LA-UR-97-851

CONF-970669--2

Title: OPTIMIZATION OF CIRCUITS USING A CONSTRUCTIVE LEARNING ALGORITHM

Author(s): Valeriu Beiu

Submitted to: International Conference on Engineering Applications of Neural Materia
Stockholm, Sweden
June 16-18, 1997

Los Alamos
NATIONAL LABORATORY

Optimisation of Circuits Using a Constructive Learning Algorithm

Valeriu Beiu *

Los Alamos National Laboratory, Division NIS-1, MS D466, Los Alamos, NM 87545, USA

Abstract—*The paper presents an application of a constructive learning algorithm to optimisation of circuits. For a given Boolean function f, a fresh constructive learning algorithm builds circuits belonging to the smallest $F_{n,m}$ class of functions (n inputs and having m groups of ones in their truth table). The constructive proofs, which show how arbitrary Boolean functions can be implemented by this algorithm, are shortly enumerated. An interesting aspect is that the algorithm can be used for generating both classical Boolean circuits and threshold gate circuits (i.e. analogue inputs and digital outputs), or a mixture of them, thus taking advantage of mixed analogue/digital technologies. One illustrative example is detailed. The size and the area of the different circuits are compared (special cost functions can be used to closer estimate the area and the delay of VLSI implementations). Conclusions and further directions of research are ending the paper.*

Keywords—*constructive learning, circuit optimisation, VLSI complexity.*

Introduction

In this paper we shall consider a neural network (**NN**) constructive algorithm (i.e. which determines both the number of layers and number of neurons, and their synaptic *weights*), which is guaranteed to converge in finite time. In particular, it can be used for synthesising Boolean circuits. We shall use the notations for feedforward NNs which classify m examples of n bits each $X_k = (x_0, \dots, x_{n-1}) \in \mathbb{R}^n$, $k = 1, \dots, m$ where each neuron computes $out\,(\boldsymbol{In}) = \sigma\left(\sum_{i=0}^{n-1} w_i \times \boldsymbol{In}_i + \theta\right)$, with $w_i \in \mathbb{R}$ called the synaptic *weights*, $\theta \in \mathbb{R}$ known as the *threshold*, and σ a non-linear activation function. The cost functions commonly in use are *depth* and *size*. These can be linked to T ($\approx$ *depth*) and A ($\approx$ *size*) of a VLSI chip, but NNs do not closely follow such proportionalities (as the *area* of the connections counts and the *area* of one neuron is related to its associated *weights*, not to mention the complexity of the function realised by each neuron). That is why better alternative measures are: (i) the *fan-in* as the *"area required for inter-node connectivity grows like the cube of a node's fan-in"* [13]; (ii) the total number of connections [17]; (iii) the *total number of bits needed to represent the weights* [11, 20]; or even more precise approximations like (iv) the *sum of all the weights and thresholds* $\sum\left(\sum_{i=0}^{n-1} |w_i| + |\theta|\right)$ [3, 5], or $\sum\sum_{i=0}^{n-1} w_i^2$ [21]. Such cost measures can easily be understood as the *area* is related to the *weights* and the *thresholds*, as they have to be physically realised (transistors, conductances, resistances, etc.), and have been used as an optimum criteria for linear programming synthesis [16], or for defining the minimal integer realisation of one threshold gate (**TG**) [14].

A Direct Synthesis Algorithm

A novel synthesis algorithm has evolved from the decomposition of COMPARISON.

Proposition 1 [1] *The computation of* COMPARISON *of two n-bit numbers can be realised by a Δ-ary tree of size $\boldsymbol{O}(n/\Delta)$ and depth $\boldsymbol{O}(\log n/\log\Delta)$ for any integer fan-in $2 \le \Delta \le 2n$.*

Proposition 2 [1] *The* COMPARISON *of two n-bit numbers can be computed by a Δ-ary tree neural network with polynomially bounded integer weights and thresholds, having size $\boldsymbol{O}(n/\Delta)$ and depth $\boldsymbol{O}(\log n/\log\Delta)$ for any integer fan-in (Δ) in the range 3 to $\boldsymbol{O}(\log n)$.*

Proposition 3 [3] *The neural network with polynomially bounded integer weights (and thresholds) computing the* COMPARISON *of two n-bit numbers occupies an area of $\boldsymbol{O}(n \cdot 2^{\Delta/2}/\Delta)$ for all the values of the fan-in (Δ) in the range 3 to $\boldsymbol{O}(\log n)$.*

* On leave of absence from the Department of Computer Science , "Politehnica" University of Bucharest, Splaiul Independenței 313, RO-77206 Bucharest, România.

We have used these results for synthesising functions of n input variables having m groups of ones.
Proposition 4 [2, 3] *Any function $f \in F_{n,m}$ can be computed by a neural network with polynomially bounded integer weights (and thresholds) of depth $O\,[\log(mn)/\log\Delta]$ and size $O\,(mn/\Delta)$, and occupying $O\,(mn \cdot 2^{\Delta}/\Delta)$ area if $2m \le 2^{\Delta}$, for all the values of the fan-in (Δ) in the range 3 to $O\,(\log n)$.*
The well-known AT^2 becomes $O\,[(mn \cdot \log^2(mn) \cdot 2^{\Delta})/(\Delta \cdot \log^2\Delta)]$, which is minimised for small constant *fan-in* values 6...9 [7]. This method can be directly used to learn k functions from m examples.
Proposition 5 [3, 5] *Any finite set of k functions $f \in F_{n,i}$ defined by m examples ($i \le m \le ik$), can be computed by a neural network with polynomially bounded integer weights and thresholds having size $O\,[m\,(2n+k)/\Delta]$ and depth $O\,[\log(mn)/\log\Delta]$, and occupying $O\,(mn \cdot 2^{\Delta}/\Delta + mk)$ area, for all the values of the fan-in (Δ) in the range 3 to $O\,(\log n)$.*
If $2m > 2^{\Delta}$ the tree structure has more COMPARATORs than all the different COMPARISONs (of limited *fan-in*), and by deleting the redundant ones the *size* and the *area* can be reduced by huge factors.
Proposition 6 [6] *The dichotomy of m examples from $\mathbb{R}^n$ can always be solved with:*

$$\#bits < m \cdot \lceil n \log(D/d) + 2.0471\,n - (\log n)/2 - 0.8257 \rceil / 2 .$$

Proposition 7 [6] *The entropy of a dichotomy of m examples from $\mathbb{R}^n$ is bounded by $2^{m\lceil \log m \rceil}$.*
Proposition 8 [9] *The dichotomy of $m = m_+ + m_-$ examples from $\mathbb{R}^n$ can always be solved with:*

$$\#bits < \max(m_+, m_-) \cdot \lceil n \log(D/d) + 1.8396\,n - 1.0800 \rceil .$$

Proposition 9 [12] *The number-of-bits necessary for the classification of m patterns in general position in $\mathbb{R}^n$ using weights in the set $\{-p, -p+1, \ldots, 0, \ldots, p\}$ is:*

$$\#bits > m\,n \cdot \lceil \log(2pD) \rceil = m\,n \cdot \lceil \log(D/d) \rceil .$$

Proposition 10 [10] *The dichotomy of $m = m_+ + m_-$ examples in general position in $\mathbb{R}^n$ using weights in the set $\{-p, -p+1, \ldots, 0, \ldots, p\}$ requires:*

$$\#bits > m \cdot \lceil n \log(D/d) + n + 0.6515 \rceil / 2 .$$

Proposition 11 [10] *The dichotomy of $m = m_+ + m_-$ examples in general position in $\mathbb{R}^n$ using weights in the set $\{-p, -p+1, \ldots, 0, \ldots, p\}$ requires:*

$$\#bits > \max(m_+, m_-) \cdot \lceil n \log(D/d) - 0.4667n + \log n + 0.0665 \rceil .$$

A first algorithm has been described [2, 4, 5, 8] as:
1. Quantize the space as described in *Propositions 6* and *8*; for the moment a logarithmic factor is lost as the intersection of two hypercubes is used instead of the intersection of two balls (see [6, 9, 19]).
2. From the set of examples, determine the 'constants' to be used by the COMPARATORs (the first layer).
3. Reduce the number of COMPARATORs by grouping together more hypercubes, or by sometimes using AND gates instead of COMPARATORS.

This is equivalent to *finding hyperplanes parallel to the axes*, which are COMPARISONs (first layer) with values deduced from the data-set. The desired function can be synthesised either by one more layer of TGs (in fact MAJORITY gates, see Fig. 1.a), or by an additional classical 2-layer AND-OR structure: a second hidden layer of AND gates (each gate corresponds to one hypercube, and at most 2 COMPARATORS are needed for each dimension), and a third layer of k OR gates represents the outputs (Fig. 1.b). For minimising the *area* some COMPARATORs can be replaced by AND gates [3] (see first layer in Fig. 1.b).

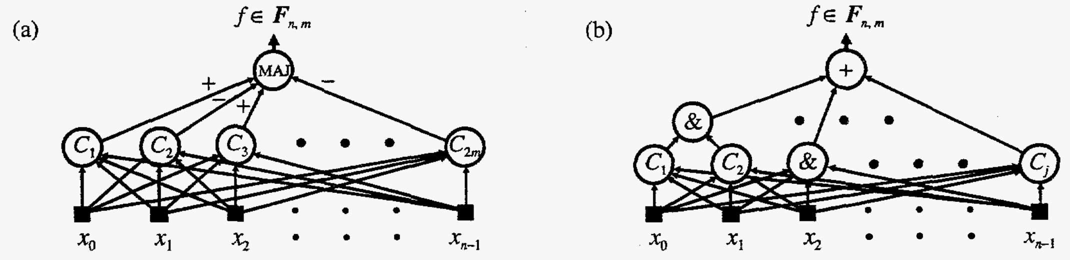

Figure 1. Two solutions for implementing $F_{n,m}$ functions: (a) COMPARISONs and one MAJORITY gate; (b) COMPARISONs or AND gates (for minimising *size* and *area*), followed by a classical AND-OR tree structure.

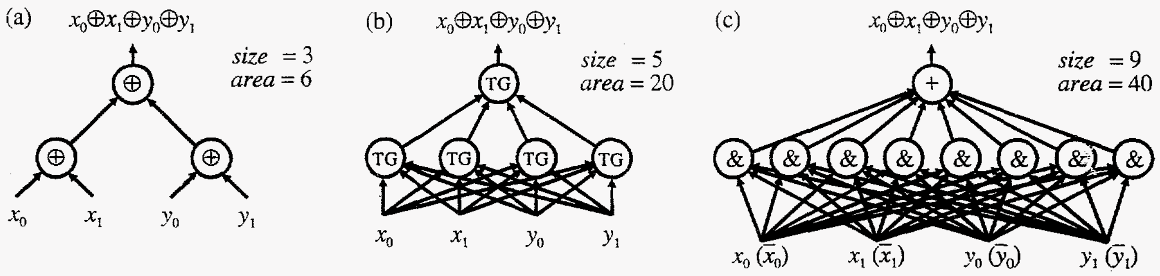

Figure 2. The PARITY problem: (a) solution with XORs; (b) TG solution; (c) classical AND-OR solution (brute force).

An immediate but significant extension for analog inputs has been to use analogue COMPARATORs in the first layer [5]. Such an extension—beside allowing analog inputs—has also several other advantages: (i) the overall *area* is reduced (a 2-input analog COMPARATOR is smaller than a digital COMPARATOR requiring several digital inputs); (ii) it is faster (smaller *depth*); (iii) dissipates less power (due to reduced *area* and fewer transistors needed to implement analog COMPARATORs). The only disadvantage is that the design and the technology is more complex.

Illustrative Example

As an example, we consider the PARITY function of four bits. Here we estimate the *area* as the sum of the *fan-ins* [17] (see [5, 8] for more precise cost functions). It is known that PARITY can be implemented with just three 2-input XOR gates (Fig. 2.a), or with five 4-input TGs (Fig. 2.b). It is also known that, in general, a 4-input Boolean function requires $2\sqrt{16}+3=11$ TGs [15, 18]. A classical Boolean solution requires eight (one for each minterm) AND gates (Fig. 2.c). Using the constructive algorithm we obtain a brute force solution (see Fig. 3.a and 3.b); a hand crafted solution can be seen in Fig. 3.c.

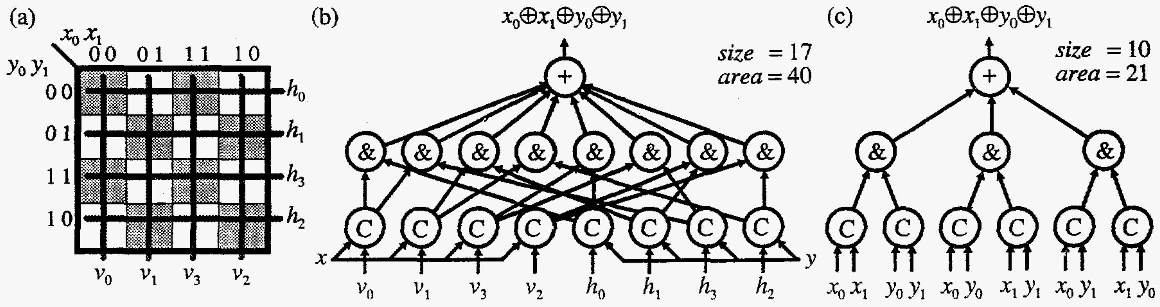
(a)
$x_0\,x_1$
$y_0\,y_1$
00 01 11 10
00
01
11
10
h_0
h_1
h_3
h_2
v_0 v_1 v_3 v_2
(b)
$x_0 \oplus x_1 \oplus y_0 \oplus y_1$
size = 17
area = 40
x
y
v_0 v_1 v_3 v_2 h_0 h_1 h_3 h_2
(c)
size = 10
area = 21
$x_0 x_1$ $y_0 y_1$ $x_0 y_0$ $x_1 y_1$ $x_0 y_1$ $x_1 y_0$

Figure 3. The PARITY problem: (a) Karnaugh map (for COMPARATORS); (b) brute force solution; (c) hand crafted.

If the inputs are considered 'analogues', the solution given by our algorithm is improved, and looks like the one in Fig. 4.b, or Fig. 4.c (using MAJORITY gates, the *size* could still be reduced).

Finally, going closer towards the TG solution from Fig. 2.b, we can allow COMPARATORs between

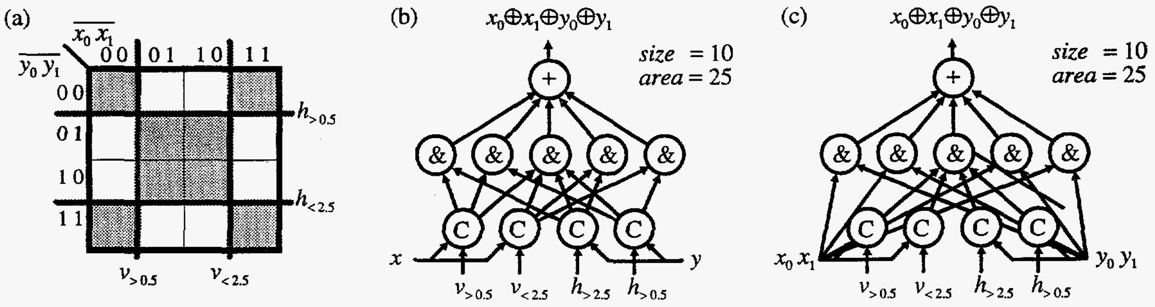

Figure 4. The PARITY problem: (a) Karnaugh map (for analogue COMPARATORS); (b) solution using COMPARATORS (with constants); (c) mixed solution using COMPARATORS with constants and classical digital circuitry.

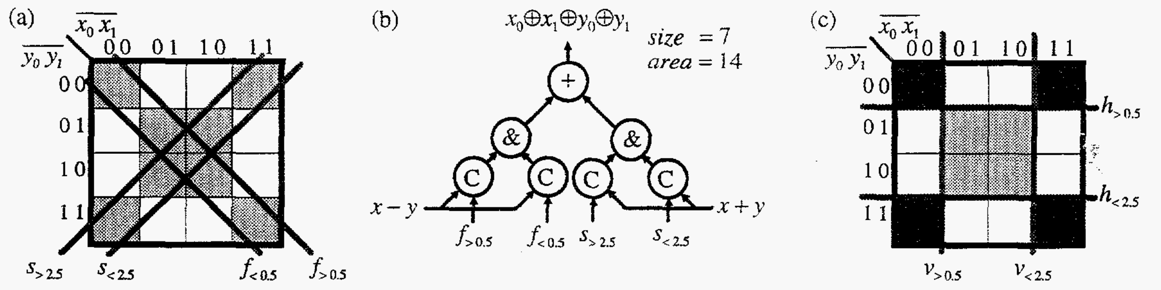

Figure 5. The PARITY problem: (a) Karnaugh map for analogue COMPARATORS between inputs [1]; (b) the resulting circuit; (c) Karnaugh map for a possible mixed solution (hand crafted).

several inputs and a constant (e.g. the inputs are combined by a pre-processing layer of op-amps); this is equivalent to allowing hyperplanes which are equally spaces from the axes (Fig. 5.a). Now, the algorithm generates an even smaller circuit (Fig. 5.b); a mixed solution can be hand crafted (Fig. 5.c).

Conclusions

This paper has shown that constructive learning could be used for optimising circuits. Such an approach can take advantage of mixed analogue/digital implementations as it builds either TG circuits, Boolean circuits, or a mixture of them (for a solution to the two-spirals problem see [5, 8]). For the particular example considered, the solutions given by our algorithm can still be enhanced (see hand crafted versions), but the starting points are always in the solutions given by the algorithm. We are currently working on designing another constructive algorithm based on *Propositions 9, 10* and *11*.

References

[1] V. Beiu, J. Peperstraete, J. Vandewalle and R. Lauwereins. COMPARISON and Threshold Gate Decomposition. In D.J. Myers and A.F. Murray (eds.): *MicroNeuro '93*, UnivEd Tech. Ltd., Edinburgh, 83-90, 1993.

[2] Beiu, V., J.A. Peperstraete, J. Vandewalle and R. Lauwereins. Learning from Examples and VLSI Implementation of Neural Networks. In R. Trappl (ed.): *Cybernetics and System Research '94*, World Scientific, Singapore, 1767-1774, 1994.

[3] V. Beiu, J.A. Peperstraete, J. Vandewalle and R. Lauwereins. Area-Time Performances of Some Neural Computations. In P. Borne, T. Fukuda and S. G. Tzafestas (eds.): *Symp. on Signal Proc., Robotics and Neural Networks SPRANN'94*, GERF EC, Lille, 664-668, 1994.

[4] V. Beiu. Optimal VLSI Implementations of Neural Networks. In *Proc. Applied Decision Tech. Conf. ADT'95*, 193-207, 1995. Also in J.G. Taylor (ed.): *Neural Networks and Their Applications*, John Wiley and Sons, Chichester, 255-276, 1996.

[5] V. Beiu and J.G. Taylor. Direct Synthesis of Neural Networks. *Proc. MicroNeuro'96*, IEEE CS Press, Los Alamitos, CA, 257-264, 1996.

[6] V. Beiu. Entropy Bounds for Classification Algorithms. *Neural Network World*, **6**(4), 497-505, 1996.

[7] V. Beiu. Constant Fan-In Neural Networks Are VLSI-Optimal. *Int. Conf. on Maths. of Neural Nets and Appls. MANNA'95* (Oxford, U.K., July 1995). Also in S.W. Ellacott, J.C. Mason and I.J. Anderson (eds.): *Mathematics of Neural Networks — Models, Algorithms and Applications*, Kluwer Academic Publishers, 1997 (in press).

[8] V. Beiu. *VLSI Complexity of Discrete Neural Networks*. Gordon and Breach, Newark, New Jersey, 1997 (in press).

[9] V. Beiu and T. de Pauw. Tight Bounds on the Size of Neural Networks for Classification Problems. Submitted to the *Intl. Work-Conf. on Artif. and Natural Neural Networks IWANN'97* (Lanzarote, Spain), June 4-6, 1997.

[10] V. Beiu and S. Draghici. Limited Weights Neural Networks: Very Tight Entropy Based Bounds. Submitted to the *2nd Intl. ICSC Symp. on Soft Computing, Fuzzy Logic, Artif. Neural Nets, Genetic Algs. SOCO'97* (Nîmes, France), September 14-17, 1997.

[11] J. Bruck and J.W. Goodmann. On the Power of Neural Networks for Solving Hard Problems. *J. of Complexity*, **6**, 129-135, 1990.

[12] S. Draghici and I.K. Sethi. On the Possibilities of the Limited Precision Weights Neural Networks in Classification Problems. Submitted to the *Intl. Work-Conf. on Artif. and Natural Neural Nets IWANN'97* (Lanzarote, Spain), June 4-6, 1997.

[13] D. Hammerstrom. The Connectivity Analysis of Simple Association –or– How Many Connections Do You Need. In D.S. Touretzky (ed.): *Advances in Neural Information Processing Systems*, Morgan Kaufman, San Mateo, 533-541, 1989.

[14] S. Hu. *Threshold Logic*. University of California Press, Berkeley and Los Angeles, 1965.

[15] O.B. Lupanov. On a Method of Circuit Synthesis. *Izvestia VUZ (Radiofizika)*, **1**, 120-140, 1958.

[16] R.C. Minnik. Linear-Input Logic. *IRE Trans. on Electronic Computers*, **10**, 6-16, 1961.

[17] R. Paturi and M. Saks. On Threshold Circuits for Parity. *Proc. IEEE Symp. on Foundations of Comp. Sci.*, 1990.

[18] N.P. Red'kin. Synthesis of Threshold Circuits for Certain Classes of Boolean Functions. *Kibernetika*, **5**, 6-9, 1970.

[19] R. Smolensky. Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity. *Proc. ACM Symp. on Theory of Computing STOC'87*, 77-82, 1987.

[20] R.C. Williamson. ε-Entropy and the Complexity of Feedforward Neural Networks. In D.S. Touretzky (ed.): *Advances in Neural Information Processing Systems*, Morgan Kaufman, San Mateo, 946-952, 1990.

[21] B.-T. Zhang and H. Mühlenbein. Genetic Programming of Minimal Neural Networks Using Occam's Razor. *Tech. Rep. Arbeitspapiere der GMD 734*, Schloß Birlinghoven, Sankt Augustin, 1993.

[1] Many thanks to Kalpak Dighe who suggested the particular approach from Fig. 5.a for the 'analogue' 4-input PARITY problem.