Parallel Execution of Neural Networks for Solving SAT

Kairong Zhang and Masahiro Nagamatu

Graduate School of Life Science and Systems Engineering, Kyushu Institute of Technology,
2-4 Hibikino, Wakamatsu-ku, Kitakyushu 808-0196, Japan
e-mail: choh-kaie@edu.brain.kyutech.ac.jp

Abstract— **The satisfiability problem (SAT) is one of the most basic and important problems in the computer science. We have proposed a recurrent analog neural network called LPPH (Lagrange Programming neural network with Polarized High-order connections) for the SAT. In this paper, a method of parallel execution of the LPPH is proposed. Experimental results show that high speedup radio is obtained. Furthermore this method is very easy to be realized by hardware. In the dynamics of the LPPH, there is an important parameter named attenuation coefficient, which is already know to strongly affect the speed of execution of the LPPH. However determining the good value of the attenuation coefficient is very difficult. In this paper, we also show that our method of the parallel execution can ease the difficulty of determining the good value of the attenuation coefficient.**

I. INTRODUCTION

The satisfiability problem (SAT) is an important fundamental problem in the computer science as well as in wide fields of practical applications, such as decision making, scheduling, designing , etc. On the other hand the SAT is a famous NP-complete problem. It requires a lot of time to solve as the size of problem becomes large. We have proposed a neural network for the SAT named LPPH [1] [2] [3]. The LPPH is based on the Lagrangian method, and it was proved theoretically that the solutions of the SAT are the equilibrium points of the LPPH and vice versa, and almost all solutions of the SAT are stable equilibrium points of the LPPH. This means that the LPPH is not trapped by any point which is not a solution of the SAT. The dynamics of the LPPH consists of the differential equations of the neuron outputs and the weights of connections between neurons.

Similar to many other neural networks, it is easy for the LPPH to achieve a parallel processing when neurons are implemented individually by electronic circuits or on different computers (neuron-level parallel processing). Many researches of neuron-level parallel processing had been done [4] [5]. In this case several kinds of communication overheads are needed, such as space overhead or time overhead.

In the dynamics of the LPPH, it is known although only slightly different at initial points, the trajectory of them are different completely from the experimental results. In this paper, we propose a parallel technique of the LPPH, which means preparing plural neural networks of the LPPH in order to find out a solution more efficiently from the different initial states (network-level parallel processing). And the experimental results show that high speedup ratio is obtained by using this parallel technique of the LPPH.

There is an important parameter named attenuation coefficient in the dynamics of the LPPH. The attenuation coefficient is similar to the one which is the essence of "Tabu learning" of Beyer [6]. And this parameter has strong influence on the speed of execution of the LPPH. Furthermore the good value of the attenuation coefficient strongly depends on the given problems, and it is difficult to decide the good value in advance. In this paper, we also show that our method of the parallel execution can ease the difficulty of determining the good value of the attenuation coefficient.

II. SAT AND CNF

A. Conjunctive Normal Form (CNF)

CNF (conjunctive normal form) is a logical formula which is a conjunction of clauses. The clause is a disjunction of literals. The literal is a variable (positive literal) or a negation of a variable (negative literal). Each variable has Boolean value 0 (False) or 1 (True). The rth clause C_r is written as follows:

$$C_r = L_{r1} \vee L_{r2} \vee \cdots \vee L_{ri} \vee \cdots \vee L_{rl} , \quad (1)$$

where L_{ri} means the ith literal of clause C_r. And the CNF is written as follows:

$$E = C_1 \wedge C_2 \wedge \cdots \wedge C_m . \quad (2)$$

In this paper, we can make the following assumption without loss of generality, because $x \vee x = x$ and $x \vee \bar{x} = 1$.

Assumption: No variable appears more than once in each clause.

B. Satisifiability problem (SAT)

A vector $\boldsymbol{x} \in \{0,1\}^n$ is called a Boolean assignment. When $\boldsymbol{x}$ is assigned to CNF E and the result is true, we say '$\boldsymbol{x}$ is a satisfying assignment of E' or '$\boldsymbol{x}$ satisfies E'. Satisfiability problem (SAT) is a problem to find a solution of the given CNF if any solution exists. The definition of the SAT is stated as follows:

$$
\begin{aligned}
&(SAT)\quad find \quad \boldsymbol{x},\\
&such\ \ that\ \ \boldsymbol{x}\ satisfied\quad C_r\quad r=1,2,\cdots m\\
&\boldsymbol{x}\in\{0,1\}^n
\end{aligned}
\tag{3}
$$

An example of the SAT is given as follows:

$$
E=(\bar{x}_1\vee\bar{x}_2\vee\bar{x}_3)\wedge(\bar{x}_1\vee x_2\vee x_3)\wedge(x_1\vee x_2\vee x_3). \tag{4}
$$

The solutions of the CNF E $\boldsymbol{x}\in\{0,1\}^3$ are (0,0,1), (0,1,0), (0,1,1), (1,0,1) and (1,1,0) .

III. LPPH

A.Continuous Valued Satisfiability Problem (CONSAT)

SAT is a discrete valued problem. Here we will convert the SAT into an arithmetic continuous valued problem.

Let us consider $\boldsymbol{x}=(x_1,x_2,\cdots,x_n)$ to be a vector of arithmetic continuous valued variables. For each $i=1,2,\cdots,n,$ and $r=1,2,\cdots,m,$ a function $g_{ir}:[0,1]^n\to[0,1]$ is defined as follows:

$$
g_{ir}(\boldsymbol{x})=\begin{cases}x_i & if\ x_i\ appears\ in\ C_r\ as\ a\ negative\ literal,\\ 1-x_i & if\ x_i\ appears\ in\ C_r\ as\ a\ positive\ literal,\\ 1 & otherwise.\end{cases} \tag{5}
$$

Corresponding with $C_r\,(r=1,2,\cdots,m)$ a function $h_r:[0,1]^n\to[0,1]$ is design as follows:

$$
h_r(\boldsymbol{x})=\prod_{i=1}^{n}g_{ir}(\boldsymbol{x}). \tag{6}
$$

So the CONSAT can be defined as follows,

$$
\begin{aligned}
&(CONSAT)\quad find(\boldsymbol{x}),\\
&\qquad\quad such\ \ that\quad h(\boldsymbol{x})=0\ \ \forall r=1,2,\cdots,m,\\
&\qquad\quad \boldsymbol{x}\in[0,1]
\end{aligned}
\tag{7}
$$

Note that a solution of the CONSAT is in $[0, 1]^n$, while a solution of the SAT is in $\{0, 1\}^n$.

B. Definition of LPPH

The LPPH is expressed by Lagrange function as follow:

$$
F(\boldsymbol{x},\boldsymbol{w})=\sum_{r=1}^{m}w_r h_r(\boldsymbol{x}),\quad \boldsymbol{x}\in[0,1]^n,\ \boldsymbol{w}\in(0,\infty)^m \tag{8}
$$

Here, $F(\boldsymbol{x},\boldsymbol{w})\ge 0$ is clearly. And when $\boldsymbol{x}\in[0,1]^n$ is a solution of the CONSAT, $F(\boldsymbol{x},\boldsymbol{w})=0$. Therefore if the minimum of above expression is found, it is the solution of the CONSAT. The LPPH will be defined as follows by using $F(\boldsymbol{x},\boldsymbol{w})$.

$$
\begin{aligned}
&\frac{dx_i}{dt}=-x_i(1-x_i)\frac{\partial F(\boldsymbol{x},\boldsymbol{w})}{\partial x_i} \qquad i=1,2,\cdots,n,\\
&\frac{dw_r}{dt}=-\alpha\boldsymbol{w}_r+h_r(\boldsymbol{x}) \qquad\qquad r=1,2,\cdots,m.
\end{aligned}
\tag{9}
$$

By solving the above differential equations, the LPPH can find a solution of the SAT. α is called attenuation coefficient.

C. Features of LPPH

Some features are proved about the LPPH when α=0.

(1) The LPPH does not have equilibrium points other than the solutions of the CONSAT, while a solution of the CONSAT is an equilibrium point of the LPPH.
(2) The dynamics of the LPPH converges to a solution of the CONSAT when it approaches the solution.

The features do not guarantee dynamics of the LPPH to find a solution anytime although a solution exists. However experimental results show the LPPH can find a solution effectively if the CONSAT is satisfiable. We also know from experiments that if small appropriate positive value of α significantly improve the efficiency of the LPPH.

IV. PARALLEL EXECUTION OF THE LPPH

A. Definition of the Parallel Execution of the LPPH

We have known that only slightly difference of initial points ultimately cause the completely difference of the trajectories of the LPPH. In this paper, we propose a parallel execution of the LPPH. In this technique,

(1) Prepare plural neural networks of the LPPHs.
(2) Start the LPPHs simultaneously from different initial points to each others.
(3) When any of the LPPHs finds a solution , halt all the LPPHs and return the solution.

The image of the parallel execution of the LPPH is shown as Fig.1.

It is easy to realize the parallel execution of the LPPH by hardware. Only we have to do is preparing plural neural networks. The total system is very simple and executable at high-speed.

B. Experiment

Suppose that the parallel execution of p LPPHs is done. Let t_j be the execution time of jth LPPH for finding a solution. Then, $T_p=\min_j\{t_j\,|\,1\le j\le p\}$ is the execution time of the parallel execution. We will call T_p and pT_p, "execution time" and "total execution time", respectively. In some experiment we use "number of update" instead of "execution time", because these are proportional when the problem is fixed.

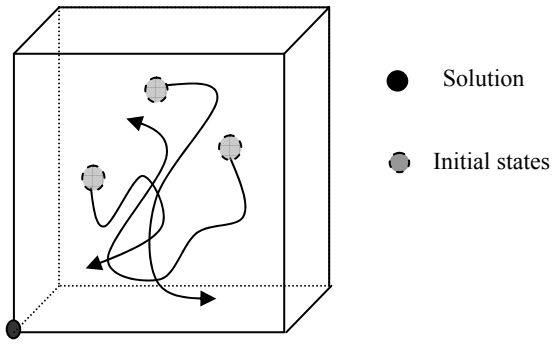

Fig.1. The image of the parallel technique of the LPPH

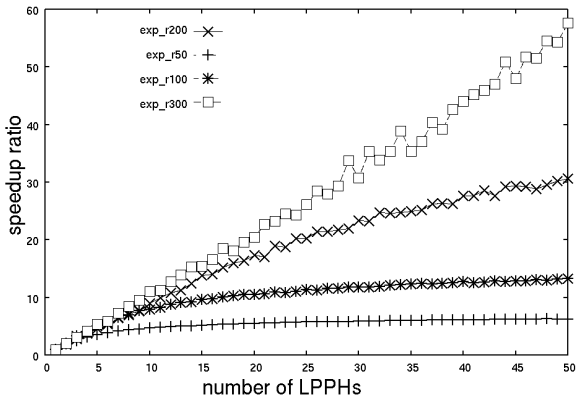

Fig.2 The speedup ratio of random 3-SAT

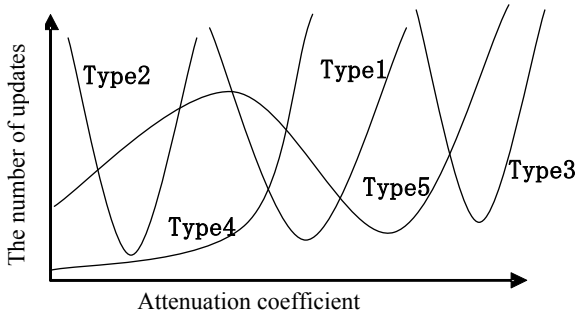

Fig.3 Several types of dependency on the attenuation coefficient

The result of experiment is shown in Fig.2. The horizontal axis indicates the number of LPPHs, and the vertical axis indicates the speedup ratio, namely $E(T_p)/E(T_1)$, where E means the average. In this experiment, parallel execution of P (P=1, 2, …,50) neural networks of the LPPH is used. Randomly generated 3-SAT problems are used in this experiments. They are exp-r300 (300 variables and 1275 clauses), exp-r200 (200 variables and 860 clauses), exp-r100 (100 variables and 430 clauses), exp-r50 (50 variables and 215 clauses). From Fig. 2, it is shown that the high speedup ratio is obtained. This is remarkable for large and difficult problems, e.g. exp-r300.

C. Problem dependency of the attenuation coefficient

The value of attenuation coefficient influences the execution time of the LPPH strongly. The experimental results show that the speedup can be achieved by using the optimal value of the attenuation coefficient. However, the optimal value strongly depends on the problems in hand. Furthermore it is known to be very difficult to decide the optimal value of attenuate coefficient in advance.

When the value of attenuation coefficient is too small, the values of weights become large, and speed of the change of the weights become slow, hence, the execution time will increase [3]. Adversely the value of attenuation coefficient is too large, the dynamics of the LPPH will fall into a limit circle frequently. The dependency on the attenuation coefficient is very strong in general, for difficult problems. Several types of dependency are shown in the Fig.3. Type 1 represents the typical dependency mentioned above. For some problems, the optimal value of the attenuation coefficient is small (Type 2), while for some other problems, the optimal value is large (Type 3). Furthermore there are problems for which the optimal value is 0 (Type 4), or the dependency is bimodal (Type 5).

D. Advantage of the parallel execution

To examine the relation between the number of the LPPHs and the dependency of the execution time on the attenuation coefficient, the following experiments are done. In the experiments, we examined how the execution time depends on the attenuation coefficients for several numbers of LPPHs. The experimental results are shown in Fig.4 and Fig.5. Fig.4 is the result for a random 3-SAT problem of 200 variables, and Fig.5 is the result for an unique-solution random 3-SAT problem of 50 variables. The horizontal axes indicate the value of the attenuation coefficient and the vertical axes indicate the average of the total number of updates. From the experimental results, we can see, for the random 3-SAT problem, the optimal value of the attenuation coefficient increases as the number of LPPHs increases. Additionally, it can be seen that the dependency on the attenuation coefficient is eased. For the unique-solution random 3-SAT problem, although it can be seen that the dependency on the attenuation coefficient is eased, the optimal value of the attenuation coefficient is not affected by the number of the LPPHs. From the experimental results, we can see the dependency of the execution time on the attenuation coefficient can be eased by the parallel execution of the LPPH.

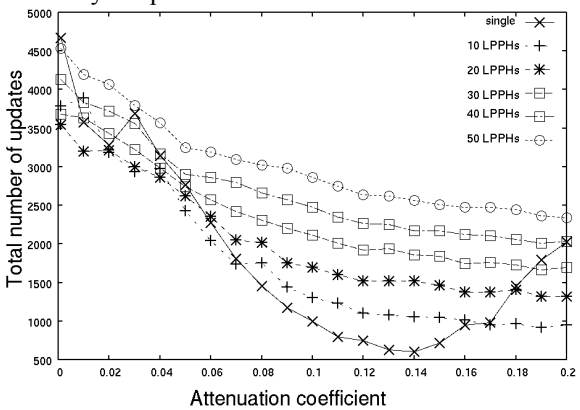

Fig.4 The dependency of total execution updates on the attenuation coefficient for a random 3-SAT problem of 200 variables and 860 clauses

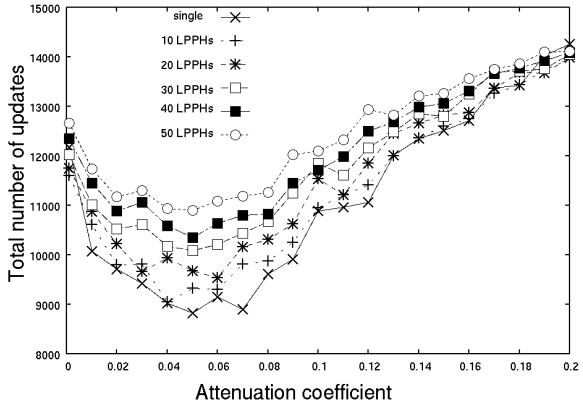

Fig.5 The dependency of total execution updates on the attenuation coefficient for an unique-solution random 3-SAT problem of 50 variables and 120 clauses

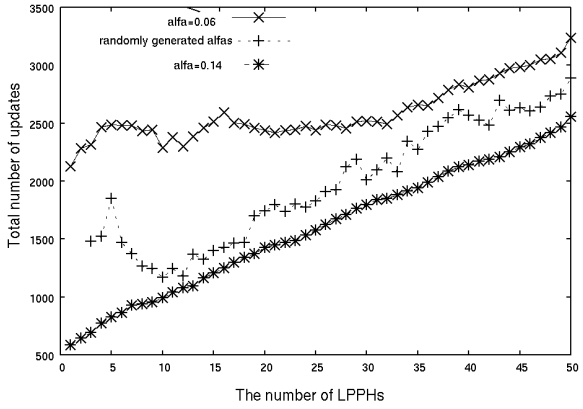

Fig.6 Comparison of several parallel execution methods for random 3-SAT problem of 200 variables and 860 clauses

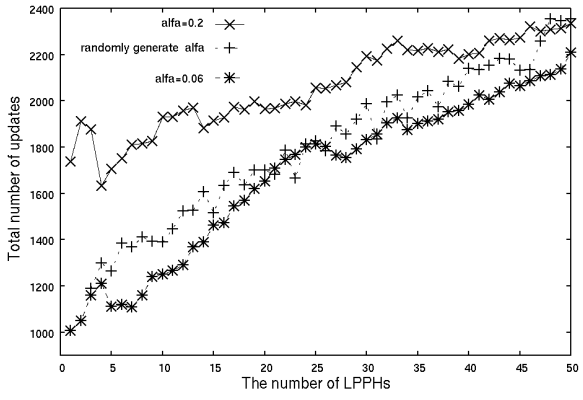

Fig.7 Comparison of several parallel execution methods for an unique-solution random 3-SAT problem of 30 variables and 64 clauses

V. GENERATING ATTENUATION COEFFICIENTS RANDOMLY

It is known that the optimal value of the attenuation coefficients for each problem is very different as shown in Fig.3. And it is very difficult to decide the good value of the attenuation coefficient for each given problem. So far we have used a fixed value of the attenuation coefficient for each LPPH in the parallel execution. In this section we will examine the effectiveness of assigning different value to each LPPH.

Fig.6 and Fig.7 show the experimental results that comparing with the parallel executions in which the attenuate coefficient are fixed and the one in which the attenuation coefficients are generated by uniform random number between 0 and 0.2. In Fig.6~7, the horizontal axes indicate the number of LPPHs and the vertical axes indicate total number of updates. In this experiment, the random 3-SAT problem of 200 variables and 860 clauses and an unique-solution random 3-SAT problem of 30 variables and 64 clauses are used.

In Fig.6, for the fixed values of the attenuation coefficient, α=0.06 and α=0.14 are used. From the result of Fig.4, it is known that α=0.14 is the best value of the attenuation coefficient, and α=0.06 is a bad value for the random 3-SAT problem. From Fig.6 we can see that the result of the parallel execution with randomly generated attenuation coefficients is better than the result of one with α=0.06, and close to the result of one with α=0.14 for some cases.

In Fig.7, for the fixed values of the attenuation coefficient, α=0.06 (the best value of attenuate coefficient), α=0.2(a bad value of the attenuate coefficient) are used. We can also see that the result of randomly generated attenuation coefficients is closed to the result of the best attenuation coefficient (α=0.06).

VI. CONCLUTION

We propose a technique of parallel execution of the LPPH (network level parallel processing). It is very easy to realize this technique by hardware. Only we have to do is preparing plural LPPHs, and execute them simultaneously. From the experimental results, the high speedup ratio is obtained especially for the difficult problems.

It is known that the attenuation coefficient has strong influence on the execution time of the LPPH. And it is also known to be very difficult to find the optimal value of the attenuation coefficient in advance. From the experimental results of this paper, the dependency of the execution time on the attenuation coefficient can be eased by the parallel execution of the LPPH.

We also examined the efficiency of the parallel execution with randomly generated attenuation coefficients. Experimental results show this technique is more efficient than using a bad value for the attenuation coefficient.

REFERENCE

[1] M. Nagamatu and T. Yannaru, “On the stability of Lagrange programming neural networks of satisfiability problems of propositional calculus”, Neurocomputing, 13, 119-133, 1995.

[2] M. Nagamatu and T. Yannaru, “Parallel state space search for SAT with Lagrange Programming Neural Network”, proceedings of the fifth International Conference on Neural Information Processing. October, 1998

[3] M. Nagamatu and T. Yanaru, ”Solving SAT by Lagrange Programming Neural Network with Long and Short Term Memories”,

ch. 11 in “Information Modelling and Knowledge Bases”, IOS Press, pp. 289-301, 2000

[4] W. Chrabakh and R. Wolski, “GrADSAT: A Parallel SAT Solver for the Grid”, UCSB Computer Science Technical Report Number 2003-05

[5] C. Sinz, W. Blochinger and W. Kuchlin, “PaSAT-Parallel SAT-Checking with Lemma Exchange: Implementation and Applications”, In Proceedings of SAT2001, pages 212-217, 2001.

[6] D. A. Beyer and R. G. Ogier, “Tabu learning: a neural network search method for solving nonconvex optimization problem,” Proc. 1991 IEEE Int. Joint Conf. Neural Networks, pp. 953-961,1991.