ニューラルネットワークの並列実行によるＳＡＴの解法

5ＡＧ－3　　工藤　道明　　永松　正博　　矢鳴　虎夫

九州工業大学工学部

1．はじめに

代表的なNP完全問題の一つにSAT（充足可能性問題）がある．SATとは，与えられた論理式を真にする変数の割り当てが存在するかどうかを判定する問題である．著者らは，ラグランジュの方法を利用することによってSATを解くニューラルネットワークLPPH(Lagrange programming neural network)を提案している[1]．LPPHは，局所探索法の一つであり，初期値によって解を見つける速さの差が大きい．また，LPPHはDPや分岐限定法などに比べて高速に解を見つけることができるが，それでも大規模な問題を解く場合は非常に時間がかかる．そこで，LPPHを並列実行して解くことが考えられる．本論文では，各プロセッサがそれぞれ異なる初期値に対する解の探索を独立に並列実行する方法を提案する．この方法で並列実行を行った場合，どれか一つのプロセッサが解を見つければ良い．この並列実行ではプロセッサ間で頻繁にデータをやり取りすることがないので並列化によって生じるオーバーヘッドが小さい．本論文では，LPPHを上述の方法で並列実行してSATを解いた場合の計算時間について考察を行う．

2．ＳＡＴ

論理式Eは以下のような和積標準形で与えられる．

$$E = C_1 \wedge C_2 \wedge \cdots \wedge C_m, \tag{1}$$

$$C_r = L_{r1} \vee L_{r2} \vee \cdots \vee L_{rl_r} \qquad r = 1,2,\cdots,m. \tag{2}$$

C_rはr番目の節，L_{rp}はC_rのp番目のリテラルである．また，変数を$x_1, x_2, \cdots, x_n$とする．

ここで，離散的な問題であるSATを連続的なCONSAT（Continuous valued SAT）と呼ばれる問題に置き換える．

CONSAT：　find x,
such that　$\forall r \; h_r(x) = 0,$

$$0 \le x_i \le 1 \qquad i = 1,2,\cdots,n.$$

ここで，

$$h_r(x) = \prod_{p=1}^{n} g_{rp}(x) \qquad r = 1,2,\cdots,m.$$

$$g_{rp}(x) = \begin{cases} x_i & L_{rp}\text{が負のリテラルとして}C_r\text{に現れたとき,} \\ 1 - x_i & L_{rp}\text{が正のリテラルとして}C_r\text{に現れたとき,} \\ 1 & \text{その他のとき.} \end{cases}$$

3．ＬＰＰＨ

ラグランジュ関数$F(x,w)$を以下のように定義する．

$$F(x,w) = \sum_{r=1}^{m} w_r h_r(x). \tag{3}$$

ここで，w_rは節rの重みである．以下にLPPHの状態方程式を示す．

$$\frac{dx_i}{dt} = -x_i(1 - x_i)\frac{\partial F(x,w)}{\partial x_i}, \tag{4}$$

$$\frac{dw_r}{dt} = \frac{\partial F(x,w)}{\partial w_r} = h_r(x) \qquad r = 1,2,\cdots,m. \tag{5}$$

この方程式の平衡点はCONSATの解と対応し，しかも漸近安定であることが知られている[2]．従って，この方程式を与えられた初期値に対して数値解法を行うことによりCONSATの解を求めることができる．式(5)の代わりに，各w_rの減衰を考慮した次に示す式を用いるとLPPHの解を探す時間を短縮できることが分かっている[3]．

$$\frac{dw_r}{dt} = -\alpha w_r + h_r(x) \qquad r = 1,2,\cdots,m. \tag{6}$$

ここで，αは重みw_rの減衰率である．

4．ＬＰＰＨの並列実行の方法とその解析法

次に，LPPHの並列実行を行う手順を示す．

ステップ１：プロセッサ１に問題を与える．

ステップ２：プロセッサ１は，他のプロセッサにその問題を送信する．

ステップ３：各プロセッサは，自分のプロセッサ番号を乱数の初期値として各変数の初期値をランダムに作り，LPPHを用いて問題を解く．

Solving SAT by Parallel Simulation of Lagrange Programming Neural Networks

Michiaki Kudou, Masahiro Nagamatu, Torao Yanaru

Kyusyu Institute of Technology

Tobata, Kitakyusyu, Fukuoka 804, Japan

ステップ４：一番速く問題を解いたプロセッサが，プロセッサ１に解を送信する．

以上の方法による並列実行の効率の解析を行うために次の実験を行う．

ステップ１：初期値をランダムに 10000 回変えて，それぞれの初期値に対して LPPH を用いて問題を解いた時間を計測し，10000 個の時間データを得る．

ステップ２：10000 個の時間データの中から n 個の時間データをランダムに取り出す．ただし，n は並列実行した場合のプロセッサ数である．

ステップ３：取り出した n 個の時間データの中の最小値を調べる．

ステップ４：ステップ２，３を 3000 回繰り返し，得られた 3000 個の最小値の時間データの平均を求めて，それを $T(n)$ とする．この $T(n)$ を n 個のプロセッサで LPPH を用いて並列実行して問題を解いたときの計算時間と考える．

ステップ５：n が１から 200 それぞれの場合についてステップ２，３，４を実行して $T(1), T(2), \cdots, T(200)$ を得る．

５．実験結果

５つの SAT の例題に対して上述の方法により求めた $T(n)$ に基づいて並列実行の効率を調べたものを以下に示す．ただし，微分方程式の数値計算にはオイラー法を用いており，$T(n)$ としてオイラー法の繰り返し回数を用いている．図１に n と $T(n)$ の関係を示す．また，並列実行の効率を調べるために n と $n \cdot T(n)$ の関係を図２に示す．これらの図では減衰率 α の値は 0.08 に固定している．図３に減衰率 α を０以上１以下のランダム値とした場合の n と $n \cdot T(n)$ の関係を示す．

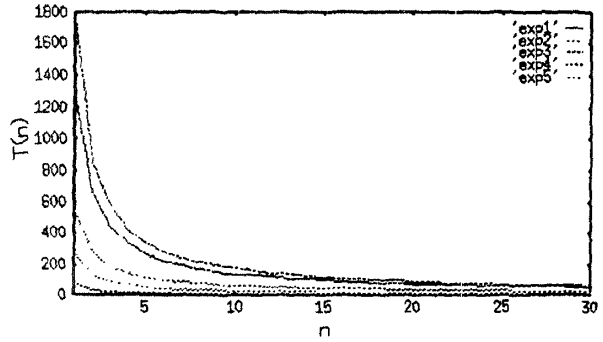

図１ n と $T(n)$ の関係

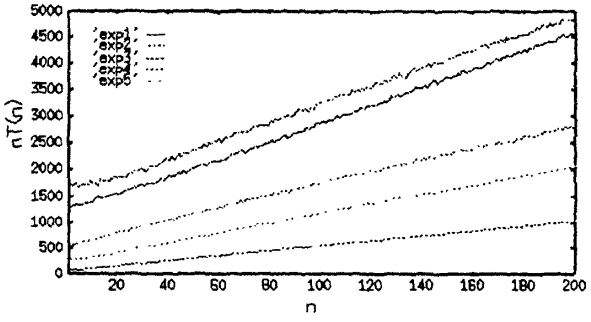

図２ n と $n \cdot T(n)$ の関係

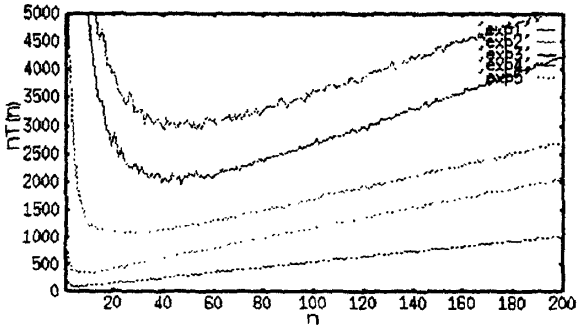

図３ 減衰率 α を固定しない場合の n と $n \cdot T(n)$ の関係

６．考察

図１より並列実行することによって計算時間を短縮できることがわかる．しかし，図２より n 個のプロセッサで並列実行した場合に n 倍のスピードアップが得られていないことがわかる．また，図２と図３との比較から，n が大きい場合は最適な α の値をあらかじめ知っておく必要はなくて α の値をランダムに決めてよいことがわかる．

参考文献

[1] M. Nagamatu, and T. Yanaru, "Lagrange programming neural networks with polarized high-order connections for satisfiability problems of propositional calculus," *IIZUKA '94, The 3rd International Computing*, pp. 233-235, August 1994.

[2] M.Nagamatu, and T.Yanaru,"On the stability of Lagrange programing neural networks for satisfiability problems of propositional calculus "in *Proc.ANNES '95*,pp.71-74, November 1995.

[3] M. Nagamatu, and T. Yanaru, "Lagrangian Method for satisfiability problems of propositional calculus," in *Proc. ANNES '95*, pp.71-74, November 1995.