6R−8 ニューラルネットワークを用いたソートアルゴリズムとそのハードウェア化について

山田 学　中川 徹　北川 一　村上勝彦
(豊田工業大学)

1 はじめに

ソートはコンピュータサイエンス, エンジニアリングにおいてもっとも基本的な操作の一つである. しかし従来からのシーケンシャルなアルゴリズムでは, 問題の規模 (n) に対する計算時間のオーダは最速のものでも $O(n \log n)$ になると言われている [1]. Takefuji と Lee は Hopfield 型ニューラルネットワーク [2][3] (以下, NN と略す) で $O(n^2)$ 個のニューロンを用い, 問題の規模に関わらず 2 ステップ すなわち $O(1)$ で解を得る並列処理ソートアルゴリズムを示した [4]. しかし, このアルゴリズムでは多入力のアナログ加算器を必要とした. 今回, この Takefuji らのアルゴリズムとバイナリ結合のニューラルネットワーク $SDNN$[5] を基礎として加算器を 1 つも必要としないモデルを新たに考え, 実験を行った. また NN でソータを作ろうとする場合, 実際にハードウェアとして実現できることが重要であり, 今回, 4 要素ソータの試作および, そのハードウェア量の定式化を行った.

2 NN によるソート問題の表現

ソート問題は, $(n-1)$ 個の正の整数 $N_1, N_2, \ldots, N_{n-1}$ と 0 (ダミー N_n) があるとき, $0 < N_{\pi_1} < N_{\pi_2} < \ldots < N_{\pi_{n-1}}$ となるような ($\pi_1, \pi_2, \ldots, \pi_{n-1}$) を見つけることである.

NN によるソート問題は, $n \times (n-1)$ 個のニューロンを図 1 に示すように配置する. そして, X 方向の各要素の次に大きな値になる Y 方向要素との交差する位置のニューロンが結果として ON になるように各ニューロンを結合する. 図 1 はその NN を収斂させ, 結果として $N_4 < N_3 < N_2 < N_1$ となった状態を示す.

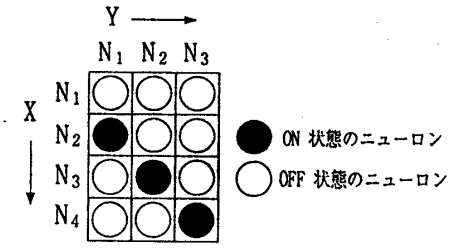


図 1: NN による 3 要素ソート問題

3 Takefuji アルゴリズム

このアルゴリズムで使用されるニューロンはバイナリニューロンと呼ばれるものである. バイナリニューロンの出力は (1) 式によって与えられる.

$$V_{ij} = \begin{cases} 1, & \text{if } U_{ij} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

ここで V_{ij} は (i,j) 番目のニューロンの出力で, U_{ij} は (i,j) 番目のニューロンの入力を示す. このバイナリニューロンの運動方程式は (2) 式によって表される.

$$\frac{dU_{XY}}{dt} = -f(N_Y, N_X)\left(\sum_{i \neq X}^{n} f(N_Y, N_i) V_{Xi} - 1\right) \quad (2)$$

ここで, $$f(L,R) = \begin{cases} 0, & \text{if } L < R \\ 1, & \text{otherwise} \end{cases}$$

である. 1 ステップ目で N_Y が N_X より大きいとき (2) 式は正の値をとり, ニューロンの出力 V_{XY} は 1 になる. また, N_Y が N_X より小さいときには負の値をとるので, 同出力は 0 になる. 2 ステップ目で, N_Y が N_X の次に大きな値の時, (2) 式が 0 となるので, ニューロンの出力 V_{XY} は ON のままになるが, それ以外の時は (2) 式が全て負の値となるので同出力は OFF になる. このようにしてニューロンは 2 ステップでソートされた状態になる. ただし, ニューロンの入力 U_{XY} の初期値はすべてが小さな負の値になっていなければならない.

ここで, 3 要素 (3, 2, 1) ソートにおける NN の動作を図 2 に示す. この実行結果は $0 < 1 < 2 < 3$ となっていることを示している.

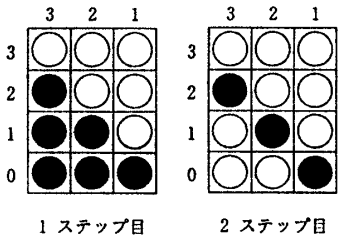


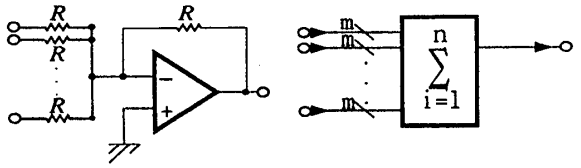
図 2: NN の動作 (3 要素)

A Super Parallel Sorting Algorithm with a Binary Neural Network and its Hardware Implementation
Manabu YAMADA, Tohru NAKAGAWA, Hajime KITAGAWA, and Katsuhiko MURAKAMI
Toyota Technological Institute

4 ハードウェア化に関する問題点

Takfuji らのアルゴリズムで使用する (2) 式は $\sum$ (総和) を含んでいる. 図 3 に基本的な n 入力加算器を示す. 大規模なソータを構成する時, (2) 式の総和を図 3 (a) に示すオペアンプのアナログ加算器によりハードウェアとして実現する事は大変困難であると考えられる. 一方, 図 3 (b) に示すようなディジタル加算器で計算をさせることもまた困難である. なぜなら, m ビット長の n 入力全加算器を構成するのに m ビット長の 2 入力全加算器を使用した場合, 計算時間は $O(\log_2 n)$ かかるからである. また, n 入力の総和計算を $O(1)$ で行うにはハードウェア規模が非常に大きくなる. それゆえに, オリジナルのアルゴリズムは大規模なソータを作るのには向いていないと考えられる.

そこで今回バイナリ結合のみを用い, かつ, ニューロンの入力における加算器を 1 つも必要としない新しいニューラルネットワーク (以下 *SBNN*: Strictly Binary NN と略す) を提案する.



(a) n 入力アナログ加算器　(b) n 入力 m ビット長ディジタル加算器

図 3: 基本的な加算器

5 簡略化したアルゴリズム

ここで *SBNN* を使って簡略化したアルゴリズムについて述べる. *SBNN* は Takefuji らのアルゴリズムと *SDNN* に基づいたもので, ニューロンも (1) 式で示されるバイナリニューロンを使用する. しかしながら, *SBNN* は総和の代わりに多入力 *OR* を使用し, 積の代わりに *AND* を使用している. この簡略化されたアルゴリズムにおいて, バイナリニューロンの入力 U_{XY} は (3) 式によって表される.

ただし, $\sum_1^{max}$ は多入力 *OR* である.

$$U_{XY} = \begin{cases} 0, & \text{if } X = Y \\ AND(f(N_Y, N_X), M_{XY}), & \text{otherwise} \end{cases} \quad (3)$$

ただし,

$$M_{XY} = \sum_{i=1}^{n-2}{}_1^{max} \sum_{j=i+1}^{n-1}{}_1^{max} AND(f(N_Y, N_i), V_{Xi}, f(N_Y, N_j), V_{Xj})$$

6 シミュレーションによる検証

この簡略化したアルゴリズムを検証する目的で. ソフトウェアによるシミュレーションを行った. この実験での問題の規模は 10 から 2000 要素までで, その要素が保持する値は乱数により与えた. その結果, 実験した全ての場合において原論文と同様に 2 ステップで解が得られた.

7 ハードウェア化について

ハードウェアの規模に関して, *SBNN* を使った簡略化アルゴリズムと Takefuji らのオリジナルアルゴリズムとを比較すると表 1 のようになる. 表 1 に示すように,

表 1: ハードウェアの規模

Items	*Number of Components*	
	Original	*BNN*
Binary neuron	$n \times (n-1)$	$n \times (n-1)$
Comparator	$\frac{n \times (n-1)}{2}$	$\frac{(n-1) \times (n-2)}{2}$
Adder†	$n \times (n-1)$††	none
OR†	none	$(n-1)^2$
AND†	none	$(n-1)^3$

†: Multiple input, ††: n-input analogue adder

SBNN は加算器を 1 つも必要とせず AND-OR ゲートのみで構成される. それゆえ *SBNN* を用いることにより, $O(1)$ で計算する大規模ソータを実際にハードウェアで構成することが可能であると考えられる. 最後に, 現在試作している 4 要素ソータの外観を図 4 に示しておく.

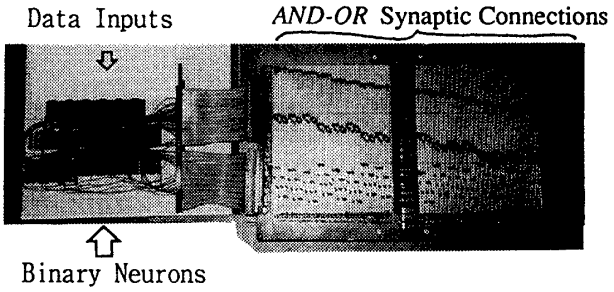


図 4: ソータの外観 (4 要素)

参考文献

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman: "The Design and Analysis of Computer Algorithms," Addison-Wesley.

[2] J. J. Hopfield: "Neurons with graded response have collective computational properties like those of two-state neurons," *Biophysics, Proc. Natl. Acad. Sci. USA*, Vol. 81, pp. 3088-3092, 1984.

[3] J. J. Hopfield and D.W. Tank: "Neural Computation of Decisions in Optimization Problems," *Biolog. Cybern.*, Vol. 52, pp. 141-152, 1985.

[4] Y. Takefuji and K. C. Lee: "A Super-Parallel Sorting Algorithm Based on Neural Networks," IEEE Trans. on Circuits and Systems, Vol. 37, No. 11, pp. 1425-1429, Nov. 1990.

[5] T. Nakagawa and H. Kitagawa: "SDNN-3: A Simple Processor Architecture for O(1) Parallel Processing in Combinatorial Optimization with Strictly Digital Neural Networks," *Proc. IJCNN-91*, Vol. 3, pp. 2444-2449, Singapore, 1991.