IEICE Proceeding Series

A Modular Neural Network for Parallel Computation

Yoshihiro Hayakawa, Daisuke Sasaki, Koji Nakajima

Vol. 1 pp. 723-726
Publication Date: 2014/03/17
Online ISSN: 2188-5079

2012 International Symposium on Nonlinear Theory and its Applications
NOLTA2012, Palma, Majorca, Spain, October 22-26, 2012

A Modular Neural Network for Parallel Computation

Yoshihiro Hayakawa†, Daisuke Sasaki‡ and Koji Nakajima‡

†Sendai National College of Technology
4-16-1 AyashiChuoh, Aoba-ku, Sendai-shi, 989-3128, Japan
‡The Laboratory for Brainware/Laboratory for Nanoelectronics and Spintronics,
Research Institute of Electrical Communication, Tohoku University,
2-1-1 Katahira, Aoba-ku, Sendai-shi, 980-8577, Japan
Email: hayakawa@sendai-nct.ac.jp, sasaki@nakajia.riec.tohoku.ac.jp, hello@riec.tohoku.ac.jp

Abstract—Some of combinatorial optimization problems cause exponential increases of calculation time in terms of a problem size. In case of using a neural network solver, faster calculation is prospective because each neuron is updated in essentially parallel. In fact, however, a neural network solver is performed by using the ordinary differential equation in a computer, so that a calculation speed is not enough yet. Hence, in this paper we propose a modular neural network for implementing parallel computation with a MPI technique. Each module is assigned to each processor and these are calculated in parallel in our proposed modular neural network and we discuss the possibility of improvement of calculation by decreasing communication frequency between processors.

1. Introduction

The combinatorial optimization problems such as the Traveling Salesman Problems (TSPs) or Quartic Assignment Problems (QAPs) include a lot of applications, for example, the packet routing , delivery planning, etc. However, it is not easy to obtain useful solutions of these problems in real time by using conventional computers. Neural networks having a energy function, which are derived from symmetric synapse connection, can be used to solve combinational optimization problems and are an object of research to aim a fast solver.

As one of the solvers of these problems, we have proposed the Inverse function Delayed (ID) network [1]. The ID network is a neural network that has a negative resistance effect in its dynamics, and the output space of the network has a negative resistance region. The network state is updated along the gradient of the quadratic form energy function if the state is outside the region; otherwise the state is not updated along the gradient of the energy function. Hence we can destabilize the undesirable states through appropriate setting of the negative resistance. Especially, N-Queen problems or 4-Color problems are able to be solved with 100% success rate by using the ID network [1]. Additionally, for solving TSPs or QAPs, a Higher-order Connection ID (HC-ID) network has been proposed and we have also demonstrated solver having the fourth-order energy function for these problems[2]. However, the problem size of these demonstration is very small. Although a dedicated neuro-chip is required to apply to practical problems, we don't have it with enough size and speed yet. Under the present situation, we have to execute a simulation of neural networks by computer systems.

Recently, a message passing interface (MPI)[3] has been developed and it is pretty commonly now to use in the field of a computer simulation. We expected to be able to use this MPI technique when a big size neural network is calculated. However, if we apply a MPI technique to a neural network having full-connected synapse weights, a communication cost increases with increasing neuron size. In this paper, hence, we propose a modular neural network which is a new method reducing a communication cost.

2. A modular neural network

Both Hopfield model and ID model have the same inner potential dynamics. A inner potential of these is described as below,

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

where τ , w_{ij},u_i,x_i and h_i are time constant, synapse weight from $j-th$ neuron to $i-th$ one, inner potential, output and bias of $i-th$ neuron, respectively.

When we do parallel computation by using plural computers (processors), mutual communication is needed. A Message Passing Interface(MPI)[3] is a library of these communication functions, we can easily program for a parallel computation. When we use a MPI, we have to divide neurons in the network into modules equally. For example, let us consider that a neural network is divided by m modules. A neuron set of $\gamma - th$ module is defined as M_γ. In such a case, we rewrite Eq.(1) as a following equation,

$$\begin{aligned} \frac{du_i}{dt} &= \sum_{j \in M_\gamma} w_{ij} x_j + \sum_{j \notin M_\gamma} w_{ij} x_j + h_i & (2) \\ &= \sum_{j \in M_\gamma} w_{ij} x_j + \theta_i^{ex}, & (3) \end{aligned}$$

where a neuron i is a member of M_γ, the first term of Eq.(2) means a calculation inside the same module, the second term is a calculation of inputs from other modules. Here, θ_i^{ex} is defined as a total input from other modules and a bias. Eq.(2) changes to Eq.(3).

The calculation of the first term of Eq.(3) can perform in perfectly parallel as shown in fig.1(A) because each module assigns to each computer. However θ_i^{ex} needs information from other modules, so communications between mutual modules happen as shown in fig.1(B) and hence these process cannot be performed in parallel. Furthermore, much time is required because these communications are done through an Ethernet line or a BUS.

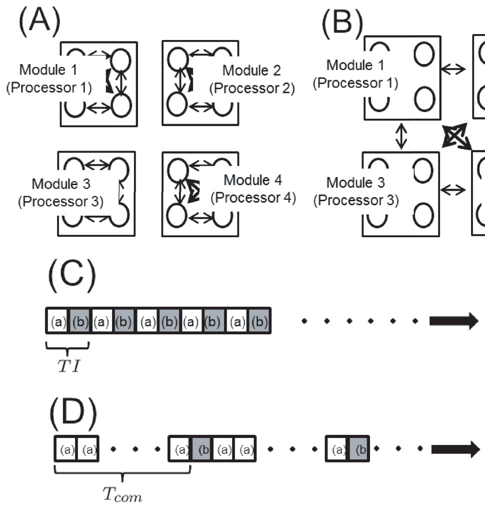

Figure 1: Operation timing. (A) Inside module calculation. These operation can perform in perfectly parallel. (B) θ_i^{ex} calculation. These operation need communications of other modules. (C) MPI normal operation. (D) Modular neural network operation.

In a case of a normal MPI programming we require to update $\theta_i^{ex}$of Eq.(3) every time interval (TI) of simultaneous ordinary differential equations as shown in fig. 1(C), so a communication cost between each module is no small matter. In this paper, hence, we propose a modular neural network that is a new method to reduce this communication cost. The modular neural network updates θ_i^{ex} for a far longer period T_{com} than TI as shown in Fig.1(D). We label T_{com} as a communicating time interval.

3. Minima of an energy function (equilibrium points) and a converging process

Minima of energy function of conventional network are satisfied with $du/dt = 0$ in Eq.(1). Equilibrium points of Eq.(3) are the same ones of Eq.(1). Hence, our proposed modular neural network also has the same minimum positions.

Here, let us consider the converging process of a modular neural network. The converging process depends on a communicating time interval T_{com}. When $T_{com} \ll \tau$ ($T_{com} \approx TI$), the calculation method is practically equivalent to a normal MPI programming. Hence, we expect that its dynamics is also equivalent to a conventional Hopfield model.

An energy function of a conventional Hopfield model is

$$E_{hop} = -\frac{1}{2}\sum_i\sum_j w_{ij}x_ix_j - \sum_i h_ix_i + \sum_i\int^{x_i} f^{-1}(x)dx. \quad (4)$$

In case of $T_{com} \gg \tau$, during a parallel operation as shown in fig.1(A), an energy function E_{mod} of a modular neural network is given by

$$\begin{aligned} E_{mod} &= \sum_\gamma\left\{-\frac{1}{2}\sum_{i\in M_\gamma}\sum_{j\in M_\gamma} w_{ij}x_ix_j - \sum_{i\in M_\gamma}\theta_i^{ex}x_i\right. \\ &\quad + \left.\sum_{i\in M_\gamma}\int^{x_i} f^{-1}(x)dx\right\}, \end{aligned} \quad (5)$$

and its time evolution is given by

$$\frac{dE_{mod}}{dt} = -\sum_\gamma\sum_{i\in M_\gamma}\left(\frac{du_i}{dt}\right)^2\frac{df(u_i)}{du_i} \le 0. \quad (6)$$

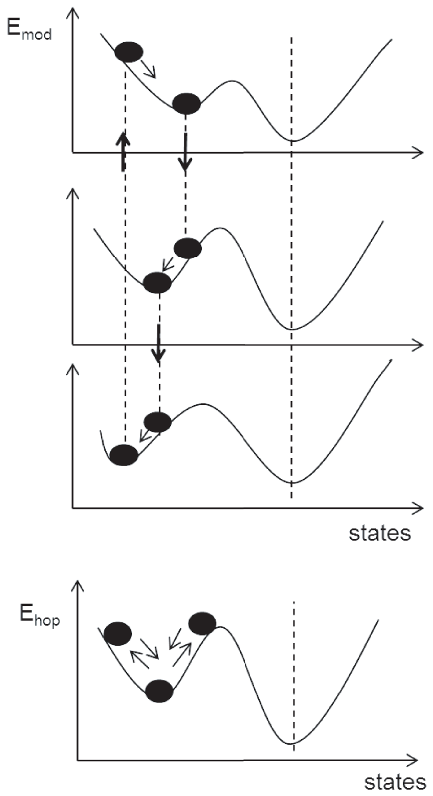

Figure 2: A state transition of modular neural network. ($T_{com} \gg \tau$)

During this period, therefore, E_{mod} also decreases with time and it converges to minimum states because T_{com} is enough long to converge. After that, the mutual communications in fig.1(B) occurs and then the shape of E_{mod} changes dramatically in accordance with changing θ_i^{ex}. As a result, it is easy assume that a periodic behavior occurs as shown in fig 2. We consider a poor performance with $T_{com} \gg \tau$ because it reaches to periodic states instead of solution states. Of course, local or global minimum states of eq.(4) are the same ones of eq.(5), so that these states remain stationary even after the mutual communications.

Hence, Our most interesting situation is $T_{com} \approx \tau$. In this case, we investigate the performance of modular neural networks by using a computer simulation.

4. Simulation

In this paper, we use the n-Queen problem which is one of combinatorial optimization problems to solve by neural networks. The n-Queen problem is the problem of placing n chess queens on an $n \times n$ chessboard so that no two queens attack each other. In case of this problem, we prepare $n \times n$ neurons on chessboard and we interpret that a queen is put on this grid only if an output of neuron is 1.

To apply this problem to our proposed modular neural network, we have to divide this neurons to some modules. In case of a neural network for n-queen problem, we divide column and row into c and r parts, respectively. Fig.3 shows an example of a dividing network into 4 parts ($c = 1$ and $r = 4$).

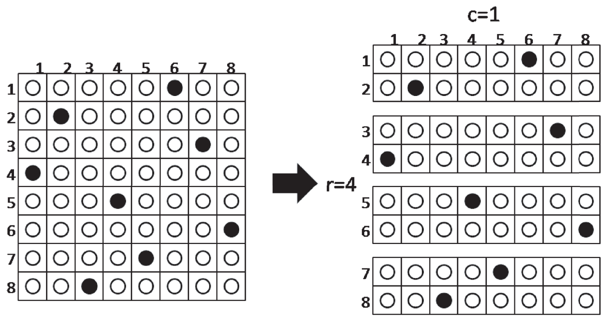

Figure 3: An example of modularizing a network.

4.1. Hopfield type modular neural network

First, we consider to use a conventional Hopfield model and a relation between u and x is given by $x = (\tanh(\beta u) + 1)/2$. These simulations were investigated by 8 machines system (3.4GHz Pentium4, 1GB memory and Gigabit Ethernet). Figure 4 shows results of 8 and 16-Queen problem with $c \times r = 8 \times 1$ and 4×1, respectively. The lines in the graph are success rate of conventional hopfield model. These are 41%($n = 8$) and 29%($n = 16$).

As described above, the success rate of a modular neural network is almost the same one of Hopfield when $T_{com} << \tau$. On the other hand, in case of $T_{com} \approx \tau$ or $T_{com} \leq \tau$, the performance is predictably poor.

However, we can confirmed an increase of success rate when $T_{com} \approx 0.3 \times \tau$.

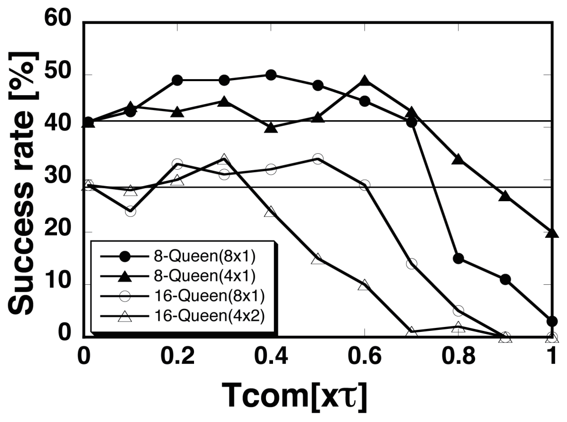

Figure 4: Success rate as a function of T_{com} by using a Hopfield type modular neural network.

Figure 5 shows dependency of calculation speed rate on the number of processors with 10- and 18-Queen problems. Two black filled marks indicate calculation speed rate of a normal MPI programming, for these results, a calculation speed does not increase with increasing processors because the communication cost among processors is big. Two open marks indicate results of modular neural networks with $T_{com} = 0.3 \times \tau$ and $c = 1$, the calculation speed is remarkably improved in comparison with a normal MPI programming.

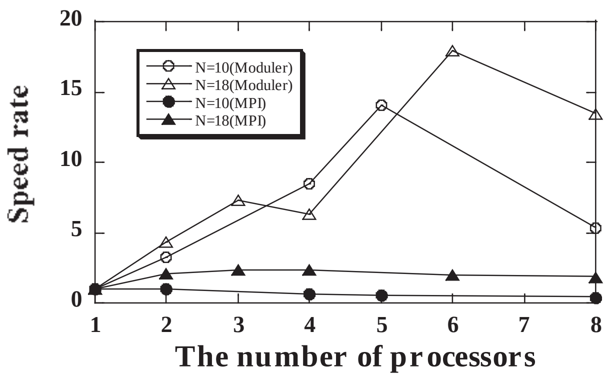

Figure 5: Speed rate as a function of the number of processors by using a Hopfield model.

4.2. ID type modular neural network

We also attempt applying an ID model network to our proposed modular neural network. In case of an ID model network, a relation between u and x is given by a differential equation as following,

$$\tau_x \frac{dx_i}{dt} = u_i - g(x_i), \quad (7)$$

where $g(x) = f^{-1}(x) - \alpha \times (x - 0.5)$ and τ_x is a time constant of output. These simulations were investigated by 4 machines system (2.8GHz PhenomII X6, 8GB memory and Gigabit Ethernet). In this network, α is a control parameter of negative resistance (please see reference [1]). Here, 8-Queen problem is used and we set parameters to achieve 100% success rate in a normal ID network. The modular neural network is modularized by $c = 1$ and $r = 8$.

Figure 6 shows the dependency of success rate on T_{com} with various control parameter α. As results, 100% success rate is kept with an increasing T_{com} differently from a Hopfield type modular neural network. Moreover, the range keeping it becomes wider as α increases.

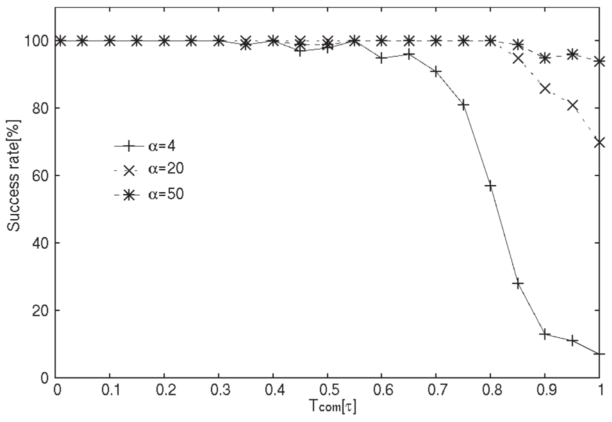

Figure 6: Success rate as a function of T_{com} by using an ID model.

Figure 7 shows the calculation speed rate as a function of the number of processors. These circle marks indicate it by using a normal MPI programming, the time of mutual communication is longer than inner module calculation, so that this speed decreases with an increasing processors. On the other hand, a modular neural network shows a good performance.

5. Conclusion

In this paper, we proposed a modular neural network for achieving speed up of neural network simulations and we could demonstrate that this way of reducing frequencies of mutual communications among modules (processors) is very effective.

In case of Hopfield model type modular neural networks, when a communication time interval T_{com} was about $0.3\times\tau$, a performance was improved. Though this phenomenon is very interesting, the detail is future work.

On the other hand, an ID model type one could keep 100% success rate in wide range of T_{com} and the speed rate was also high.

We consider that this concept of our proposed modular neural networks is not only used in computer simulations but also in a multi-chip system based on a digital hardware effectively. This subject is also future work.

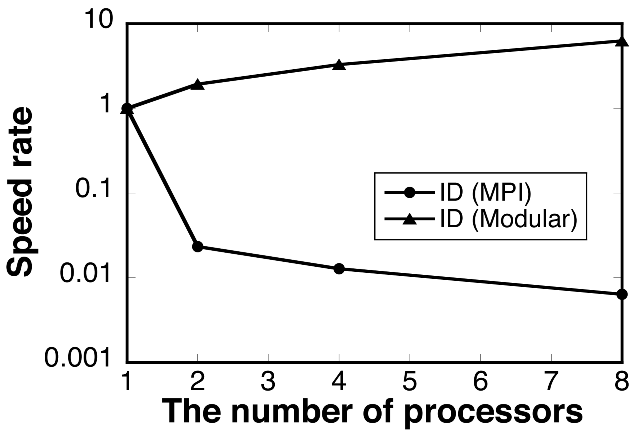

Figure 7: Speed rate as a function of T_{com} by using an ID model.

Acknowledgments

This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C).

References

[1] Y. Hayakawa and K. Nakajima, "Design of the inverse function delayed neural network for solving combinatorial optimization problems", *IEEE Trans. Neural Netw.*, vol. 21, no. 2, pp. 224–237, 2010.

[2] T. Sota, Y. Hayakawa, S. Sato and K. Nakajima, "An application of higher order connection to inverse function delayed network", *Nonlinear Theory and Its Applications*, IEICE, vol. 2, no. 2, pp. 180–197, 2011.

[3] Peter S. Pacheco, "PALALLEL PROGRAMMING with MPI", Morga Kaufmann Publishers, 1997.