$O(n\log^3 n)$. Note that this algorithm produces the correct result provided the input is valid.

ACKNOWLEDGMENTS

We wish to thank Prof. Sartaj Sahni for suggesting this problem and the anonymous referee who suggested an alternate way of merging local segments (Section IV.C), that resulted in an improvement of a $O(\log^{1/2} n)$ factor. The research of Ramakrishna Thurimella was supported in part by National Science Foundation grant CCR-9210604.

REFERENCES

[1] P.K. Agarwal, "Partitioning arrangements of lines I: An efficient deterministic algorithm," *Discrete Comput. Geom.*, vol. 5, pp. 449-483, 1990.

[2] P.K. Agarwal, "Partitioning arrangements of lines II: Applications," *Discrete Comput. Geom.*, vol. 5, pp. 533-573, 1990.

[3] A.V. Aho, J.E. Hopcroft, and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, 1974.

[4] H. Edelsbrunner, L.J. Guibas, and J. Stolfi, "Optimal point location in a monotone subdivision," *SIAM J. Comput.*, vol. 15, no. 2, pp. 317-340, 1986.

[5] H. Edelsbrunner, L.J. Guibas, and M. Sharir, "The complexity and construction of many faces in an arrangement of lines and of segments," *Discrete Comput. Geom.*, vol. 5, pp. 161-196, 1990.

[6] H. Edelsbrunner, J. O'Rourke, and R. Seidel, "Constructing arrangements of lines and hyperplanes with applications," *SIAM J. Comput.*, vol. 15, no. 2, pp. 341-363, 1986.

[7] H. Edelsbrunner, J. van Leeuwen, T. Ottmann, and D. Wood, "Computing connected components of simple rectilinear geometrical objects in *d*-space," *RAIRO Theoretical Informatics*, vol. 18, no. 2, pp. 171-183, 1984.

[8] L.J. Guibas and J. Saxe, "Problem 80-15," *J. Algorithms*, vol. 4, pp. 176-181, 1983.

[9] L.J. Guibas and M. Sharir, *Computing the Unbounded Component of an Arrangement of Line Segments*, manuscript, (1987).

[10] M. Lopez, R. Janardan, and S. Sahni, "A fast algorithm for VLSI net extraction," *Proc. IEEE/ACM Int'l Conf. CAD*, pp. 770-774, 1993.

[11] J.S.B. Mitchell, *On Computing a Single Face in an Arrangement of Line Segments*, manuscript, (1990).

[12] H.G. Mairson and J. Stolfi, "Reporting and counting intersections between two sets of line segments," *NATO ASI series, Theoretical Foundations of Computer Graphics and CAD*, Springer-Verlag, pp. 307-325, 1988.

[13] J. Matoušek, "Cutting hyperplane arrangements," *Discrete Comput. Geom.*, vol. 6, pp. 385-406, 1991.

[14] J. Matoušek, "Range searching with efficient hierarchical cuttings," *Proc. 8th Ann. ACM Symp. Comput. Geom.*, pp. 276-285, 1992.

[15] S. Nahar and S. Sahni, "Time and space efficient net extractor," *Computer Aided Design*, vol. 20, no. 1, pp. 17-26, 1988.

[16] F. Preparata and M.I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1988.

On a Constant-Time, Low-Complexity Winner-Take-All Neural Network

Yuen-Hsien Tseng and Ja-Ling Wu

***Abstract*—A nearly cost-optimal winner-take-all (WTA) neural network derived from a constant-time sorting network is presented. The resultant WTA network has connection complexity $O(n^{2^s/(2^s-1)})$ where s is the depth of cascaded sorting networks. Application of the WTA network to other problems such as nonbinary majority is also included.**

***Index Terms*—Quadratic perceptron, sorting, winner-take-all, nonbinary majority, complexity.**

I. INTRODUCTION

Winner-take-all (WTA) is an important operation in many neural network models [1], [2], [3] to identify the neuron with the maximum (or minimum) activation among a set of n neurons. There are a number of neural networks proposed for this purpose. Some are recurrent, the output of a neuron feedbacks to itself and others. The recurrent networks [3], [4], [5], [6] with constant inhibitive weights converge in O(n*log*n) iterations, and converge in O(n) time steps with variable weights [4]. The magnitude preserving recurrent MAX net converges in O(*log*n) iterations for many commonly occurring distributions [5]. Another class of WTA nets use feedforward networks [6], [7], [8]. One realization is a binary tree made up of two-input comparators [6], [7], which has *log*n levels of circuit delay. The network in [8] implemented in MOS VLSI circuits has O(n) network complexity. This circuit demands the maximum input be larger than all the others by a significant amount for the winner to suppress the others to near zero.

In this paper, the perceptron networks with quadratic polynomials as their discriminant functions are used to solve the winner-take-all problems. The basic idea used in constructing a neural network to solve these problems is to express the solution in logic forms and then convert them into a set of discriminant functions. The resulting networks have low network complexity and short propagation delay, making them applicable as primitives of more complex computations.

The output function of a general perceptron can be described as [9]

$$z = \mathrm{sgn}_0(g(X)) \tag{1}$$

In the above equation, $X = [x_1, x_2, ..., x_n] \in R^n$ is the network input; sgn_0 is a dichotomy function: $\mathrm{sgn}_0(a) = 1$ if $a > 0$, 0 if $a \le 0$; and g, called a discriminant function, is an *r*th-order polynomial:

$$g(X) = w_1 f_1(X) + w_2 f_2(X) + ... + w_N f_N(X) + w_0 \tag{2}$$

where w_i are called weights and each product term $f_i(X)$ is of the form: , $x_{k_1} x_{k_2} \cdots x_{k_r}$, $k_1, k_2, ..., k_r \in \{1, 2, ..., n\}$. A linear perceptron has a discriminant function of order $r = 1$. A perceptron with $r = 2$ is called a quadratic perceptron, whose discriminant is also called a gating activation function in [10] since the two variables in a product term can be seen as one gates the other. Especially, the quadratic perceptron presented below has the property that in each two-variable term, the weight of the product term is +1 and one variable is

Manuscript received December 21, 1992; revised February 1994.

Y.-H. Tseng is with the Republic of China Army General Headquarters, Lung Tan, Taoyun, Taiwan, ROC.

J.-L. Wu is with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei 10764, Taiwan, ROC.

IEEECS Log Number C95026.

binary, the other is a real number, making possible the binary variable gating the flow of the other variable when they are implemented in VLSI circuits.

The WTA net is derived from a previously developed constant-time sorting network [11] of connection complexity $O(n^2)$. We first describe it in the next section. We then present the WTA net and illustrate a divide-and-conquer technique in this paper to reduce the network complexity to approximate its optimal solution, namely O(n). Application of the WTA net to nonbinary majority is described in Section IV.

II. Sorting Networks

Among all computational models studied over the past, sorting appears to be a vital operation included as a primitive in many computing tasks. Previous works on application of neural networks to sorting problems used Hopfield model and multilayer perceptrons for constant-time computing [12], [13], [14], [15]. The recurrent network [12] uses $O(n^2)$ neurons to compute a permutation matrix indicating an increasing sequence of n positive numbers in two iterations. The networks [13], [14] of linear perceptrons whose weights' magnitudes are bounded by a polynomial in n computes a sorted sequence in constant depth and polynomial network size. An improvement on the depth of such polynomial-size sorting networks has been made in [15]. Our approach uses quadratic perceptrons to exploit their expressive power in representing solutions of problems and ease of implementation.

An unordered sequence $X = \{x_1, x_2, ..., x_n\}$ of real numbers can be sorted into an ordered sequence $S = \{s_0, s_1, ..., s_{n-1}\}$ by first determining the ranks of the numbers and then outputting them according to their ranks [16]. The rank of number x_i is the number of elements in S proceeding x_i. So it can be expressed as:

$$R(x_i) = \sum_{j=1}^{n} \mathrm{sgn}_0(x_i - x_j) \tag{3}$$

The smallest number has rank 0, the second is of rank 1, and so on. This rank will lead to a sorted sequence in increasing order and is thus called an *increasing rank*. In the same way, one can define a *decreasing rank* to result in a decreasing sequence as follows.

$$R^*(x_i) = \sum_{j=1}^{n} \mathrm{sgn}_0(x_j - x_i) \tag{4}$$

The above rank function results in a two-layer network: the first layer is a set of n linear perceptrons each of which has only two inputs; the second layer performs linear sums of the comparison results. The perceptrons in the second layer are those without the dichotomy function and thus their outputs can be immediately directed to the next layer.

The next step is to output the number with rank k to the correct position. A direct translation of this operation into a logic expression takes the form:

$$s_k = \sum_{i=1}^{n} EQ(k = R(x_i))x_i \tag{5}$$

where EQ(s = t) is a predicate, EQ = 1 if s = t, and EQ = 0 if $s \neq t$. The expression is further converted into a form implementable by quadratic perceptrons using a technique similar to those in [13], [14]:

$$s_k = \sum_{i=1}^{n} \left[\mathrm{sgn}_0(k+1-R(x_i)) + \mathrm{sgn}_0(R(x_i)-k+1) - 1\right]x_i \tag{6}$$

It can be seen that the expression in the square brackets is 1 if $k-1 < R(x_i) < k+1$ and 0 if $R(x_i) \geq k+1$ or $R(x_i) \leq k-1$, exactly meets the requirement of the predicate EQ. The resulting sorting network is shown in Fig. 1.

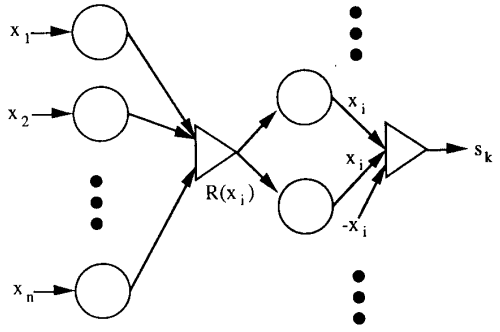

Fig. 1. The sorting network, where circles denote linear perceptrons and triangles denote direct sums of input signals without going through dichotomy functions. This figure shows the part that computes the *k*th element in the sorted sequences.

In parallel computing, an algorithm is often evaluated by a cost function defined as the product of hardware complexity and time complexity. For the sorting problem, there are n! possible permutations of n inputs and log(n!) (i.e. of order n*log*n) bits are needed to distinguish among them. In the worst case, any algorithm requires on the order of n*log*n steps at least to recognize a particular output. Thus the optimal cost for sorting is O(n*log*n) [16]. The above sorting network contains n rank functions, each has O(n) connections, and n outputs, each also has O(n) connections. The connection complexity is $O(n^2)$ but the computation time is a constant function of input size. Thus its cost is $O(n^2)$. Although it is not cost optimal, it has a simple structure for implementation.

III. Winner-Take-All Networks

With the idea of the above sorting network, winner-take-all problems can be solved accordingly. We consider the problem in two versions: MIN, select the minimum; MAX, the maximum.

The MIN network can be derived from choosing the increasing rank defined in (3) and the first output s_0 of the sorting network. If the index instead of the value is needed, s_0 can be modified to output the index of the minimum number:

$$\mathrm{MIN} = s_0 = \sum_{i=1}^{n} EQ(0 = R(x_i))i \tag{7}$$

The index can be coded in such a way that if there are more than one number whose rank is zero, the output s_0 in (7) still indicates which one contains the minimum. For example, we can use one-out-of-n code and interpret each index in binary number, which is equivalent to replacing i in (7) with 2^i. So any two numbers added together will lead to another distinct number.

Note if there are at least two equal maxima in the unordered sequence X, there will be no number whose rank is n − 1, as is defined in (3). Thus to obtain the MAX network, we have to choose the decreasing rank in (4) and the first output s_0.

The optimal solution for the selection problem is of cost O(n) [16]. The above winner-take-all network requires constant-time computation and $O(n^2)$ connection complexity. The network complexity

can be reduced to approximate the optimal solution by use of a divide-and-conquer method. The idea is to divide the n elements into n^{1-x} groups, each of which has n^x elements, as is shown in Fig. 2. Each group selects its minimum with a MIN network of complexity provisionally $O(n^y)$. Since there are n^{1-x} groups, the connection complexity needed is $O(n^{1-x+xy})$. The minimum over all the n elements is then chosen from these n^{1-x} values by a similar MIN network, for which the network complexity is $O(n^{y(1-x)})$. We would like to minimize both the orders $1 - x + xy$ and $y(1 - x)$. The value that makes them minimum when $y = 2$ is at $x = 1/3$, so the total network complexity is $O(n^{4/3})$.

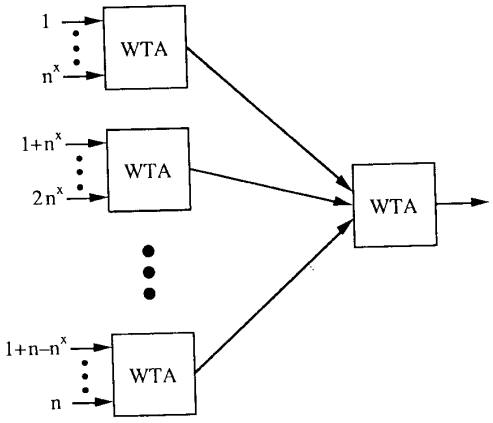

Fig. 2. The multi-stage winner-take-all network.

The newly obtained WTA net can be recursively applied in Fig. 2 to yield a multi-stage WTA net of lower complexity. After the *ith* application, the network has 2^i stages, the partition factor

$$x = \frac{1}{2^{2^{i-1}}+1} \tag{8}$$

and the connection complexity of order

$$y = \frac{2^{2^i}}{2^{2^i}-1} \tag{9}$$

The above order approaches to 1 in an exponential rate, which is faster than the rate of the propagation delay due to the growth of the stages.

IV. Non-Binary Majority

Binary majority operation is a linearly separable problem and can be trivially solved by a linear perceptron by summing all its inputs [17]. However, the solution for the majority of nonbinary inputs is not so obvious without the help of the above idea of expressing solutions representable by perceptrons. The majority of m candidates cast by n voters is computed by a network which first counts the number of votes for each candidate and then feeds these numbers to the MAX network. The votes for candidate i are denoted as V_i, for i = 1, 2, ..., m, and

$$V_i = \sum_{j=1}^{n} EQ(i = x_j) = \sum_{j=1}^{n} \left[sgn_0(i+1-x_j) + sgn_0(x_j - i + 1) - 1 \right] \tag{10}$$

The set of numbers $\{V_1, V_2, ..., V_m\}$ is then fed to the MAX network to result in a correct output. The constant-time majority network, of complexity approximately O(nm), can be applied to decode those majority-decodable codes of nonbinary case [18], for example.

V. Conclusion

We have presented efficient neural network solutions to sorting, winner-take-all, and nonbinary majority problems which are important primitives for neural networks themselves and other computation models. The main idea used is to convert the solution, expressed in logic form, into a set of discriminant functions of the perceptrons. This is quite a systematic method to construct neural networks in solving problems since it is about the same level of complexity as writing an algorithm. Application of this approach to other problem such as decoding error-correcting codes has been presented in [19].

Extension of the above result to a k-winner network is directly. A k-winner-take-all operation is to identify k neurons with larger activation values among n neurons [20], [21]. It can be performed by choosing the first k outputs of the sorting network. The resultant constant-time k-winner network requires the n inputs be distinct because equal inputs, having the same ranks, are passed to the same outputs, leaving some s_k void.

References

[1] T. Kohenon, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

[2] D.E. Rumelhart and D. Zipser, "Feature discovery of competitive learning," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D.E. Rumelhart and PDP research group, MIT Press, Cambridge, Mass., 1986.

[3] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architecture," *Neural Networks*, vol. 1, pp. 17-61, 1988.

[4] B.W. Suter and K.D. Reilly, "On the performance of maximum-picking neural networks," *Int'l J. Neural Networks*, vol. 3, no. 3, pp. 85-96, Sept. 1992.

[5] B.W. Suter and M. Kabrisky, "On a magnitude preserving iterative MAXnet algorithm," *Neural Computation*, no. 4, pp. 224-233, 1992.

[6] R.P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, pp. 4-22, Apr. 1987.

[7] G.R. Gindi, A.F. Gmitro, and K. Parthasarathy, "Winner-take-all networks and associative memory: Analysis and optical implementation," *Proc. Int'l Conf. Neural Networks*, vol. III, pp. 607-614, San Diego, Calif., 1987.

[8] J. Lazzaro, M. Rychebush, A. Mahowald, and C.A. Mead, "Winner-take-all networks of O(n) complexity," *Advances in Neural Information Processing Systems*, vol. 1, pp. 703-711, Palo Alto, Calif., Morgan Kaufmann, 1989.

[9] N.J. Nilsson, *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*, McGraw-Hill, 1965.

[10] R.J. Williams, "The logic of activation functions," *Parallel Distributed Processing*, D.E. Rumelhart and PDP research group, MIT Press, Cambridge, Mass., 1986.

[11] Y.-H. Tseng and J.-L. Wu, "Solving sorting and related problems by quadratic perceptrons," *Electronics Letters*, vol. 28, no. 10, pp. 906-908, 1992.

[12] Y. Takefuji and K.-C. Lee, "A super-parallel sorting algorithm based on neural networks," *IEEE Trans. Circuits and Systems*, vol. 37, no. 11, pp. 1425-1429, Nov. 1990.

[13] A.K. Chandra, L. Stockmeyer, and U. Vishkin, "Constant depth reducibility," *SIAM J. of Computing*, vol. 13, no. 2, pp. 423-439, May 1984.

[14] K.-Y. Siu and J. Bruck, "Neural computation of arithmetic functions," *IEEE Proc.*, vol. 8, no. 10, pp. 1661-1675, Oct. 1990.

[15] K.-Y. Siu and J. Bruck, "On the power of threshold circuits with small weights," *SIAM J. of Discrete Math.*, vol. 4, no. 3, pp. 425-435, Aug. 1991.
[16] G. Selim, *AKL, The Design and Analysis of Parallel Algorithms*, Prentice Hall, 1989.
[17] B. Widrow and R. Winter, "Neural nets for adaptive filtering and adaptive pattern recognition," *Computer*, pp. 25-39, Mar. 1988.
[18] R.E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, 1983.
[19] Y.-H. Tseng, J. Shiu, and J.-L. Wu, "Perceptron-based neural networks for decoding error-correcting codes," *IEEE Int'l Workshop Intelligent Signal Processing and Comm. Systems*, Taipei, pp. 11-22, 1992.
[20] W.J. Wolfe, D. Mathis, C. Anderson, J. Rothman, M. Gottler, G. Brady, R. Walker, G. Duane, and G. Alaghband, "K-winner networks," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 310-315, 1992.
[21] E. Majani, R. Erlanson, and Y. Abu-Mostafa, "On the k-winner-take-all networks," *Advances in Neural Information Processing Systems I*, D.S. Touretzky, ed., pp. 634-624, Los Altos, Calif., Morgan Kaufmann, 1989.

An Optimal Algorithm for Permutation Admissibility to Multistage Interconnection Networks

Xiaojun Shen, *Member, IEEE*, Mao Xu, and Xiangzu Wang

***Abstract*—This paper introduces a simple $O(NlogN)$ sequential algorithm that determines the admissibility of an arbitrary permutation to an $N \times N$ Multistage Cube-Type Network (MCTN) implemented by 2×2 switching elements (SEs) in contrast to previous $O(Nlog^2N)$ algorithms. It is proven that the new algorithm is optimal in the sense that any algorithm, based on bit-operations, has to examine at least $(N/4)logN$ different bits among the total $NlogN$ bits in the binary representations of the destinations numbered from 0 through $N-1$.**

***Index Terms*—Adversary strategy, multistage interconnection network, permutation admissibility, transition matrix, window method.**

I. Introduction

An $N \times N$ Multistage Interconnection Network (MIN) provides connections between N sources and N destinations. The class of Multistage Cube-Type Networks (MCTNs) refers to those MINs which are topologically equivalent to the Generalized Cube, including Omega, Baseline, Indirect Cube, and Regular SW Banyan with S = F = 2 [3]. Because of the equivalence, this paper studies Omega networks only, and the results will be applicable to any MCTNs. An $N \times N$ ($N = 2^n$) Omega network [12] is implemented by $n = logN$ stages of 2×2 switching elements (SEs). We assume:

1) The stages are numbered 0 through $n-1$ from left to right.
2) The N inputs to (and outputs from) each stage are addressed 0 through $N-1$, top to bottom, in binary, $p_0 p_1 \ldots p_{n-2} p_{n-1}$.
3) The SEs at each stage are labeled 0 through $(N/2)-1$, top to bottom, in binary.
4) The perfect shuffle [13] is used to connect two adjacent stages. That is, the output $p_0 \ldots p_{n-2}\, p_{n-1}$ from stage i is connected to the input $p_1 \ldots p_{n-1}\, p_0$ of stage $i+1 (0 \le i \le n-2)$. The N source inputs are also shuffled before connecting to stage 0.

Manuscript received January 22, 1993; revised August 1993.
The authors are with the Computer Science Telecommunications Program, University of Missouri-Kansas City, 5100 Rockhill Road, Kansas City, MO 64110; e-mail xshen@cstp.umkc.edu.
IEEECS Log Number C95027.

Given a source $\mathbf{S} = s_0 \ldots s_{n-2}\, s_{n-1}$ and a destination $\mathbf{D} = d_0 \ldots d_{n-2}\, d_{n-1}$, there is a unique path from **S** to **D**. The sequence $s_0 \ldots s_{n-2}\, s_{n-1}\, d_0 \ldots d_{n-2}\, d_{n-1}$ (denoted by [**S**, **D**]) is called the transition sequence of this path, because the exit address taken by this path at stage i is exactly equal to the binary number, $s_{i+1}\, s_{i+2} \ldots s_{n-1}\, d_0\, d_1 \ldots d_i$, of consecutive n bits in this trasnsition sequence starting from s_{i+1} ([10]). The SE at stage i which meets the path has the label $s_{i+1} \ldots s_{n-2}\, s_{n-1}\, d_0\, d_1 \ldots d_{i-1}$. The setting of this SE depends on the ith bit of $\mathbf{S} \oplus \mathbf{D}$ ($\oplus$ is the bit-wise exclusive-or operation), i.e., $s_i \oplus d_i$. If $s_i \oplus d_i = 0$, this SE is set to *through*; otherwise, it is set to *cross*. Two such paths will conflict if and only if they meet at a common exit port of some SE (Fig. 1). Thus, the path from **S** to **D** conflicts with the path from **S′** to **D′** if and only if

$$s_{i+1} \cdots s_{n-2} s_{n-1} d_0 d_1 \cdots d_i = s'_{i+1} \cdots s'_{n-2} s'_{n-1} d'_0 d'_1 \cdots d'_i$$

for some i.

An N-permutation defines N paths by specifying a distinct destination for each of the N sources. A permutation π is said to be admissible to an MCTN if N conflict-free paths, one for each input-output pair, can be established simultaneously. The identity permutation, which maps each source **S** to the destination with the same address **D** = **S**, is obviously admissible. Moreover, because $S \oplus D = 0$, every SE is set to *through* for the identity permutation.

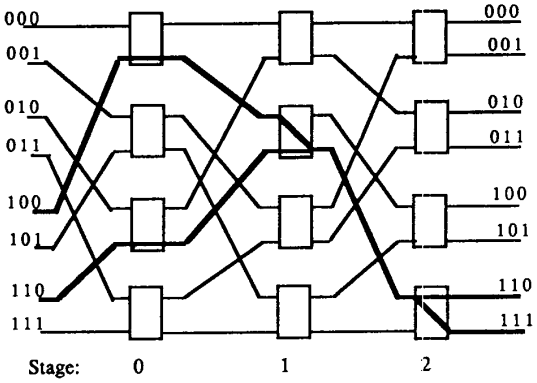

Fig. 1. Path (100, 111) conflicts with path (110, 110) at stage 1.

Because there are only $N^{N/2}$ admissible permutations among $N!$ possible permutations, determining the admissibility of a permutation to an MCTN is an important problem in optimally supporting application needs. In 1990, Hsia and Chen summarized previous $O(Nlog^2N)$ algorithms [6], [7], [8] in their paper [10]. In addition, they introduced an $O(N)$ algorithm in the same paper. However, this complexity is not correct. A detailed analysis of this algorithm will be given in Section IV of this paper.

Given a permutation π, if we list all transition sequences, one for each pair $(k, \pi(k))$ $(0 \le k \le N-1)$, then we obtain an $N \times 2n$ matrix $\mathbf{T}[0 .. 2n-1]$ called *transition matrix*. We define a *window* as a collection of n consecutive columns in **T**. Thus, window $W_i = \mathbf{T}[i .. n+i-1]$ $(0 \le i \le n)$. Obviously, permutation π is admissible if and only if the N rows (i.e. the N n-bit numbers) in any window W_i are distinct. As an example, Fig. 2 shows all windows of the follow-