O PROBLEMA DO DOBRAMENTO
OPTIMAL DE PLAS

Afonso Galvão Ferreira

DISSERTAÇÃO APRESENTADA
AO
INSTITUTO DE MATEMÁTICA E ESTATÍSTICA
DA
UNIVERSIDADE DE SÃO PAULO
PARA OBTENÇÃO DO GRAU DE MESTRE
EM
MATEMÁTICA APLICADA

ÁREA DE CONCENTRAÇÃO: CIÊNCIAS DA COMPUTAÇÃO

ORIENTADOR: PROF. DR. SIANG WUN SONG

Parte deste trabalho recebeu apoio financeiro
da FAPESP - Proc. Nº 83/2157-7 e Nº 84/2477-4

- SÃO PAULO, Junho de 1986 -

*Ao meu orientador,
Siang Wun Song.
Por ter me orientado.*

AGRADECIMENTOS

A Renata, pela compreensão; à Eunice, Walnice, Oliveiros e Rui, pelo apoio sempre que necessário; ao Grupo de Computação Paralela e VLSI, Inês, Miguel e Alceu, por possibilitarem a discussão; ao Cláudio, por uma dica imprescindível; ao Imre, por um remoto puxão de orelhas; ao Jayme, por sua paixão por algoritmos e grafos; ao Paulo, pelo interesse; à família, como de hábito; ao prof. Altamiro Suzim, da UFRGS, pelas PLAs do 8085; aos funcionários do MAP, Xerox e Biblioteca do IME, sem os quais nada teria andado; aos colegas, pelo incentivo; e "last but not least" - muito pelo contrário - ao Siang, por tudo.

INDICE

pg

Apresentação.....	1
I. O Problema do Dobramento Optimal - PDO.....	4
II. Uma Solução Baseada na Teoria dos Grafos.....	18
III. Uma ótima Solução Optimal para o Dobramento Múltiplo.....	35
IV. Dobramento Múltiplo Restrito: O Dobramento Simples.....	48
V. Conclusão.....	58
Anexos.....	60
Referências Bibliográficas.....	66

APRESENTAÇÃO

Uma técnica para implementar circuitos lógicos é o uso de PLAs (Programmable Logic Arrays). Esta técnica mostrou-se uma ferramenta efetiva na implementação, em pastilhas, de funções lógicas combinatórias com muitas saídas.

O Problema do Dobramento Optimal de PLAs (PDO) tem por finalidade reduzir a área de PLAs, tendo sido proposto em 1981 por Hachtel, Newton e Sangiovanni-Vincentelli [1]. Movidos por motivações práticas, engenheiros elétricos e eletrônicos ligados à Computação e mesmo pesquisadores de áreas teóricas da Computação passaram a buscar soluções para o problema que se mostrava de grande aplicabilidade na construção em larga escala de "chips", ou pastilhas, em VLSI.

Provado ser NP-completo, o PDO admite dois caminhos para soluções algorítmicas: algoritmos exaustivos (do tipo "branch and bound") ou heurísticos. Sua primeira resolução foi heurística e baseou-se numa formulação através da Teoria dos Grafos. O interessante é que todos os algoritmos heurísticos por nós pesquisados - que não foram poucos -, aproximaram-se do problema do mesmo modo que Hachtel et alii. Houve apenas um trabalho, de autoria de Paillotin [2], que inovou a abordagem do problema, abrindo espaço para nossas descobertas.

A principal contribuição desta dissertação reside na utilização

da Teoria dos Grafos e Grafos Perfeitos, seus algoritmos e teoremas específicos para a elaboração de soluções originais e eficientes para o PDO, tanto na modalidade de dobramento simples como múltiplo, visto sob a ótica da Paillotin.

No capítulo I apresentamos as PLAs e o PDO. Trata-se basicamente do artigo em que Hachtel et alii primeiro expuseram o problema, completado com algumas outras informações, compiladas durante nossas pesquisas.

A heurística por eles desenvolvida para o dobramento simples, bem como seu algoritmo, foram reunidos no capítulo seguinte, onde colocamos também uma heurística que criamos sobre a teoria até então analisada.

A menina de nossos olhos é o capítulo III. Lá compilamos resultados originais para o PDO múltiplo. A "ótima solução optimal" do título refere-se a um algoritmo quadrático (assintoticamente optimal) que resolve, otimamente certas instâncias do problema, da classe NP-completo. Os exemplos estudados deram-nos a impressão de que, em geral, nossa solução é excelente, estando muito próxima da ótima na grande maioria dos casos. Embora curtos e simples, os teoremas e lemas propostos nesse capítulo são de extrema utilidade na solução do problema. Esta característica deveu-se principalmente à nova abordagem do PDO, lá descrita.

O último capítulo (IV) é como que uma continuação do anterior. O dobramento simples é analisado e um algoritmo muito semelhante ao do capítulo III é indicado. Pequenas alterações foram necessárias devido a características intrínsecas do problema. De qualquer modo, os resul-

tados permaneceram originais e muito eficientes.

Esperamos que o texto que ora apresentamos seja ágil e claro, não confundindo o leitor com passagens supérfluas ou obscuras. Por outro lado, talvez se encontrem alguns trechos em que a linguagem utilizada pareça um tanto direta. Os capítulos III, IV e parte do II provêm de artigos que apresentamos entre julho de 1985 e julho de 1986 [15,16,24].

Nosso primordial intento durante a fase de pesquisa, cujo resultado é esta dissertação, foi gerar contribuições originais para o problema estudado, não nos contentando em realizar apenas um "survey" dos trabalhos pesquisados.

1. Introdução

Na era do VLSI ("very large scale of integration"), circuitos e sistemas tornaram-se tão complexos que técnicas estruturadas de projeto passaram a ser necessárias visando-se ao aprimoramento da correção lógico-elétrica sob um custo razoável.

Por possuir uma estrutura regular que possibilita enorme eficiência no projeto de circuitos VLSI, lógica matricial tem sido bastante utilizada para este fim. Infelizmente esta regularidade é também responsável por uma considerável sub-utilização da área das pastilhas. Mesmo assim, matrizes lógicas programáveis ("programmable logic arrays (PLAs)") mostraram-se altamente proveitosas na implementação de funções lógicas combinatórias em pastilhas [22].

A implementação de uma função lógica através de uma PLA pode ser concebida em três etapas [10]:

- .Projeto Funcional
- .Projeto Topológico
- .Projeto Físico

O Projeto Funcional consiste na transformação de um conjunto de equações booleanas em um conjunto minimal de implicantes lógicos na forma de somas de produtos. O objetivo é atingir a velocidade máxima de transmissão numa área mínima de silício.

Projeto Topológico é a fase em que a lógica resultante do processo anterior é mapeada numa representação topológica da estrutura final da PLA: tabela de símbolos, matriz de personalidade ou diagrama de pauzinhos ("stick diagram"), por exemplo.

Já no Projeto Físico, faz-se a transposição da representação topológica para uma máscara de traçado, segundo a tecnologia de implementação escolhida, por exemplo NMOS.

O interesse deste trabalho estará voltado para a otimização do Projeto Topológico, numa tentativa de reduzir a área de PLAs através de algoritmos polinomiais. Para tanto comecemos por entender como representá-las.

2. PLAs

A estrutura geral de uma PLA é a seguinte:

-fig 1-

As entradas - e seus complementos - correm verticalmente através de uma matriz de elementos de circuito chamada PLANO E. Este plano gera combinações das entradas e seus complementos e estes sinais tornam-se entradas para outra matriz de elementos de circuitos, o PLANO OU. Suas saídas, por sua vez, são as funções booleanas das variáveis, na forma canônica de soma de produtos. Cada linha horizontal, de interligação entre os dois planos, leva um termo-produto.

Para melhor ilustrar, imaginemos que ao final da fase Funcional temos as funções $F1 = AC + \bar{A}\bar{C} + D + \bar{B}$ e $F2 = \bar{C}\bar{D} + CD$ para implementar por meio de uma PLA 1. $F1$ e $F2$ serão saídas (somas de produtos) e

A, B, C, D variáveis.

Há várias maneiras de representarmos esta PLA 1. Vejamos duas delas.

a. Diagrama da PLA 1:

PLA 1:

-fig 2-

Uma cruz no plano E indica a presença da entrada correspondente num termo-produto, enquanto uma cruz no plano OU indica a presença do correspondente termo-produto na saída.

b. Matriz de Personalidade da PLA 1:

PLA 1:		A	B	C	D	F1	F2
	1	1	N	1	N	1	N
	2	0	N	0	N	1	N
	3	N	N	N	1	1	N
	4	N	N	0	0	N	1
	5	N	N	1	1	N	1
	6	N	0	N	N	1	N
		<----E---->				<--OU-->	

-fig 3-

É uma matriz com elementos de três tipos, indicando:

1: conexão da entrada ou saída.

0: conexão do complemento da entrada.

N: não há conexão (sem importância)

O resultado desta etapa é transformado, durante o Projeto Físico, em minúsculos fios, transistores e portas lógicas. Vejamos, na próxima figura, como ficaria a PLA 1 pronta para a implementação:

-fig 4-

A figura 4 deixa claro que as entradas de uma PLA são apenas as variáveis; seus complementos são obtidos através de portas NÃO colocadas na linha ao lado daquela em que o sinal da variável será inserido. Por razões ligadas às técnicas estudadas para o processo de otimização, estaremos interessados apenas em saber se um elemento da Matriz de Personalidade (MP) representa ou não uma conexão - não importando se da variável ou de seu complemento. Redefinindo a matriz de personalidade, criamos a MPM (Matriz de Personalidade Modificada).

Um elemento da MPM é: 1, se o elemento correspondente (a variável ou seu complemento) na MP é 0 ou 1.

0, se o elemento é "não há conexão".

MPM da PLA 1:

PLA 1:	A	B	C	D	F1	F2
1	1	0	1	0	1	0
2	1	0	1	0	1	0
3	0	0	0	1	1	0
4	0	0	1	1	0	1
5	0	0	1	1	0	1
6	0	1	0	0	1	0

-fig 5-

3. Métodos de Otimização

A maioria das PLAs é muito esparsa. Uma implementação direta no Projeto Físico resultará num significativo desperdício da área de silício, i.e., área que será ocupada apenas por cruzamentos de fios e que não contribuirá para a formação das funções lógicas. Esta área não utilizada reduzirá o rendimento do circuito e até mesmo o desempenho da PLA será afetado pela introdução de distorções elétricas (parasitas) desnecessárias.

Para a solução deste entrave, desenvolveram-se métodos de otimização da área de PLAs. Devido à sua já mencionada esparsidade, sur-

giram - entre outras - técnicas de dobramento para a compactação de PLAs, cuja intenção principal é alocar ligações efetivas no lugar de simples fios sem conexões.

Há dois tipos de dobramento e algumas estratégias para cada um deles.

Os tipos:

-Dobramento de colunas

as entradas e saídas não mais se posicionam em apenas um lado da PLA.

-Dobramento de linhas

altera-se a arquitetura da PLA original (E-OU) para uma (OU-E-OU) ou ainda para uma sequência de planos OUs e Es alternados.

As estratégias:

-Dobramento Simples

permutam-se as linhas (colunas) da PLA com o intuito de permitir que pares de colunas (linhas) lógicas ocupem a mesma coluna (linha) física.

-Dobramento Múltiplo

Similar ao acima descrito, entretanto permitem-se mais de duas colunas (linhas) lógicas compartilhando uma mesma física.

-Dobramento Restrito

Devido a problemas de conexão com o circuito externo, é imperativo que se permitam restrições às posições a serem ocupadas pelas entradas e saídas após a dobra. Nesta ótica o dobramento

simples pode ser encarado como um dobramento múltiplo restrito, onde as entradas e saídas devem ser conectadas apenas acima ou abaixo da PLA.

Vejamos um exemplo. A estrutura da PLA 1 reduz-se à metade após um dobramento simples de colunas.

PLA 1		A	B	F1
dobrada:	6	0	1	1
	1	1	1	1
	2	1	1	1
	3	1	0	1
	4	1	1	1
	5	1	1	1
		D	C	F2

-fig 6-

Se efetuarmos o dobramento simples de colunas seguido de um de linhas, ou vice-versa, poderemos reduzir a área de uma PLA a um mínimo de 25%, não importando o quão esparsa ela é. No caso de dobramento múltiplo, o único limite será sua esparsidade como veremos no capítulo III.

Por serem análogos, trataremos apenas do dobramento de colunas. Todos os resultados serão válidos, mutatis mutandis, para o de linhas.

Devido a problemas de ordem técnica, os planos E e OU serão dobrados independentemente, assim os resultados e definições aplicar-se-ão a cada um em separado.

Certamente há algumas restrições que impedem o dobramento. A mais óbvia requer a disjunção das colunas, i.e., os 1s de uma coluna devem estar em linhas diferentes das daquelas que serão suas parceiras após o dobramento. Esta, porém não é a única limitação. A escolha de duas colunas introduz novas condições, pois ao dobrarmô-las forçaremos todos os termos de uma delas a estarem acima ou abaixo dos da outra.

Para dobrarmos (simples) uma PLA precisamos especificar os pares de colunas a serem dobrados e sua posição relativa (acima ou abaixo). A fim de que um dobramento seja implementável, cada posição relativa proveniente de cada alocação não pode implicar em uma restrição cíclica na posição das linhas da PLA. Só assim existirá uma permutação de tais linhas satisfazendo a todas as restrições impostas.

	X	Y	Z	W	<u>Exemplo de restrição cíclica:</u>
1	1	0	0	1	As colunas X e Y podem ser dobradas.
2	1	0	1	0	As colunas Z e W podem também ser dobradas,
3	0	1	1	0	permutando-se as linhas 2 e 4.
4	0	1	0	1	Entretanto, as duas dobras não podem ser efetuadas simultaneamente.

-fig 7-

4. Formalização do PDO

Tendo por objetivo uma abordagem mais rigorosa, Hachtel et alii [1] introduzem as seguintes definições:

par dobrável: é o par $f = \{c_i, c_j\}$, onde c_i e c_j são colunas disjuntas.

$R(c_i) = \{r_1, r_2, \dots, r_k\}$ é um subconjunto do conjunto R de linhas da MPM, constituído por aquelas que têm 1 na coluna c_i .

par ordenado dobrável: é um par dobrável $o = (c_i, c_j)$, onde todas as linhas em $R(c_i)$ devem estar acima das linhas em $R(c_j)$. É a restrição na posição relativa das linhas da MPM.

conjunto dobrável: $F = \{f_1, f_2, \dots, f_k\}$, onde os f_i 's são pares dobráveis, disjuntos dois a dois.

conjunto ordenado dobrável: $O = \{o_1, o_2, \dots, o_k\}$, onde os o_i 's são pares ordenados dobráveis, disjuntos dois a dois.

Para exemplificar estas definições, aplicá-las-emos à MPM da figura 7.

- a. $\{X,Y\}$ e $\{Z,W\}$ são pares dobráveis.
- b. $R(W) = \{1,4\}$; $R(Z) = \{2,3\}$.
- c. (X,Y) e (Z,W) são pares ordenados dobráveis.
- d. $\{ \{X,Y\}, \{Z,W\} \}$ é um conjunto dobrável.
- e. $0 = \{ (X,Y), (Z,W) \}$ é um conjunto ordenado dobrável.

-fig 8-

Cada par ordenado dobrável $o_p = (c_i, c_j)$ induz uma relação $Q_p = R(c_i) < R(c_j)$ no conjunto R das linhas da MPM. Pela definição de par ordenado dobrável, todas as linhas de $R(c_i)$ devem estar acima de $R(c_j)$.

Seja $Q(0) = \bigcup_i Q_i$ induzida pelos pares ordenados dobráveis de 0 .

Diremos então que $r_i < r_j$ se existe em 0 algum par ordenado dobrável que obrigue a linha r_i a estar acima da linha r_j . Sem perda de generalidade, adicionamos a $Q(0)$ a relação $r_i < r_i$, $i = 1, \dots, |R|$.

Definimos o fecho transitivo da relação $Q(0)$ como sendo a relação $Q^+(0)$ definida por: $r_i <^+ r_j$ se e somente se existe uma sequência $\{r_1 < r_2, r_2 < r_3, \dots, r_{k-1} < r_k\}$, onde $k \geq 2$, $r_1 = r_i$ e $r_k = r_j$.

Podemos então dizer que 0 é um conjunto ordenado dobrável implementável se $Q^+(0)$, o fecho transitivo de $Q(0)$, é uma ordem parcial (reflexiva, transitiva e anti-simétrica) sobre o conjunto R .

Na figura 8, o conjunto $\hat{O}$ obtido no ítem (e) não é implementável, pois (X,Y) induz $R(X) < R(Y)$ e (Z,W) induz $R(Z) < R(W)$ e portanto as linhas $\{1,2\} < \{3,4\}$ e $\{2,3\} < \{1,4\}$. Assim teremos linha 1 < 3 e 3 < 1. Logo $Q(\hat{O})$ não é anti-simétrico. Vide figura 7.

Neste ponto é-nos possível definir o Problema do Dobramento Optimal de PLAs (PDO), reduzido ao caso de dobramento simples de colunas:

PDO: Dada a MPM de uma PLA, encontre o conjunto ordenado dobrável implementável, $\hat{O}$, de cardinalidade máxima.

5. Considerações

O PDO foi provado ser NP-completo[1], sendo portanto de difícil computação. Este fato gerou duas correntes opostas entre os pesquisadores. Há os que preferem algoritmos do tipo "branch-and-bound", achando soluções ótimas em tempo exponencial [11,12]. Em contraposição muitos algoritmos heurísticos foram criados com o intuito de montar o conjunto $\hat{O}$ com cardinalidade maximal, gerando soluções nem sempre ótimas, porém bastante rápidas [4, 7, 8, 9, 13].

Por nosso lado, acreditamos ser mais interessante seguir a linha dos que pesquisam características peculiares das PLAs transformando-as em heurísticas para algoritmos polinomiais.

É importante notar que o Problema do Dobramento Optimal de PLAs

pode ser considerado como um problema de Permutação e Compactação. Dentre as $m!$ permutações de linhas existentes, qual conduzirá à maior redução da área necessária para a matriz? Por outro lado, dada uma determinada configuração de uma MPM, como compactá-la otimamente? quais serão as colunas lógicas ideais para compartilhar uma mesma coluna física?

O PDO visto sob este prisma será examinado nos capítulos III e IV. Por enquanto estudemô-lo a partir da teoria já exposta.

II. UMA SOLUÇÃO BASEADA NA TEORIA DOS GRAFOS

1. Teorização em Grafos

O Problema do Dobramento Optimal de PLAS foi primeiramente definido por Hachtel et alii [1] em 1981 e a solução heurística proposta baseou-se numa interpretação através de sua formulação na Teoria dos Grafos [20].

Seja o grafo intersecção de colunas (GIC) o grafo $G = (V, E)$ cujos vértices em V estão numa correspondência um-a-um com as colunas da MPM da PLA. O conjunto das arestas é definido como $E = \{e = \{v_i, v_j\} : R(c_i) \cap R(c_j) \neq \emptyset; v_i, v_j \in V \text{ e } i \neq j\}$, i.e. as arestas unem "colunas" que não são disjuntas.

Seja $ADJ(u) = \{v : v \in V \text{ e } \{u, v\} \in E\}$, $u \in V$, o conjunto de vértices adjacentes a u .

Num dobramento implementável um par dobrável qualquer é constituído de "vértices não adjacentes".

Dado um conjunto ordenado dobrável O e o grafo de intersecção de colunas $G = (V, E)$, introduzimos um grafo misto associado $G(O) = (V, E, A(O))$, onde $A(O)$ é um conjunto de arestas dirigidas tal que $A(O) = \{(v_i, v_j) : (c_i, c_j) \in O; v_i, v_j \in V\}$ e V e E são como visto.

Como todos os pares ordenados dobrados de 0 são disjuntos, $A(0)$ é um emparelhamento em $G(0)$; por outro lado, dada a MPM de uma PLA e seu GIC, um conjunto de arestas dirigidas B , formando um emparelhamento em $G = (V, E, B)$, identifica um conjunto ordenado dobrável $0(B)$ onde a orientação das arestas indica a posição relativa das duas colunas correspondentes na PLA dobrada. Teremos um exemplo na figura 2 adiante.

Estabelecida a correspondência biunívoca entre conjunto ordenado dobrável e emparelhamento, podemos caracterizar conjuntos ordenados dobráveis implementáveis em termos de grafos.

Sejam:

AP (caminho alternado), uma sequência de vértices $w = \{v_1, v_2, \dots, v_{2k}\}$ do grafo misto $G(V, E, A)$, tal que $(v_{2i-1}, v_{2i}) \in A$ e $(v_{2j}, v_{2j+1}) \in E$; $i = 1, 2, \dots, k$; $j = 1, \dots, k-1$.
AC (ciclo alternado), a sequência $w_c = \{w, v_1\}$ onde $w = \{v_1, v_2, \dots, v_{2k}\}$, $v_j \in V$, $j = 1, \dots, 2k$; é um AP e $\{v_{2k}, v_1\} \in E$.

$w = \{1, 2, 3, 4\}$ é um
caminho alternado (AP)

$w_c = \{1, 2, 3, 4, 1\}$ é um
ciclo alternado (AC)

-fig 1-

Em outras palavras, um caminho alternado no grafo misto é um caminho composto por arestas não dirigidas e dirigidas, alternadamente. Um ciclo alternado, por sua vez, é ciclo com estas mesmas características.

LEMA 1: Dado $G = (V, E, A)$, onde $GIC = (V, E)$ é o grafo de intersecção de colunas de uma PLA e onde A é um emparelhamento de G , o conjunto ordenado dobrável $O(A)$ induzido é implementável se e somente se G não tem ciclos alternados.

Dem: (somente se) Por contradição assumamos que $G = (V, E, A)$ contém um ciclo alternado $w_c = \{c_1, \dots, c_n, c_1\}$. Pela definição de ciclo alternado, $(c_1, c_2) \in A$ e então $(c_1, c_2) \in O$ e portanto:

$$R(c_1) < R(c_2) \quad (1)$$

Mas $(c_2, c_3) \in E$, logo $\exists r \in R(c_2) \cap R(c_3)$ e, de (1):

$$R(c_1) < r \quad (2)$$

Como $R(c_3) < R(c_4)$ e $r \in R(c_3)$ temos que

$R(c_1) < r < R(c_4)$ e daí $R(c_1) < R(c_4)$ pois $Q^+(O)$ é transitivo já que $O(A)$ é implementável.

Seguindo o mesmo argumento chegamos a:

$$R(c_1) < R(c_{n-2}) \quad (3)$$

Sejam agora $q \in R(c_{n-2}) \cap R(c_{n-1})$ e $p \in R(c_n) \cap R(c_1)$.

Ora, já que (3) então $p < q$.

Entretanto, $(c_{n-1}, c_n) \in A$, e assim: $q < p$.

CONTRADIÇÃO!! $Q^+(O)$ é ordem parcial por hipótese.

(se) Por contradição suponha que $Q^+(O(A))$ não é uma ordem parcial (e então $O(A)$ não é implementável).

Assim, existem duas linhas a e b tal que:

$$a < b \text{ e } b < a$$

Pela definição de fecho transitivo existe uma sequência

$$\{a, a_2, a_3, \dots, a_n, b\}$$

tal que:

$$a \in R(s_1), a_2 \in R(t_1), a_2 \in R(s_2), a_3 \in R(t_2), \dots, b \in R(t_n)$$

$$\text{e para } i = 1, 2, \dots, n, \quad o_i = (s_i, t_i) \in O(A).$$

O que implica em $R(t_i) \cap R(s_{i+1}) \neq \emptyset$, pois

$$a_{i+1} \in S = R(t_i) \cap R(s_{i+1}) \text{ para } i = 1, 2, \dots, n-1$$

$$\text{e portanto } \{t_i, s_{i+1}\} \in E, \quad i = 1, 2, \dots, n-1.$$

Analogamente temos uma sequência $\{b, b_2, b_3, \dots, b_m, a\}$

$$\text{e } b \in R(q_1), b_2 \in R(p_1), b_2 \in R(q_2), \dots, a \in R(p_m),$$

$$\text{com } o_i = (q_i, p_i) \in O(A), \quad i = 1, 2, \dots, m$$

$$\text{e } R(p_i) \cap R(q_{i+1}) = \emptyset, \quad i = 1, 2, \dots, m-1.$$

$$\text{Portanto } \{p_i, q_{i+1}\} \in E, \quad i = 1, \dots, m-1.$$

$$\text{Logo } w = \{s_1, t_1, s_2, t_2, \dots, t_n, q_1, p_1, q_2, \dots, p_{m-1}, q_m, p_m, s_1\}$$

é um ciclo alternado em $G = (V, E, A)$.

CONTRADIÇÃO!!

Para ilustrar o Lema, observemos na figura 2 o grafo de intersecção de colunas e o grafo misto associado de uma PLA, onde cada vértice corresponde a uma coluna de sua MPM. O conjunto ordenado dobrável $O(A)$, induzido pelo conjunto de arestas dirigidas no segundo grafo, não é implementável pois existe um ciclo alternado $(\{3, 2, 5, 6, 3\})$. De

fato, para dobrarmos as colunas 5 e 6 é necessário que todos os 1s da coluna 5 estejam acima dos da coluna 6. Esta coluna tem 1 na mesma linha que a coluna 3 também tem. Esta ao ser dobrada com a coluna 2 deve ter seus 1s acima dos dela, coluna 2, que tem 1 na mesma linha que a coluna 5. Solução impossível!!

MPM:

1	0	1	0	1	0
1	0	1	0	1	0
0	0	0	1	1	0
0	0	1	1	0	1
0	0	1	1	0	1
0	1	0	0	1	0

GIC:

Grafo misto:

$A = \{(1, 4), (3, 2), (5, 6)\}$

$w = \{3, 2, 5, 6, 3\}$ é um ciclo alternado

-fig 2-

TEOREMA 1: Considere o seguinte problema em Grafos:

PGRAFO: Dado um grafo de intersecção de colunas $G = (V, E)$ da MPM de uma PLA, achar um conjunto A de máxima cardinalidade tal que:

1 - A é um emparelhamento de $G = (V, E, A)$.

2 - $G = (V, E, A)$ não tem ciclos alternados.

Então PGRAFO $\equiv$ PDO, i.e., o problema em Grafos é equivalente ao Problema do Dobramento Optimal de PLAs.

Dem: Segue diretamente do Lema 1.

2.0 primeiro algoritmo

Desde que PGRAFO e PDO são equivalentes e este é NP-completo, então aquele também o é. Isto posto, as soluções conhecidas para acharmos um emparelhamento A de cardinalidade máxima, segundo as condições impostas, são exaustivas e exponenciais. Ou então podemos optar por um algoritmo aproximativo, polinomial, que construa A com cardinalidade maximal, como fizeram Hachtel et alii [1] ao serem os primeiros a proporem um algoritmo heurístico que consome tempo proporcional a $O(n^3)$.

ALGORITMO MODELO

```
begin
  A :=  $\emptyset$ ;  $\hat{U} := \hat{V} := V$ ;
  label: while  $\hat{V} \neq \emptyset$  do
    begin
      v := VSELECT( $\hat{V}$ );
      {faça  $\hat{U}(v) = \{u \in \hat{U} : u \neq v, \{u, v\} \notin E\}$ };
      while  $\hat{U}(v) \neq \emptyset$  do
        begin
          u := USELECT( $\hat{U}(v)$ );
          if CYCLE (v, u)
            then  $\hat{U}(v) := \hat{U}(v) - \{u\}$ 
            else
              begin
                TRANS := TRASCLOS (v, u);
                A := A  $\cup$  {(v, u)};
                 $\hat{V} := \hat{V} - \{v, u\}$ ;
                 $\hat{U} := \hat{U} - \{v, u\}$ ;
                go-to label
              end
            end;
           $\hat{V} := \hat{V} - \{v\}$ 
        end;
      end;
    write (A)
  end.
```

-fig 3-

A seleção heurística do par dobrável é efetuada nos procedimentos USELECT e VSELECT.

A função CYCLE verifica a criação de um possível ciclo alternado.

O procedimento TRANSCLDS computa TRANS, uma eficiente representação do fecho transitivo $Q^+(0(A))$.

Sejam:

.TRANS(v) - o conjunto dos vértices adjacentes a sorvedouros que estão em caminhos alternados começando em v.

.A(F) = {s | $\exists$ t com (s,t) $\in$ A}, o conjunto dos vértices-fonte de A.

.S(u) = {t $\in$ A(F) | t $\in$ ADJ(u)}, o conjunto dos vértices-fonte adjacentes a u.

.F(v) = {t $\in$ A(F) | v $\in$ TRANS(t)}, o conjunto dos vértices-fonte que são início de caminhos alternados que passam por v.

função CYCLE (v,u): boolean;

```
    if S(u)  $\cap$  F(v)  $\neq \emptyset$  then CYCLE := true
    else CYCLE := false
```

-fig 4-

Se houver algum vértice-fonte w, adjacente a u, início de caminho alternado que passa por v, então não se pode colocar a aresta (v,u) em A pois fechar-se-á um ciclo alternado. Logo, se $S(u) \cap F(v)$

não for vazio, teremos $CYCLE = true$ e MODELO desistirá deste par.

```
procedimento TRASCLOS (v,u);  
  begin  
    (1)   TRANS(v) := ADJ(u);  
          for {todo t  $\in$  S(u)} do  
    (2)   TRANS(v) := TRANS(v)  $\cup$  TRANS(t);  
          for {todo t  $\in$  F(v)} do  
    (3)   TRANS(t) := TRANS(t)  $\cup$  TRANS(v)  
  end.
```

-fig 5-

TRASCLOS atualiza a lista TRANS, que contém o fecho transitivo $Q^+(O(A))$, de um modo utilizável pela função CYCLE. Em TRANS(v) estão todos os vértices adjacentes a sorvedouros alcançáveis por um caminho alternado com início em v.

- (1): Colocam-se todos os adjacentes de u em TRANS(v).
- (2): Unem-se a TRANS(v) todos os não-sorvedouros alcançáveis por APs que comecem nos adjacentes de u.
- (3): Atualizam-se as listas TRANS de vértices-fonte que alcançavam v através de APs, com os novos vértices alcançáveis.

Para a prova da correção de CYCLE e TRASCLOS veja [1].

$\emptyset: \in \text{TRANS}(u)$

$\emptyset: \in \text{TRANS}(u) \cap \text{TRANS}(w)$

$\emptyset: \in \text{TRANS}(U) \cap \text{TRANS}(w) \cap \text{TRANS}(v)$

-fig 6-

3. Uma heurística

Uma seleção aleatória dos pares dobráveis provavelmente gerará uma solução maximal distante da ótima, portanto uma discussão a respeito da heurística a ser utilizada é necessária, tendo em vista um bom desempenho do algoritmo.

No capítulo III abordaremos com maior profundidade os limitantes para dobramentos. Por enquanto é fácil notar que o maior inteiro menor ou igual a $n/2$ é um limite superior físico para a cardinalidade do conjunto A, onde n é o número de vértices do Grafo Intersecção de Colunas estudado.

Ilustremos com duas seleções diferentes no grafo da figura 2.

- a. Escolhemos ao acaso a aresta (1,2) e em seguida (5,6), construindo um conjunto com dois elementos. (3,4) não pode ser alocada pois os vértices 3 e 4 são adjacentes.
- b. Selecionamos (1,4), (2,3) e (5,6), obtendo A com três elementos. Dobramento ótimo.

-fig 7-

Em a ligamos inicialmente dois vértices de grau pequeno. Isto deixa vértices com grau grande para serem emparelhados depois. Já que só nos é permitido casar nós não adjacentes, teremos dificuldades em lidar com os restantes.

Ao tomarmos u e v com grau máximo daremos conta, em princípio, do entrave acima. Contudo, aumentaremos as chances de parada do algoritmo devido à criação de circuitos alternados. De fato, muitos deles são criados como consequência desta seleção.

Assim, em qualquer heurística idealizada, devemos precaver-nos

contra duas situações específicas no instante da aposição de uma nova aresta em A. Uma é a não existência de futuros vértices a serem dobrados, outra a não criação de um ciclo alternado por esta aresta.

Se dermos preferência, no início do processo, a arestas ligadas a vértices com graus máximos, adiaremos o momento em que todos os restantes serão adjacentes. Por outro lado, quanto maior o grau de um vértice escolhido, maior a probabilidade dele tomar parte de um ciclo alternado.

Este dilema teórico não encontrou similar tão poderoso na aplicação prática. As PLAs costumam ser enormes e "muito esparsas", contendo um grande conjunto de colunas disjuntas candidatas à dobra. Em casos concretos a verdadeira restrição para a fase de seleção torna-se a não criação de ciclos alternados.

Hachtel optou pela seguinte regra: "selecione v com grau mínimo e u com grau máximo". Escolher v com grau mínimo diminui suas chances de pertencer a um ciclo alternado. Dado então que v tem poucos adjacentes, u torna-se um ótimo candidato para ser seu par.

4. Outra heurística

Quando do estudo do critério adotado acima, notamos um elevado grau de liberdade na escolha das arestas. Pensamos que talvez fosse mais eficiente impedirmos a criação de ciclos alternados, tornando desnecessários os testes em CYCLE.

Desenvolvemos, por conseguinte, uma heurística algo mais elaborada, com características construtivas ou, melhor dizendo, "destrutivas". A idéia básica consiste em destruir ciclos potenciais através da seleção de arestas específicas.

Ao atingirmos no grafo misto associado uma situação similar a:

a intenção é lograr a junção de uma aresta assim:

ou assim:

visando à alocação de ambas:

Para tanto são empregadas duas listas de vértices ainda não utilizados no emparelhamento, L e C. A primeira contendo os vértices adjacentes a sorvedouros, a segunda aqueles adjacentes a fontes.

Em poucas palavras, a heurística procura colocar em A arestas que saiam de um vértice de grau mínimo em C-L e cheguem a um de grau mínimo em L-C. Buscamos evitar um possível ciclo alternado futuro, conservando alta a probabilidade de acharmos outra aresta nas mesmas condições num próximo passo. Ressaltamos que a escolha de vértices de grau mínimo retarda o crescimento de L e C.

Seja $\hat{V}$ o conjunto de vértices de $G = (V, E, A)$ que ainda não fazem parte do emparelhamento.

Seja $D \rightarrow B$ a notação de: "tome u em B de menor grau tal que $v \in D : v \notin \text{ADJ}(u)$. Tome v em D-ADJ(u) de menor grau. Coloque a aresta (v,u) em A". Ou seja, D será o conjunto "saída", enquanto B será

o conjunto "chegada". D e B são sub-conjuntos não vazios de V.

Um algoritmo que implemente esta heurística (digamos algoritmo ALG-TIGER-1) observará num dado momento uma das seguintes situações, tomando as decisões apresentadas.

Obs: As soluções alternativas são fornecidas, em ordem de prioridade, caso todos os vértices do conjunto de saída sejam adjacentes a todos os vértices do conjunto de chegada.

a. C-L disjuncto de L-C

1. $C-L \rightarrow L-C.$

1.1. $\hat{V}-(L \cup C) \rightarrow \hat{V}-(L \cup C).$

1.2. $C-L \rightarrow \hat{V}-(L \cup C).$

1.3. $\hat{V}-(L \cup C) \rightarrow L-C.$

b. $C-L = \emptyset, L-C \neq \emptyset$

1. $\hat{V}-L \rightarrow L-C.$

1.1. $\hat{V}-L \rightarrow \hat{V}-L.$

c. $L-C = \emptyset, C-L \neq \emptyset$

1. $C-L \rightarrow \hat{V}-C.$

1.1. $\hat{V}-C \rightarrow \hat{V}-C.$

d. $C = L$

1. $\hat{V}-L \rightarrow \hat{V}-L.$

e. $C = L$ e $|\hat{V}-L| = 1$

1. $\hat{V}-L \rightarrow L.$

Se $\hat{V}-(L \cup C) = \emptyset$ e não podemos aplicar a.1, então nenhuma outra solução é ou será ainda aplicável. Neste caso retornamos à heurística original: (fonte) mínimo - (sorvedouro) máximo, utilizada em MODELO.

Lembramos novamente que o emprego da heurística mínimo-mínimo tem por objetivo o adiamento de $\hat{V}-(L \cup C) = \emptyset$, fazendo L e C crescerem mais lentamente. Em algumas PLAs é possível que o dobramento seja completado satisfatoriamente sem que tal condição ocorra.

O uso das listas L e C foi idealizado visando à simplicidade de computação. Garante-se, pelo critério de seleção, a não criação de circuitos alternados após a adição de cada aresta. A função CYCLE faz-se desnecessária, abolindo-se a verificação passo a passo.

A atualização do fecho transitivo $Q^+(O(A))$ torna-se extremamente fácil no algoritmo ALG-TIGER-I. A lista TRANS do vértice-fonte v recém escolhido compor-se-á em a, b, c, d apenas de seus adjacentes. Já em e necessitaremos de mais um passo, pois $u \in C$ e e é adjacente a alguma fonte. Contudo, neste caso, teremos $\hat{V}-(L \cup C) = \emptyset$ após e . Voltando o comando para o algoritmo MODELO não modificado, poderemos utilizar seu procedimento TRANSCLOS, já descrito, a partir desse ponto.

Em contrapartida, temos que manter atualizadas as listas $L, C, \hat{V}-(L \cup C), L-C$, e $C-L$. Operações de união, intersecção e diferença entre conjuntos fazem-se necessárias, bem como na seleção do par (v, u) . Cada operação destas tem complexidade $O(n)$ [19]. Um algoritmo que implemente esta heurística e tenha como dados uma PLA com n colunas, pode vir a rodar em $O(n^3)$, pois num péssimo caso a escolha de (v, u) consumirá $O(n^2)$.

Porém, não há porquê fazermos grandes estudos acerca da complexidade de tempo para esta heurística. Num caso ruim será imprescindí-

vel o desvio para MODELO não modificado, a fim de completarmos o dobramento, e sua complexidade é $O(n^3)$ [1].

A seguir mostramos um Grafo de Intersecção de Colunas de uma PLA, ao qual foram aplicadas as duas heurísticas discutidas acima. Note que a solução alcançada por ALG-TIGER-I é melhor que a de MODELO.

Na verdade, uma solução ser melhor que a outra não implica que a heurística também o é. Em ALG-TIGER-I o fator aleatório na escolha das arestas é muito menor que em MODELO. Somos inclusive levados a intuir que alcançaríamos melhores dobramentos através do primeiro. No entanto, devido ao grau de liberdade existente em certas instâncias da seleção dos vértices, não nos é possível provar tal intuição.

Exemplo $G(V,E,A)$

Legenda

Arestas de $G(V,E,A)$:

—: $\in E$

---: $\in A$ de ALG-TIGER-I

—: $\in A$ de alg. em [1]

$\# A(\text{ALG-TIGER-I}) = 10$

$\# A(\text{alg. em [1]}) = 9$

A numeração das arestas corresponde à ordem em que foram escolhidas para o matching.

Situação de L e C em ALG-TIGER-I após a colocação da aresta:

1. (R,H)	$L = \{K\}, \quad C = \{S\}$
2. (S,K)	$L = \{J,N\}, \quad C = \{N,P,T\}$
3. (P,J)	$L = \{N,E\}, \quad C = \{N,O,T\}$
4. (D,E)	$L = \{N,A,B\}, \quad C = \{N,L,T\}$
5. (T,B)	$L = \{N,A\}, \quad C = \{N,L\}$
6. (L,A)	$L = \{N,F\}, \quad C = \{N,G\}; \quad G,F \text{ adj's! Faço } \bar{V} \setminus (LuC) \rightarrow \bar{V} \setminus (LuC).$
7. (C,D)	$L = \{N,F,G\}, \quad C = \{N,G\}; \quad L \not\subseteq C, \bar{V} \setminus L \neq L \setminus C$
8. (I,F)	$L = \{N,G\}, \quad C = \{N,G,M\}; \quad L \subseteq C, C \setminus L \neq \bar{V} \setminus C$
9. (M,Q)	$L = \{N,G\}, \quad C = \{N,G\}; \quad L = C, \bar{V} \setminus (LuC) = \emptyset$, desvio p/ alg. em [1] que tanto pode escolher a aresta (N,G) como (G,N) sem nenhum problema.
10. (G,N)	

III UMA ÓTIMA SOLUÇÃO OPTIMAL PARA DOBRAMENTO MÚLTIPLO

1. Introdução

Neste capítulo estudaremos o dobramento múltiplo, no qual cada coluna física pode receber um número qualquer de colunas lógicas.

No capítulo I foi colocado que o PDO é também um problema de permutação das linhas de uma PLA. O algoritmo MODELO, lá apresentado, deixa transparecer um certo distanciamento desta idéia, seja com a heurística de Hachtel et alii [1] ou com a nossa. Naquela solução, uma vez conseguido o emparelhamento A, sabemos a posição relativa das colunas mas não qual a permutação exata das linhas na PLA dobrada.

Ao voltarmos nossas atenções para o enfoque "Permutação e Compactação", conseguimos desenvolver um método para efetuar o dobramento múltiplo que é ótimo para determinadas PLA,s e excelente para todas elas. Tal método leva a um algoritmo quadrático que soluciona ótima-
mente o PDO múltiplo de PLA,s que possuam certa propriedade. Considerando que o tempo necessário para fazer a leitura de uma PLA com n linhas e n colunas é quadrático, podemos concluir que nosso algoritmo é assintoticamente optimal.

Trabalhamos sobre idéias lançadas por Paillotin [2] - direta-

mente interessado em otimização de PLA,s -, passando por um algoritmo de Booth & Lueker [17] - preocupados com estruturas de dados eficientes para solucionar problemas em Grafos. Os conceitos que se seguem foram primeiramente compilados em Ferreira [15].

A partir deste ponto estudaremos o Problema do Dobramento Optimal dividindo-o em duas etapas distintas:

1. Permutação das linhas com o intuito de obter o máximo número de colunas compatíveis, onde colunas são compatíveis se elas podem ser organizadas em apenas uma coluna física.
2. Compactação da PLA: as colunas são reorganizadas e a PLA dobrada é obtida.

2. Fase de Compactação: quadrática e ótima

São necessárias algumas definições de Paillotin que ora colocamos em nossos termos. Utilizaremos novamente a Matriz de Personalidade Modificada (MPM), introduzida no capítulo I, para a representação das PLA,s.

Portão: é a parte da coluna compreendida entre os dois ls mais extremos, incluindo-os.

Função Pico: para uma dada permutação das linhas (configura-

ção), o pico de uma linha é igual ao número de portões por ela cruzados. É calculada independentemente para cada plano.

P_* : é o máximo da função Pico (para cada plano).

Relação 1: um portão p está completamente abaixo de um portão t se seu extremo de cima está abaixo do extremo de baixo de t.

Relação 2: um portão p está acima de um portão t se seu extremo de cima está acima do extremo de cima de t.

Vejamos exemplo na figura 1.

	A	B	C	D	
1	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	0	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	0	* Pico(6) = 1.
2	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	0	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	0	* $P_* = 2$.
3	0	0	0	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	* pD R1 pA.
4	0	0	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	* pC R2 pD.
5	0	0	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	
6	0	$\left \begin{array}{c} 1 \\ 1 \end{array} \right $	0	0	

-fig 1-

Dada uma certa configuração da MPM de uma PLA, Paillotin descreve um algoritmo para alocação dos portões nas colunas, ou seja, para efetuar a compactação daquela específica configuração.

Algoritmo PTIN

- a. Considere a primeira coluna física.
- b. Posicione na coluna o portão mais acima dentre todos e retire-o da lista de portões.
- c. Considere o conjunto de colunas completamente abaixo do último portão alocado. Posicione na coluna o mais acima e retire-o do conjunto de portões. Isto é repetido até que o conjunto dos portões completamente abaixo do último colocado seja vazio.
- d. Se {ainda há portões} então {considere a próxima coluna e vá para b}.
- e. Pare.

Este algoritmo é utilizado na prova do próximo teorema:

TEOREMA P: Para uma determinada configuração da MPM K de uma PLA, seus portões podem ser reorganizados em P_* colunas.

Dem: Por contradição, suponha que PTIN não aloca os portões em P_* colunas. Então existe ao menos um portão p na (P_*+1) -ésima coluna. Como P_* é o máximo da fun-

ção Pico, a linha r que cruza o 1 extremo de cima de p , cruza um máximo de $P_* - 1$ portões nas P_* colunas anteriores. Logo, existe ao menos uma coluna, q , sem portão na linha r . Todavia o algoritmo, no passo c ., deveria ter posicionado p em q .

3. Um limite para o dobramento ótimo

Através deste teorema é-nos possível determinar o número exato de colunas necessárias para dobrar-se uma configuração (permutação das linhas) de uma MPM.

Existe ao menos uma permutação de linhas que permite a melhor compactação possível, aquela que possui o P_* mínimo (P_{*min}). Assim, qualquer compactação de qualquer configuração utilizará $Z \geq P_{*min}$ colunas em sua implementação.

Se acharmos um limite inferior para P_{*min} , teremos como calcular um limite de tolerância para soluções alcançadas por algoritmos heurísticos ou mesmo exaustivos. Logo, se soubermos de antemão qual o número mínimo de colunas necessárias para implementar-se a MPM dobrada, poderemos discutir acerca da aceitabilidade de uma solução. Há inclusive a possibilidade de, dada uma PLA, não ser necessário estudarmos suas permutações. Aquela fornecida pode vir a gerar um excelente dobramento, quiçá ótimo.

Vejamos então como chegar a este limitante.

Um portão é composto por linhas que contém 1 e linhas que contém 0. Uma configuração na qual os portões são formados apenas por 1s (não há 0s entre os 1s extremos) tem o máximo da função Pico - P_* -, calculado na linha que cruza o maior número de 1s.

Seja U a função que computa, para cada linha, o número de 1s que ela contém. Seja U_* o máximo desta função. Observe que, dada uma MPM, os valores da função U e seu máximo são fixos para qualquer configuração.

O próximo teorema é, apesar de bastante simples, de suma importância para o PDO, devido a seus dois corolários.

TEOREMA T1: Para qualquer MPM, $U_* \leq P_{\min}$, i.e., U_* é limite inferior para qualquer dobramento múltiplo efetuado em PLAs.

Dem: A primordial restrição ao dobramento é a disjunção das colunas (não existência de 1s em linhas comuns), assim nenhum dobramento conseguirá reduzir a MPM a menos de U_* colunas.

COROLARIO 1: Um dobramento que implemente uma MPM em U_* colunas é um dobrimento ótimo.

COROLARIO 2: Uma permutação das linhas de uma MPM cujos portões contenham apenas 1s é uma permutação (configuração) ótima, no sentido de que viabiliza um dobramento igual ao ótimo.

4. Fase de Permutação: linear e ótima para certas PLAs

Uma permutação de uma MPM que forneça portões formados apenas por 1s é ótima. Ela obtida, saberemos reduzi-la a exatas $P_* = U_*$ novas colunas. Resta-nos, então, descobrir um algoritmo que ache tal configuração ótima.

Durante nossas pesquisas deparamos com um algoritmo linear que executa esta tarefa para algumas PLAs especiais; aquelas cujas MPMs possuam uma determinada propriedade.

Uma matriz na qual uma permutação de suas linhas contenha os portões como acima descritos, é conhecida na teoria dos Grafos Perfeitos como possuidora da "Propriedade dos Uns Consecutivos" (PUC) para colunas. Tais matrizes são objeto de estudo detalhado, pois Golumbic [19] provou que se a matriz-clique de um grafo possui a PUC, então o grafo é um grafo-intervalo.

Na mesma obra, Golumbic cita um algoritmo (de Booth & Lueker [17]) que testa se uma dada matriz possui ou não a PUC.

Sejam m e n as dimensões da matriz dada e f o número de entradas não nulas. O algoritmo de Booth & Lueker roda em $O(m + n + f)$ e pára com uma resposta positiva ou negativa, na dependência de a matriz possuir ou não a PUC. Ao final do processamento estará também disponível, se existir, uma permutação com os portões com os uns consecutivos. Referir-nos-emos a este algoritmo por ACHAPPUC.

Com o visto neste item e no anterior, fica estabelecido o resultado a seguir.

TEOREMA T2: Seja uma MPM ($m \times n$). Se ela possui a PUC para colunas então, através de ACHAPPUC conseguimos sua configuração ótima em tempo linear, $O(m + n + f)$.

Dem: Segue do já exposto.

É importante ressaltarmos que, se por um lado o limite inferior U_* não é necessariamente igual a P_{*min} , em muitas instâncias acharemos configurações onde $P_* = U_*$, e portanto iguais a P_{*min} , sem que estejamos na permutação dos uns consecutivos.

Infelizmente não sabemos qual a abrangência da classe de matrizes com PUC dentre as PLAs. Num primeiro momento ela assemelhou-se grande; entretanto Tucker [18] demonstra que se uma matriz possui alguma permutação que contenha uma das submatrizes reproduzidas abaixo, então ela não possui PUC para colunas.

-fig 2-

Este resultado é muito forte e nos indica que tal classe talvez seja, em verdade, pouco abrangente. Nosso resultado, apesar de excelente em termos teóricos, pareceu-nos de pouca aplicabilidade prática da maneira como proposto. Desenvolvemos, por conseguinte, uma heurística sobre o ACHAPPUC que se mostrou realmente útil nos exemplos pesquisados.

5.0 algoritmo PROCMP

Havia várias soluções polinomiais para o problema dos uns consecutivos. Para chegar a um algoritmo linear, Booth & Lueker desenvolveram uma estrutura de dados especial chamada "árvores PQ" que opera muito bem o posicionamento relativo de subconjuntos de um conjunto dado.

A idéia subjacente ao algoritmo ACHAPPUC é de fácil concepção. As colunas são examinadas uma a uma, da esquerda para a direita. Para cada coluna procura-se uma permutação das linhas que permita que seus 1s fiquem consecutivos sem, contudo, atrapalhar a consecutividade nas colunas já examinadas. Para tanto cria-se uma família de subconjuntos (subconjuntos restrição) do conjunto de linhas. Cada subconjunto é associado a uma coluna, contendo as linhas que nela têm 1. Trata-se dos mesmos subconjuntos $R(c_j)$ de I.4.

Devido às restrições advindas do estudo das colunas anteriores, num determinado momento pode não existir permutação em que ocorram

consecutivamente também os ls da coluna ora analisada. Aí ACHAPPUC pára com resposta "não possui PUC".

Caso o algoritmo processe todas as colunas com sucesso, a resposta será "possui PUC".

Ao final da execução, em ambos os casos, haverá nas folhas da árvore PQ uma permutação que torna consecutivos os ls das colunas processadas até a parada. O algoritmo é portanto construtivo.

Nosso algoritmo (PROCMP) procurará a melhor permutação para a MPM dada, sem se preocupar com a PUC. Com este fim, nossa heurística busca aumentar o número de colunas examinadas pelo algoritmo. Uma pequena alteração em ACHAPPUC e a criação de um critério de escolha da ordem em que os subconjuntos restrição serão analisados são suficientes para levar-nos a permutações com P_* muito próximo de U_* .

Como vimos, ACHAPPUC pára assim que examinar um subconjunto restrição para o qual não haja permutação das linhas que o satisfaça simultaneamente com os demais já analisados. Em contrapartida, nosso PROCMP verificará toda a família de subconjuntos restrição. Se porventura a restrição de um deles não puder ser satisfeita, o algoritmo ignorá-lo-á e iniciará o processamento do próximo. Na configuração final da MPM a coluna a ele associada não terá os ls consecutivos, o que não impede que ela venha a ser dobrada na fase de Compactação.

Há pelo menos duas maneiras de ordenarmos os subconjuntos restrição dentro de uma mesma linha de raciocínio:

- A. Tomam-se os subconjuntos em ordem crescente de cardinalidade.
- B. Tomam-se os subconjuntos em ordem decrescente de cardinalidade.

Ao optarmos por uma ou por outra devemos pensar que quanto maior o número de 1s em um subconjunto restrição já satisfeito pelo algoritmo, maior a dificuldade para obter-se sucesso com os próximos. As restrições criadas serão muitas sobre muitas linhas a um mesmo tempo. Ademais uma coluna com muitos 1s gerará um grande portão, mesmo que seus 1s estejam consecutivos. Assim, a aplicação da opção A pareceu-nos teoricamente melhor que a de B.

Na prática porém, B mostrou-se mais eficiente quando tentamos dobrar PLAs densas. No entanto, como a aplicação do algoritmo é relativamente "barata", se comparada com a totalidade dos custos envolvidos, uma solução talvez seja testar as duas heurísticas separadamente e utilizar o melhor resultado obtido.

Acompanhe em detalhes nosso algoritmo processando a MPM de uma PLA do micro-processador 8085 no anexo I.

6. Comentários

A aplicação do algoritmo ACHAPPUC a MPMs nos parece desinteressante. Se sua resposta for "não possui PUC", várias colunas podem não ter sido rearranjadas sem, no entanto, haver impedimentos para tal.

É mais aconselhável o uso direto de PROCMP que, além de executar exatamente o mesmo que ACHAPPUC no caso ótimo, continua perseguindo boas soluções ainda que a MPM não possua a PUC.

A alteração no algoritmo original de Booth & Lueker não modifi-

ca sua complexidade, pois cada processamento de PROCMP varrerá todos os subconjuntos restrição; o que representaria estarmos sempre no caso ótimo em ACHAPPUC. Nosso algoritmo gasta, então, $O(n \log n)$ na ordenação dos subconjuntos restrição e $O(m + n + f)$ na procura da melhor permutação.

Concluindo, propomos o esquema DOBRABEM abaixo para o dobramento múltiplo de PLAs:

a. ler a MPM - $O(m.n)$	:	
b. rodar PROCMP - $O(n.\log n)$	}	fase de Permutação
c. realocar a MPM - $O(m.n)$	:	
d. rodar PTIN - $O(n^2)$	}	fase de Compactação
e. gravar a MPM dobrada $O(m.P_*)$	:	

Supondo $m > n$, bastante usual nos casos concretos, teremos um algoritmo em $O(m^2)$ para executar o dobramento de MPMs de PLAs. O dobramento ótimo quando alcançado é, através do limitante inferior U_* , algumas vezes indicado como sendo o ótimo. Por outro lado, como U_* é limite inferior da função Pico e não seu mínimo, não podemos garantir a existência de uma permutação das linhas da MPM que nos leve a uma implementação com U_* colunas.

Simulamos o esquema DOBRABEM para dobrar alguma PLAs reais - também do micro-processador 8085 -, e compilamos os resultados na tabela a seguir. As MPMs destas PLAs e suas representações após o dobramento múltiplo ter sido executado estão no anexo II.

As entradas da tabela indicam: número da MPM sendo dobrada; com LIN linhas, CO colunas e ENN entradas não nulas; limite inferior U_* ;

P_* encontrado, i.e., número de colunas em que DOBRABEM organiza a MPM dobrada; FOE fator ótimo de economia $[(C_0 - U_*)/C_0]$; FEE fator de economia encontrado $[(C_0 - P_*)/C_0]$.

MPM	LI	C0	ENN	U*	P*	FOE	FEE
1	45	06	061	03	03	50%	50%
2	53	29	097	05	09	83%	69%
3	58	19	063	02	03	89%	84%
4	57	19	122	03	05	84%	74%

-Tabela 1-

IV DOBRAMENTO MÚLTIPLO RESTRITO: O DOBRAMENTO SIMPLES

1. Variações sobre um mesmo tema

Após a execução de um dobramento múltiplo, conectar as colunas mais internas da PLA pode significar um real problema para a engenharia elétrico-eletrônica. Apesar de já existirem estudos e mesmo algoritmos para a resolução de problemas de "routing" e conexão dos sinais com o circuito externo [3], é mister concentrarmos esforços na procura de boas soluções também para o dobramento simples de PLAs.

Um máximo de duas colunas lógicas compartilharão da mesma coluna física. Isto restringirá a alocação dos sinais de entrada e saída ao plano vertical, ou acima da PLA ou abaixo dela, facilitando a fase do Projeto Físico, introduzida no capítulo I.

O limite inferior para o dobramento, antes característico da PLA dada, dependerá agora também de aspectos físicos, pois metade das colunas será conectada a um lado da PLA e metade ao outro. Portanto, a melhor dobra possível não será implementada em menos de $n/2$ colunas.

Na busca da melhor solução para o PDO simples, apresentamos no capítulo II duas heurísticas baseadas numa mesma concepção do problema. Agora mudaremos o enfoque e faremos uma abordagem semelhante à

apresentada no capítulo anterior. Alguns resultados lá obtidos verificar-se-ão válidos e mesmo o algoritmo PROCMP será utilizado para a fase de Permutação. Devido à restrição exigida, entretanto, a fase de Compactação teve que ser reformulada e sua complexidade tornou-se algo usualmente bem melhor que $O(n^{2.5})$.

2. Compactação: Novamente ótima

Para uma determinada configuração de uma MPM de uma PLA, podemos associar o Grafo de Intersecção de Portões (GIP), $G = (V, E)$, onde os vértices em V estão numa correspondência um-a-um com as colunas da MPM e $\{v_i, v_j\}$ é uma aresta de E se e somente se o portão da coluna i cruza ao menos uma linha cruzada também pelo portão da coluna j .

MPM:	A	B	C	D
1	1	0	1	0
2	1	0	1	0
3	0	0	0	1
4	0	0	1	1
5	0	0	1	1
6	0	1	0	0

GIP:

-fig 1-

No processo de alocação das colunas lógicas em físicas, não podemos organizar numa mesma coluna física duas colunas lógicas cujos portões se sobreponham. A única restrição ao dobramento de duas colunas lógicas é que seus respectivos vértices no GIP sejam adjacentes.

LEMA T1: Dada uma configuração de uma MPM, um emparelhamento no grafo de intersecção de portões induzido fornece uma compactação daquela configuração.

Dem: Basta dobrarem-se os portões associados às arestas do emparelhamento, alocando dois portões por coluna.

Há várias soluções polinomiais para o problema do emparelhamento máximo em grafos quaisquer. A mais famosa, de autoria de Edmonds [23], gasta tempo proporcional a $O(|V|^4)$, enquanto que a mais rápida, atualmente, acha o melhor emparelhamento em $O(|V|^{1/2} \cdot |E|)$ e é de autoria de Micali & Vazirani [14]. Pode-se dizer que o segundo algoritmo é um melhoramento do primeiro, pois ambos se aproximam do problema da mesma maneira [21].

TEOREMA T1: Dado $G=(V,E)$, o grafo de intersecção de portões de uma configuração de uma MPM, podemos obter a compactação ótima desta configuração em $O(|V|^{0.5} \cdot |E|)$.

Dem: Através do algoritmo de Micali & Vazirani conseguimos um emparelhamento máximo no GIP com a complexi-

dade acima. O Lema T1 indica que os pares de vértices correspondentes ao emparelhamento podem ser encarados como os pares de colunas a serem compactados. Como o emparelhamento é máximo, teremos a compactação ótima.

A cardinalidade do conjunto V dos vértices do grafo de intersecção de portões é n (colunas da MPM). Logo $O(|V|^{0.5} \cdot |E|) \leq O(n^{2.5})$.

Ao aplicarmos este Teorema ao caso específico de PLAs reais, devemos recordar-nos de que o conjunto E será usualmente pequeno, pois as MPMs serão muito esparsas. Desta forma o algoritmo não estará muito longe de possuir ordem quadrática, como no capítulo anterior.

No dobramento simples é imediato que qualquer dobramento que reduza a MPM a $\lceil n/2 \rceil$ colunas (o menor inteiro maior ou igual a $n/2$) é um dobramento ótimo. Chamaremos este limitante de físico, pois não depende das características da PLA como no dobramento múltiplo. Por outro lado, permanece impossível compactarmos uma configuração de uma MPM em menos de P_* colunas já que não há como organizarmos conjuntamente dois portões que se interceptam. Assim, conseguimos o mesmo limite inferior lógico também para o dobramento simples.

PROP. 1: Dada uma configuração de uma MPM, o dobramento simples ótimo compacta-la-á em não menos que $\max\{\lceil n/2 \rceil, P_*\}$ colunas.

Prova: Direto do exposto acima.

Neste caso, infelizmente, P_* colunas são necessárias mas nem sempre suficientes para compactar os n portões da MPM. Vejamos um exemplo desta afirmação:

MPM:

	A	B	C	D	E	F
1	1	0	1	0	0	0
2	0	1	0	1	0	0
3	1	1	0	0	1	0
4	0	1	0	0	0	1
5	0	0	1	0	0	0
6	1	1	0	0	0	0

$P_* = 4.$

Não podemos, entretanto, compactar esta configuração em menos de 5 colunas. Experimente!!

-fig 2-

3. Permutação: Novamente a PUC

Já que nem sempre compactaremos uma MPM em exatas P_* colunas, não podemos lançar mão da função U e seu máximo U_* para provar algo em relação à melhor permutação. Provaremos que a permutação dos 1s conse-

cutivos é a que permite a melhor compactação possível através de um outro caminho.

Sejam:

M : uma configuração de uma MPM.

$B(M)$: conjunto dos pares de colunas de M que foram efetivamente dobradas na compactação ótima de M .

$RP_M(k)$: conjunto das linhas de M que cruzam o portão da coluna k .

$RU(k)$: conjunto das linhas da MPM que têm 1 na coluna k .

PROP. 2: Dada uma MPM, se $r \in RU(c)$, então $r \in RP_N(c)$ para qualquer configuração N da MPM. Isto é, se a linha r tem 1 na coluna c , então r sempre cruzará o portão da coluna c , não importa a configuração.

Prova: Imediato.

Lema T2: Seja M uma configuração qualquer de uma MPM.

Se $\{a, c\} \in B(M)$, onde a e c são colunas da MPM, então $\{a, c\}$ podem ser dobradas na permutação dos 1s consecutivos (PC), se esta existir.

Dem: Como $\{a, c\} \in B(M)$, temos $RP_M(a) \cap RP_M(c) = \emptyset$.

Assim $RU(a) \cap RU(c) = \emptyset$ (Prop. 2).

Logo, se PC existe então, também pela Prop. 2:

$RP_{PC}(a) \cap RP_{PC}(c) = \emptyset$.

Portanto a e c podem ser dobradas também em PC.

Temos agora condições de provar uma afirmação semelhante àquela do capítulo anterior.

TEOREMA T2: Dada uma MPM de uma PLA, a permutação dos 1s consecutivos, se existir, permite - após a fase de Compactação -, um dobramento simples igual ao ótimo.

Dem: Seja PC a permutação dos 1s consecutivos.
Seja COT uma configuração ótima da MPM, que permite o dobramento simples ótimo.

Como COT é ótima, $|B(PC)| \leq |B(COT)|$ (1)

Por outro lado, para todo par de colunas em B(COT) vale o Lema T2. Logo:

$|B(COT)| \leq |B(PC)|$ (2)

De (1) e (2) vem que:

$|B(COT)| = |B(PC)|$

e portanto PC leva a um dobramento simples igual ao ótimo.

COROLARIO: Uma permutação das linhas de uma MPM cujos portões contenham apenas 1s é uma permutação (configuração) ótima, no sentido de que viabiliza um dobramento ótimo.

A esta altura já sabemos qual é o desfecho que se prepara, entretanto faz-se necessário afirmar que:

TEOREMA T2: Seja dada uma MPM ($m \times n$) com f entradas não nulas. Se ela possui a PUC para colunas então através de ACHAPPUC conseguimos sua configuração ótima também para o caso de dobramento simples em tempo linear, $O(m + n + f)$.

Dem: Vide Teorema T2 do capítulo III.

Todas as considerações feitas no capítulo anterior sobre o algoritmo ACHAPPUC aplicam-se também para este caso. Desta maneira utilizaremos PROCMP aqui como lá. Vejamos como.

4.0 esquema DOBRABEM

Assim como criamos o esquema DOBRABEM para o dobramento múltiplo, indicaremos também aqui um esquema semelhante para solucionar o problema do dobramento simples.

Um algoritmo para o dobramento simples de colunas em uma MPM deve executar o seguinte:

- a. ler a MPM - $O(m.n)$
- b. rodar PROCMP - $O(n.\log n)$
- c. realocar a MPM - $O(m.n)$
- d. gerar o GIP associado - $O(n^2)$
- e. achar emparelhamento máximo - menor que $O(n^{2.5})$
- f. compactar a MPM - $O(n^2)$
- g. gravar a MPM dobrada - menor que $O(m.n)$

Logo após o Teorema T1 dissemos que, em casos concretos, o conjunto de arestas do GIP (grafo de intersecção de portões) associado é bem menor que n^2 (n é o número de portões da MPM ou de vértices do GIP). Temos assim um meio para executar o dobramento de MPMs de PLAs que possui complexidade de tempo bem melhor que $O(n^{2.5})$, talvez praticamente quadrático.

Através da Prop. 1 podemos algumas vezes indicar que chegamos a um dobramento ótimo. O grande problema enfrentado agora é que P_* não mais representa o mínimo para o número de colunas necessárias após o dobramento, mas antes seu limite inferior. Vimos na figura 2 um bom exemplo disto. Por outro lado resultados de aplicações mostraram a possibilidade do problema não ser tão grave na prática.

Testamos nosso esquema em quatro PLAs do micro-processador 8085 e em uma outra fornecida em Hachtel et alii [1]. A exceção desta última em que dobramos 17 colunas, em todas as outras chegamos ao dobramento ótimo com $n/2$ colunas dobradas, e mesmo no caso daquela PLA (N. 4 na tabela a seguir), 17 colunas foi provado ser dobramento ótimo [7].

Na TABELA 1 temos as MPMS com CO colunas, LI linhas, ENN entradas não nulas e CD pares de colunas dobradas por DOBRABEM simples. Em todas elas o dobramento ótimo foi alcançado.

PLA	CO	LI	ENN	CD
1	19	57	122	09
2	19	58	063	09
3	29	53	097	14
4	42	52	252	17
5	06	45	061	03

-Tabela 1-

CONCLUSÃO

O Problema do Dobramento Optimal de PLAs (PDO) foi aqui apresentado e discutido.

A modalidade do dobramento simples havia sido proposta por Hachtel et alii [1]. Para uma PLA de n linhas e n colunas, o algoritmo deles é $O(n^3)$ e, sendo heurístico, apresenta soluções não necessariamente ótimas. O algoritmo por nós elaborado é melhor que $O(n^{2.5})$. Além de possuir menor complexidade de tempo, ele alcança soluções ótimas para certas PLAs e, como vimos, estima-se que para a grande maioria delas os resultados sejam excelentes.

A modalidade do dobramento múltiplo recebera uma nova abordagem por parte de Paillotin [2], cujo algoritmo é de $O(n^4)$. Aproveitando-nos de suas idéias, formulamos um novo método para a resolução do problema. Chegamos a partir daí a um algoritmo em $O(n^2)$ que apresentou resultados surpreendentes. Lembramos que necessitamos de no mínimo $O(n^2)$ para operações de entrada e saída em uma PLA ($n \times n$), logo nosso algoritmo é ótimo em termos assintóticos.

Nossas soluções não vieram a ser implementadas, entretanto simulâmo-las à mão para termos uma idéia de como funcionariam. Apesar de ter sido um processo bastante trabalhoso foi também gratificante. Se os dobramentos obtidos com as PLAs do micro-processador 8085 mostra-

rem-se usuais para quaisquer PLAs, teremos criado um método para resolver o PDO que é ótimo em termos teóricos e "quase-ótimo" na prática.

Sabemos que uma questão pode ter surgido nos leitores mais atentos: qual seria a distância entre as soluções obtidas por nosso algoritmo e as ótimas possíveis?

Nesta dissertação não nos arriscamos a responder tal pergunta, embora discussões anteriores tenham nos deixado a impressão de que nossas soluções estarão sempre muito próximas das ótimas. Quiçá o mais próximo possível para algoritmos heurísticos.

Talvez possamos vir a fazer disto uma linha de pesquisa futura. Nossa ou de eventuais orientandos.

ANEXO I

PLA do processador 8085

G01							
G02							
G03					1	1	
G04							
G05					1	1	
G06	1		1	1			
G07							
G08	1		1	1			
G09						1	
G10						1	
G11	1			1			
G12					1		
G13				1			
G14				1			
G15					1	1	
G16		1				1	
G17		1				1	
G18		1					1
G19					1		
G20			1		1		
G21					1		
G22					1		
G23					1		
G24					1		
G25					1	1	
G26	1		1	1			
G27					1		
G28			1				
G29					1		
G30					1		
G31					1		
G32					1		
G33			1				
G34			1	1			
G35			1	1			
G36				1			
G37			1				
G38			1				
G39							
G40		1				1	
G41	1					1	
G42					1	1	
G43					1		
G44	1					1	
G45					1	1	
G46							
G47							
G48							
G49							

INPUTS: FROM DECODER OF PLA1

PLA - 8085
48 linhas
06 colunas
61 entradas
não nulas
 $P^* = P(27) = 6$
 $U^* = U(27) = 3$

ST1 ST2 ST3 ST4 ST5 ST6

OUTPUTS: TO M-CAT-S GENERATORS

Os subconjuntos restrição das colunas serão:

SRC(1)={6,8,11,27,41,44}; SRC(2)={16,17,18,40}; SRC(3)={6,8,20,27,29,33,34,35,37,38}; SRC(4)={6,8,11,13,14,27,34,35,36}; SRC(5)={3,5,12,15,19,20,21,22,23,24,25,28,30,31,32,42,43,45}; SRC(6)={3,5,9,10,15,16,17,18,25,40,41,42,44,45}.

Podemos tornar consecutivas as colunas de SRC(5), SRC(6), SRC(3), SRC(4), falhamos para SRC(1) e temos sucesso também com SRC(2).

Estratégia utilizada: C, subconjuntos em ordem decrescente de cardinalidade.

A falha é devida a: as linhas 41 e 44 de SRC(1) devem ficar juntas com SRC(6) que ficará abaixo de SRC(5) pois há intersecção. Esta por sua vez ficará abaixo de SRC(3) - causa: linha 20 -, que deve estar junto de SRC(1). Logo é impossível tornar os uns de SRC(1) consecutivos após alocar SRC(6), (5) e (3).

		1	2	3	4	5	6
Configuração da PLA após a exe- cução do algo- ritmo:	11	1	0	0	1	0	0
	13	0	0	0	1	0	0
	14	0	0	0	1	0	0
	36	0	0	0	1	0	0
	06	1	0	1	1	0	0
	08	1	0	1	1	0	0
	27	1	0	1	1	0	0
	34	0	0	1	1	0	0
	35	0	0	1	1	0	0
	38	0	0	1	0	0	0
	37	0	0	1	0	0	0
	33	0	0	1	0	0	0
	29	0	0	1	0	0	0
	20	0	0	1	0	1	0
	12	0	0	0	0	1	0
	19	0	0	0	0	1	0
	21	0	0	0	0	1	0
	22	0	0	0	0	1	0
	23	0	0	0	0	1	0
	24	0	0	0	0	1	0
	28	0	0	0	0	1	0
	30	0	0	0	0	1	0
	31	0	0	0	0	1	0
	32	0	0	0	0	1	0
	43	0	0	0	0	1	0
	03	0	0	0	0	1	1
	05	0	0	0	0	1	1
	15	0	0	0	0	1	1
	25	0	0	0	0	1	1
	42	0	0	0	0	1	1
	45	0	0	0	0	1	1
	09	0	0	0	0	0	1
	10	0	0	0	0	0	1
	41	1	0	0	0	0	1
	44	1	0	0	0	0	1
	16	0	1	0	0	0	1
	17	0	1	0	0	0	1
	18	0	1	0	0	0	1
	40	0	1	0	0	0	1

P*=3=P(35)

U*=3=U(27)

Logo P*=U* =>

Configuração Ótima

Dobram-se as colu

nas (4,5,2); (1);

(3,6) ou (1,2);

(4,5);(3,6).

Dobramento múlti-

plo igual ao sim-

ples:

TODOS ÓTIMOS !!!

ANEXO II

Exemplo da aplicação de ACHAPPUC a
PLA's reais - do processador 8085.

A large grid of graph paper with a pattern of small '1' characters scattered across it, resembling a sparse matrix or a binary image. The '1's are distributed in a way that suggests a complex, non-random pattern, possibly representing a specific data set or a mathematical structure.

.58 columnas

• 63 entradas não nulas

$$P^*=P(22)=13$$
$$U^*=U(29)=02$$

À direita, a mesma PLA após ACHAPUC

$$P^* = P(34) = 0.3$$

Limite para Redução Ótima: 89% (D. Múltiplo)

Redução alcançada: 84%

Estratégia: B - ordem decrescente

[illegible]

A large grid of graph paper, approximately 30 units wide and 40 units high. The word "OUTPUTS" is written vertically along the right edge of the grid, oriented sideways. The letters are in a bold, sans-serif font. The grid lines are thin and black, forming a uniform pattern of squares. The paper appears to be a standard sheet of graph paper used for technical drawing or data plotting.

INPUTS: FROM DECODING OF FLAG

Acima, PLA com 29 linhas, 53 colunas e 97 entradas não nulas.

$$P^* = P(28) = 17$$
$$U^* = U(33) = 0.5$$

Ao lado, a mesma PLA após a execução de ACHAPPUC.

$$P^* = P(33) = 0.9$$

Limite p/ Redução Ótima: 83% (D. Múlt.)

Redução Alcançada: 69%

Estratégia: A - ordem crescente.

[illegible]

REFERENCIAS BIBLIOGRAFICAS

1. Hachtel, G.D. et alii, An algorithm for optimal PLA folding, Research Report RC8668, IBM Thomas J. Watson Research Center, Jan 81.
2. Paillotin, J.F., Optimization of the PLA area, Proc. 18th Des. Autom. Conf., 1981, pp 406-410.
3. Chuquillanqui, S., Internal connection problem in large optimized PLAs, Proc. 20th Des. Autom. Conf., 1983, pp 795-802.
4. Hu, T.C. & Kuo, Y.S., Optimum reduction of programmable logic array, Idem, idem, pp 553-558.
5. Micheli, G.D. & Sangiovanni-Vincentelli, A., PLEASURE: A computer program for simple/multiple constrained/unconstrained folding of programmable logic arrays, Idem, idem, pp 530-537.
6. Idem, Multiple constrained folding of programmable logic arrays: Theory and applications, IEEE Trans. on Comp. Aided Des., vol. CAD-2, n. 3, jul 83, pp 151-167.
7. Egan, J.R. & Liu, C.L., Bipartite folding and partitioning of a PLA, Idem, vol. CAD-3, n. 3, jul 84, pp 191-199.
8. Idem, Optimal bipartite folding of PLAs, Proc. 19th Des. Autom. Conf., 1982, pp 141-146.
9. Chuquillanqui, S. & Segovia, T.P., PAOLA: A tool for topological optimization of large PLAs, Idem, idem, pp 300-306.

10. Hachtel, G.D. et alii, Techniques for programmable logic array folding, Idem, idem, pp 147-155.
11. Grass, W., A depth-first branch-and-bound algorithm for optimal PLA folding, Idem, idem, pp 133-140.
12. Lewandowski, J.L. & Liu, C.L., A branch and bound algorithm for optimal PLA folding, Proc. 21st Des. Autom. Conf., 1984, pp 426-433.
13. Kuo, Y.S. et alii, A heuristic algorithm for PLA block folding, Proc. 22nd Des. Autom. Conf., 1985, pp 744-747.
14. Micali, S. & Vazirani, V.V., An $O(|V|^{1/2} |E|)$ algorithm for finding maximum matching in general graphs, Proc. 21st Ann. Symp. Found. of Comp. Sci., IEEE(80), pp 17-27.
15. Ferreira, A.G., An optimal $O(n^2)$ -algorithm to fold special PLAs, Optimization Days 1986: Program and Abstracts, mai 86, pp 23-24.
16. Idem, O problema do dobramento optimal de PLAs: Uma nova abordagem e seu algoritmo, Anais do XIII SEMISH do VI Congr. Bras. de Comp., 1986, no prelo.
17. Booth, K.S. & Lueker, G.S., Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms, J. Comput. Syst. Sci., 13, pp 335-379.
18. Tucker, A.C., A structure theorem for the consecutive 1s property, J. Combin. Th. 12(B), pp 153-162.
19. Golumbic, M.C., Algorithm graph theory and perfect graphs, Academic Press, 1980.
20. Harary, F., Graph theory, Addison-Wesley Publishing Company, Inc, 2. Ed., 1971.
21. Papadimitriou, C.H. & Steiglitz, K., Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, Inc, 1982.

22. Mead, C. & Conway, L., Introduction to VLSI systems, Addison-Wesley Publishing Co., 1980.
23. Edmonds, J., Paths, trees and flowers, Canad. J. Math., 17 (1965), pp 449-467.
24. Ferreira, A.G., Algoritmos heurísticos para a otimização de PLAs, Anais do V Simp. Bras. de Microeletrônica, 1985, pp 511-520.