases} u_i & \text{if } x_i = 1 \\ -u_i & \text{if } x_i = 0 \end{cases} \quad (6)$$

and ζ is a uniform random of [0,1). Fig.2 shows a diagram of the function $\frac{1}{1+exp(\frac{\Delta_i}{T})}$ which is a sigmoid function. Select move i with probability

$$\frac{d_i}{\sum_{j=1}^{N} d_j} \quad (7)$$

Then make the move, and update the d_i value of all neurons accordingly. The sequence of states generated by the rejectionless method is probabilitically equivalent to the sequence of states generated by the Boltzmann machine[1].

An algorithm of implementation of the rejectionless method on computer is

(1) A number $\zeta \in [0, 1)$ is generated.

(2) The integer i satisfied with

$$d_i = max\{\sum_{j=1}^{N} d_j\} \quad (8)$$

is the number of the i-th neuron state movement

(3) An updating d_i, $i = 1, 2, \ldots N$

4 Parallelization

From the above mentioned section, it is clear that a volume of calculation is $O(n^2)$ for sorting the $\frac{n^2-n}{2}$ data to select one neuron's movement, where n is the number of cities. With the increase of n, the calculational volume gets larger and larger.

A Parallelization of the rejectionless method is that the neuron network consists of individual units (neurons) which themselves calculate Δ_i and d_i independently. So, the neuron network does only the detection of maximum from all candidates of state movement, which makes the calculational volume decreased into $O(\log(n))$.

5 Experiment

TSP (Traveling Salesman Problem) is a well-known combinatorial optimization problem. With TSP, the availability of the rejectionless method was confirmed.

In the experiment, the number of cities is 100, the initial temperature is 1,000, the iterations of calculation is 60,000 and the temperature drops off with a magnification of 0.99 in 1,000 iterations.

For the experiment, at hight temperatures the standard Boltzmann machine is used; when the ratio of neurons' state movement reached the crossover point, the rejectionless method was employed. The crossover point was selected at the point of 20%. Fig.3 shows the relationship between temperature T and movement's number L, where the crossover point is at $T = 696$.

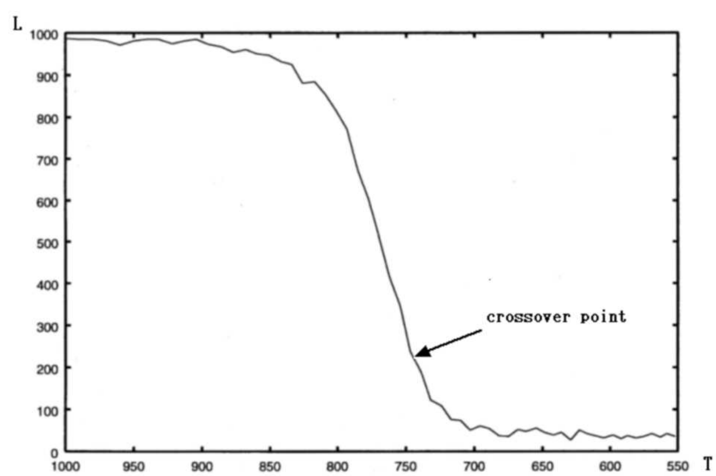

Fig.3 Movement's ratio as a function of temperature

The final movement's ratio was 10%, for which the rejectionless method runs at five times the speed of the standard Boltzmann machine.

Our experiment was finished on one computer with an Alpha cpu (600MHz) and Linux OS. We employed the processer's command "exec", "fork" "wait" and "exit" to produce sub-processers playing a role of individual units (neurons) to implement the parallelization of the rejectionless method.

Fig.4 and Fig.5 show the experimental results of using the standard Boltzmann machine and using both the standard Boltzmann machine and the rejectionless method respectively. From them, it is known that the result of the latter is much more close to minimum than one of the former.

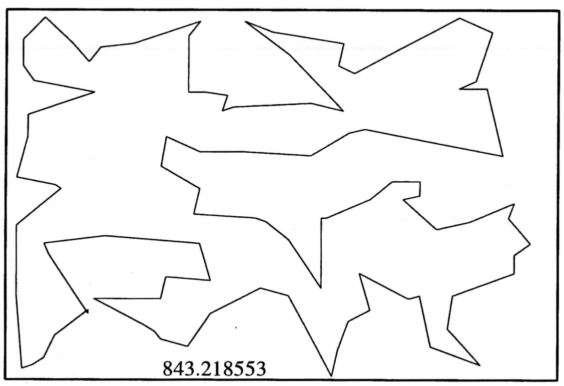

Fig.4 Result of TSP without parallelization

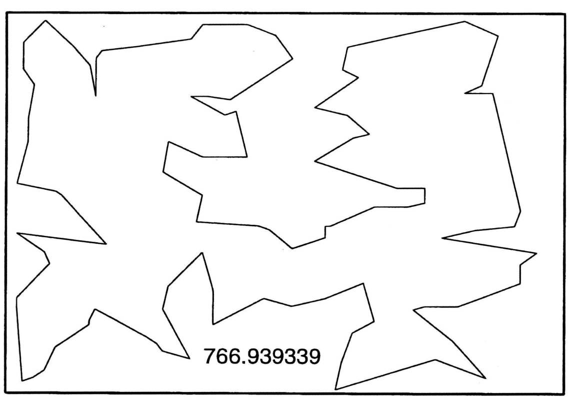

Fig.5 Result of TSP with parallelization

6 Analysis

Here, we give an analysis of the parallelization and the number of cities. Assume that in the standard Boltzmann machine and the parallelized rejectionless method the required time of one neuron's state movement is a and for the parallelized rejectionless method the the the required time of comparison between two candidates moving their state is b. The sum of calculational time for the the parallelized rejectionless method is as

$$\begin{aligned} & a + b \cdot \log_2 \frac{n^2 - n}{2} \\ & = a + b \cdot \{\log_2(n^2 - n) - \log_2 2\} \end{aligned} \tag{9}$$

As generally $n \gg 1$, we can omit $\log_2 2$ and take

$$\log_2(n^2 - n) \simeq \log_2(n^2)$$

So, Equ.(9) is

$$\begin{aligned} & \simeq a + b \cdot \log_2(n^2) \\ & = a + 2b \cdot \log_2 n \end{aligned} \tag{10}$$

By our experiment of TSP using the standard Boltzmann machine, the ratio of state movement is 3%, that is one movement of 31 trials. The calculational time takes $31a$, $a > b$. Therefore, we can get

$$\begin{aligned} \log_2(n) &< 15 \\ n &< 2^{15} \end{aligned}$$

It follows that the efficiency of the parallelized rejectionless method for speed-up is till the number of cities is 2^{15}. Our experiments presented that the ratio of state movement is 50% for the case of 10 cities and 8% for 40 cities. So, when the number of cities is over 40 the introduction of parallelization is of great advantage to the speed-up.

7 Conclusions

A parallelization of the rejectionless method ,which make all units of network calculate their changes in the cost function Δ_i and probabilities of state movement d_i, $i = 1, 2, \ldots N$, is proposed. By the parallelization, the calculational volume was decreased greatly, which was confirmed with an experiment of TSP. And from an analysis of the parallelization and the number of cities, the efficiency of the parallelized rejectionless method under 2^{15} cities was presented. The further work is to make an experiment on the parallelized rejectionless method with parallel computers and implement the hardware.