Achieving Optimality for Gate Matrix Layout and PLA Folding: a Graph Theoretic Approach*

Afonso G. Ferreira[1] *and Siang W. Song*[2]

[1] LIP/IMAG - CNRS, École Normale Supérieure de Lyon and University of São Paulo, Institute of Mathematics and Statistics, Department of Computer Science
[2] University of São Paulo, Institute of Mathematics and Statistics, Department of Computer Science

1 Introduction

The Programmable Logic Array (PLA) is a regular layout of logical gates that is commonly used to implement multiple-output combinatorial logic circuits. Logical functions in the canonical form of sum of products can easily be implemented by a PLA. As we shall see later, a PLA consists of a number of rows and columns. At each intersection of a row with a column a logical gate can be placed. The number of logical gates actually placed is often very small relative to the total number of row-column intersections. PLA *folding* is a technique that, given a PLA, attempts to produce an equivalent folded PLA that occupies less space. PLA folding problem and the gate matrix layout problem have been extensively discussed in the literature [2, 6, 7, 10]. As noted in [3], the same ideas for the problem of multiple folding of PLA also apply to the gate matrix layout problem. For uniformity in notation, we adopt the PLA problem formulation.

A PLA is represented by a 0-1 personality matrix where a 1 denotes the presence of a logical gate. We assume $n = \max(r, c)$ for a PLA personality matrix with r rows and c columns. The optimal multiple/simple PLA folding and the gate matrix layout problems have been proven to be NP-complete [8, 11]. We present $O(n^3)$ heuristic algorithms for PLA folding based on a new approach. The key point of our approach is the separation of the folding problem in two distinct phases: a permutation step in which we search for a good configuration of the personality matrix, and a compaction step where the actual folding takes place. With this we realize that the real difficulty of the problem stems from the permutation step, since the optimality of the compaction step is always achieved in polynomial time.

As for the permutation step, we give a sufficient condition for the solution to be optimal. Verification of the sufficient condition constitutes part of the proposed algorithms, and is in fact used to obtain the solution. The sufficient condition for optimality is that a certain submatrix of the personality matrix, called its reduced personality matrix, possesses the consecutive ones property for columns. The relation between these layout problems and matrices with the consecutive ones property has been established independently by

* Supported by the Program BID/USP - Project 30.01, FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) - Proc. No. 88/3095-9 and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) - Proc. No. 306063/88-3 and PROTEM-CC/SP. This paper was written during a visit of the second author to LIP-IMAG/ENS Lyon.

Deo, Krishnamoorthy and Langston [2] and Ferreira [4]. Certain matrix classes have been shown in the literature that do not possess the consecutive ones property. We show that the class of personality matrices that do not satisfy the sufficient condition for optimality is strictly smaller than the class of matrices that do not possess the consecutive ones property. We also discuss some lower bounds of the optimal solution, based on the maximum number of 1's in each row, the size of the maximum clique and the chromatic number of the associated column adjacency graph. Furthermore, we show that the proposed algorithms do not suffer from the shortcomings of a number of other heuristic algorithms, as pointed out by Deo et al. in [2].

Many heuristic algorithms for the multiple/simple PLA folding problems have been proposed. Performance studies of such heuristic algorithms are based only on test runs on experimental data. Lecky et al. [10] considered disjoint column-pair compatibility graphs where cliques correspond to PLA folding sets. Near-optimal cliques can often been obtained by their proposed heuristic algorithms of $O(n^4)$ and $O(n^5)$. Performance evaluation was done through numerous test runs and experimental results show the proximity of the obtained results to the optimal solution, which was known beforehand. Hwang et al. [7] proposed $O(n^3)$ and $O(n^4)$ heuristic algorithms based on longest paths on the associated disjoint graph generated from the personality matrix. Through experimental testing, they compared their work with that of Hachtel et al. [6]. See [10] for a brief history of the development of PLA folding heuristics. See [9] and [13] for some recent works related to gate matrix layout.

2 The Optimal PLA Folding Problem

In a PLA the input signals and their complements run, say vertically, through a matrix of circuit elements called the *AND plane*. Combinations of the inputs and their complements are selected in this AND plane. These signals then run horizontally and become inputs to another matrix of circuit elements called the *OR plane*. The outputs of the OR plane are the canonical sum-of-products form of Boolean functions of the PLA inputs. Consider the following logic functions to be implemented by a PLA:

$$f_6 = \bar{v_1}v_5 + \bar{v_2}\bar{v_3} + v_1v_3 + \bar{v_5}$$
$$f_7 = v_2v_4 + \bar{v_2}\bar{v_4}$$

where v_1, v_2, v_3, v_4 and v_5 are variables and $\bar{v_i}$ denotes the complement of v_i. The corresponding PLA is illustrated in Fig. 1. In the illustration, a dot denotes the presence of a logical circuit. In the AND plane input variables and their complements drive logical circuits to form the corresponding product terms. These then run horizontally to drive logical circuits in the OR plane to form the results f_6 and f_7.

Personality matrix Consider input variables v_1, v_2, …, v_k and output functions f_{k+1}, f_{k+2}, …, f_c with a total of r product terms. The corresponding PLA can be represented by a $r \times c$ *personality matrix* $M = (m_{ij})$, a (0,1)-matrix defined as follows.

$$\text{for} \quad 1 \leq i \leq r, 1 \leq j \leq c, m_{ij} = \begin{cases} 1 \text{ if a logical circuit is present at } \; v_j, \bar{v_j} \text{ or } f_j \\ 0 \text{ otherwise} \end{cases}$$

The columns of matrix M will be called *logical columns*. The personality matrix M of the PLA of Fig. 1 is shown below.

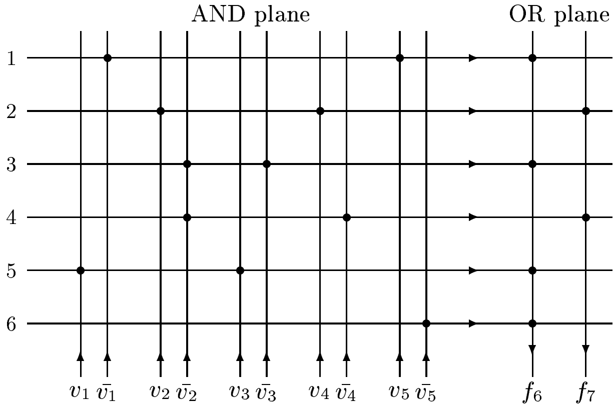


Fig. 1. An example PLA

$$M = \begin{array}{c} \text{row 1} \\ \text{row 2} \\ \text{row 3} \\ \text{row 4} \\ \text{row 5} \\ \text{row 6} \end{array} \begin{pmatrix} 1\,0\,0\,0\,1\,1\,0 \\ 0\,1\,0\,1\,0\,0\,1 \\ 0\,1\,1\,0\,0\,1\,0 \\ 0\,1\,0\,1\,0\,0\,1 \\ 1\,0\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,1\,1\,0 \end{pmatrix}$$

For a logical column j of a given configuration $M' = \left(m'_{ij}\right)$, obtained by row permutation of M, we denote by $\text{Int}(M', j)$ the interval $[a, b]$, where

$$a = \min i \ \text{ such that } m'_{ij} = 1 \ \text{ and}$$
$$b = \max i \ \text{ such that } m'_{ij} = 1.$$

We say that a subset of logical columns of M', denoted by s_j,

$$s_j = \{j_1, j_2, \ldots, j_m\}$$

is *foldable* to a single *physical column j* if and only if

$$\text{Int}(M', j_1) \cap \text{Int}(M', j_2) \cap \ldots \cap \text{Int}(M', j_m) = \phi \ .$$

Given a personality matrix M of zeros and ones, we want to find the configuration, among all the row permutations of the given matrix, whose columns can be foldable to the minimum number of physical columns or subsets s_j.

For our example, we can obtain the following configuration M', with the rows of M permuted as indicated

$$M' = \begin{array}{c} \text{row 2} \\ \text{row 4} \\ \text{row 3} \\ \text{row 5} \\ \text{row 1} \\ \text{row 6} \end{array} \begin{pmatrix} 0\,1\,0\,1\,0\,0\,1 \\ 0\,1\,0\,1\,0\,0\,1 \\ 0\,1\,1\,0\,0\,1\,0 \\ 1\,0\,1\,0\,0\,1\,0 \\ 1\,0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,1\,0 \end{pmatrix}$$

and obtain the subsets $s_1 = \{1, 2\}, s_2 = \{3, 4, 5\}, s_3 = \{6, 7\}$. The folded PLA is shown in Fig. 2.

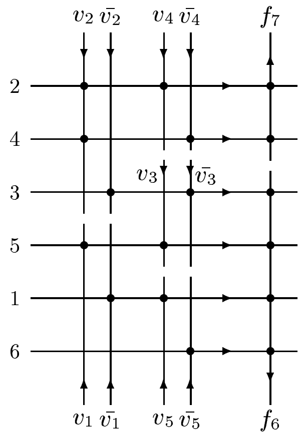


Fig. 2. The folded PLA

The simple PLA folding problem adds the constraint of not folding more than two logic columns in one physical column. In other words, the cardinality of each subset s_j should not be greater than two. Hence, while in multiple folding the best result possible is to fold all the c logical columns to one single physical column, in simple folding we should not expect to do better than $\lceil c/2 \rceil$ physical columns.

Reduced personality matrix: Each row of M consists of 1's and 0's. For each row, we can define an associated set whose elements are the column positions where we have 1's. The following considerations are due to Deo, Krishnamoorthy and Langston [2]. Let us first remove all duplicate rows, so that only distinct rows remain. For the remaining rows, a row is said to be *dominant* if its associated set of column positions with 1's is not a proper subset of those associated with any other row. A row that is not dominant is said to be *dominated*. We will now proceed to remove all dominated rows so as to obtain a final matrix, which we call the *reduced personality matrix* or simply *reduced matrix*, with only dominant rows.

As noted in [2], any given personality matrix can be folded using the same number of physical columns as its reduced matrix. The dominated (resp. duplicate) rows can be repositioned next to their corresponding dominant (resp. original) rows. Hence, for the discussions from now on, we consider the personality matrix already in the reduced form, unless explicitly stated. It can be seen that the cost for removing rows and their subsequent repositioning is $O(n^3)$.

Consider a $r \times c$ personality matrix M and let $n = \max(r, c)$. It is easy to see that if two logical columns of M have a 1 in the same row, then they cannot share the same physical column and hence they cannot belong to the same subset s_j. In other words, they cannot be folded together. On the other hand, once we have found all the subsets s_j it is relatively easy to assign them to new physical columns. So we divide the optimal PLA folding problem into two steps:

1. **Permutation step:** find the permutation of the rows which leads to the minimum number of subsets s_j. The exponential nature of the problem is due to this step.
2. **Compaction step:** compact the matrix obtained by step 1 by merging its logical columns into physical columns. This step is solvable in polynomial time.

The *permutation step* is performed for the entire matrix M, involving both the AND and OR planes. However, due to implementation constraints, the AND plane and the OR plane must be folded independently. Thus the *compaction step* must be performed separately for each plane. Hence the definitions and results for the compaction step hold for each plane separately.

We will use the following terminology, according to Paillottin [14]:

Gate: the part of a logical column between (and including) the two extreme 1's.
Peak function: For a given configuration of matrix M, the peak o row is equal to the number of gates it crosses.
$P*$: the maximum of the peak function taken over all the rows.

Using this terminology, the following relations can be defined over the set of gates.

- Relation COMPLETELY-BELOW: a gate g_1 COMPLETELY-BELOW a gate g_2 if the top of g_1 is below the bottom of g_2.
- Relation ABOVE: the gates can be ordered according to their top. We say that a gate g_1 ABOVE a gate g_2 if the top of g_1 is above the top of g_2.

In the following example, the gates are indicated by the vertical lines. For instance, we have peak(1)=3, peak(6)=2 and $P* = 7$. The gate at column 4 lies between the rows 2 and 4, that is, $\mathrm{Int}(M, 4) = [2, 4]$.

$$\begin{array}{r}
\text{row } 1 \\
\text{row } 2 \\
\text{row } 3 \\
\text{row } 4 \\
\text{row } 5 \\
\text{row } 6
\end{array}
\left(\begin{array}{ccccccc}
|1| & 0 & 0 & 0 & |1| & |1| & 0 \\
|0| & |1| & 0 & |1| & |0| & |0| & |1| \\
|0| & |1| & |1| & |0| & |0| & |1| & |0| \\
|0| & |1| & |0| & |1| & |0| & |0| & |1| \\
|1| & 0 & |1| & 0 & |0| & |1| & 0 \\
0 & 0 & 0 & 0 & |1| & |1| & 0
\end{array}\right)$$

We say that a gate is the *topmost* of a set if there is no other gate ABOVE it. Of course there can be more than one topmost gate in a set.

3 Compaction Step: Polynomial

For a given row permutation of matrix M, given by the permutation step, the compaction step assigns the logical columns to new physical columns.

3.1 Multiple Compaction: $O(n \log n)$

Paillotin [14] describes an $O(n^2)$ algorithm to place the gates of a matrix M into new physical columns. In our terms, it is an algorithm that performs multiple compaction of a specific configuration.

Algorithm 3.1 *Multiple compaction algorithm to allocate a given set of gates to physical columns* (Paillotin [14])*:*

While the set of gates is not empty do the following steps

1. Consider a new physical column.
2. Position the topmost gate on the column and remove it from the set of gates.
3. While the set of the gates COMPLETELY-BELOW the last positioned gate is not empty do
   (a) Position the topmost one of this set on the physical column.
   (b) Remove it from the set of gates.

Such an algorithm led to

Theorem 1. (Paillotin [14])
For a given configuration of a personality matrix M, the gates can be positioned on $P*$ physical columns, which is optimal.

Note that the $O(n^2)$ complexity of Algorithm 3.1 can easily be improved as follows. The algorithm consists roughly of the repetition of the following two steps.

- Step (a): search for a gate satisfying a certain condition (the topmost of all or the one immediately below a given gate).
- Step (b): position it on a certain physical column and remove it from consideration.

We can first sort the gates in descending order of their top positions. Then we construct a balanced binary search tree of these gates with the top positions as key. This is easily done since all the keys are given and already sorted. Steps (a) and (b), which can be performed in $O(\log n)$, are repeated until the tree becomes empty. This gives an $O(n \log n)$ algorithm that shall be referred to as **Algorithm 3.2**.

The problem then is to find the best configuration of M, which is the row permutation that leads to the minimum $P*$. Theorem 1 with Algorithm 3.2 yield the following corollary.

Corollary 2. *A given configuration of a personality matrix M can be optimally compacted in $O(n \log n)$ time.*

3.2 Simple Folding: $O(n^{2.5})$

In the simple PLA folding problem, we position at most two logical columns on each physical column. A matrix with n columns has an obvious lower bound of $\lceil n/2 \rceil$ on the number of physical columns of the folded PLA. Most of the ideas of the multiple folding problem still apply here. We will notice, however, some differences.

For a given personality matrix configuration, Algorithm 3.2, unfortunately, can no longer produce the best compaction. This is easily shown in the example of Fig. 3, where obviously two columns suffice.

We will hence take a different approach. We associate to a given matrix configuration M' a *gate compatibility graph*, $G_c = (V_c, E_c)$, where the vertices v_i in V_c correspond to the columns i of M' and (v_i, v_j) is an edge of E_c if and only if

$$\mathrm{Int}(M', i) \cap \mathrm{Int}(M', j) = \phi \ .$$

$$\begin{pmatrix} 1_a & 0 & 0 & 0 \\ 0 & 1_b & 0 & 0 \\ 0 & 1_b & 1_c & 0 \\ 0 & 0 & 1_c & 1_d \end{pmatrix} \Rightarrow \begin{pmatrix} 1_a & 0 & 0 \\ 1_b & 0 & 0 \\ 1_b & 1_c & 0 \\ 0 & 1_c & 1_d \end{pmatrix}$$

Fig. 3. Algorithm 3.2 fails to produce the best compaction

When compacting the PLA a physical column cannot receive two logical columns whose gates overlap. The only condition for folding two logical columns is that the corresponding vertices in the associated gate compatibility graph are adjacent.

Lemma 3. *Given a configuration of a personality matrix M, a matching on the associated gate compatibility graph induces a simple compaction of that configuration.*

There are several polynomial-time solutions for the maximum matching problem in general graphs. Here we refer to the algorithm by Micali and Vazirani [12] which is $O(\mid V \mid^{1/2} \mid E \mid)$. Hence we have:

Theorem 4. *Given a configuration of a matrix M, with its associated gate compatibility graph $G_c = (V_c, E_c)$, we can obtain the optimal simple compaction in $O(\mid V_c \mid^{1/2} \mid E_c \mid)$, which is at most $O(n^{2.5})$.*

4 Complexity Results

By Theorem 1 we determine the exact number of necessary physical columns for a given matrix configuration in the multiple folding. There is at least one row permutation of matrix M that yields the best possible multiple compaction, i.e., the one which leads to the minimum $P*$, denoted by $P*_{\min}$. In this way any compacted configuration will need $P*_{\min}$ or more physical columns to be implemented.

If we find a lower bound for $P*_{\min}$ we will have a bound on how far a heuristic solution is from the optimal. By knowing beforehand the mininum number of necessary physical columns we can decide whether or not a solution is good.

A gate may have 1's and 0's. In a permutation in which there are only 1's in the gates (with no intervening 0's in between), the maximum of the peak function occurs at the row crossing the maximum number of 1's.

Let Ω be the function that gives for each row, the number of 1's it crosses. Let $\Omega*$ be the maximum of Ω computed over all the rows. Note that Ω and $\Omega*$ are characteristic of a given matrix, independent of any permutation of its rows. Hence we have:

Proposition 5. *For any given personality matrix M, $\Omega* \leq P*_{\min}$, i.e., a multiple folding that positions the logical columns of M in $\Omega*$ physical columns is optimal.*

In the simple folding, the use of the gate compatibility graph and a maximum matching algorithm give us the optimal number of physical columns for compacting a given matrix configuration. There is at least one row permutation of the given personality matrix that gives the best possible compaction.

It is clear that whenever a folding reduces a personality matrix of n columns to $\lceil n/2 \rceil$ physical columns, the folding is optimal. This lower bound independs on the characteristics of the PLA. On the other hand, $\Omega*$, the maximum number of 1's computed over all the rows, is always a lower bound for the folding. Hence the proposition below follows.

Proposition 6. *Given a personality matrix M of n columns, the optimal simple folding compacts it in no less than* $\max\{\lceil n/2 \rceil, \Omega*\}$ *columns.*

Unfortunately, in the simple folding, $P*$ columns are necessary but not always sufficient for compacting the n gates of the matrix. Such an example is shown in Fig. 4.

$$\begin{pmatrix} |1| & 0 & |1| & 0 & 0 & 0 \\ |0| & |1| & |0| & |1| & 0 & 0 \\ |1| & |1| & |0| & 0 & |1| & 0 \\ |0| & |1| & |0| & 0 & 0 & |1| \\ |0| & |0| & |1| & 0 & 0 & 0 \\ |1| & |1| & 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 4. The simple folding needs 5 columns (more than $P* = 4$)

Since we cannot always compact a matrix into $P*$ columns exactly, we cannot use the function Ω and its maximum $\Omega*$ to show a sufficient condition for an optimal simple folding, as for the multiple folding. We will show this in another way.

Consider a given matrix configuration M' of M and the following notation.

Best-column-pairs(M'): the set of the column pairs of M' that are folded in an optimal simple compaction of M'

Rows-with-1(M', j): the set of the rows from the matrix M' with a 1 in column j.

We note without proof the following proposition.

Proposition 7. *Given a personality matrix M, if a row r has a 1 in column j, then r crosses the gate of column j in any matrix configuration M', i.e., if $r \in$ Rows-with-1(M, j) then $r \in$ Int(M', j).*

And therefore we have the following lemma.

Lemma 8. *Let M' be a configuration of a personality matrix M. If $\{i, j\} \in$ Best-column-pairs(M') then $\{i, j\}$ can be folded together in a row permutation $\bar{M}$ in which the gates are zero-free (i.e., have only 1's), if it exists.*

Proof. As $\{i, j\} \in$ Best-column-pairs(M'), we have

$$\text{Int}(M', i) \cap \text{Int}(M', j) = \phi \ .$$

Thus,

$$\text{Rows-with-1}(M', i) \cap \text{Rows-with-1}(M', j) = \phi \ .$$

Hence if the consecutive ones permutation $\bar{M}$ exists, then, from Proposition 7:

$$\mathrm{Int}(\bar{M}, i) \cap \mathrm{Int}(\bar{M}, j) = \phi \ .$$

So columns i and j can be folded together in $\bar{M}$. □

Now we are ready to state the main result for the simple folding problem.

Theorem 9. *Given a personality matrix M, a row permutation $\bar{M}$ in which the gates are zero-free, if it exists, is an optimal configuration, meaning that it leads to an optimal folding after the compaction step.*

Proof: In the multiple folding setting, the proof is straightforward with the use of Proposition 5, since $P* \geq \Omega*$.

For the simple folding, let $\bar{C}$ be a configuration of the given matrix that leads to an optimal simple folding. Since $\bar{C}$ is optimal, we must have

$$\mid \text{Best-column-pairs}(\bar{M}) \mid \leq \mid \text{Best-column-pairs}(\bar{C}) \mid \ .$$

On the other hand, Lemma 8 holds for all column pairs $\{i, j\} \in$ Best-column-pairs($\bar{C}$). Therefore, we also have

$$\mid \text{Best-column-pairs}(\bar{C}) \mid \leq \mid \text{Best-column-pairs}(\bar{M}) \mid \ .$$

Thus it follows that

$$\mid \text{Best-column-pairs}(\bar{C}) \mid = \mid \text{Best-column-pairs}(\bar{M}) \mid \ .$$

So $\bar{M}$ leads to an optimal simple folding of the given matrix M. □

4.1 Further Lower Bounds

Consider the *column adjacency graph* $G_a = (V_a, E_a)$ associated with a given personality matrix M. The vertices v_j of V_a correspond to the columns j of M. Two vertices v_i and v_j are adjacent if both columns i and j of M have a 1 in a same row. Figure 5 shows some matrices and their associated column adjacency graphs.

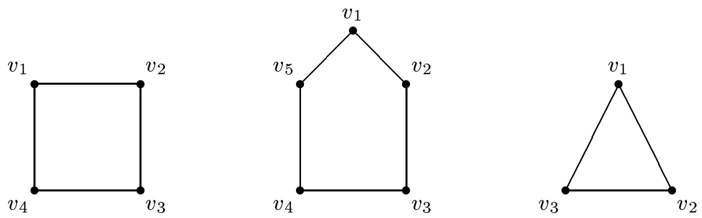
Observe that if two vertices v_i and v_j are adjacent in the column adjacency graph, then $\mathrm{Int}(M', i) \cap \mathrm{Int}(M', j) \neq \phi$ for any configuration M' of M. So a necessary condition to be able to position two columns i and j on a same physical column is that vertices v_i and v_j are not adjacent. Note that this condition is not sufficient as can be easily shown in Fig. 5(a): v_2 and v_4 are not adjacent but $\mathrm{Int}(M, 2) \cap \mathrm{Int}(M, 4) \neq \phi$. The following theorem can be easily shown.

Theorem 10. *For any given personality matrix M and its associated column adjacency graph, we have*

$$\Omega* \leq \mid \textit{max clique} \mid \leq \ \textit{Chromatic number} \ \leq P*_{\min} \ .$$

All the inequalities of the above theorem can be strict, as shown by Fig. 5. Figure 5(a) shows a case where $P*_{\min} >$ Chromatic number. Figure 5(b) shows a case where Chromatic number $> \mid$ max clique $\mid$. Figure 5(c) shows a case where $\mid$ max clique $\mid > \Omega*$.

$$M = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad M = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$



(a)	(b)	(c)
$P{*}_{\min} = 3$	Chromatic no. =3	$\lvert$max clique$\rvert$ =3
Chromatic no. =2	$\lvert$max clique$\rvert$ =2	$\Omega{*} = 2$

Fig. 5. Some column adjacency graphs

5 Heuristics for the Permutation Step

A row permutation of M whose gates have only 1's is optimal for either folding. Hence it suffices to provide an algorithm to find such an optimal configuration, if it exists.

A matrix in which a row permutation has zero-free gates possesses the *Consecutive Ones Property* for columns [5]. Consider a $r \times c$ matrix, with d non-zero entries. An $O(r + c + d)$ algorithm by Booth and Lueker [1] tests if a given matrix possesses the consecutive ones property. The algorithm gives, if it exists, a row permutation of the matrix whose gates have consecutive ones. We are now able to state the following theorem, which stems directly from Theorem 9 and from what has been mentioned above.

Theorem 11. *Given a $r \times c$ personality matrix M with d non-zero entries, if it has the consecutive ones property for columns, then we can get its optimal configuration in $O(r + c + d)$, i.e., at most $O(n^2)$ time.*

The sufficient condition for optimality is that matrix M, in its reduced form meets the condition of Theorem 11.

5.1 Bad Matrices

Tucker[17] shows a class of matrices that do not possess the consecutive ones property for columns. A matrix in such a class has a permutation that contains as its submatrix one of the following five matrices S_1 to S_5. S_1 is a $k \times k$ matrix with $k \geq 3$, S_2 is $k \times (k + 1)$ with $k \geq 3$ and S_3 is $k \times k$ with $k \geq 4$.

$$
S_1 = \begin{pmatrix}
1 & & & & & 1 \\
1 & 1 & & 0 & & 0 \\
 & 1 & & & & 0 \\
 & & \ddots & & & \vdots \\
 & & & 1 & & 0 \\
 & 0 & & 1 & 1 & 0 \\
 & & & & 1 & 1
\end{pmatrix}
\quad
S_2 = \begin{pmatrix}
1 & & & & & 0 \\
1 & 1 & & 0 & & 1 \\
 & 1 & & & & 1 \\
 & & \ddots & & & \vdots \\
 & & & 1 & & 1 \\
 & 0 & & 1 & 1 & 1 \\
 & & & & 1 & 0 \\
0 & 0 & \dots & 0 & 0 & 1
\end{pmatrix}
$$

$$
S_3 = \begin{pmatrix}
1 & & & & & 1 & 0 \\
1 & 1 & & 0 & & 1 & 1 \\
 & 1 & & & & 1 & 1 \\
 & & \ddots & & & \vdots & \vdots \\
 & & & 1 & & 1 & 1 \\
 & 0 & & 1 & 1 & 1 & 1 \\
 & & & & 1 & 0 & 1 \\
0 & 0 & \dots & 0 & 0 & 1 & 1
\end{pmatrix}
\quad
S_4 = \begin{pmatrix}
1 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 \\
0 & 0 & 0 & 1
\end{pmatrix}
\quad
S_5 = \begin{pmatrix}
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
0 & 0 & 1 & 1
\end{pmatrix}
$$

Notice that the personality matrix, in its reduced form, has no dominated rows. Notice also that the last two rows of S_2 are dominated by other rows of S_2. Therefore, in order for a reduced matrix of an original matrix M to possess a configuration that contains S_2 as submatrix, it is necessary that the corresponding two rows in M are dominant rows. Similary, this is the case of the last two rows of S_3, rows 2, 3 and 5 of S_4, and rows 1, 3 and 5 of S_5. Therefore, the class of personality matrices, not in its reduced form, that do not satisfy the sufficient condition of for folding optimality is strictly smaller than the class of matrices that do not possess the consecutive ones property.

5.2 The Algorithm

We propose to utilize Booth and Lueker's algorithm [1] to produce an acceptable permutation, even if the sufficient conditions for optimality are not verified.

The heuristic to be proposed attempts to get a good permutation by minimizing the distance between the extreme 1's, generating small gates. We first give a brief explanation of how their algorithm works. Each column yields a subset whose elements are the row positions with 1 in it. These subsets are called restriction subset in the sense that we want all the row positions of the subset to be consecutive in the final configuration. The following procedure from [1] implements a test for the consecutive ones property.

Boolean procedure REDUCTION(M)
begin
  $\Pi := \{\pi \mid \pi \text{ is a row permutation of } M\}$
  for each $S_i \in S$ do
    $\Pi := \Pi \cap \{\pi \mid \text{all 1 in } S_i \text{ are consecutive within } \pi\}$
  return $\Pi \neq 0$
end

S is the set composed by the restriction subsets S_i. Π denotes possible row permutations in M.

Columns are sequentially examined. For each column, row permutations are searched such that they satisfy the restriction subset associated to the current column without disturbing the consecutivity already achieved for the columns examined so far. If there is no such a row permutation then the matrix has not the consecutive ones property, and the algorithm will stop. On the other hand, if the algorithm examines successfully all columns to completion then the matrix possesses the desired property.

In both cases, when the algorithm stops, a row permutation will be available in which every processed column so far has its 1's arranged consecutively. Our heuristic attempts to increase the number of processed columns. The first immediate modification is to change the termination strategy, so that all the restriction subsets are examined.

If a restriction subset cannot be satisfied, our algorithm ignores it and takes the next one to be processed. In the final configuration this column will not have its 1's arranged consecutively. It may still be possible, however, to fold it at the compaction step.

This approach leads to several possible heuristics, regarding the order of examining the columns. We propose to first sort the restriction subsets in descending order of its cardinality. The columns with more 1's should be treated first, since it would be easier to arrange their 1's consecutively at an earlier stage. On the other hand, we can also argue that the larger the number of 1's in a column already treated by the algorithm, the more difficult it becomes to obtain success with the next columns, since there will be many induced restrictions upon a lot of rows at the same time. Besides that, a column with many 1's will generate a long gate any way, even if its 1's occur consecutively. So we can start processing columns with small number of 1's, by sorting the restriction sets in ascending order.

Some examples showed that the first strategy (descending order) is superior to the ascending order strategy especially for dense matrices. Since this step is relatively "cheap", we may use both heuristics separately and choose the one that gives the better solution, i.e., the one that leads to the least $P*$.

We will call what we have discussed above the *modified Booth and Lueker's algorithm.*

Both the following algorithms are $O(n^3)$ time because of the removal step, to obtain the reduced matrix. As already noted, the sufficient condition to achieve optimality is that the reduced personality matrix possesses the consecutive ones property.

Algorithm 5.1 *Heuristic algorithm for multiple folding.*

a. read in the personality matrix, not in reduced form	$O(n^2)$
b. remove duplicate and dominated rows	$O(n^3)$
c. run modified Booth and Lueker's algorithm on the reduced matrix	$O(n^2)$
d. rearrange the matrix	$O(n^2)$
e. reinsert the rows removed at step b	$O(n^2)$
f. compact the matrix by Algorithm 3.2	$O(n \log n)$
g. output the folded matrix	$O(nP*)$

Algorithm 5.2 *Heuristic algorithm for simple folding.*

a. read in the personality matrix, not in reduced form	$O(n^2)$
b. remove duplicate and dominated rows	$O(n^3)$
c. run modified Booth and Lueker's algorithm on the reduced matrix	$O(n^2)$
d. rearrange the matrix	$O(n^2)$
e. reinsert the rows removed at step b	$O(n^2)$
f. generate associated gate compatibility graph	$O(n^2)$
g. find maximum matching	$< O(n^{2.5})$
h. compact the matrix	$O(n)$
i. output the folded matrix	$< O(n^2)$

5.3 Discussion on the Shortcomings of Heuristic Algorithms

Deo et al. [2] pointed out several shortcomings of a number of greedy heuristic algorithms, for multiple folding, based on repeated search for local optima. They show that a number of greedy strategies that consider the columns one at a time to form a solution may attempt to rearrange the 1's of the first column of the matrix below, thus producing a bad permutation. Note that the matrix on the left can be folded onto four physical columns, while the one on the right requires six. We note that our proposed algorithms, using the ascending order on the cardinality of restriction subsets, produces the optimal result that uses four physical columns.

$$\begin{pmatrix} 1\,1\,0\,0\,0\,0 \\ 0\,1\,0\,0\,1\,1 \\ 1\,0\,1\,0\,0\,0 \\ 0\,0\,1\,0\,1\,1 \\ 1\,0\,0\,1\,0\,0 \\ 0\,0\,0\,1\,1\,1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1\,1\,0\,0\,0\,0 \\ 1\,0\,1\,0\,0\,0 \\ 1\,0\,0\,1\,0\,0 \\ 0\,1\,0\,0\,1\,1 \\ 0\,0\,1\,0\,1\,1 \\ 0\,0\,0\,1\,1\,1 \end{pmatrix}$$

6 Conclusions

We have presented efficient heuristic algorithms for the PLA multiple/simple folding problem, and established sufficient conditions for the obtained solutions to be optimal. One sufficient condition is that the reduced personality matrix possesses the consecutive ones property. Verification of the sufficient conditions constitutes part of the proposed algorithms, and is in fact used to construct the heuristic solution, even if the sufficient conditions are not met.

As an illustrative example, consider the 58×19 personality matrix of Fig. 6(a). It corresponds to a real PLA used in the 8085 processor.

Algorithm 5.1 obtains first the reduced personality matrix which possesses the consecutive ones property, as shown in Fig. 6(b). Thus we know we will get the optimal solution. Rows of the reduced matrix are marked with an asterisk. The removed rows are repositioned below their corresponding identical ou dominating rows, as shown in Fig. 6(c). The optimal solution obtained is shown in Fig. 6(d). In Fig. 6(c) the columns are labeled by letters a, b, c, ... In Fig. 6(d) we use the same letters to denote the corresponding gates of the folded columns.

01	0000000000000100000
02	0000000010000000000
03	0000001000000000000
04	0000000000000000010
05	0001000000000000000
06	0000000100001000000
07	0000000100000000001
08	0000000100000000000
09	0000000000000000001
10	0000000000000001000
11	1000000000000000000
12	0000000000100000000
13	0000000000100000100
14	0000000000010000000
15	0000000000010000000
16	0000000000000100000
17	0000000100000000000
18	0000000000100000000
19	0000000000100000000
20	0000000001000000000
21	0000000100000000000
22	0010000000000000000
23	0000000000000000001
24	0000000000000001000
25	0000000000001000000
26	0000000000010000000
27	0000000000100000000
28	0010000000000000000
29	0000000100000100000
30	0000000100000000000
31	0000000000000001000
32	0000000000001000000
33	0000000000001000000
34	0000000100000100000
35	0000000000100000000
36	0000000000000100000
37	0000000010000000000
38	0000000001000000000
39	0000000010000000000
40	0000000100000000000
41	0000000000010000000
42	1000000000000000000
43	0000000000100000000
44	0000000000100000000
45	0000000000000001000
46	0000000000000000001
47	0000000000000010000
48	0000100000000000000
49	0000000010000000000
50	0000000010000000000
51	0000000010000000000
52	0000001000000000000
53	0100000000000000000
54	0100000000000000000
55	0000000000010000000
56	0000000000000000100
57	0000010000000000000
58	0000000100000000000

(a)

*02	0000000010000000000
*03	0000001000000000000
*04	0000000000000000010
*05	0001000000000000000
*06	0000000100001000000
*07	0000000100000000001
*29	0000000100000100000
*10	0000000000000001000
*11	1000000000000000000
*13	0000000000100000100
*14	0000000000010000000
*20	0000000001000000000
*22	0010000000000000000
*47	0000000000000010000
*48	0000100000000000000
*53	0100000000000000000
*57	0000010000000000000

(b)

	abcdefghijklmnpqrst	
*02	0000000010000000000	i 0
37	0000000010000000000	i 0
39	0000000010000000000	i 0
49	0000000010000000000	i 0
50	0000000010000000000	i 0
51	0000000010000000000	i 0
*03	0000001000000000000	g 0
52	0000001000000000000	g 0
*04	0000000000000000010	s 0
*05	0001000000000000000	d 0
*06	0000000100001000000	h m
08	0000000100000000000	h m
17	0000000100000000000	h m
21	0000000100000000000	h m
25	0000000000001000000	h m
30	0000000100000000000	h m
32	0000000000001000000	h m
33	0000000000001000000	h m
40	0000000100000000000	h 0
58	0000000100000000000	h 0
*07	0000000100000000001	h t
09	0000000000000000001	h t
23	0000000000000000001	h t
46	0000000000000000001	h t
*29	0000000100000100000	h n
01	0000000000000100000	h n
16	0000000000000100000	h n
34	0000000100000100000	h n
36	0000000000000100000	0 n
*10	0000000000000001000	q 0
24	0000000000000001000	q 0
31	0000000000000001000	q 0
45	0000000000000001000	q 0
*11	1000000000000000000	a 0
42	1000000000000000000	a 0
*13	0000000000100000100	k r
12	0000000000100000000	k r
18	0000000000100000000	k r
19	0000000000100000000	k r
27	0000000000100000000	k r
35	0000000000100000000	k r
43	0000000000100000000	k r
44	0000000000100000000	k r
56	0000000000000000100	0 r
*14	0000000000010000000	l 0
15	0000000000010000000	l 0
26	0000000000010000000	l 0
41	0000000000010000000	l 0
55	0000000000010000000	l 0
*20	0000000001000000000	j 0
38	0000000001000000000	j 0
*22	0010000000000000000	c 0
28	0010000000000000000	c 0
*47	0000000000000010000	p 0
*48	0000100000000000000	e 0
*53	0100000000000000000	b 0
54	0100000000000000000	b 0
*57	0000010000000000000	f 0

(c) (d)

Fig. 6. An illustrative example

Acknowledgments: We would like to thank Jean Duprat for his careful reading of this paper and many helpful suggestions. The second author would also like to thank Michel Cosnard for his invitation to visit LIP-IMAG/ENS Lyon (Laboratoire de l'Informatique du Parallélisme).

References

1. Booth, K. S. and Lueker, G. S., "Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms", *Journal of Computer and System Sciences* 13, pp. 335-379, 1976.
2. Deo, N., Krishnamoorthy, M. S. and Langston, M. A., "Exact and approximate solutions for the gate matrix layout problem", *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 79-84, January 1987.
3. Fellows, M. R. and Langston, M. A., "Nonconstructive advances in polynomial-time complexity", *Information Processing Letters*, 26, pp. 157-162, 1987.
4. Ferreira, A. G., "An Optimal $O(n^2)$ algorithm to fold special PLA's", in *Lecture Notes in Economics and Mathematical Systems*, No. 302, Proceedings of the Conference "Optimization Days 86", Montreal, Canada, April 30 - May 2, 1986, H. A. Eiselt and G. Pederzoli (eds.), pp. 92-102, Springer-Verlag, 1988.
5. Golumbic, M. C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, 1980.
6. Hachtel, G. D., Newton, A. R. and Sangiovanni-Vicentelli, A. L., "An Algorithm for optimal PLA folding", *IEEE Trans. Computer-Aided Design*, vol. CAD-1, pp. 63-76, January 1982.
7. Hwang, S. Y., Dutton, R. W. and Blank, T., "A best-first search for optimal PLA folding", *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 433-442, July 1986.
8. Kashiwabara, T. and Fujisawa, T., "An NP-complete problem on interval graph", *Proc. 1979 IEEE Int. Symposium on Circuits and Systems*, pp. 82-83, 1979.
9. Kinnersley, N. G. and Langston, M. A., "Obstruction set isolation for the gate matrix layout problem", Technical Report, Department of Computer Science, University of Tennessee, Knoxville, January, 1991.
10. Lecky, J. E., Murphy, O. J. and Absher, R. G., "Graph-Theoretic Algorithms for the PLA folding problem", *IEEE Trans. Computer-Aided Design*, vol. CAD-8, pp. 1014-1021, September 1989.
11. Luby, M., Vazirani, U., Vazirani, V. and Sangiovanni-Vicentelli, A., "Some theoretical results on the optimal PLA folding problem", in *Proceedings IEEE Conf. on Circuits and Computers*, pp. 165-170, 1982.
12. Micali, S. and Vazirani, V. V., "An $O(\mid V \mid^{1/2} \mid E \mid)$ algorithm for finding maximum matching in general graphs", *Proceedings 21st Annual Symposium on Foundations of Computer Science*, IEEE, pp. 17-27, 1980.
13. Möhring, R. H., "Graph problems related to gate matrix layout and PLA folding", Technical Report no. 223, Technical University of Berlin, 1989.
14. Paillotin, J. F., "Optimization of the PLA area", *Proceedings 18th Design Automation Conference*, pp. 406-410, June 1981.
15. Pape, U. and Conradt, D. "Maximales matching in Graphen", in H. Späth (ed.) *Augewählte operations research in Fortran*, Oldenburg, Munich, pp. 103-114, 1980.
16. Syslo, M. M., Deo, N. and Kowalik, J. S. *Discrete Optimization Algorithms*, Prentice Hall, 1983.
17. Tucker, A. C., "A structure theorem for the consecutive ones property", *Journal Combinatorial Theory* 12(B), pp. 153-162, 1972.