

COMPACTAÇÃO DE CIRCUITOS INTEGRADOS :
COM APLICAÇÃO DE RELAXAÇÃO LAGRANGEANA

Paulo César Parga Rodrigues

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO

Aprovada por :



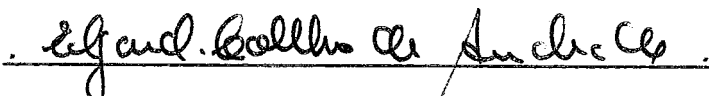
Prof. Nelson Maculan Filho (D.Sc.)
(Presidente)



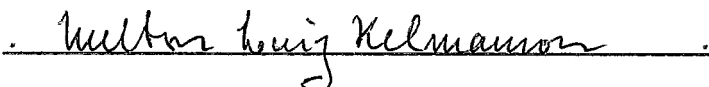
Prof. Eber A. Schmita (Ph.D.)
(co-orientador)



Prof. Edil S. T. Fernandes (Ph.D.)



Prof. Edgard C. do Andrade (D.Sc.)



Prof. Milton L. Kelmanson (Ph.D.)

RIO DE JANEIRO , RJ - BRASIL

NOVEMBRO DE 1988

RODRIGUES , PAULO CÉSAR PARGA

Compactação de Circuitos Integrados :
com Aplicação de Relaxação Lagrangeana ,
(Rio de Janeiro) 1988 .

X , 137 P. 29.7 cm (COPPE/UFRJ , D.Sc. ,
Engenharia de Sistemas e Computação ,
1988) .

Tese - Universidade Federal do Rio de
Janeiro , COPPE .

I. Programação Inteira I. COPPE/UFRJ

II. Título (série)

Agradecimentos :

- Ao professor Nelson Maculan Filho pelo apoio .
- Ao professor Eber Schmitz pelo incentivo .
- Ao departamento de matemática da PUC - RJ pelo use de seus equipamentos .
- A CAPES por bolsa concedida .
- Ao CNPq por bolsa concedida .

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências (D. Sc.) .

Compactação de Circuitos Integrados :
com Aplicação de Relaxação Lagrangeana .

Paulo César Parga Rodrigues

NOVEMBRO de 1988

Orientador : Nelson Maculan Filho

Programa : Engenharia de Sistemas e Computação

Compactar o leiaute de um circuito integrado é minimizar a área do menor retângulo circunscrito ao desenho do circuito, obedecendo regras tecnológicas pré-estabelecidas. O leiaute do circuito integrado é inicialmente representado por um diagrama simbólico , denominado diagrama de varetas.

Como modelo para o problema de compactação são utilizados três grafos . Um destes grafos é usado na direção horizontal e o segundo na direção vertical de modelo . Cada um destes dois grafos uni-direcionais é resolvido pelo algoritmo do caminho mais longo (ACML) . Os dois grafos uni-direcionais são ligados pelo terceiro grafo , que também pode ser visto como um problema linear com variáveis inteiras e 0-1 .

O problema de compactar um circuito bi-dimensionalmente é NP-completo. Com objetivo de contornar a complexidade

do problema , é feita uma busca em profundidade no terceiro grafo . Esta busca em profundidade leva a um ótimo local . Empregando-se uma pós-otimização nos grafos uni-direcionais o ótimo local é melhorado . Além disso , as folgas das variáveis são usadas para obter uma solução inicial melhor .

A aplicação de relaxação lagrangeana nesse problema permitiu avaliar com mais eficiência a solução encontrada . A maneira como o problema foi equacionado e a modificação que foi feita nessa forma de equacioná-lo deram bons resultados . A escolha da função lagrangeana não só fez com que o problema ficasse simples de ser resolvido , como também melhorou sensivelmente o tempo de computação .

Com a aplicação de uma nova regra no passo do método do subgradiente, melhorou-se a solução e a convergência do método .

Finalmente foi feita uma comparação entre otimizar a área ou o semi-perímetro do leiaute do circuito integrada . Esta comparação nos problemas que foram resolvidos, mostrou que a utilização da área ou do semi-perímetro produziu uma diferença insignificante no resultado .

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Doctor of Science (D. Sc.) .

Compaction of Integrated Circuits :
with Application of Lagrangean Relaxation .

Paulo César Parga Rodrigues

NOVEMBER of 1988

Thesis Supervisor : Nelson Maculan Filho

Department : Computing and Systems Engineering

To compact the layout of an integrated circuit is to minimize the area of the smallest circumscribed rectangle to the circuit , observing pre-established technological rules . Initially a symbolic diagram is used to represent the layout of the integrated circuit . This representation is named stick diagram .

Three graphs are used as model to the compactification problem . One of this graphs is used in the horizontal direction and another in the vertical direction of the model . Each one of this one-directional graphs is solved by the longest path algorithm . This two one-directional graphs are joined by the third graph , which can be viewed also as an integer linear problem with 0-1 variables .

The problem to compact a bi-dimensional circuit is NP-complete . A deep search is done to avoid the complexity of the problem. This search produce a local optimum , which is improved by using a pos-optimization in the one-directional graphs. The variables` slack are used ta obtain a better initial solution .

The application of lagrangean relaxation method in this problem permits a more efficient estimation of the obtained solution . Good results were obtained by the way the problem was set-up and later msdified . The choice of the lagrangean function not only transformed the problem into a much simpler one but also shortened considerably the computational time .

With the use of a new rule in each stage of the sub-gradient method the solution waa impsaved and the computational time shortened .

Finally , it is investigated the difference in optimizing area and semi-perimeter . This comparison shows that at least in the problem solved, the difference is very small .

ÍNDICE

| | | | |
|-------------|---|---|----|
| CAPÍTULO I | - | <u>Introdução</u> | 1 |
| I.1 | - | Os Circuitos Integrados | 1 |
| I.2 | - | O Projeto de um Circuito Integrado . | 2 |
| I.3 | - | Diagramas de Varetas e o Problema de Compactação | 7 |
| I.4 | - | Trabalhos Realizados na Área de Compactadores | 12 |
| I.5 | - | Plano da Tese | 13 |
| CAPÍTULO II | - | <u>Método de Compactação</u> | 15 |
| II.1 | - | Introdução | 15 |
| II.2 | - | Modelo | 15 |
| II.3 | - | O Problema | 23 |
| II.4 | - | Gerando as Restrições | 25 |
| II.5 | - | Algoritmo do Caminho Mais Longo (ACML) | 26 |
| II.5.1 | - | A Arvore Geradora Máxima | 28 |
| II.5.2 | - | As Folgas dos Nós | 32 |
| II.5.3 | - | Uma Pós-Otimização no Grafo | 35 |
| II.6 | - | Algoritmo de Compactação | 36 |
| II.6.1 | - | Buscando um ótimo Local | 39 |
| II.6.2 | - | Gerando uma Solução Inicial e Obtendo Um Limite Inferior | 42 |
| II.6.2.1 | - | Uma Solução Inicial Rápida | 42 |
| II.6.2.2 | - | Uma Solução Inicial Usando as Folgas | 46 |
| II.7 | - | Conclusão | 51 |

| | | | |
|--------------|---|---|-----|
| CAPITULO III | - | <u>Relaxação Lagrangeana</u> | 52 |
| III.1 | - | Introdução | 52 |
| III.2 | - | Adaptação do Problema | 52 |
| III.3 | - | Um Pouco de Teoria | 55 |
| III.4 | - | O Método do Sub-Gradiente | 64 |
| III.5 | - | O Método do Sub-Gradiente no Problema de Compactação | 70 |
| III.6 | - | O Método de Compactação com M Variável | 73 |
| III.7 | - | Área ou Semi-Perímetro ? | 75 |
| CAPITULO IV | - | <u>Experiência Computacional</u> | 80 |
| IV.1 | - | Introdução | 80 |
| IV.2 | - | Descrição das Rotinas Empregadas | 81 |
| IV.3 | - | Obtendo um ótimo Local | 87 |
| IV.3.1 | - | Resultado Sem as Folgas | 89 |
| IV.3.2 | - | Resultado Usando as Folgas | 91 |
| IV.4 | - | Aplicando Relaxação Lagrangeana | 95 |
| IV.4.1 | - | A Cota Inferior com Um único R | 98 |
| IV.4.2 | - | A Cota Inferior com o Valor de M Variável | 103 |
| IV.4.2.1 | - | Cálculo dos Valores de M_x e de M_y | 103 |
| IV.4.2.2 | - | Resultados Com M Variável | 104 |
| IV.4.2.3 | - | Adotando Outra Regra | 108 |
| IV.5 | - | Resolvendo Problemas Reais | 112 |
| CAPITULO V | - | <u>Conclusão</u> | 119 |

| | |
|--|-----|
| REFERÊNCIAS BIBLIOGRÁFICAS | 123 |
| APÊNDICE | 128 |
| A.1 - Arquivos para Entrada de Dados ... | 128 |
| A.2 - Arquivo Gerado | 129 |
| A.2.1 - Método de Geração das Restrições | |
| Tipo "e" Horizontais | 130 |
| A.2.2 - Método de Geração das Restrições | |
| Tipo "e" Verticais | 133 |
| A.2.3 - Método de Geração das Restrições | |
| Tipo "ou" | 133 |

CAPITULO IINTRODUÇÃO

1.1 Os Circuitos Integrados

a) O que são e as formas de classificá-los :

Os circuitos integrados são dispositivos eletrônicos contendo milhares de componentes (transistores , capacitores³ dentro de uma única cápsula . Estes dispositivos fazem as funções de processadores , memórias , etc .

Os circuitos integrados podem ser classificados quanto ao número de componentes colocados na cápsula (densidade de integração) e quanto ao tipo de transistor .

A seguir colocamos a classificação quanto à densidade de integração :

| | | | | | |
|------|---------------------------------|-------------------|--------|---|--------|
| LSI | "Large-Scale-Integration" | contem entre | 10^3 | e | 10^4 |
| | | transistores , | | | |
| VLSI | "Very-Large-Scale-Integration" | contém entre | 10^6 | e | 10^8 |
| | | 10 transistores , | | | |
| ULSI | "Ultra-Large-Scale-Integration" | contém mais de | 10^6 | | |
| | | transistores . | | | |

Os circuitos integrados são classificados quanto ao tipo de transistor da seguinte maneira :

BIPOLAR (por exemplo TTE , ECL , etc .)

MOS "Metal-Oxide-Semiconductor" (por exemplo NMOS ,
CMOS)

b) Como são feitos :

Um circuito integrado é um conjunto de camadas , constituídas de diversos materiais. Essas camadas são separadas por camadas de isolantes . As camadas são colocadas sobre uma lâmina de silício , denominada substrato .

Para criar um circuito integrado deve-se repetir, para cada uma das camadas , os denominados passos de processo . Um passo de processo consiste de :

1. Depositar o material da camada .
2. Fazer a gravação do material da camada usando uma máscara . (Este processo é semelhante ao que é usado fotografias)

Assim para fazer um circuito integrado **devemos obter** as máscaras . Para cada uma das camadas temos uma máscara .

1.2 O Projeto de um Circuito Integrado

a) A Fase de Projeto

Quando pretendemos fabricar um circuito integrado , devemos projetar as máscaras das camadas envolvidas no pro-

cesso de fabricação .

O desenho de cada máscara consiste de um conjunto de retângulos com os lados paralelos entre si . As dimensões desses retângulos dependem de parâmetros, que são definidos pelo processo de fabricação .

Para projetar um circuito integrado devemos seguir um processo . Este processo inicia com a especificação da função a ser executada pelo circuito e termina com o desenho em escala de todas as máscaras necessárias para as camadas. A seguir descrevemos as etapas intermediárias do processo :

- Definir os blocos componentes (quais funções serão executadas por cada bloco) .
- Definir a rede de transistores de cada bloco .
- Desenhar as máscaras de cada bloco (leiaute a nível de blocos) .
- Fazer a composição final (montagem) .

b) A etapa do leiaute

Para gerar o leiaute final de um bloco , parte-se da rede de transistores .

A seguir colocamos as técnicas que podem ser usadas para fazer o leiaute de um bloco lógico. O projetista pode:

1. Fazer o desenho na tela de um computador , usando um editor gráfico .
2. Usar uma biblioteca de células , no caso de ter uma a sua disposição . Neste caso o problema consiste em

fazer as interligações das células .

3. Utilizar pacotes de CAD ("Computer-Aided-Design") que fazem a geração automática do leiaute de um bloco.
4. Através de um desenho simbólico , denominado diagrama de varetas ("Sticks") , fazer uma representação do leiaute do bloco . Com essa representação simbólica usar um algoritmo de compactação para gerar o leiaute final .

c) Representação do circuito por suas máscaras

A representação final de um circuito é feita por suas máscaras . A representação por máscaras , consiste em desenhar os retângulos dos diversos materiais. Estes retângulos estão dispostos de forma que seus lados são verticais ou horizontais, não se permitindo retângulos inclinados. Assim estamos empregando a mesma tecnologia que foi usada em MEAD 8 COMWAY C181 . Na figura 1.1 , colocamos a forma de representar os diversos materiais para circuitos construídos na tecnologia NMOS .

Quando dois retângulos de difusão e polisilício se cruzam , eles formam um transistor . Um exemplo deste tipo de elemento do circuito esta desenhado na figura 1.2 .

Por outro lado , quando retângulos de metal e de polisilício (ou de difusão) se sobrepõem , não há interferência entre eles. Para que haja interferência é necessário um contato . O contato consiste de um furo na camada de isolante . Na figura 1.3 pode ser vista um exemplo de contato entre um retângulo de difusão e um de metal .

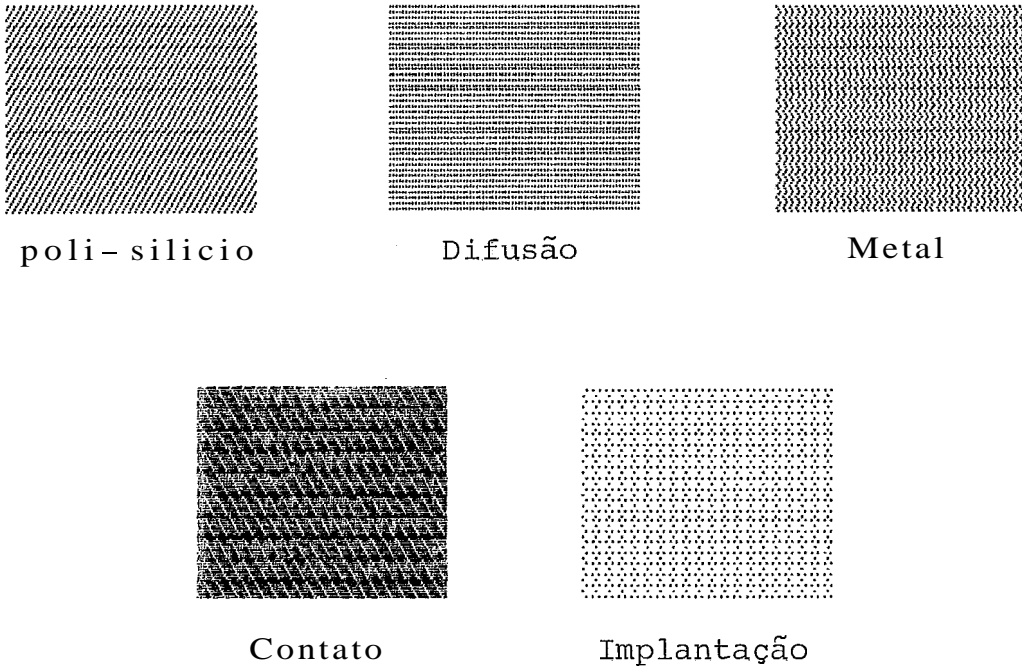


FIGURA I.1 : Representação das camadas na tecnologia N-MOS

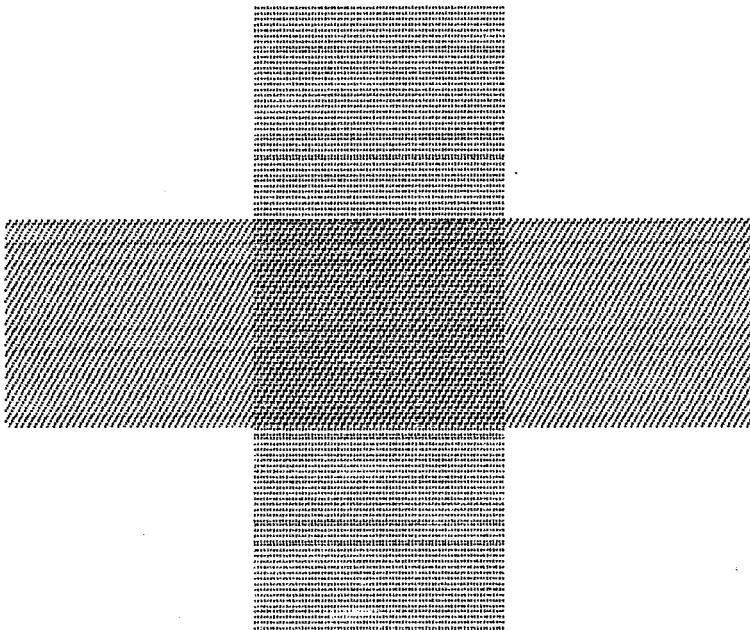


FIGURA I.2 : Transistor

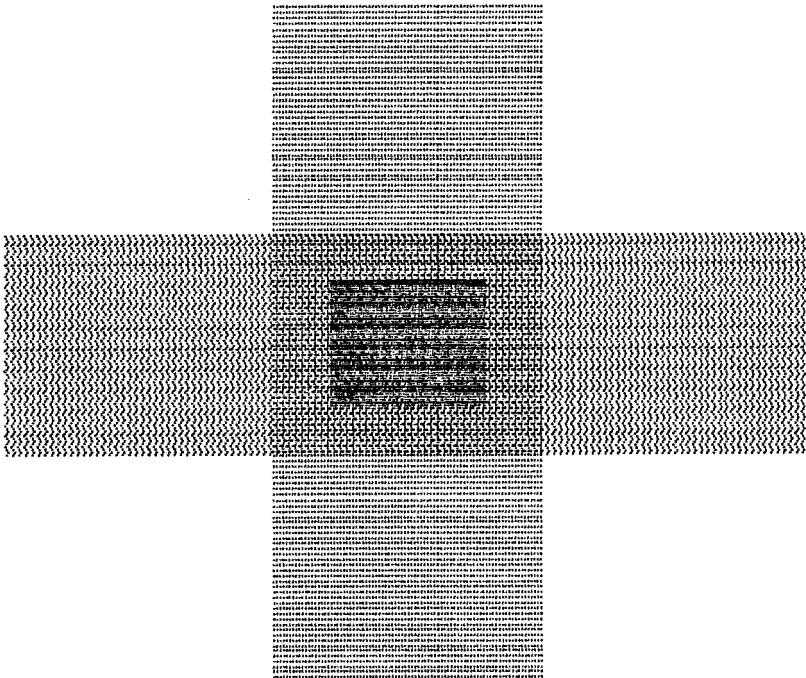


FIGURA 1.3 : Contato

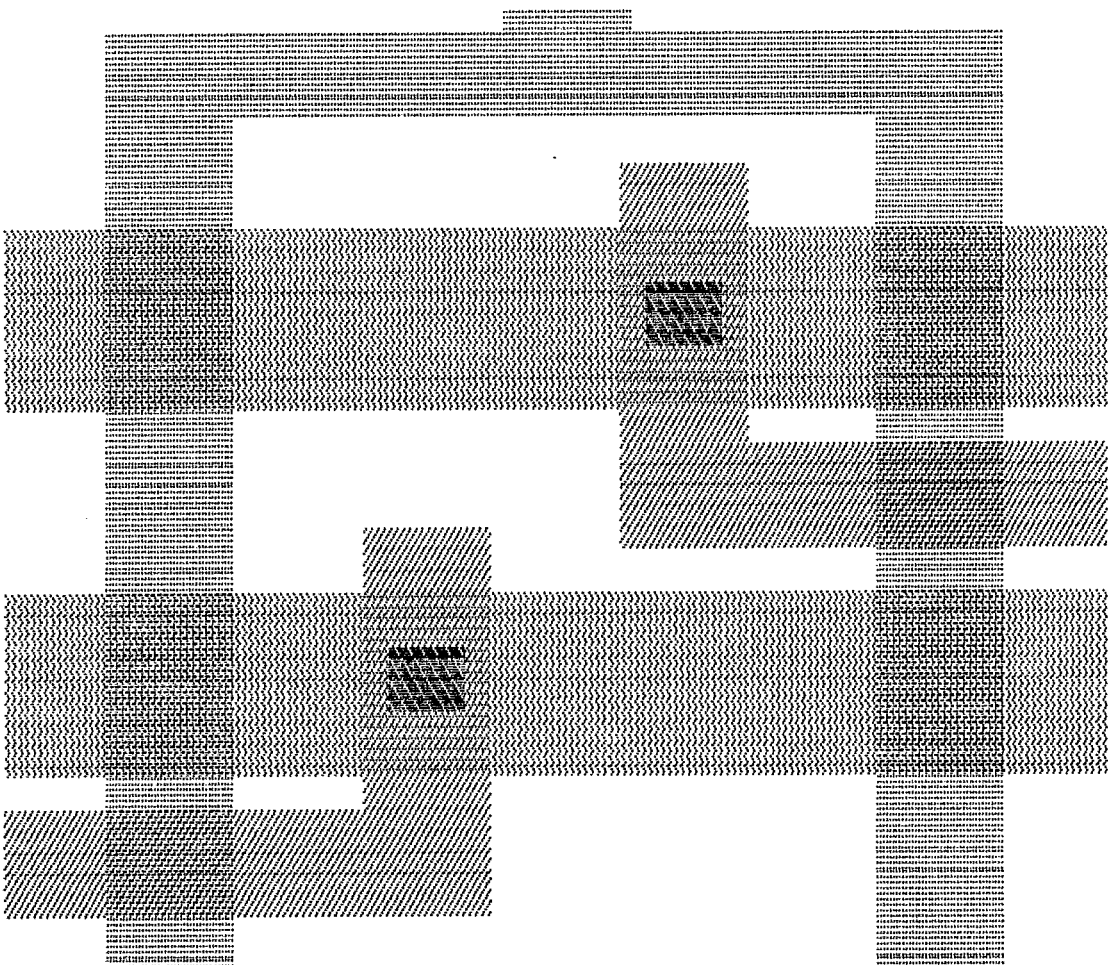


FIGURA 1.4 : Multiplexador

Na figura 1.4 temos um desenho de um circuito , representado por suas máscaras .

As figuras 1.1 , 1.2 , 1.3 e 1.4 foram feitas com o pacote TEBMOS desenvolvido no Núcleo de Computação Eletrônica da Universidade Federal do Rio de Janeiro .

Cada um dos elementos do circuito tem uma largura , e entre eles existe uma distância mínima . Considerando que a unidade em questão é d , por exemplo a largura de um fio de metal é no mínimo $3d$. A distância mínima entre um fio de polisilício e de difusão deve ser $2d$, ou a distância mínima entre dois fios de metal, cujos potenciais elétricos são diferentes deve ser $3d$, etc... . Estas larguras e distâncias dependem da tecnologia empregada . A tabela para circuitos NMOS pode ser encontrada no livro de MEAB & CONWAY [18] .

1.3 Diagramas de Varetas e o Problema de Compactação

a) Simplificando o projeto .

Para poder dimensionar os retângulos , de cada uma das máscaras , o projetista deve levar em conta dois tipos de regras de projeto :

- Elétricas ; Para definir o tamanho dos retângulos em função de características elétricas, tais como corrente , capacitância , etc .
- Geométricas ; Indicam qual a largura mínima que um elemento pode ter , ou qual a distância

mínima entre dois elementos (da mesma camada ou não) .

Se o leiaute fôr feito manualmente pelo projetista , o problema dele consiste em posicionar os retângulos de forma que os blocos executem as funções desejadas . Cabe então colocar os elementos numa posição tal que :

- As regras de projeto sejam obedecidas .
- A área total seja mínima .

Uma alternativa para o leiaute manual é utilizar um algoritmo que , a partir de um esboço do circuito faça o leiaute final . Isso permitiria ao projetista explorar as diversas possibilidades de colocação dos retângulos , e com isso descobrir (se possível) qual a posição que daria a área mínima .

Desta forma , a rotina de projeto um bloco fica muito simplificada . Mostramos a seguir como fica a rotina de projeto usando diagramas de varetas :

1. Define-se padrões de cores ou de linhas para os elementos de cada máscara .
2. Abstrai-se a informação de largura e de comprimento dos retângulos (estas informações podem ser obtidas das regras de projeto) . Um retângulo passa a ser um segmento de reta de uma dada cor ou um ponto , representando sua camada .
3. O projetista faz um desenho, que é um esboço do leiaute (denominado diagrama de varetas) . Neste desenho

se representa apenas a topologia do circuito C sem métrica) .

4. Aplica-se o algoritmo de compactação :
 - Partindo das regras de projeto descobre-se o tamanho dos elementos .
 - Procura a posição dos elementos de forma a minimizar a área total .
5. Gera-se o leiaute final usando uma rotina simples .

b) O diagrama de varetas

O diagrama de varetas ("Sticks") é uma representação simbólica do circuito integrado . É uma representação por linhas , sem levar em conta a largura e o comprimento dos elementos do circuito . O diagrama de varetas é um esboço do posicionamento relativo dos elementos do circuito . Esta representação foi criada por WILLIAMX C303 , que desenvolveu o compactador Sticks .

Este diagrama usa linhas para representar os diversos fios do circuito e pontos para os contatos. A representação dos fios tem linhas diferentes para as diversas camadas , como podemos ver na figura 1.5 .

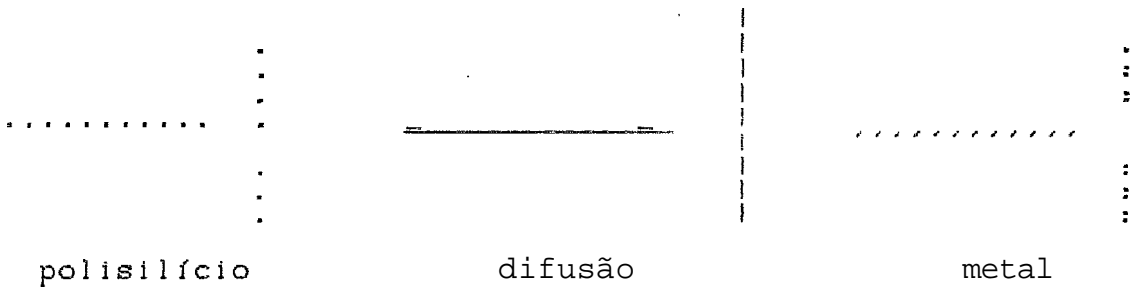


FIGURA 1.5 : Representação das Camadas
no Diagrama de Varetas

Como não levamos em conta, as larguras e as distâncias entre os elementos, guardamos estes dados em um arquivo. Na literatura, como em ULLMAM [28] e em MEAD & CONWAY [18] este arquivo é guardado como uma linguagem CIF (que é a abreviação de "Caltech Intermediate Form"). Esta linguagem serve para armazenar os dados tecnológicos e também para fornecer dados para a fabricação automática do circuito. Para dar uma ilustração melhor da representação tipo "stick", re-desenhemos o circuito da figura 1.4, usando esta representação na figura 1.6.

Na figura 1.6, denominamos por X as coordenadas horizontais dos elementos do circuito e por Y as coordenadas verticais. Essas coordenadas são as do centro de cada um dos elementos do circuito da figura 1.4. Os fios verticais têm somente coordenadas horizontais e os fios horizontais somente as coordenadas verticais. No capítulo II daremos mais detalhes sobre essas coordenadas, lá elas são usadas para modelar o problema.

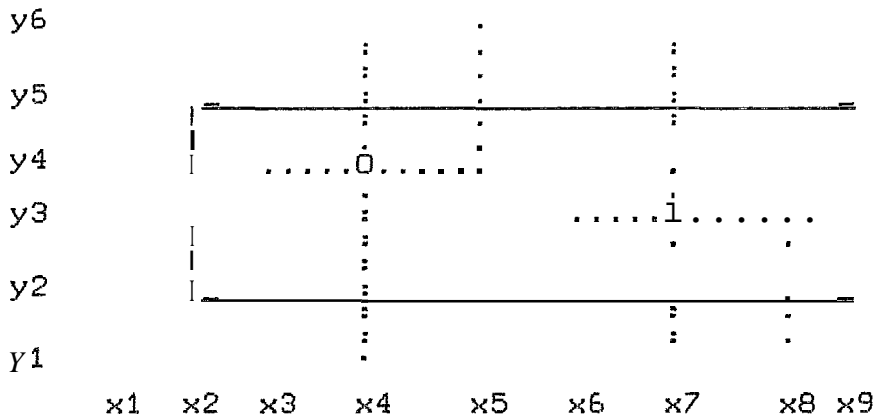


FIGURA 1.6 : Representação de um Multiplexador Usando Diagrama de Varetas

Para melhor ilustrar a figura 1.6 , x_1 é a coordenada horizontal do ponto mais a esquerda e x_9 a do ponto mais a direita . Outro exemplo de coordenada horizontal é x_6 , que fornece a posição do extremo esquerda do fio horizontal de polisilício que tem coordenada vertical y_3 . As coordenadas verticais têm um ponto extremo inferior y_1 e um superior y_6 .

c) Compactação de grupos de células previamente desenhadas

Alguns compactadores usam células para representar o circuito , como por exemplo WILLIAMS [30] e SCWLAG [25]. Devido à grande complexidade do desenho ele é decomposto em células , que são circuitos pré-desenhados . Estas células funcionam como caixas pretas e são representadas por retângulos . Esses retângulos são olhados como elementos do circuito. A largura e a distância mínima entre os elementos são fornecidos em um arquivo de dados .

Na caso do algoritmo que apresentaremos nos outros capítulos há diferença entre usarmos este tipo de representação ou a representação simbólica. O que nos interessa como dados, para fazer a compactação, são as coordenadas dos elementos do circuito. No capítulo II fazemos um comentário sobre as coordenadas e no apêndice apresentamos um algoritmo que partindo de uma representação simbólica ou por células obtemos as coordenadas.

d) A Compactação do layout de Circuitos Integrados

A topologia inicial de um circuito integrado, como já dissemos na seção 1.1, é feita quando são resolvidos os problemas de disposição e ligação. Compreendemos como topologia do circuito a posição relativa entre seus elementos, sem levar em conta as larguras e as distâncias entre os elementos. Compactar um circuito é partir de um desenho inicial e reduzir sua área, obtendo um desenho que respeite as regras pré-estabelecidas e que não altere sua topologia inicial. A invariância da topologia se faz necessária, para manter o circuito que foi previamente desenhado usando-se os problemas de disposição e ligação. Permite-se então que os fios sejam esticados ou encolhidos, e que os elementos mudem de posição, sem alterar suas posições relativas.

1.4 Trabalhos Realizados na Área de Compactadores

Um quadro comparativo contendo os diversos algoritmos sobre compactação de circuitos integrados, pode ser encontrado no artigo de MLYNSKI & SUNG [19], onde foi feito um pequeno "survey" sobre compactadores.

A maioria dos algoritmos desenvolvidos até o momento usam compactação unidirecional, ou melhor duas compactações distintas unidirecionais. Podemos ressaltar alguns autores que usam este tipo de modelo para compactar, WILLIAMS E903 e LIAO & WONG C151 dentre outros.

O maior problema com os compactadores bi-dimensionais é que, como mostraram SCWLAG, LIAO & WONG [25], o problema de compactar um circuito bi-dimensionalmente é NP-completo. Isto é, os algoritmos não são polinomiais.

Os autores KEDEB & WATAMABE C121 por exemplo, usaram um modelo bi-dimensional, e para compactar aplicaram o algoritmo "Branch-and-Bound". Quando o número de elementos do circuito é muito grande, o algoritmo "Branch-and-Bound" fica praticamente impossível de ser realizado. Outros autores também desenvolveram compactadores bi-dimensionais, tais como WOLF, MATHEWS, NEWKIRK & DUTTON C311 e também SHLAG, LIAO & WONG C253. Todos estes artigos usam variáveis 0-1 para modelar seus algoritmos.

1.5 Plano da Tese

No capítulo II apresentamos o modelo que foi usado neste trabalho, para compactar um circuito Si-dimensionalmente. Também neste capítulo, mostramos como foi empregada uma pós-otimização nas grafos para melhorar o algoritmo de compactação.

Técnicas de relaxação lagrangeana foram aplicadas ao problema, para obter uma análise mais precisa dos resulta-

dos obtidos na compactação . Esta aplicação e o desenvolvimento da teoria necessária foram feitos no capítulo III .

As experiências computacionais e suas análises estão feitas no capítulo IV . Também nesse capítulo são feitas algumas alterações no equacionamento do problema , com referência ao que foi apresentado no capítulo III .

As conclusões finais e a avaliação do trabalho estão no capítulo V .

CAPITULO IIMÉTODO DE COMPACTAÇÃO

II.1 Introdução

Neste capítulo descreveremos o método utilizado para a compactação de um circuito VLÇI . A modelagem que usaremos aqui , apresentada na seção II.2 , será a mesma que foi usada na tese do WATANABE C293 . Na seção II.3 discutiremos o problema de compactar o circuito. O problema de gerar as restrições para o problema , bem como o tipo de arquivo que é usado pelo programa de compactação está descrito na seção II.4. Na seção II.5 , obteremos uma árvore geradora máxima para o grafo, com o algoritmo do caminho mais longo (ACML). Este algoritmo será utilizado no método de compactação . Ainda na seção II.5 , faremos uma pós-otimização no grafo a partir das folgas dos nós . Finalmente , na seção II.6 , obtemos um ótimo local com o algoritmo de compactação.

II.2 Modelo

Os elementos do circuito estão dispostos no primeiro quadrante do espaço Euclidiano R^2 . Porém os elementos do circuito são desenhados somente sobre os números inteiros positivos . Isto é , consideramos que o "layout" do circuito esteja sobre o espaço $N \times N$ (produto carte-

siano \mathbb{N} , onde \mathbb{N} é o conjunto dos números naturais $\{0, 1, \dots\}$.

Os elementos do circuito são representados por retângulos de lados paralelos aos eixos coordenados, isto é, verticais e horizontais. Cada elemento i é representado pelas coordenadas de seu centro (x_i, y_i) . São denominados fios as interligações entre os elementos do circuito. Da mesma forma que os elementos esses fios só podem ser verticais ou horizontais, não se permitindo fios inclinados. Os fios verticais só terão coordenadas horizontais x_i e os fios horizontais só coordenadas verticais y_i . As coordenadas x_i ou y_i dos fios são suas posições centrais.

As regras que dão as distâncias entre dois elementos ou entre dois fios, ou ainda a largura dos elementos são se do livro de MEAD & CONWAY C183.

Considere dois elementos quaisquer i e j dispostos como na figura 11.1, cujas coordenadas são (x_i, y_i) e (x_j, y_j) , suas larguras são L_i e L_j e a distância mínima entre eles é w_{ij} . Temos que a distância mínima entre seus centros é dada por: $x_j - x_i \geq d_{ij}$, onde $d_{ij} = (L_i/2) + (L_j/2) + w_{ij}$.

Caso os elementos estejam dispostos como na figura 11.2, devemos ter restrições entre as coordenadas verticais dos dois elementos i e j . Neste caso a restrição fica: $y_j - y_i \geq d_{ij}$, onde d_{ij} é a distância mínima entre os centros dos elementos.

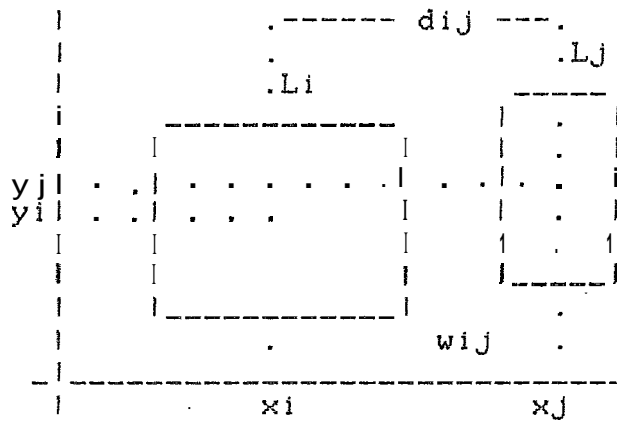


FIGURA 11.1 : Restrição horizontal

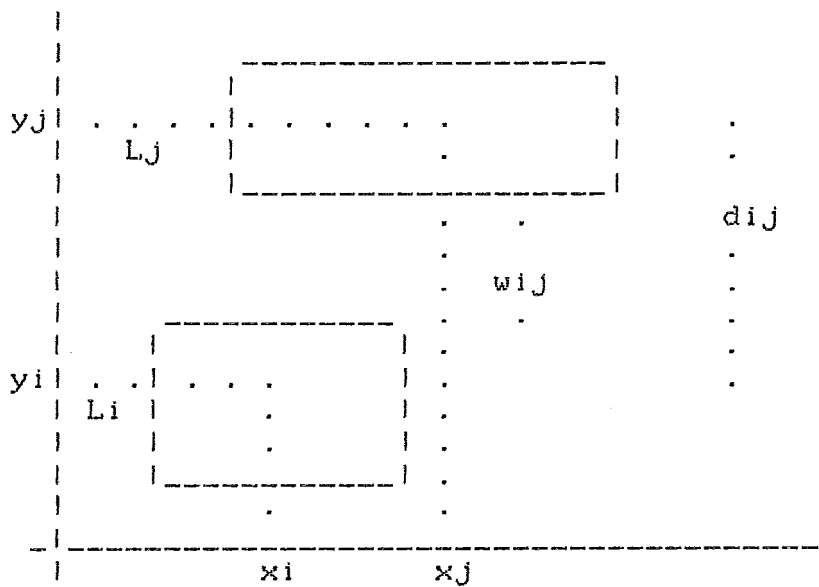


FIGURA 11.2 : Restrição vertical

Vimos até agora que temos dois tipos de restrições em X e em Y . Estes conjuntos de restrições são aparentemente independentes. Porém **esta** independência não ocorre, já que **esses** dois conjuntos de restrições podem ser ligados de **duas** maneiras.

Ao compactar um circuito os elementos A e B podem estar dispostos de duas formas, lado a lado ou um acima do outro. Neste caso aparece uma das duas restrições, uma para afastar A e B horizontalmente ou uma outra para afastá-los verticalmente (veja figura II.3).

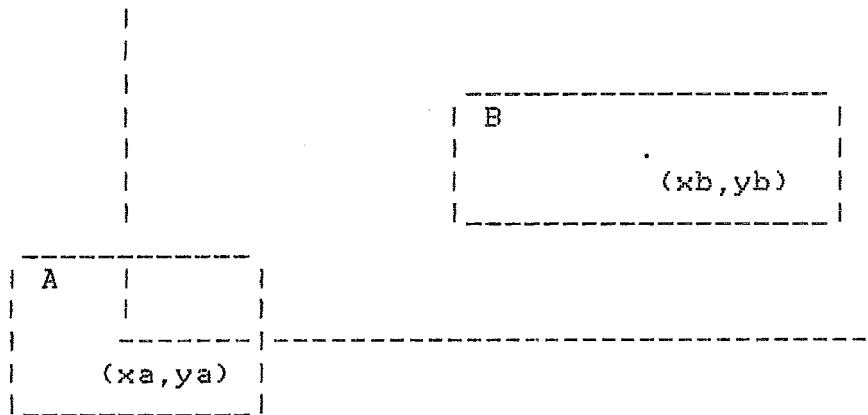


FIGURA II.3 : Posição do centro do elemento A relativamente ao elemento B

Neste primeiro caso as restrições ficam da seguinte forma :

$$x_b - x_a \geq d_{ab} \quad \text{ou} \quad y_b - y_a \geq d_{ab} ,$$

fazendo uma ligação entre as restrições em X e Y .

O segundo caso se dá entre três elementos A, B e C (ou mais) como mostrado na figura II.4. Dependendo da posição vertical que o elemento C se encontre, devemos considerar o afastamento mínimo de A ou de B. Este caso é considerado quando há um grupo de elementos verticais.

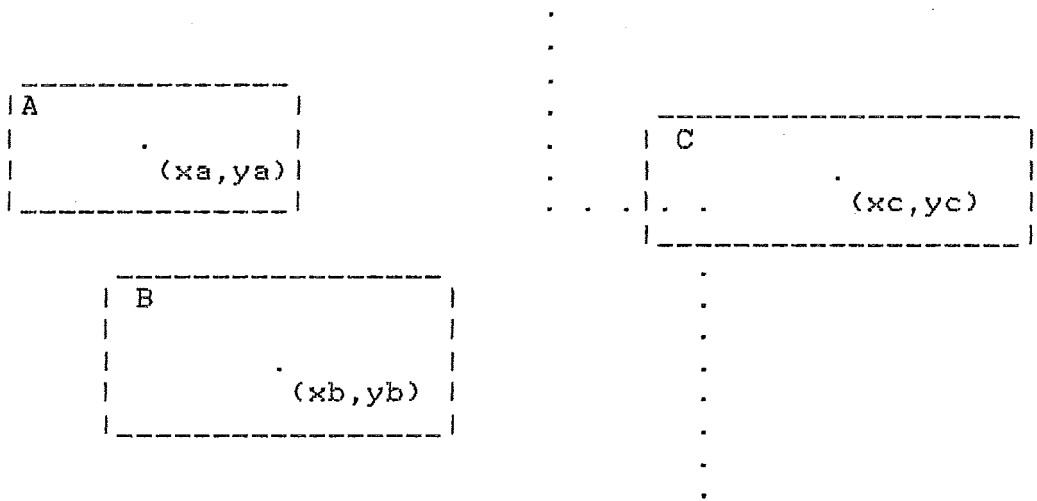


FIGURA II.4 : Posição do centro do elemento C em relação aos elementos A e B

Neste segundo caso , as restrições ficam da seguinte forma :

se $y_c - y_b \geq \bar{d}_{bc}$ então $x_c - x_a \geq d_{ac}$

senão $y_c - y_b \leq \bar{d}_{bc}$ e então $x_c - x_b \geq d_{bc}$

onde \bar{d}_{bc} , d_{bc} e d_{ac} são os respectivos afastamentos mínimos entre B e C e entre A e C (a barra é para diferenciar o afastamento vertical do horizontal) .

O mesmo tipo de restrição aparece , quando há grupos horizontais .

Porém para que o algoritmo não fique demasiadamente complicado pode ser usado apenas o primeiro caso . Quando aparecerem grupos verticais ou horizontais, o distanciamento mínimo entre o grupo e um elemento fora do grupo , pode ser considerado como o maior distanciamento dentre as

elementos do grupo e o elemento de fora . Perdemos um pouco da compactação mas ganhamos em tempo de computação , como poderá ser visto posteriormente . O esboço da região onde pode ficar o centro de elemento C , quando abandonamos as restrições deste tipo, está feito na figura 11.5 . A região da figura 11.5 é menor que a da figura 11.4 .

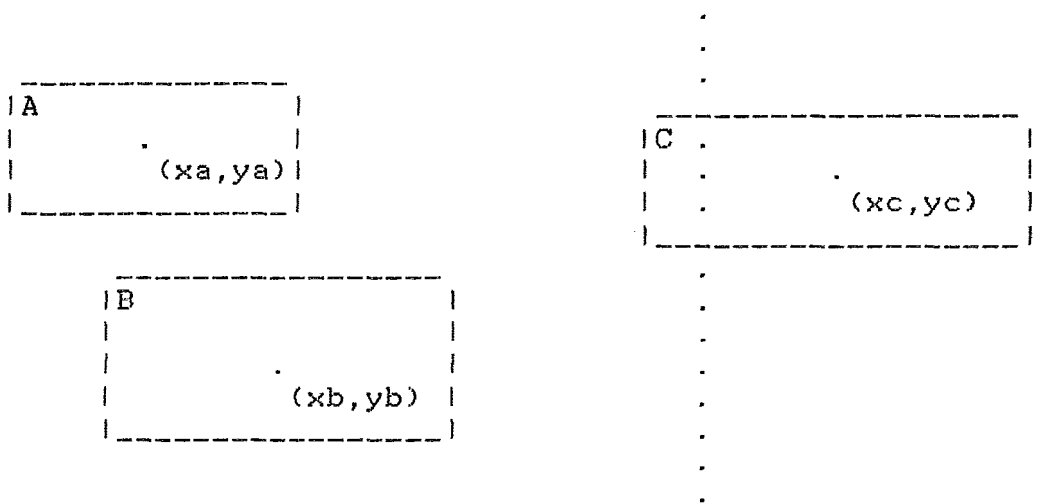


FIGURA 11.5 : Posição do centro de C ao usar apenas o afastamento horizontal de B

Se o projetista achar que há um ganho razoável no desenho do circuito , poderá substituir o segundo caso por duas ou mais restrições (como no exemplo acima) do primeiro caso . As duas restrições são

$$(y_c - y_b \geq \bar{d}_{bc} \quad \text{ou} \quad x_c - x_b \geq d_{bc}) \quad \text{e}$$

$$(y_c - y_a \geq \bar{d}_{ac} \quad \text{ou} \quad x_c - x_a \geq d_{ac})$$

que equivale a alterar a região possível para o centro de C para a região da figura 11.6 . É importante que as po-

sições relativas de A e de B estejam como na figura 11.6 . Isto é , A e B devem estar de tal forma que $x_b + d_{bc} \geq x_a + d_{ac}$, Caso isto não ocorra , perdemos espaço para a posição relativa para o centro de C , se usamos este tipo de substituição . E neste caso o desenho da região fica como na figura 11.7 . Podemos ver que esta região é equivalente a da figura 11.5 .

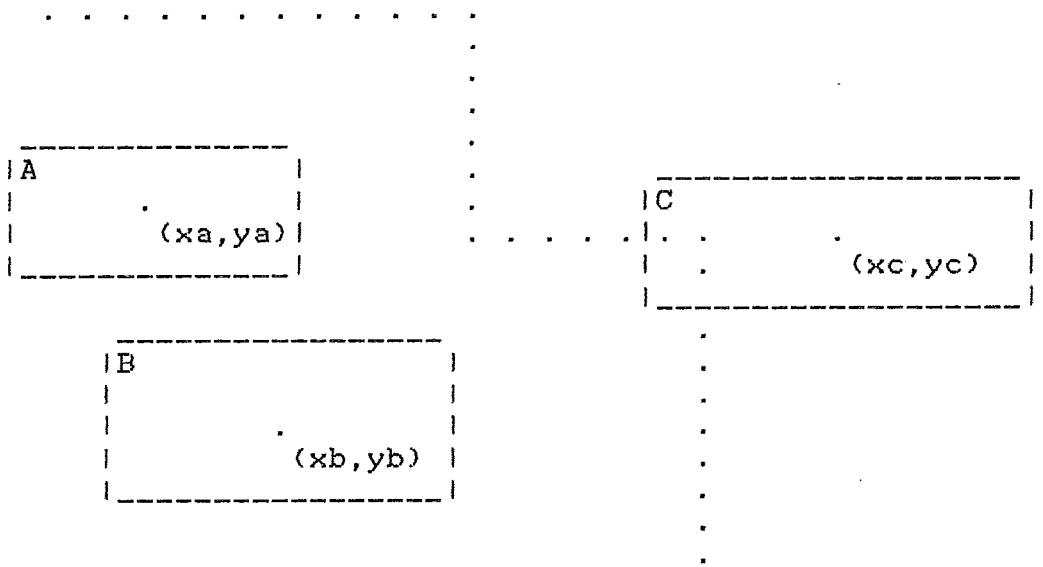


FIGURA 11.6 : Posição do **centro** do elemento C ao considerar os afastamentos de A e B verticais e horizontais

Podemos pensar que o projetista estaria desenhando o leiãute do circuito integrado sobre um reticulado no plano . Ou melhor , é como se ele estivesse colorindo os elementos A_{ij} de uma matriz A . Estas cores representam os tipos de elementos do circuito . As cores podem ser substituídas por números . Por exemplo o número 0 significa que não há elementos naquela posição . O elemento

$A(3,9)$ da matriz ser igual a 1 (representado em um desenho pela cor azul), diz que na posição $X = 3$ e $Y = 9$ devemos ter , por exemplo , o centro de um retângulo que pertence a camada de meta! , etc... .

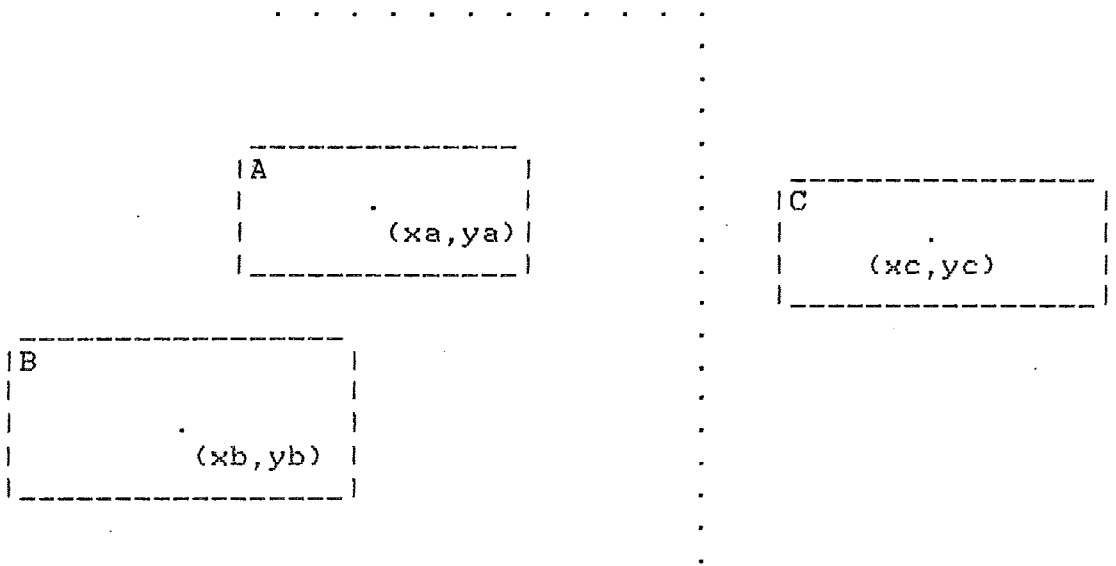


FIGURA 11.7 : Posição do centro do elemento C com o elemento B à esquerda do do elemento A

Ao desenharmos o circuito consideramos dois elementos artificiais . O primeiro mais à direita de todas os outros, com coordenada horizontal x_nx e um outro mais alto , com coordenada vertical y_ny . Além disso, não colocamos nenhum elemento nas coordenadas x_1 e y_1 . Estas duas coordenadas são reservadas para os eixos X e Y , respectivamente , e a elas é atribuído a valor zero . Estes elementos são os delimitadores do leiaute . O circuito deve estar inscrito em um retângulo , cujos lados estão sobre os eixos coordenados e sobre as retas $X = x_nx$ e $Y = y_ny$.

11.3 O Problema

O problema a resolver é minimizar a área do leiaute , ou seja :

Minimizar a área do retângulo acima (descrito no final da seção anterior) que é $x_n x_n^* y_n y_n$, sujeito às restrições tecnológicas (comprimento , distancia entre os elementos , etc ...) .

OTTEN [21] sugere minimizar o semi-perímetro que é $x_n x_n + y_n y_n$ no lugar ds área , para que o problema fique linear . Na seção III.7 discutimos a questão de usar a área ou o semi-perímetro e suas implicações.

Como vimos na seção anterior existem dois tipos de restrição "ou" e "e" . Se considerarmos apenas as restrições do tipo "e" e aceitarmos a sugestão de OTTEN 6213 , ficamos com um problema de programação linear :

$$\begin{aligned} & \text{minimizar } x_n x_n + y_n y_n \\ & \text{sujeito a } A_1 X \geq b_1 , A_2 Y \geq h_2 , X \geq 0 \text{ e } Y \geq 0 , \end{aligned}$$

onde A_1 e A_2 são matrizes $m_x \times n_x$ e $m_y \times n_y$, respectivamente , b_1 matriz $m_x \times 1$, b_2 matriz $m_y \times 1$ e X e Y são as variáveis do problema com n_x e n_y coordenadas .

Este problema pode ser resolvido , por exemplo pelo método Simplex . Porém as matrizes A_1 e A_2 são totalmente unimodulares . As linhas destas matrizes tem todos os elementos iguais a zero exceto dois deles que são 1 e -1.

Neste caso , fica mais fácil usar o algoritmo ACHE , " Algoritmo do Caminho Mais Longo " (seção 11.5) , do que usar o método Simplex . Isto se deve ao fato de que no algoritmo ACML necessitamos apenas comparar e fazer m somas , onde m é o número de restrições , enquanto que no método Simplex temos que inverter pelo menos uma matriz $m \times m$. A solução ótima é encontrada facilmente, e como os elementos das matrizes b_1 e b_2 são números inteiros , a solução ótima também será inteira . Ainda mais , quando não usamos as restrições tipo "ou" isto é , quando fixamos estas restrições em X ou em Y , o problema fica separável em dois problemas independentes , um em X e outro em Y que são :

$$\begin{aligned} & \text{minimizar } x_n x \\ & \text{sujeito a } A_1 X \geq b_1 \text{ e } X \geq 0 \end{aligned}$$

$$\begin{aligned} & \text{minimizar } y_n y \\ & \text{sujeito a } A_2 Y \geq b_2 \text{ e } Y \geq 0 \end{aligned}$$

O problema fica difícil de ser resolvido quando são consideradas as restrições do tipo "ou" . Cada uma destas restrições nos diz que devemos considerar uma restrição do tipo " e^p " em X ou uma em Y . Podemos assim criar uma variável 0-1 , que nos informa se estamos usando a restrição em X (se o valor desta variável é zero) ou em Y (e neste caso é igual a um) . Suponhamos que o número de tais variáveis é p . Para cada uma delas temos duas possi-

bilidades de escolha . Logo o número total de problemas somente com restrições do tipo "e" é 2^p ,

11.4 Gerando as Restrições

Quando desenhamos o circuito aparece o problema de gerar as restrições tecnológicas das distâncias mínimas entre seus elementos e das larguras destes . Estas restrições estão descritas na seção 11.2 . Temos em primeiro lugar restrições horizontais (em X) e depois as verticais (em Y) . As restrições em X devem estar ordenadas segundo os índices das variáveis x_i . As restrições em Y também devem ficar ordenadas da mesma forma .

Devem ser fornecidos dados sobre a tecnologia empregada, o esboço do layout e suas "esquinas" . As "esquinas" são as restrições do tipo "ou" que discutimos na seção 11.2 . Dois tipos de informações são necessárias para definir estas "esquinas" : Onde estão localizadas inicialmente e de que tipo são .

As "esquinas" podem ser de quatro tipos :

1 nordeste (NE) , 2 noroeste (NW) , 3 sudoeste (SW) e 4 sudeste (SE) . Seguem o sentido trigonométrico como podemos ver na figura 11.8 .

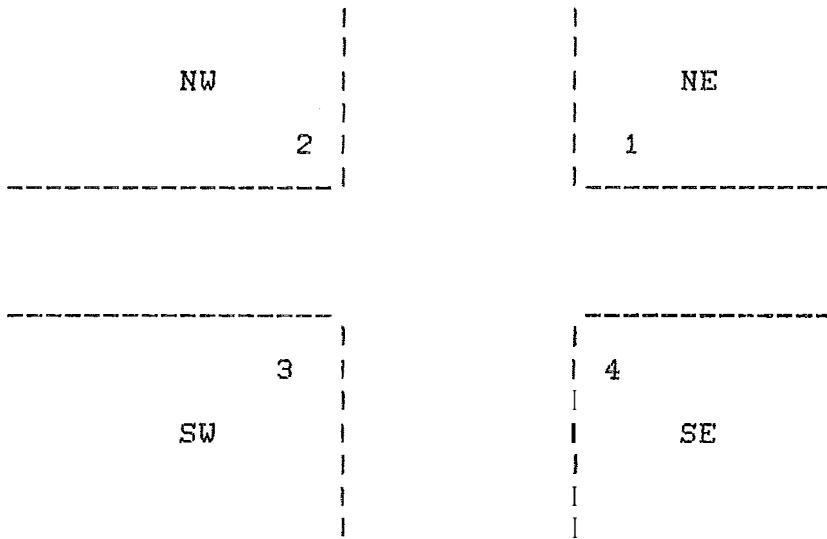


FIGURA 11.8 : Tipos de "esquinas"

No apêndice colocamos um algoritmo para gerar as restrições . Este algoritmo embora ainda sem implementação , **gude** servir **de** base para uma futura implementação .

Quando geramos as restrições várias delas são **supérfluas** . No apêndice colocamos uma maneira de eliminar tais restrições , permitindo diminuir a tempo de execução nos algoritmos de compactação .

11.5 Algoritmo do Caminho Mais Longo (ACML)

Consideremos um grafo $G = \{N,R\}$, sem circuitos , direcionado , simples , finito , com custos positivos (inteiros) , com uma fonte **e** um alvo . Desejamos obter o caminho mais longo ligando a fonte ao alvo . Este caminho mais longo ou caminho de maior custo é denominado caminho

crítico . Na figura 11.9 , podemos ver um exemplo de tal grafo .

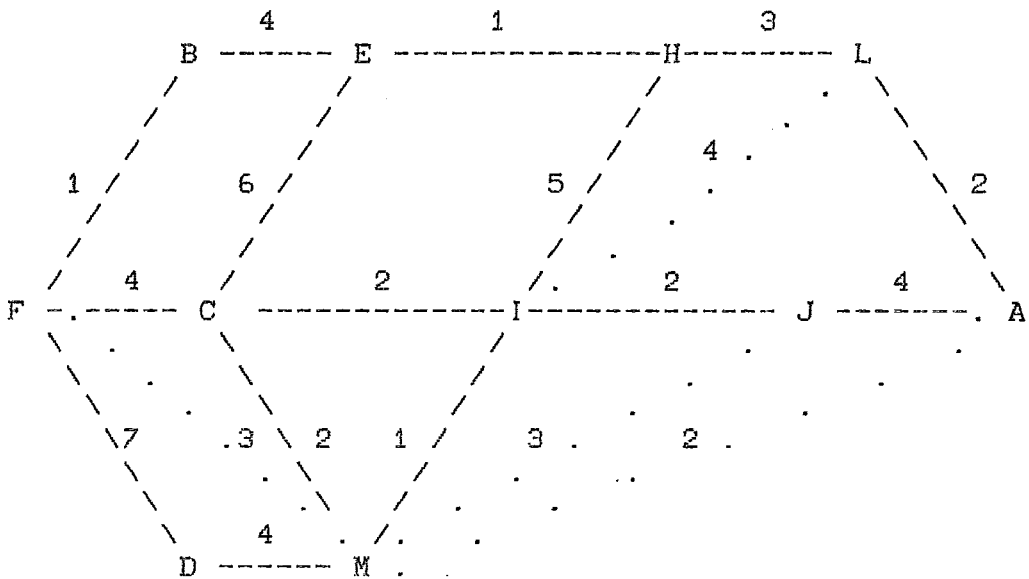


FIGURA 11.9 : Exemplo de um grafo

Os nós do grafo G na figura 11.9 , são as elementos do conjunto

$$N = \{F, B, C, D, E, M, I, H, J, L, A\}$$

onde F é a fonte e A o alvo . Os ramos de G são dados pelo conjunto

$$R = \{ (F, B, 1), (F, C, 4), (F, D, 7), (B, E, 4), (C, E, 6), (F, M, 3), \\ (C, M, 2), (D, M, 4), (C, I, 2), (M, I, 1), (E, H, 1), (I, H, 5), \\ (M, J, 3), (I, J, 2), (I, L, 4), (H, L, 3), (M, A, 2), (J, A, 4), \\ (L, A, 2) \}$$

onde cada terno acima é formado pelo nó inicial, o nó final

e o custo deste ramo do grafo .

A rigor , no nosso caso , não usaremos um grafo como o do exemplo acima , pois como podemos ver na seção II.4 e no apêndice , alguns ramos pontilhados podem ser eliminados quando geramos as restrições. Quando fazemos as eliminações no método de gerar as restrições (ver apêndice) , o ramo pontilhado $(M,A,2)$ não é eliminado (porém neste exemplo podemos eliminá-lo por simples inspeção). Este fato ocorre devido às restrições do tipo "ou" (veja seção II.6) . Porém estes ramos continuam no grafo a título de exemplo.

II.5.1 A Árvore Geradora Máximal

Para que obtenhamos o caminho crítico de F a A , vamos dar a cada um dos nós , o custo do caminho mais caro para ir de F até este nó . A nossa intenção neste caso , é obter a árvore geradora máximal do grafo C de maior custo . Como não sabemos estes custos , inicialmente vamos atribuir aos nós um custo nulo . Depois olhando para cada um dos ramos de G , comparamos o custo do nó final do ramo com o do nó inicial . Colocamos como custo do nó final do ramo , o maior valor entre o custo pré-existente do nó final e o custo do nó inicial mais o custo do ramo em questão .

Consideremos que o número de ramos em R é q , que estes estão na forma (i,j,C_{ij}) , onde i e j são elementos do conjunto de nós N e que C_{ij} é o custo deste ramo . A seguir colocamos o algoritmo ACML :

Algoritmo <do caminho mais longo (ACML)> :

(Obtém uma árvore geradora maximal , fornecendo os caminhos críticos para ir da fonte a todos os demais nós)

(N =: conjunto dos nós do grafo ;

R =: conjunto dos ramos do grafo ;

F =: fonte ;

m =: contador dos ramos ;

i =: nó inicial , j =: nó final e C_{ij} =: custo do ramo (i,j,C_{ij}) ;

q =: número de ramos em R ;

v_j =: maior custo entre os caminhos obtidos para ir de F até o nó j ;

a_j =: antecessor do nó j na Arvore maximal obtida ;)

inicia

i <- F ;

m <- 1 ;

Para todo j em N , v_j <- 0 e a_j <- 0 ;

1: CONSIDERE O RAMO $m = (i,j,C_{ij})$ em R ;

Se $v_i + C_{ij} > v_j$

então : v_j <- $v_i + C_{ij}$

a_j <- i ;

m <- m + 1 ;

Se m = q + 1

então : PARE ; (Para todo j em N , o valor v_j é o custo do caminho ótimo para ir da fonte F ao nó j e a_j são os antecessores do nó j na árvore geradora máxima .)

senão : VÁ PARA 1 ;

fim

Para que o algoritmo funcione , é necessário que o conjunto N esteja de tal forma que a fonte seja o primeiro nó , o alvo o último e os demais nus estejam ordenados de maneira que os sucessores venham sempre depois de seus antecessores . Além disso , o conjunto R dos ramos também deve ter uma ordenação. Seguindo a urdem de N , o conjunta R deve ser ordenado pelos nós finais de cada ramo . Estas ordenações são possíveis porque exigimos que o grafo não tenha circuitos .

Para resolver o exemplo , montamos o quadro 11.1 a seguir , onde atualizamos os valores de v_j e de a_j conforme vamos percorrendo o conjunto R , segundo o algoritmo ACML acima:

| | | | | | | | | | | | | |
|-------|---|---|---|---|----|----|----|----|----|----|----|---|
| . | | | | | | D | | | | | L | |
| . | | | | | | C | C | M | I | | H | J |
| a_j | | F | F | F | B | F | C | E | M | I | M | |
| . | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| Nós : | F | B | C | D | E | M | I | H | J | L | A | |
| . | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| . | | 1 | 4 | 7 | 5 | 3 | 6 | 11 | 14 | 16 | 13 | |
| v_j | | | | | 10 | 6 | 12 | 17 | | 20 | 18 | |
| . | | | | | | 11 | | | | | 22 | |

QUADRO 11.1

Com auxílio do quadro 11.1 podemos construir a árvore geradora máxima, recuperando os caminhos críticos através dos antecessores de cada nó que estão na parte do quadro de aj na posição mais acima. Na figura 11.10 desenhamos esta árvore, onde os números ao lado dos nós (valores de v_j , são os custos máximos para se ir da fonte F até aquele nó.

Este algoritmo é usado em PERT/CPM com o nome de "Longest Path Algorithm" e pode ser encontrado com mais detalhes no livro de CONDRAN e MINOUX C071. Aqui nos limitamos à forma que será empregada no algoritmo de compactação.

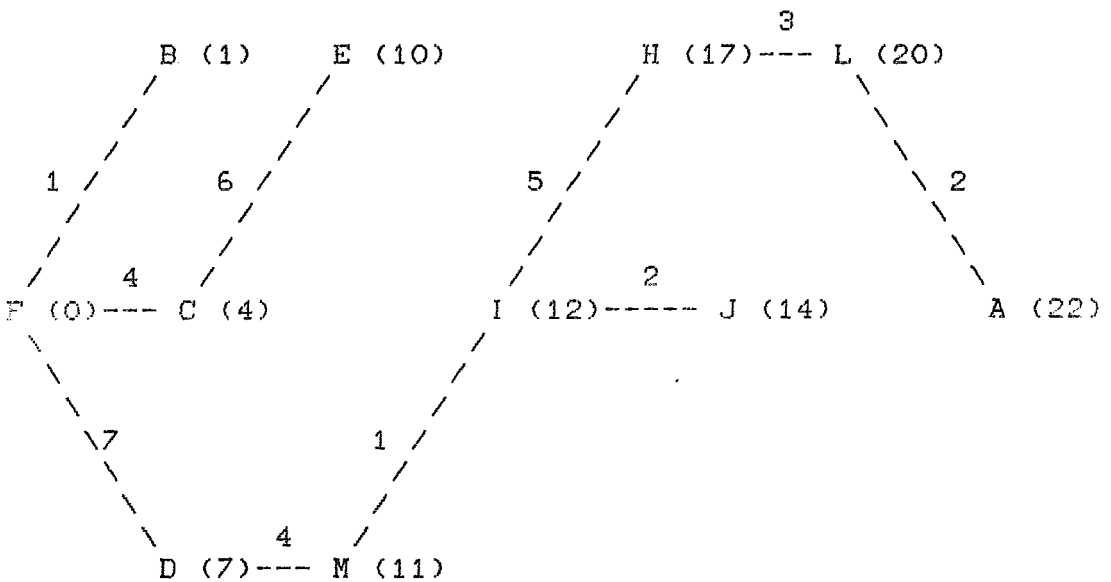


FIGURA 11.10 : Árvore geradora maximal

11.5.2 As Folgas dos Nós

Gostaríamos de saber qual é o maior custo que cada um dos nós do grafo C pode ter sem que o custo do alvo aumente. Denominamos este valor para cada um dos nós, o custo máximo do nó.

Denominamos o algoritmo que calcula estes custos de LDATE. Se pensamos nos custos como datas, isto é, nos custos de cada arco como tempo, estamos procurando a última data i (em inglês "last date") de cada nó que não altere a data final. Ou melhor, que a data do alvo continue a mesma.

O algoritmo que descreveremos a seguir pode ser encontrado no livro de HIRSHFELD [113], onde é feita uma discussão mais profunda sob o ponto de vista de PERT/CPM.

Para obter este custo máximo dos nós, inicialmente começamos com o alvo, atribuindo a próprio custo dele ao seu custo máximo. Depois percorrendo o conjunto dos nós N em ordem decrescente, isto é, do alvo A para a fonte F , procuramos os sucessores de cada nó. Colocamos como custo máximo de cada nó, o mínimo dentre todos os sucessores deste nó, da diferença entre o custo máximo do sucessor e o custo do arco entre o nó e o sucessor.

Na seção 11.5.1, o algoritmo ACML obtém os custos v_j de cada um dos nós j pertencentes a N . Denominaremos por x_j os custos máximos dos nós j . c_{ij} será o custo do arco ligando i e j . Suporemos que o conjunto dos nós N tenha n elementos, assim o custo do alvo é x_n .

Algoritmo <LDATE> ;

(Calcula o custo máximo de cada um dos nós , fornecendo as folgas dos nós . Os nós devem estar numerados com a fonte igual a 1 e o alvo igual a n .)

(N =: conjunto dos nós do grafo ;

n =: número de nós em N ;

m =: contador de nós ;

i =: nó inicial , j =: nó final e C_{ij} =: custo do ramo (i, j, C_{ij}) ;

J_i =: conjunto dos sucessores do nó i ;

j =: sucessor de i ;

x_n =: custo do alvo ;

X_1 =: vetor com n coordenadas com os custos máximos dos nós ;

x_{1j} =: j-ésima coordenada de X_1 , cujo valor é o custo máximo do nó j ;

inicio

$x_{1n} \leftarrow x_n$;

m \leftarrow 1 ;

1: i \leftarrow n - m ;

OBTENHA J_i ;

Para todo j em J_i , $x_{1i} \leftarrow$ mínimo ($x_{1j} - C_{ij}$) ;

m \leftarrow m + 1 ;

Se m = n + 1

então : PARE ; (os custos máximos estão em X_1 .)

senão : VÁ PARA 1 ;

fim

A folga de cada nó é a diferença entre, o custo máximo e o custo de cada nó , $x_{ij} - x_j$. Esta é a quantidade que o custo do nó pode crescer sem aumentar o custo do nó final . O caminho crítico é aquele que tem folga zero . Pode haver mais de um caminho crítico no grafo .

Na figura II.11 , colocamos em cada um dos nós do exemplo um terno que representa seu custo, seu custo máximo e sua folga .

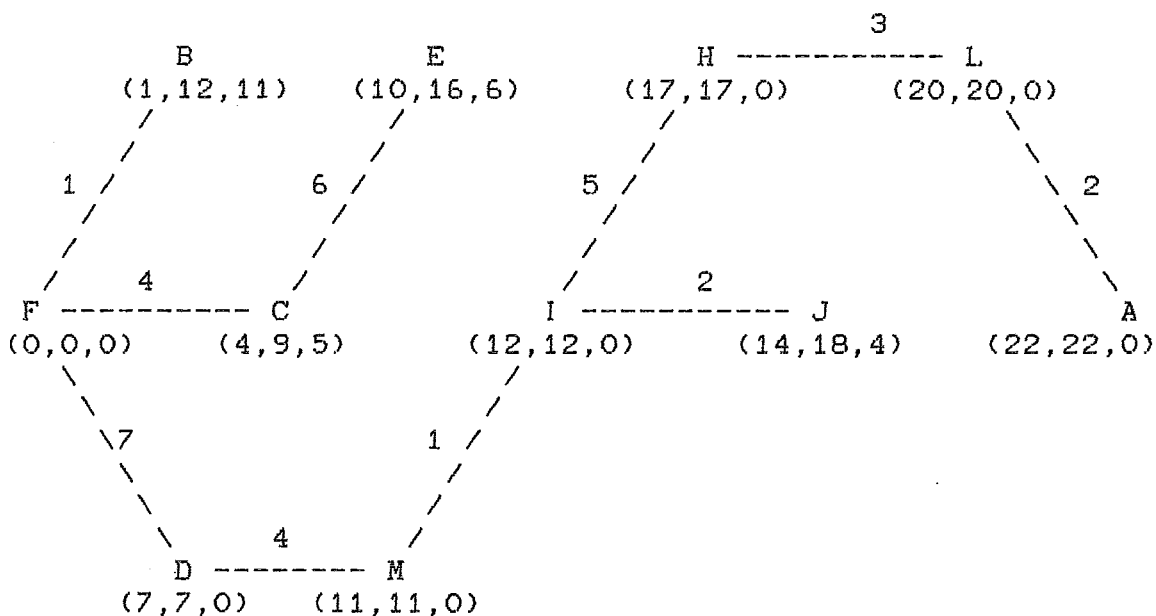


FIGURA II.11 : A folga dos nós

Olhando para a figura II.11 , vemos nós que têm folga zero . Estes são exatamente aqueles nós , que formam o caminho crítico . Assim o caminho crítico , é a caminho descrito pelo conjunto $\{ E, F, D, M, I, H, L, A \}$.

11.5.3 Uma Pós-Otimização no Grafo

Usando as folgas podemos fazer uma pós-otimização no grafo. Uma primeira pós-otimização já foi feita quando definimos as folgas. Sabemos até quanto podemos aumentar o custo de um arco sem alterar o custo do alvo. Se este valor ultrapassar a folga, o custo do alvo crescerá tanto quanto for o valor ultrapassado, não importando se o referido arco está ou não no caminho crítico. A única diferença é que se o arco não está na caminho crítico, no novo grafo ele estará. Após a alteração as folgas mudarão de valor.

Ao acrescentarmos um novo arco no grafo, qual será a mudança que ocorrerá no custo total? Desejamos colocar um novo arco (i, j, C_{ij}) , onde conhecemos o custo do nó i , x_i , e o custo máximo do nó j , x_{lj} . Este novo arco não vai mudar o custo total do grafo, se o novo custo do nó j não ultrapassar x_{lj} . Isto é, o custo do nó i mais C_{ij} deve ser no máximo igual ao custo máximo do nó j . Podemos escrever que se

$$x_{lj} - x_i \geq C_{ij}$$

não haverá mudança no custo do alvo. Caso esta equação não seja satisfeita, o custo do alvo aumentará de

$$dx = C_{ij} + x_i - x_{lj}.$$

Este valor dx é o que o novo valor do nó j ultrapassará x_{lj} . Neste caso, o valor do alvo x_n mudará para

$$x_n + dx.$$

Quando retiramos um arco do grafo, qual a mudança no custo total? Se o arco não pertence ao caminho crítico o custo do alvo não será alterado. Se o arco está no caminho crítico, também pode não ter nenhuma alteração, pois pode existir um outro caminho crítico diferente deste. Podemos neste caso saber apenas, qual seria a alteração máxima que poderia ocorrer no custo total. Retirando o arco (i,j,C_{ij}) pertencente a um caminho crítico do grafo, o custo do alvo no máximo diminuirá de C_{ij} .

11.6 Algoritmo de Compactação

Como vimos as restrições do problema são de dois tipos "e" e "ou". Consideremos um vetor V com tantas coordenadas quantos são as restrições do tipo "ou". Cada uma das coordenadas de V , $V(i)$, está associada a uma restrição do tipo "ou". Estas coordenadas recebem os valores 0, 1 ou -1. O valor de $V(i)$ é 0 se a restrição "ou" número i a ser considerada é a restrição em X , vale 1 se for considerada a restrição em Y e -1 se não considerarmos nenhuma das duas restrições (neste caso o problema pode ser inviável, isto é, os requisitos tecnológicos do circuito podem não ser atendidos).

Ao fixarmos $V(i)$ com valor 0 ou 1 obtemos dois grafos como o da seção II.5, um para 0 e outro para 1 da seguinte forma:

Quando as restrições tipo "ou", são fixadas em X ou em Y , todas as restrições passam a ser do tipo "e",

Consideremos as coordenadas horizontais dos elementos do circuito

$$x_1, x_2, x_3, \dots, x_n.$$

Os nós do grafo são os índices destas coordenadas

$$NX = \{1, 2, 3, \dots, n\}.$$

Cada restrição $x_j - x_i \geq d_{ij}$ representa um ramo (i, j, d_{ij}) em RX do grafo

$$GX = (NX, RX).$$

O mesmo procedimento que fizemos com as restrições horizontais para obter o grafo GX , devemos fazer com as restrições verticais para obter o grafo

$$GY = (NY, RY) \quad \text{a partir de } y_1, y_2, \dots, y_n.$$

Para obtermos o leiaute mais compacto que obedeça a estas restrições, basta aplicar o algoritmo ACML da aos grafos GX e GY , independentemente.

Os grafos GX e GY não tem circuitos porque todos seus ramos (i, j, d_{ij}) , de cada um dos grafos, estão sempre orientados de i para j onde $i < j$.

Ma seção 11.5 comentamos que não seria possível eliminar todos os ciclos do grafo em questão, devido às restrições do tipo "ou". Na figura 11.12 vemos um exemplo de um ciclo de um grafo que não pode ser eliminado. Eliminamos da figura 11.12.a a ramo $(C, A, 2)$, obtendo a figura 11.12.b.

Quando fixamos as restrições tipo "ou" , estas aparecem em alguns grafos e em outros não . Se no grafo da figura II.12 o ramo $(Z,C,5)$, associado a uma restrição tipo "ou" não aparece , este pode aparecer no grafo da figura

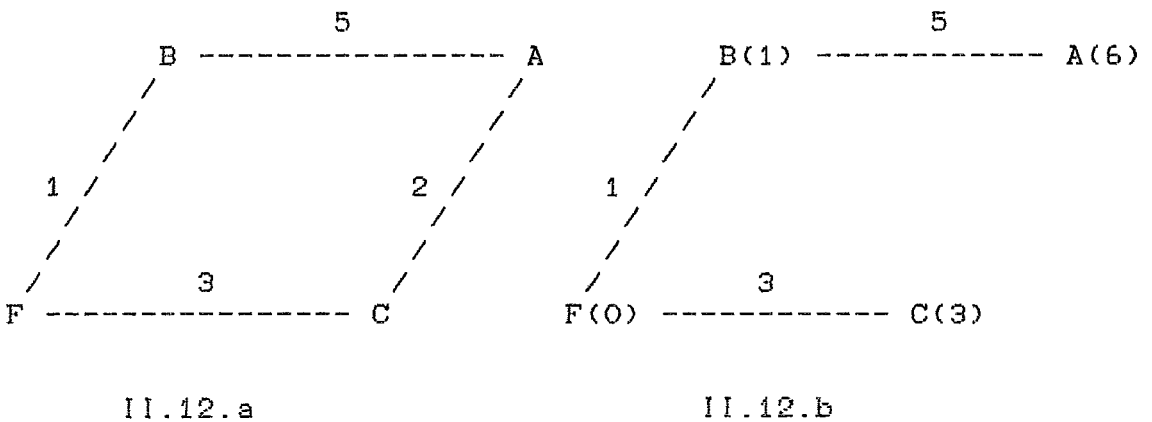
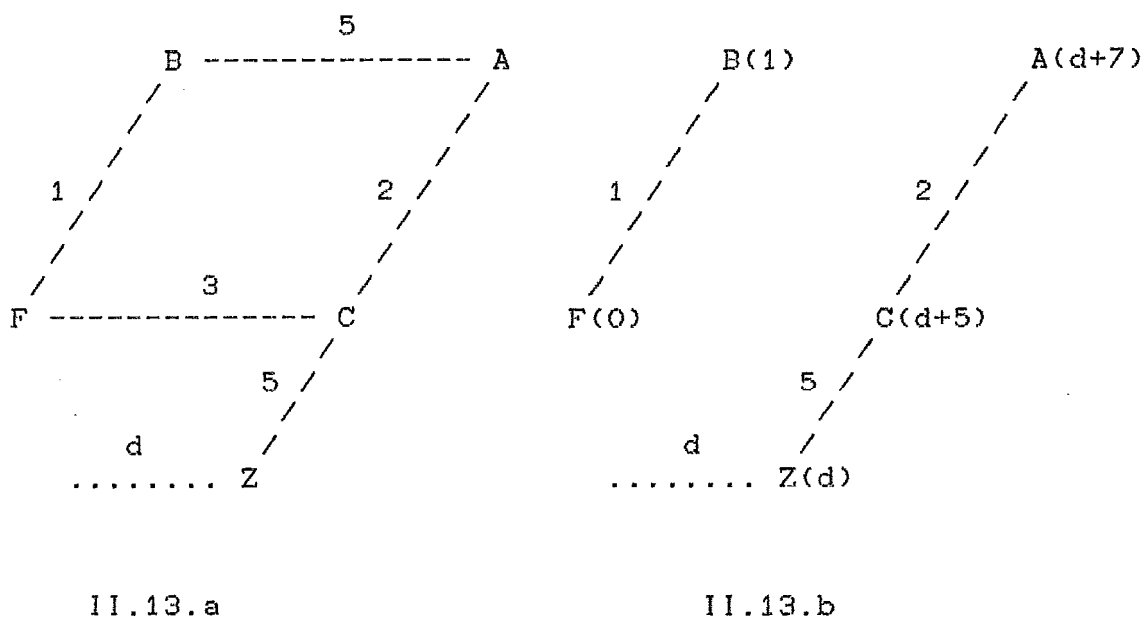


FIGURA 11.12 : Eliminando o ramo $(C,A,2)$

II.13 . E neste caso a árvore geradora maximal que aparece na figura II.12.b , é diferente daquela que aparece na figura II.13.b . Por isso, na seção II.4 não podemos fazer mais eliminações do que as que já estão propostas lá , sem maiores considerações .



11.13.a

11.13.b

FIGURA 11.13 : O ramo (C,A,2) não é eliminado

Vimos que quando fixamos as variáveis tipo "ou" em X ou em Y o problema é simples de ser resolvido. O que fica difícil de se resolver é quando variamos estas restrições. Se o número de restrições "ou" é p , temos que aplicar $\frac{(p+1)}{2}$ vezes o algoritmo ACML, uma vez que fixando o vetor V , ficamos com dois grafos G_X e G_Y .

11.6.1 Buscando Um Ótimo Local

Usaremos um vetor V (como o descrito na seção 11.6) com suas coordenadas fixas por V_i , para cada valor fixo. Neste caso estaremos considerando que as coordenadas de V_i são 0 ou 1. Convém chamar a atenção do leitor sobre a notação que estamos usando. Denotamos por V_i um vetor fixo, e por $V(i)$ a i -ésima coordenada do vetor V .

Diremos que o vetor V_i é vizinho do vetor V_j , se os dois vetores são iguais exceto em uma coordenada.

O algoritmo procure entre os vizinhos de um vetor dado V_0 , um outro vetor V_j que seja melhor que ele. Depois tomamos o vetor V_j no lugar de V_0 e repetimos o processo até que não encontremos nenhum vizinho melhor.

Consideremos um vetor V . Os vetores X e Y são as soluções obtidas aplicando o algoritmo ACML para V e para os grafos G_X e G_Y , respectivamente. O valor da coordenada de X de maior índice é x_n e o de Y é y_n . Assim a área que obtemos a partir de V é $x_n * y_n$.

Com este algoritmo estamos usando um terceiro grafo G cujos nós são os vetores V_i e os ramos são formados com os vizinhos. Isto é, (V_i, V_j, C_{ij}) é um ramo se V_i é vizinho de V_j . O custo C_{ij} deste ramo é fornecido pela diferença entre, a área obtida de V_j e a área obtida de V_i . Ou seja: $C_{ij} = (\text{área obtida de } V_j) - (\text{área obtida de } V_i)$. O algoritmo bi-dimensional procura o caminho de custo mínimo partindo de um nó inicial V_0 . Este caminho existe pois os custos são inteiros, o custo de um caminho é a diferença das áreas obtidas dos nós final e inicial e as áreas estão logicamente limitadas.

A seguir colocamos o algoritmo que foi usado para obter um ótimo local, quando são consideradas as restrições do tipo "ou",

Algoritmo <Compactação bi-dimensional> :

(Faz uma busca em profundidade no grafo G)

CG =: grafo descrito na seção II.6.1 ;

V_i =: nó de G . Vetor de coordenadas 0 ou 1 ;

p =: número de coordenadas de V_i ;

V_0 =: nó inicial ;

área =: valor da área obtida de V_i ;

X =: solução obtida da algoritmo ACML para G_X e V_i ;

xnx =: valor da coordenada de mais alto índice de X ;

Y =: solução obtida do algoritmo ACML para G_Y e V_i ;

yny =: valor da coordenada de mais alto índice de Y ;)

inicio

Use o ACML para V_0 obtendo X e Y :

1: área <- xnx * yny ;

j <- 1 ;

2: Modifique a coordenada j de V_0 obtendo o vizinho V_j ;

Use o ACML para V_j obtendo novos valores de X e Y ;

Se área > xnx * yny

então : V_0 <- V_j

vá para 1 :

senão : Se j = p

então : PARE ; (V_0 é um ótimo local)

senão : j <- j + 1

v para 2 ;

fim

I . . Gerando Uma Solução Inicial e Obtendo Um Limite Inferior

Para que obtenhamos a solução inicial , que nos referimos na seção II.6.1 , vamos apresentar dois algoritmos diferentes , que são usados independentemente de acordo com o problema em questão .

O primeiro , descrito na seção II.6.2.1 , obtém uma solução inicial mais rapidamente e não usa as folgas das nós . O segunda algoritmo , que veremos na seção II.6.2.2 , usa as folgas dos nós e apesar de levar mais tempo em termos computacionais , porque a cada passo re-calcula as folgas , obtém uma solução inicial mais próxima do ótimo local .

Todos dois algoritmos obtém uma cota inferior para a área , da seguinte maneira . Consideramos o vetor V com todas as suas coordenadas iguais a -1 , isto é , estaremos usando inicialmente somente as restrições tipo "e"
Aplicamos o algoritmo ACHL para G_X e G_Y obtendo respectivamente X e Y para este vetor inicial V . O valor $x_n x * y_n y$ é um limite inferior para a área da circuito , uma vez que não estamos considerando nenhuma restrição tipo "ou"
Ao colocarmos estas restrições no problema , a área tenderá a aumentar .

II.6.2.1 Uma Solução Inicial Rápida

Para obtermos uma solução inicial, olhamos para as restrições tipo "ou", digamos a de número k :

$$x_i - x_j \geq d_{ij} \quad \text{ou} \quad y_i - y_j \geq d_{ij} .$$

Dependendo do valor de $x_i - x_j - d_{ij}$ e de $y_i - y_j - d_{ij}$ escolhemos o valor 0 ou 2 para $V(k)$. Se as duas restrições falham, colocamos em $V(k)$ o valor 0 se o valor em X for maior do que o em Y , e o valor 1 se ocorrer o oposto. Se uma das duas não falhar não alteramos o vetor V nesta fase. Repetimos o processo reaplicando o ACML para os novos valores de V até que não alteremos mais os valores das coordenadas do vetor V . As coordenadas de V que ainda ficaram iguais a -1, são tornadas iguais a 0 se a restrição em X for satisfeita, senão serão colocadas com o valor 1. A seguir descrevemos o algoritmo:

Algoritmo <Solução inicial rápida> :

{Obtem rapidamente uma solução inicial para o algoritmo de
compactação bi-dimensional }

(G =: grafo descrito na seção II.6.1 ;

V =: vetor de coordenadas 0 , 1 ou -1 . É um nó de G
quando $V(i)$ é diferente de -1 , para tudo i ;

p =: número de coordenadas de V ;

área =: valor da área obtida de V_i ;

X =: solução obtida do algoritmo ACML para G_X e V_i ;

xnx =: valor da coordenada de mais alta índice de X ;

Y =: solução obtida do algoritmo ACML para G_Y e V_i ;

yny =: valor da coordenada de mais alto índice de Y ;

muda =: variável indicativa de que houve alteração em $V(i)$
para algum $i = 1, \dots, p$;)

inicio

Vara todo $k = 1, \dots, p$, $V(k) \leftarrow -1$;

Use a ACML para V obtendo X e Y ;

{ uma cota inferior para a área é $xnx * yny$ }

k $\leftarrow 0$;

muda $\leftarrow 0$;

1 : k $\leftarrow k + 1$;

Se $k = p + 1$

então : vá para 3 ;

2: OBTENHA A RESTRIÇÃO "ou" EM X NÚMERO k ;
 (($x_i - x_j \geq d_{ij}$))

Se $x_i - x_j \geq d_{ij}$

então : v_6 para 2 ;

senão : OBTENHA A RESTRIÇÃO "ou" EM Y NÚMERO k
 (($y_s - y_t \geq d_{st}$))

Se $y_s - y_t \geq d_{st}$

então : $v_ã$ para 1

senão : Se

$d_{ij} - x_i + x_j > d_{st} - y_s + y_t$

então : $V(k) \leftarrow 0$

$muda \leftarrow 1$

v_6 para 1 ;

senão : $V(k) \leftarrow 1$

$muda \leftarrow 1$

$v_á$ para 1 ;

3: Se $muda = 1$

então : $muda \leftarrow 0$

$k \leftarrow 0$

$v_á$ para 1 ;

senão : $k \leftarrow 0$

4: $k \leftarrow k + 1$;

Se $k = p + 1$

então : PARE ;

(V é uma solução inicial)

senão : Se $V(k) = -1$

então : $v_á$ para 5 ;

senão : $v_á$ para 4 ;

5: OBTENHA A RESTRIÇÃO "ou" EM X NÚMERO k ;
 (($x_i - x_j \geq d_{ij}$))

Se $x_i - x_j \geq d_{ij}$

então : $V(k) \leftarrow 0$;

senão : $V(k) \leftarrow 1$;

$v_á$ para 4 ;

fim

II.6.2.2 Uma Solução Inicial Usando as Folgas

Da mesma forma que fizemos na seção II.6.2 , para obter uma cota inferior usamos todas as coordenadas do vetor V iguais a -1 .

Em primeiro lugar , verificamos dentre as restrições do tipo "ou" aquelas que satisfazem as restrições do tipo "e" , isto é , aplicamos o algoritmo ACML para G_X e G_Y somente com as restrições "e" e depois verificamos se as restrições tipo "ou" em X satisfazem a elas . Se satisfizerem mudamos a coordenada correspondente do vetor V para 0 . Depois fazemos o mesmo para Y , alterando neste caso , a coordenada para 1 .

Depois , analisamos uma a uma as restrições em X com folga suficiente para não aumentar a área . Toda vez que uma coordenada de V é alterada de -1 para 0 , neste caso , re-aplicamos o algoritmo ACHL e re-calculamos as folgas . Após a análise com X , fazemos o mesmo tipo de análise para Y , transformando as coordenadas do vetor V de -1 para 1 .

Finalmente escolhemos dentre as coordenadas de V que ainda ficaram iguais a -1 , as restrições "ou" em X ou em Y aquelas que aumentam menos a área. Isto é feito usando a pós-otimização que nos referimos na seção II.5.3 quando colocamos um novo arco em um grafo . Para cada alteração do vetor V , ou o que é a mesma coisa dos grafos G_X ou G_Y , os vetores X ou Y são re-calculados e também as suas folgas .

Algoritmo <Solução inicial com as folgas> :

{Obtem uma solução inicial para o algoritmo de compactação bi-dimensional , usando as Folgas das variáveis}

{G =: grafo descrito na seção 11.6.1 ;

V =: vetor de coordenadas 0 , 1 ou -1 . É um nó de G quando $V(i)$ é diferente de -2 , para todo i ;

p =: número de coordenadas de V ;

área =: valor da área obtida de V_i ;

X =: solução obtida do algoritmo ACHL para G_X e V_i ;

xnx =: valor da coordenada de mais alto índice de X ;

Y =: solução obtida do algoritmo AÇML para G_Y a V_i ;

yny =: valor da coordenada de mais alto índice de Y ;

XL =: vetor das folgas de X ;

xli =: i-ésima coordenada de XL ;

YL =: vetor das folgas de Y ;

yli =: i-ésima coordenada de YL ;

arenox =: valor da área nova, ao acrescentar uma restrição em X ;

arenoy =: valor da área nova, ao acrescentar uma restrição em Y ; }

inicio

Para tudo $k = 1, \dots, p$, $V(k) \leftarrow -1$;

Use o ACML para V obtendo X e Y ;

{ uma cota inferior para a área é $xnx * yny$ }

k $\leftarrow 0$;

1: $k \leftarrow k + 1$;

Se $k = p + 1$

então : vá para 3 ;

senão : vá para 2 ;

2: OBTENHA A RESTRIÇÃO "ou" EM X NÚMERO k ;

(($x_i - x_j \geq d_{ij}$))

Se $x_i - x_j \geq d_{ij}$

então : $V(k) \leftarrow 0$

vá para 1 ;

senão : vá para 1 ;

3: $k \leftarrow 0$;

4: $k \leftarrow k + 1$;

Se $k = p + 1$

então : vá para 6 ;

senão : vá para 5 ;

5: OBTENHA A RESTRIÇÃO "ou" EM Y NÚMERO k ;

(($y_i - y_j \geq d_{ij}$))

Se $V(k) = -1$

então : Se $y_i - y_j \geq d_{ij}$

então : $V(k) \leftarrow 1$;

vá para 4 ;

6: $k \leftarrow 0$;

7: $k \leftarrow k + 1$;

Se $k = p + 1$

então : vá para 9 ;

8: OBTENHA A RESTRIÇÃO "ou" EM X NÚMERO k ;

(($x_i - x_j \geq d_{ij}$))

se $V(k) = -1$

então : CALCULE AS FOLGAS DE X , XL

Se $x_{li} - x_j \leq d_{ij}$

então : vá para 7 ;

senão : $V(k) \leftarrow 0$

use o ACML para calcular X ;

vá para 7 ;

9: $k \leftarrow 0$

10: $k \leftarrow r + 1$

Se $k = p + 1$

então : vá para 12

11: OBTENHA A RESTRIÇÃO "ou" EM Y NÚMERO k ;

E ($y_i - y_j \geq d_{ij}$))

Se $V(k) = -1$

então : CALCULE AS FOLGAS DE Y , YL

Se $y_{li} - y_j \leq d_{ij}$

então : vá para 10 ;

senão : $V(k) \leftarrow 1$

use o ACML para calcular Y ;

vá para 10 ;

12: $k \leftarrow 0$;

13: $k \leftarrow k + 1$;

Se $k = p + 1$

então : PARE ; (V é uma solução inicial)

14: OBTENHA A RESTRIÇÃO "ou" EM X NÚMERO k ;

$((x_i - x_j \geq d_{ij}))$

Se $V(k) = -1$

então : CALCULE AS FOLGAS DE X , XL

$dx \leftarrow d_{ij} - x_{li} + x_i$

$arenox \leftarrow (x_{nx} + dx) \wedge y_{ny}$

OBTENHA A RESTRIÇÃO "ou" NÚMERO k em Y

$((y_i - y_j \geq d_{ij}))$

CALCULE AS FOLGAS DE Y , YL

$dy \leftarrow d_{ij} - y_{li} + y_i$

$arenoy \leftarrow x_{nx} * (y_{ny} + dy)$

Se $arenox \leq arenoy$

então : $V(k) \leftarrow 0$;

senão : $V(k) \leftarrow 1$;

vá para 15 ;

senão : vá para 13 ;

15: RECALCULE X e Y , usando o algoritmo ACML ;

vá para 13 ;

fim

II.7 Conclusão

O tipo de modelo que usamos neste trabalho, é interessante, pois torna o problema bastante palpável. Queremos dizer com palpável, o fato de encontrarmos facilmente uma solução viável para o problema e de ser possível fazer uma ligação, entre os processos iniciais do desenho do circuito e sua fabricação.

Já dissemos anteriormente, que a maior dificuldade deste problema é quando o número variáveis do tipo 0-1 é muito grande. O método de compactação não leva necessariamente a um ótimo absoluto, e além do que é difícil dizer se os ótimos locais obtidos são também ótimos absolutos. WATANABE C293, usa o método Branch-and-Bound para obter ótimos absolutos. Porém quando o problema é muito grande, este método fica impraticável. O que nós fazemos, é tentar melhorar as soluções obtidas variando a solução inicial. Algumas experiências neste sentido estão descritas no capítulo IV.

Outra informação útil, é saber até que ponto devemos melhorar a solução. Temos que levar em consideração o tempo de máquina que é gasto, bem como quanto pode ser melhorada a solução. Para sabermos o quanto podemos melhorar a solução, obtemos uma cota inferior para o problema e tentamos melhorar esta cota. Esta informação pode nos levar a saber, em alguns casos, que a solução do método de compactação que obtivemos é ótima. Fazemos a procura, de cotas inferiores melhores nos capítulos III e IV.

CAPITULO IIIRELAXAÇÃO LAGRANGEANA

III.1 Introdução

Neste capítulo , pretendemos aplicar relaxação lagrangeana , com o intuito de obter uma cota inferior para o problema descrito no capítulo II . Já havíamos obtido , naquele capítulo , uma cota inferior . O que pretendemos agora neste capítulo , é melhorar esta cota para sabermos quão boa é a solução que obtivemos quando aplicamos o algoritmo de compactação . É interessante que esta cota inferior seja melhorada , pois não sabemos se a solução em questão é ótima ou se ela é um ótimo local .

Na seção III.2 , escrevemos o problema descrito no capítulo II de forma que , ele fique de uma maneira mais fácil de aplicar o método de relaxação lagrangeana . A teoria necessária ao método de relaxação lagrangeana , é mostrada na seção III.3, sempre levando em conta o problema em questão . Na seção III.4 , mostramos como este fica no caso do nosso problema .

III.2 Adaptação do Problema

Vimos no capítulo II , que o problema em questão tem dois tipos de restrição , do tipo "e" e do tipo "ou"

Vamos agora escrever o problema matematicamente .

As restrições do tipo "e" são separáveis , como restrições em X e como restrições em Y . Podemos assim , escrever essas restrições matricialmente como

$$A1 X \geq b1 \quad e \quad (III.1)$$

$$A2 Y \geq b2 \quad (III.2)$$

onde $A1$ e $A2$ são matrizes $m_x \times n_x$ e $m_y \times n_y$, respectivamente , $b1$ matriz $m_x \times 1$, $b2$ matriz $m_y \times 1$ e X , Y são as variáveis do problema com n_x e n_y coordenadas . Isto significa , que temos n_x variáveis horizontais e n_y verticais . Ainda mais , o número de restrições horizontais do tipo "e" é m_x e o número de restrições verticais é m_y .

As restrições do tipo "ou" aparecem aos pares , uma em X e outra em Y , e são da forma

$$x_i - x_j \geq d_{ij} \quad ou \quad (III.3)$$

$$y_s - y_t \geq d_{st} \quad . \quad (III.4)$$

Colocamos índices diferentes em x e em y , porque esse fato pode ocorrer , quando Consideramos o segundo caso descrito na seção 11.2 . Em geral , $i = s$ e $j = t$ com $d_{ij} \neq d_{st}$. Sem perda de generalidade , deixaremos os índices diferentes .

Criando para cada par de restrições do tipo "ou" uma variável t_k com valores no conjunto $(0,1)$, podemos modificar a forma de escrever as restrições III.3 e III.4

para

$$x_i - x_j + M t_k \geq d_{ij} \quad e \quad (III.5)$$

$$y_s - y_t + M (1 - t_k) \geq d_{st} \quad (III.6)$$

onde , M é um número real grande . **As** equações III.3 e III.4 e as equações III.5 e III.6 tem as mesmas soluções . Pois fazendo a variável $t_k = 0$, a equação III.5 fica igual a equação III.3 . Tomamos M suficientemente grande para que, qualquer que seja o valor de y_s e y_t , a equação III.6 que neste caso fica

$$M \geq d_{st} + y_t - y_s ,$$

seja sempre satisfeita . Da mesma forma , se $t_k = 1$ a equação III.6 e a equação III.4 ficam iguais . E tomamos M grande o bastante, para que também a equação III.5 seja satisfeita para todos os valores possíveis de x_i e de x_j .

Considerando que temos p restrições do tipo "ou" , podemos escrever estas matricialmente , da seguinte forma

$$D1 X + M T \geq b3 \quad e \quad (III.7)$$

$$D2 Y + M (1I - T) \geq b4 \quad (III.8)$$

onde $D1$ e $D2$ são matrizes $p \times n_x$ e $p \times n_y$, respectivamente , $X = (x_1, x_2, \dots, x_{n_x})$, $Y = (y_1, y_2, \dots, y_{n_y})$, $1I$ é a matriz $p \times 1$ cujos elementos são todos iguais a 1 , $b3$ e $b4$ são matrizes $p \times 1$ e $T = (t_1, t_2, \dots, t_p) \in \{0,1\}$.

Assim , o problema que temos a resolver considerando o semi-perimetro no lugar da área , fica da seguinte forma :

$$\text{minimizār } xnx + yny \quad (\text{III.9})$$

sujeito a

$$A1 X \geq b1 \quad (\text{III.1})$$

$$A2 Y \geq b2 \quad (\text{III.2})$$

$$D1 X + M T \geq b3 \quad (\text{III.7})$$

$$D2 Y + M (11 - T) \geq b4 \quad (\text{III.8})$$

$$X \geq 0, Y \geq 0 \text{ e } T \in (0,1)^P.$$

III.3 Um Pouco de Teoria

Nesta seção adaptaremos o problema de relaxação lagrangeana ao problema III.9. Uma abordagem mais geral, pode ser encontrada no livro de ROCKAFELLAR [23], onde é desenvolvida teoria a respeito de funções convexas. Podemos usar a mesma teoria trocando funções convexas por côncavas. O método de relaxação lagrangeana também é encontrado nos livros de GONDRAN e MINOUX [07], LASDON [14] ou na publicação de MACULAN, CAMPELLO e LOPES [163] utilizado em outras aplicações.

Vamos considerar o problema III.9, denominado problema primal (P), escrito da seguinte forma:

$$a = \text{mínimo } xnx + yny \quad (\text{P})$$

sujeito a

$$A1 X \geq b1 \quad (\text{III.1})$$

$$A2 Y \geq b2 \quad (\text{III.2})$$

$$g_i(X, T) \geq 0 \quad (i = 1, \dots, p) \quad (\text{III.10})$$

$$h_i(Y, T) \geq 0 \quad (i = 1, \dots, p) \quad (\text{III.11})$$

$$X \geq 0, Y \geq 0 \text{ e } T \in (0,1)^P.$$

onde as equações III.1 e III.2 são as mesmas da seção III.2 e as funções g_i e h_i são definidas pelas equações III.7 e III.8.

Para cada uma das restrições III.10 e III.11, associamos números reais u_{g_i} e u_{h_i} não negativas, totalizando L números, denominados Multiplicadores de Lagrange.

Definimos uma função \mathcal{L} , denominada Lagrangeana, como segue:

$$\mathcal{L}(X, Y, T, U_g, U_h) = x_n x + y_n y - U_g \cdot G(X, T) - U_h \cdot H(Y, T)$$

onde $U_g = (u_{g_1}, \dots, u_{g_p})$, $U_h = (u_{h_1}, \dots, u_{h_p})$,
 $G(X, T) = (g_1(X, T), \dots, g_p(X, T))$ e
 $H(Y, T) = (h_1(Y, T), \dots, h_p(Y, T))$.

Esta é uma função Lagrangeana. Definimos desta forma, para que as variáveis T do tipo 0-1 não fiquem mais nas restrições do problema como podemos ver no problema III.12.

Denominaremos agora, uma relaxação lagrangeana do problema (P), o problema III.12, com $U_g \geq 0$ e $U_h \geq 0$:

$$L(U_g, U_h) = \text{mínimo } \mathcal{L}(X, Y, T, U_g, U_h) \quad (\text{III.12})$$

sujeito a

$$A_1 X \geq b_1 \quad (\text{III.1})$$

$$A_2 Y \geq b_2 \quad (\text{III.2})$$

$$X \geq 0, Y \geq 0, T \in \{0, 1\}^p$$

Mostraremos a seguir que o problema III.12 é realmente uma relaxação do problema (P).

Proposição III.1 : $L(Ug,Uh) \leq z$.

Demonstração :

$\mathcal{L}(X,Y,T,Ug,Uh) = xnx + yny - (Ug.G(X,T) + Uh.H(Y,T))$
 com $Ug \geq 0$, $Uh \geq 0$, $G(X,T) \geq 0$ e $H(Y,T) \geq 0$,
 logo $Ug . G(X,T) + Uh . H(Y,T) \geq 0$,
 e assim $\mathcal{L}(X,Y,T,Ug,Uh) \leq xnx + yny$

Seja (X^*, Y^*, T^*) uma solução ótima do problema (P) ,
 com $z = xnx^* + yny^*$ logo
 $L(Ug,Uh) \leq \mathcal{L}(X^*, Y^*, T^*, Ug,Uh) \leq xnx^* + yny^*$
 de onde concluímos que $L(Ug,Uh) \leq z$.

A proposição III.1 mostra que $L(Ug,Uh)$ é uma cota inferior para o problema (P) . Como pretendemos melhorar as cotas , tomemos a maior dentre estas cotas , variando os multiplicadores de Lagrange a fim de maximizar as relaxações lagrangeanas . Denominamos o problema (D) a seguir , problema dual de (P) :

$L(Ug,Uh) = \text{máximo } L(Ug,Uh)$ (D)
 sujeito a $Ug \geq 0$ e $Uh \geq 0$

Pela proposição III.1 , temos que $L(Ug,Uh) \leq z$ e em geral , nos problemas de otimização combinatória não acontece a igualdade . Neste caso dizemos que há uma folga primal-dual .

Proposição III.2 : A função $L(Ug,Uh)$ é afim por partes e é côncava .

Demonstração :

Vamos simplificar para esta demonstração , sem perda de generalidade , a forma de escrever a função $L(Ug,Uh)$. Colocaremos como $L(U) = \text{mínimo } f(X,Y) - U \cdot W(X,Y,T)$, onde $U = (Ug,Uh)$, $f(X,Y) = xnx + yny$ t $W(X,Y,T) = (G(X,T) , H(Y,T))$

Mostraremos que L é afim por partes fixando U e T . Fazendo isto ficamos com um problema de programação linear. Pois neste caso , f , G e H são transformações lineares . Quando fixamos T com um valor T_0 e variamos U , o que estamos fazendo é uma pós-otimização em programação linear . Esta pós-otimização acontece , porque estamos modificando a função objetivo . Assim , se $l = L(U)$ obtemos hiperplanos da forma

$$l = f(X_0,Y_0) - U \cdot W(X_0,Y_0,T_0) ,$$

onde X_0 e Y_0 são valores da solução ótima quando fixamos T em T_0 . Ao variarmos T obtemos 2^p conjuntos de hiperplanos , onde $L(U)$ é a menor envoltória destes hiperplanos .

Para verificarmos que L é côncava tomemos $U_1, U_2 \geq 0$ e $t \in [0,1]$. Sejam $U = t \cdot U_2 + (1-t) \cdot U_1$ e (X, Y, T) tal que $L(U) = f(X, Y) - U \cdot W(X, Y, T)$ então $L(U_1) \leq f(X, Y) - U_1 \cdot W(X, Y, T)$ e $L(U_2) \leq f(X, Y) - U_2 \cdot W(X, Y, T)$

Multiplicando a primeira inequação por $1-t$, a segunda por t e somando as duas , obtemos

$$t L(U_2) + (1-t) L(U_1) \leq f(X, Y) - U \cdot W(X, Y, T) = L(U) .$$

Mostramos então que L é côncava .

Para uma função escalar f com domínio em \mathbb{R}^n , côncava e diferenciável, o plano tangente ao gráfico num ponto X_0 fica acima do gráfico de f . Uma equação do plano tangente é

$$z = f(X_0) + \nabla f(X_0) \cdot (X - X_0),$$

onde $\nabla f(X_0)$ é o gradiente de f em X_0 . Isto quer dizer que

$$f(X) \leq f(X_0) + \nabla f(X_0) \cdot (X - X_0),$$

para todo $X \in \mathbb{R}^n$.

Por outro lado se consideramos uma função escalar com n variáveis reais, f ainda côncava e diferenciável, e a equação

$$f(X) \leq f(X_0) + D_n(X - X_0), \quad (\text{III.14})$$

$X, X_0, D \in \mathbb{R}^n$

podemos transformá-la na equação

$$\frac{f(X) - f(X_0)}{\|X - X_0\|} \leq \frac{D \cdot (X - X_0)}{\|X - X_0\|}, \quad (\text{III.15})$$

onde as barras $\|\cdot\|$ significam a norma euclidiana. Tomando o limite uni-direcional quando X tende a X_0 na equação III.15, obtemos a igualdade

$$f'(X_0; u) = D \cdot u. \quad (\text{III.16})$$

Onde $f'(X_0; u)$ é a derivada direcional em X_0 , na direção do vetor unitário $u = \frac{X - X_0}{\|X - X_0\|}$. A igualdade na equação

ção III.16 ocorre pois na equação III.14 temos a igualdade quando $X = X_0$, a equação III.15 é uma transformação da equação III.14 para $X \neq X_0$, tomamos o limite na equação III.15 quando X tende a X_0 e ainda convém ressaltar que os limites existem porque consideramos f diferenciável. Ainda mais, se usarmos a fórmula de cálculo da derivada direcional para funções diferenciáveis (APOSTOL E023), temos que

$$\nabla f(X_0) \cdot u = D \cdot u, \text{ para todo vetor unitário } u.$$

Temos então que $D = \nabla f(X_0)$.

De uma outra maneira, mostramos que

$$\nabla f(X_0) \cdot u \leq D \cdot u \text{ para todo vetor unitário } u,$$

mas como D e $\nabla f(X_0)$ são vetores fixos isso ocorre se e somente se $\nabla f(X_0) = D$.

Demonstramos assim, a seguinte proposição:

Proposição III.3: O único vetor D , que satisfaz a equação III.14 para uma função f côncava e diferenciável em X_0 , é o gradiente de f em X_0 , isto é, $D = \nabla f(X_0)$.

Podemos estender este conceito, para funções côncavas mas não diferenciáveis. Para isto vamos dar a definição de sub-gradiente.

Definição III.1: Um vetor D do \mathbb{R}^n é um sub-gradiente da função côncava f de \mathbb{R}^n em \mathbb{R}^n no ponto X_0 se,

$$f(X) \leq f(X_0) + D \cdot (X - X_0) \text{ para todo } X \text{ em } \mathbb{R}^n.$$

Neste caso se f não é diferenciável não deveremos ter

a unicidade do vetor D . Para cada ponto X_0 , onde f não é diferenciável, temos um conjunto de sub-gradientes.

Definição III.2 : Ao conjunto de todos os sub-gradientes de f em X_0 , denominamos sub-diferencial de f em X_0 , e o denotamos por $Df(X_0)$.

Na proposição III.3, mostramos que se f é diferenciável e sub-diferencial é unitário. A recíproca desta proposição está demonstrada no livro de ROCKAFELLAR **C233**, na seção 25. Isto é, se o sub-diferencial for unitário num ponto, então a função é diferenciável neste ponto.

No nosso caso o sub-diferencial é não vazio. Pois consideremos $(\bar{X}, \bar{Y}, \bar{T})$ a solução ótima do problema III.12, com os multiplicadores fixos em U_{g0} , U_{h0} , isto é,

$$\begin{aligned} L(U_{g0}, U_{h0}) &= \mathcal{L}(\bar{X}, \bar{Y}, \bar{T}, \bar{U}_{g0}, U_{h0}) = \\ &= x_{n+1} + y_{n+1} - U_{g0} G(\bar{X}, \bar{T}) - U_{h0} H(\bar{Y}, \bar{T}). \end{aligned}$$

Escreveremos L como na demonstração da proposição III.2 para simplificar, ou seja

$$L(U) = f(\bar{X}, \bar{Y}) - U_0 W(\bar{X}, \bar{Y}, \bar{T}). \quad (\text{III.17})$$

Proposição III.3 : $-W(\bar{X}, \bar{Y}, \bar{T})$ é um sub-gradiente de L em U_0 .

Demonstração :

Devemos mostrar que $L(U) \leq L(U_0) + D(U - U_0)$ com $D = -W(\bar{X}, \bar{Y}, \bar{T})$. Pela definição de L temos que

$$L(U) \leq f(\bar{X}, \bar{Y}) - U_0 W(\bar{X}, \bar{Y}, \bar{T}) \quad (\text{III.18})$$

subtraindo as equações III.18 e III.17 obtemos ,

$$L(U) - L(U_0) \leq (U_0 - U) W(\bar{X}, \bar{Y}, \bar{T})$$

ou

$$L(U) \leq L(U_0) + (-W(\bar{X}, \bar{Y}, \bar{T})) (U - U_0)$$

■

Mostramos assim que $DL(U) \neq \emptyset$. Com este fato , vamos ver outra propriedade do sub-diferencial :

Proposição III.4 : $Df(X_0)$ é fechado e convexo .

Demonstração :

Para mostrar que $Df(X_0)$ é fechado , consideremos uma sequencia $\{D_i\}$ de sub-gradientes em $Df(X_0)$, convergindo para D . Para cada D_i Lemos que

$$f(X) \leq f(X_0) + D_i (X - X_0) ,$$

para todo X em R^n .

Precisamos mostrar que $D \in Df(X_0)$, isto é , D é um sub-gradiente de f em X_0 .

Suponhamos que D não seja um sub-gradiente de f em X_0 . Então existe X_1 em R^n tal que

$$f(X_1) > f(X_0) + D (X_1 - X_0) ,$$

onde podemos escrever que

$$f(X_1) - \alpha = f(X_0) + D (X_1 - X_0) , \quad (\text{III.19})$$

para $\alpha > 0$.

Como para cada elemento da sequencia $\{D_i\}$,

$$f(X_1) \leq f(X_0) + D_1 (X_1 - X_0) \quad , \quad (III.20)$$

se subtraímos as equações III.20 e III.19 , ficamos com

$$\alpha \leq (D_1 - D) (X_1 - X_0) \leq \|X_1 - X_0\| \|D_1 - D\| .$$

Logo para todo i temos $\frac{\alpha}{\|X_1 - X_0\|} \leq \|D_1 - D\|$.

Isto contradiz o fato da sequência $\{D_i\}$ convergir para D . Então toda sequência em $Df(X_0)$ convergente , converge em $Df(X_0)$. O que nos diz que $Df(X_0)$ é fechado .

A convexidade de $Df(X_0)$ pode ser mostrada tomando D_1 e D_2 em $Df(X_0)$ e $D = t D_2 + (1-t) D_1$, $t \in [0,1]$. Como D_1 e D_2 estão em $Df(X_0)$

$$f(X) \leq f(X_0) + D_1 (X - X_0) \quad ,$$

para todo X em R^n e

$$f(X) \leq f(X_0) + D_2 (X - X_0) \quad ,$$

para todo X em R^n .

Multiplicando a segunda inequação por t , a primeira por $1-t$ e somando as duas ficamos com

$$\begin{aligned} f(X) &\leq f(X_0) + t D_2 (X - X_0) + (1-t) D_1 (X - X_0) = \\ &= f(X_0) + D (X - X_0) . \end{aligned}$$

Logo D é um sub-gradiente de f em X_0 , mostrando que $Df(X_0)$ é um conjunto convexo .

No caso de funções diferenciáveis, para sabermos se um ponto é um máximo, é preciso que olhemos se ele é um ponto crítico. Isto é, se neste ponto o gradiente se anula.

Para funções não diferenciáveis, basta que no ponto haja um sub-gradiente nulo, como poderemos ver na seguinte proposição:

Proposição 111.5 : Se f é uma função côncava então X_0 maximiza f se e somente se $0 \in Df(X_0)$.

Demonstração :

$0 \in Df(X_0)$ se e somente se

$$f(X) \leq f(X_0) + 0(X - X_0) = f(X_0), \text{ para}$$

todo X em \mathbb{R}^n . Isto quer dizer que X_0 maximiza f .

■

111.4 O Método do Sub-Gradiente

Para resolvermos o problema em questão, isto é, o problema dual (D), devemos obter o máximo da função L . Já vimos na proposição 111.2, que esta função é côncava. Assim neste caso, qualquer máximo local é também um máximo absoluto.

Tomando como base as funções diferenciáveis, onde o gradiente aponta no sentido de crescimento máximo da função. Adotando a direção do gradiente, a função deverá ter seu valor aumentado, caso o passo seja convenientemente escolhido. Podemos ainda ressaltar que a função L é li-

mitada superiormente (veja proposição III.1) . Desta maneira , no caso das funções não diferenciáveis , escolhemos como direção de movimento em cada ponto , a de algum sub-gradiente naquele ponto .

O método do sub-gradiente foi testado por HELD , WOLFE e CROWDER [15] . Aplicações deste método também podem ser encontradas no trabalho de MACULAN , CAMPELLO e LOPES [16].

A seguir apresentamos o método , onde é definida uma sequência de números $\{s_k\}$ convergente para 0 . Começamos com um multiplicador inicialmente arbitrado U_0 . Para cada iteração k , a partir do multiplicador U_k procuramos o multiplicador de índice $k+1$ na direção do sub-gradiente $-W(X_k, Y_k, T_k)$ no ponto U_k (veja proposição III.3) . O passo em cada direção é dado por uma sequência $\{s_k\}$, da qual falaremos mais adiante (veja a fórmula III.21 na próxima página) . As coordenadas deste novo multiplicador que forem negativas , as colocamos iguais a 0 . Usaremos a mesma notação da seção anterior , isto é ,

$$L(U) = \text{Mínimo } \mathcal{L}(X, Y, T, U) \quad , \quad \text{onde}$$

$$\mathcal{L}(X, Y, T, U) = f(X, Y) - U W(X, Y, T) \quad .$$

Algoritmo <método do sub-gradiente> :

{Obtem soluções para o problema dual (D)}

(U_k =: vetor dos multiplicadores de Lagrange ;

s_k =: elementos de uma sequência convergente para 0 ;

\mathcal{L} =: função lagrangeana ;

L =: função que se pretende maximizar ;

W =: sub-gradiente de L ;)

Início

Escolha $U_0 \in \mathbb{R}^{p+}$ arbitrário ;

Defina uma sequência $\{s_k\}$ convergente para 0 ;

$k \leftarrow 0$;

(Cálculos da iteração k :)

i: Calcule $L(U_k) = \mathcal{L}(X_k, Y_k, T_k, U_k) = \text{Mínimo } \mathcal{L}(X, Y, T, U_k)$;

se $W(X_k, Y_k, T_k) = 0$

então : PARE ; (a solução é ótima)

(Obtenção do próximo multiplicador U_{k+1} :)

$k_1 \leftarrow k + 1$;

$U_{k_1} \leftarrow U_k - s_k W(X_k, Y_k, T_k)$;

Projete U_{k_1} em \mathbb{R}^{p+} ;

Se s_k é suficientemente pequeno

então : PARE ;

$k \leftarrow k_1$;

vá para 1 ;

fim

Os multiplicadores escolhidos como no método do sub-gradiente, convergem para o máximo de $L(U)$. Isto foi demonstrado por POLYAK C223 e também por AGMON C013, com a condição que a sequência $\{s_k\}$ seja convergente para 0 e a série seja divergente para mais infinito.

A sequência pode ser escolhida da seguinte maneira:

$$s_k = r_k \frac{L(U^*) - L(U_k)}{\|W(X_k, Y_k, T_k)\|_2}, \quad (III.21)$$

*

onde U^* é o valor máximo de L , $0 < \alpha \leq r_k \leq 2$ com α fixo e $W(X_k, Y_k, T_k)$ é o sub-gradiente no ponto U_k , na iteração k .

Uma idéia geométrica de onde surge esta sequência pode ser encontrada no artigo de MOTZKIN e SCHOENBERG [20].

*

Em problemas práticos o valor $L(U^*)$ não é conhecido. Porém HELD, WOLFE e CROWDER C103 mostraram que o valor de $L(U^*)$ pode ser substituído por alguma solução viável do problema (P). Esta substituição na sequência $\{s_k\}$, definida pela fórmula III.21, faz com que o método continue convergindo. Neste caso a sequência $\{s_k\}$ poderá não convergir para 0. Para que isto ocorra WELD, WOLFE e CROWDER C101, fazem r_k tender a 0. A regra usada por eles, faz $r_k = 2$ durante as 4 primeiras iterações (o dobro do número de multiplicadores, que no nosso caso é $3p$), depois dividem r_k ao meio a cada q iterações (q fixo). Eles notaram que apesar da série da sequência

(s_k) não divergir para mais infinito, no trabalho deles o algoritmo sempre convergiu. No nosso caso, utilizamos a melhor solução viável encontrada no capítulo II, para fazer esta substituição. Se denominamos esta melhor solução por L , a fórmula 111.21 fica

$$s_k = r_k \frac{L - L(U_k)}{\|W(X_k, Y_k, T_k)\|_z}$$

No livro de GONDRAN e MINOUX (1973), são discutidas outras formas de fazer r_k tender a 0, e outra sequência (s_k).

Pudemos ainda observar, que no nosso caso o problema de calcular $L(U_k)$ consiste de dois problemas independentes de programação linear e um outro linear sem restrições cujas variáveis tem valores 0 ou 1. Assim vejamos, escrevendo a função $\mathcal{L}(X, Y, T, U_k)$ por extenso, temos

$$\begin{aligned} \mathcal{L}(X, Y, T, U_k) &= x_n x + y_n y - U_g G(X, T) - U_h H(Y, T) = \\ &= x_n x + y_n y - (u_{g1}, \dots, u_{gp})(g_1(X, T), \dots, g_p(X, T)) - \\ &\quad - (u_{h1}, \dots, u_{hp})(h_1(Y, T), \dots, h_p(Y, T)) = \\ &= x_n x + y_n y - \sum_{k=1}^P u_{gk} g_k(X, T) - \sum_{k=1}^P u_{hk} h_k(Y, T) = \\ &= x_n x + y_n y - \sum_{k=1}^P u_{gk} C_{xik} - x_{jk} + M t_k - b_{3k} - \\ &\quad - \sum_{k=1}^P u_{hk} C_{yqk} - y_{rk} + M(1-t_k) - b_{4k}. \end{aligned}$$

Reordenando os somatórios, de modo que as variáveis X , Y e T fiquem agrupadas, obtemos

$$\begin{aligned}
\mathcal{L}(X, Y, T, U_k) = & \left[x_n x - \sum_{k=1}^p u_{gk} (x_{ik} - x_{jk}) \right] + \\
& + \left[y_n y - \sum_{k=1}^p u_{hk} (y_{qk} - y_{rk}) \right] + \\
& + \left[M \sum_{k=1}^p (u_{hk} - u_{gk}) t_k \right] + \\
& + \left[\sum_{k=1}^p u_{gk} b_{3k} - u_{hk} (M - b_{4k}) \right] .
\end{aligned}$$

Verificamos então que a função \mathcal{L} se decompõem nas seguintes funções :

$$\mathcal{L}(X, Y, T, U_k) = f_1(X) + f_2(Y) + f_3(T) + C$$

onde as funções f_1 , f_2 e f_3 são lineares e C é uma constante . Isto porque U_k é constante em cada iteração .

Como precisamos minimizar E sujeito às condições

$$A_1 X \geq b_1 \quad (\text{III.1})$$

$$A_2 Y \geq b_2 \quad (\text{III.2})$$

$$X \geq 0, \quad Y \geq 0 \quad \text{e} \quad T \in (0, 1)^p,$$

podemos separar este problema em três problemas distintos :

Um problema de programação linear em X ,

$$\text{Minimizar } f_1(X) \quad (\text{III.22})$$

sujeito a

$$A_1 X \geq b_1 \quad (\text{III.1})$$

$$X \geq 0 .$$

Um outro problema de programação linear em Y ,

$$\text{Minimizar } f_2(Y) \quad (\text{III.23})$$

sujeito a

$$A_2 Y \geq b_2 \quad (\text{III.2})$$

$$Y \geq 0 .$$

É um terceiro problema linear com variáveis 0-1 , que pode ser resolvido por simples inspeção

$$\text{Minimizar } f_3(T) \quad (\text{III.24})$$

sujeito a

$$T \in (0,1)^p .$$

III.5 O Método do Sub-Gradiente no Problema de Compactação

Como vimos na seção 111.4 o problema de calcular $L(U_k)$ no caso do problema de compactação , é separável em três problemas distintos . Assim nesse caso a algoritmo fica bastante simples . Vamos re-escrever o método do sub-gradiente , para o problema de compactação :

Algoritmo <método do sub-gradiente aplicado ao problema de compactação> :

{Obtem uma cota inferior para a área do leiaute de um circuito integrado}

{Uk =: vetor dos multiplicadores de Lagrange ;
 sk =: elementos de uma sequência convergente para 0 ;
 C =: função lagrangeana ;
 L =: função que se pretende maximizar ;
 W =: sub-gradiente de L ;
 Tk =: vetor de variáveis 0-1 ;
 X , D1 , b3 , Y , D2 , b4 =: definidos no problema III.9 ;
 M =: número definido nas equações III.5 e III.6 ;
 f1 , f2 , f3 e C =: funções e constante definidas na seção III.4 ; }

início

Escolha $U_0 = (0, \dots, 0)$;

L <- (melhor solução do algoritmo de compactação) ;

T0 <- (valor das variáveis 0-1 da melhor solução) ;

Defina a sequência

$$\{s_k\} = \left(r_k \frac{L - L(U_k)}{\|W(X_k, Y_k, T_k)\|_2} \right) ;$$

k <- 0 ;

{ Cálculos na iteração inicial : }

Use o algoritmo ACML para resolver o problema III.22
 obtendo X_0 ;

Use o algoritmo ACML para resolver o problema III.23
 obtendo Y_0 ;

$$W(X_0, Y_0, T_0) <- \left((D_1 X_0 + M T_0 - b_3)^t , \right. \\ \left. , (D_2 Y_0 + M (11 - T_0 - b_4))^t \right) ;$$

Vá para 2 ;

(Cálculos da iteração k :)

1: Use o método SIMPLEX para resolver o problema III.22
obtendo X_k ;

Use o método SIMPLEX para resolver o problema III.23
obtendo Y_k ;

Resolva o problema III.24 , obtendo T_k ;

Se $u_{hi} > u_{gi}$

então : $t_i \leftarrow 0$;

Se $u_{hi} < u_{gi}$

então : $t_i \leftarrow 1$;

$W(X_k, Y_k, T_k) \leftarrow ((D1 X_k + M T_k - b3)^t ,$
 $, (D2 Y_k + M (1I - T_k) - b4)^t) ,$

Se $W(X_k, Y_k, T_k) = 0$

então : PARE ; (a solução é ótima)

É Obtenção do próximo multiplicador $U_{k1} : I$

2: $k_1 \leftarrow k + 1$;

Calcule $L(U_k) = \mathcal{L}(X_k, Y_k, T_k, U_k) = f_1(X_k) + f_2(Y_k) +$
 $+ f_3(T_k) + C;$

Calcule s_k ;

$U_{k1} \leftarrow U_k - s_k W(X_k, Y_k, T_k)$;

Projete U_{k1} em R^{p+} ;

Se s_k é suficientemente pequeno

então : PARE ;

$k \leftarrow k_1$;

vá para I ;

fim (A melhor solução obtida no algoritmo , é $L(U_k)$.I

III.6 O Método de Compactação com M Variável

Vamos adotar agora um valor de R para cada uma das equações. Assim denominaremos os novos valores de M por M_x e M_y . Para cada restrição do tipo $*ou^p$ em X , digamos a de número k , associamos um valor M_{xk} e para as equações tipo $*ou^p$ em Y M_{yk} . Podemos então reescrever as equações III.5 e III.6 da seguinte maneira:

$$x_i - x_j + M_{xk} t_k \geq d_{ij} \quad e \quad (III.25)$$

$$y_s - y_t + M_{yk} (1 - t_k) \geq d_{st} \quad . \quad (III.26)$$

Escrevendo estas equações de uma forma matricial, como fizemos nas equações III.7 e III.8, ficamos com

$$D_1 X + M_x T \geq b_3 \quad e \quad (III.27)$$

$$D_2 Y + M_y (1I - T) \geq b_4 \quad , \quad (III.28)$$

onde D_1 , D_2 , X , Y , $1I$, T , b_3 e b_4 são da mesma forma e tamanho que nas equações III.7 e III.8. E as matrizes M_x e M_y são quadradas, de tamanho $p \times p$, diagonais, sendo que os elementos das diagonais são exatamente M_{xi} , $i = 1, \dots, p$, para a matriz M_x e M_{yi} , $i = 1, \dots, p$, para M_y .

O problema primal III.9, fica neste caso escrito da seguinte maneira:

$$\text{minimizar } xnx + yny \quad (\text{III.29})$$

sujeito a

$$A1 X \geq b1 \quad (\text{III.1})$$

$$A2 Y \geq b2 \quad (\text{III.2})$$

$$D1 X + Mx T \geq b3 \quad (\text{III.27})$$

$$D2 Y + My (1 - T) \geq b4 \quad (\text{III.28})$$

$$X \geq 0, \quad Y \geq 0 \quad \text{e} \quad T \in (0,1)^P.$$

As funções $g_i(X,T)$ e $h_i(X,T)$ das equações III.10 e III.11, passam a ser definidas pelas equações III.27 e III.28. A função \mathcal{L} do problema III.12 fica um pouco modificada :

$$\begin{aligned} \mathcal{L}(X,Y,T,U_k) &= xnx + yny - U_g G(X,T) - U_h H(Y,T) - \\ &= xnx + yny - (u_{g1}, \dots, u_{gp})(g_1(X,T), \dots, g_p(X,T)) - \\ &\quad - (u_{h1}, \dots, u_{hp})(h_1(Y,T), \dots, h_p(Y,T)) = \\ &= xnx + yny - \sum_{k=1}^p u_{gk} g_k(X,T) - \sum_{k=1}^p u_{hk} h_k(Y,T) = \\ &= xnx + yny - \sum_{k=1}^p u_{gk} (x_{ik} - x_{jk} + Mx_k t_k - b_{3k}) - \\ &\quad - \sum_{k=1}^p u_{hk} [y_{qk} - y_{rk} + My_k (1-t_k) - b_{4k}]. \end{aligned}$$

Reordenando os somatórios de E de modo que as variáveis X , Y e T fiquem agrupadas, obtemos

$$\begin{aligned}
\mathcal{L}(X, Y, T, U_k) = & C \sum_{i=1}^n x_{ix} - \sum_{k=1}^p u_{gk} (x_{ik} - x_{jk}) \quad + \\
& + \left[\sum_{j=1}^n y_{ny} - \sum_{k=1}^p u_{hk} (y_{qk} - y_{rk}) \right] \quad + \\
& + \left[\sum_{k=1}^p (M_{yk} u_{hk} - M_{xk} u_{gk}) t_k \right] \quad + \\
& + \left[\sum_{k=1}^p u_{gk} b_{3k} - u_{hk} (M_{yk} - b_{4k}) I \right] .
\end{aligned}$$

Verificamos então , que da mesma forma que na **seção III.4** , a função \mathcal{L} se decompõem nas seguintes funções

$$\mathcal{L}(X, Y, T, U_k) = f_1(X) + f_2(Y) + f_3(T) + C .$$

Os problemas de minimizar f_1 e f_2 sujeitos a

$$A_1 X \geq b_1 \quad (III.1)$$

$$A_2 Y \geq b_2 \quad (III.2)$$

$$X \geq 0 \text{ e } Y \geq 0 ,$$

são os mesmos problemas III.22 e III.23 , já que as funções f_1 e f_2 são as mesmas da **seção III.4** . A função f_3 ficou um pouco modificada , porém o problema continua a ser resolvido da **mesma** maneira . A constante C também ficou diferente da que tínhamos na **seção III.4** .

III.7 Área ou Semi-Perímetro ?

A opção de usarmos a área ou a semi-perímetro quando fazemos a compactação , não é tão importante quanto no método do sub-gradiente. No caso da compactação, o tempo ao usarmos a área ou a semi-perímetro é praticamente o mesmo .

A única diferença é calcular , em cada iteração , uma adição no lugar de uma multiplicação . Porém quando usamos o semi-perímetro no método do sub-gradiente , este problema fica bem mais fácil de ser resolvido . Assim quando fazemos somente a compactação , usamos a área . Quando fazemos a compactação seguida pelo método do sub-gradiente , usamos o semi-perímetro .

Vamos ver qual é o erro que cometemos, quando usamos , no problema de compactação , o semi-perímetro no lugar da área . Consideraremos , nesta seção , somente as maiores coordenadas de X e de Y , que são x_n e y_n , e que aqui denominaremos apenas por x e por y , respectivamente . No caso , o que estamos olhando , é para a diferença de usarmos a função $z = x y$ ou a função $z = x + y$. Sabemos do capítulo II que temos duas cotas , uma inferior e outra superior . Denominaremos as coordenadas da cota inferior por $(x,y) = (a,b)$, e as da cota superior por $(x,y) = (c,d)$. Assim a região dos possíveis valores para (x,y) , onde devemos procurar as soluções , é o retângulo esboçado na figura III.1 .

Se a e b são números grandes e (a,b) é próximo da reta $y = x$, isto é , os valores de a e b são próximos , as curvas de nível da função $z = x y$ que são hipérbolas , e as da função $z = x + y$ que são retas , são muito parecidas . Isto se dá pois , a reta tangente a cada uma destas hipérbolas , nos pontos em que $x = y$, é uma curva de nível de $z = x + y$.

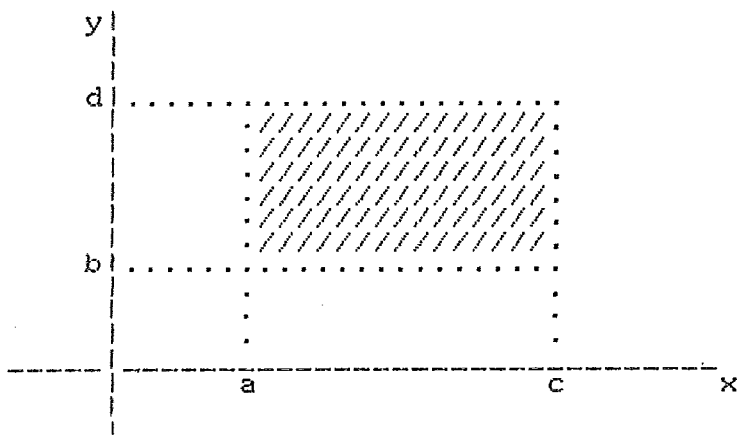


FIGURA III.1 : Região Viável

Suponhamos que o ótimo global, quando usamos o semi-perímetro, tem coordenadas (x_0, y_0) e o ótimo global quando usamos a área é (x_1, y_1) . O erro cometido quando usamos o semi-perímetro no lugar da área é

$$E = x_0 y_0 - x_1 y_1 . \quad (\text{III.30})$$

As interseções da hipérbole $xy = x_0 y_0$ com a reta, onde está o ótimo de $z = x + y$, que é $x + y = x_0 + y_0$, são próprio ponto (x_0, y_0) e o ponto (y_0, x_0) . Temos assim, duas regiões possíveis para o ponto (x_1, y_1) . Nas duas regiões o ponto (x_1, y_1) está "abaixo" da hipérbole $xy = x_0 y_0$, isto devido ao fato de (x_1, y_1) ser o ótimo global da função $z = xy$, e "acima" da reta $x + y = x_0 + y_0$, e neste caso devido a que (x_0, y_0) é o ótimo do semi-perímetro.

Denominaremos as regiões por , região da direita e da esquerda. A região da direita é aquela que contém os pontos (x, y) onde $x > y$, e a da esquerda a que contém

os pontos tais que $x < y$. Se $x_0 = y_0$ a reta $x + y = 2x_0$ será tangente à hipérbole $xy = x_0^2$, e neste caso (x_0, y_0) é também um ponto de ótimo da função $z = xy$.

Para discutirmos a existência das regiões, consideraremos dois casos. O primeiro é quando $x_0 > y_0$, ou seja o ponto (x_0, y_0) pertence a região da direita. Como estamos obviamente considerando que a região da direita exista, devemos ter $y_0 \geq b$. Neste caso, para que a região da esquerda exista é preciso que o ponto (y_0, x_0) , que é o outro ponto de interseção da reta com a hipérbole, seja viável. Isto é, temos de ter $y_0 \geq a$. O segundo caso é quando (x_0, y_0) está na região da esquerda, e assim temos $y_0 > x_0 \geq a$. Para que neste caso, a região da direita exista, devemos ter $x_0 \geq b$.

Suponhamos inicialmente que (x_0, y_0) esteja na região da direita. Se o ponto (x_1, y_1) também estiver na região da direita, a pior hipótese que podemos levar em conta, é aquela em que o ponto (x_1, y_1) estaria na hipérbole de menor nível desta região. Esta hipérbole é a que contém a interseção das retas $x + y = x_0 + y_0$ e $y = b$. Este ponto tem coordenadas $(x_0 + y_0 - b, b)$. Assim, considerando esta região e a equação III.30, o erro seria no máximo

$$E_1 = x_0 y_0 - (x_0 + y_0 - b) b \quad (\text{III.31})$$

A outra possibilidade ruim é se o ponto (x_1, y_1) estiver na região da esquerda, e pertencer a interseção da

reta $x = a$ com a reta $x + y = x_0 + y_0$. Então, o pior erro que podemos considerar neste caso é quando, $(x_1, y_1) = (a, x_0 + y_0 - a)$, e assim

$$E_2 = x_0 y_0 - a (x_0 + y_0 - a) . \quad (\text{III.32})$$

Considerando as duas regiões, o erro cometido quando usamos o semi-perímetro no lugar da área, é sempre menor ou igual ao maior valor entre E_1 e E_2 obtidos nas equações III.31 e III.32, ou seja, o erro absoluto E é tal que

$$E \leq \text{maior valor entre } (E_1; E_2) . \quad (\text{III.33})$$

A outra possível localização do ponto (x_0, y_0) , é na região da esquerda. Mas por simetria, podemos ver que o erro cometido é o mesmo calculado acima.

O erro relativo a considerar é

$$ER = E / (x_1 y_1) < E / (a b) . \quad (\text{III.34})$$

Na seção IV.4 fazemos uma análise numérica do erro.

CAPÍTULO IVEXPERIÊNCIA COMPUTACIONAL

IV.1 Introdução

Estão descritas neste capítulo , as experiências computacionais , usando a teoria desenvolvida nos capítulos II e III . Foi feito um programa em FORTRAN IV , que contém tanto o método de compactação desenvolvido no capítulo II , quanto à aplicação do método do sub-gradiente descrito no capítulo III . O método do sub-gradiente é usado após a compactação , e pudemos fazer a compactação sem usar o método do sub-gradiente .

```
-----  
| método de compactação |  
-----
```

```
|  
|  
|
```

```
-----  
| método do sub-gradiente |  
-----
```

A parte do programa para compactar um circuito é muito rápida . Tanto assim que passamos esta parte para um micro-computador da linha PC-XT/AT , e neste micro-computador um problema com 120 variáveis 0-1 , demorou cerca de 40

segundos de tempo total (no quadro IV.3 colocamos os tempos das principais subrotinas) . Porém a maior parte das experiências foram feitas em um computador de grande porte , um CYBER 170/835 . Na seção IV.3 e na IV.4 colocamos exemplos de utilização , com os tempos de CPU para executar , no CYBER , as rotinas mais importantes . A experiência com micro-computadores mostra que , para problemas não muito grandes , é viável a utilização destes equipamentos , para ilustrar problemas didáticos .

O maior problema que aplicamos o programa faz em um problema fictício com 120 variáveis com valores 0-1 . Um problema deste tamanho tem 2^{120} nós . Para termos idéia do tamanho do problema $2^{120} = 2^{10 \times 12}$, mas $2^{10} = 1024$ que é aproximadamente $1000 = 10^3$. Então o número de nós deste problema é $10^{3 \times 12} = 10^{36}$, aproximadamente . No capítulo II vimos que para cada nó , temos dois problemas de programação linear , um em X e outro em Y , que são resolvidos pelo algoritmo ACML (veja seção 11.5) . Na seção IV.3 , colocamos o resultado deste problema .

Na seção IV.5 estão os resultados da compactação do layout de dois circuitos reais Scell e T-FE10-Flop .

IV.2 Descrição das Rotinas Empregadas

Nesta seção vamos descrever sucintamente as subrotinas e as funções , usadas no programa de compactação e de aplicação de relaxação lagrangeana .

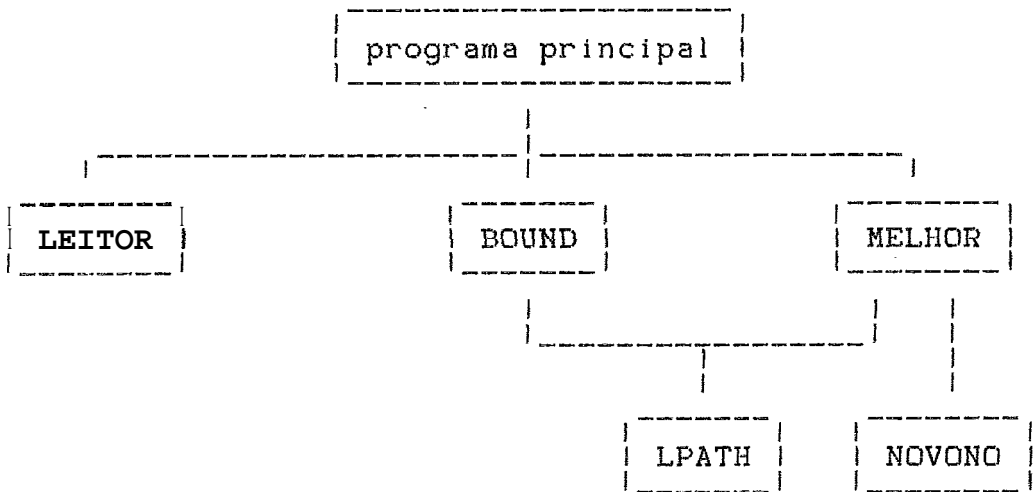


FIGURA IV.1 : Diagrama de Estrutura do Método de Compactação

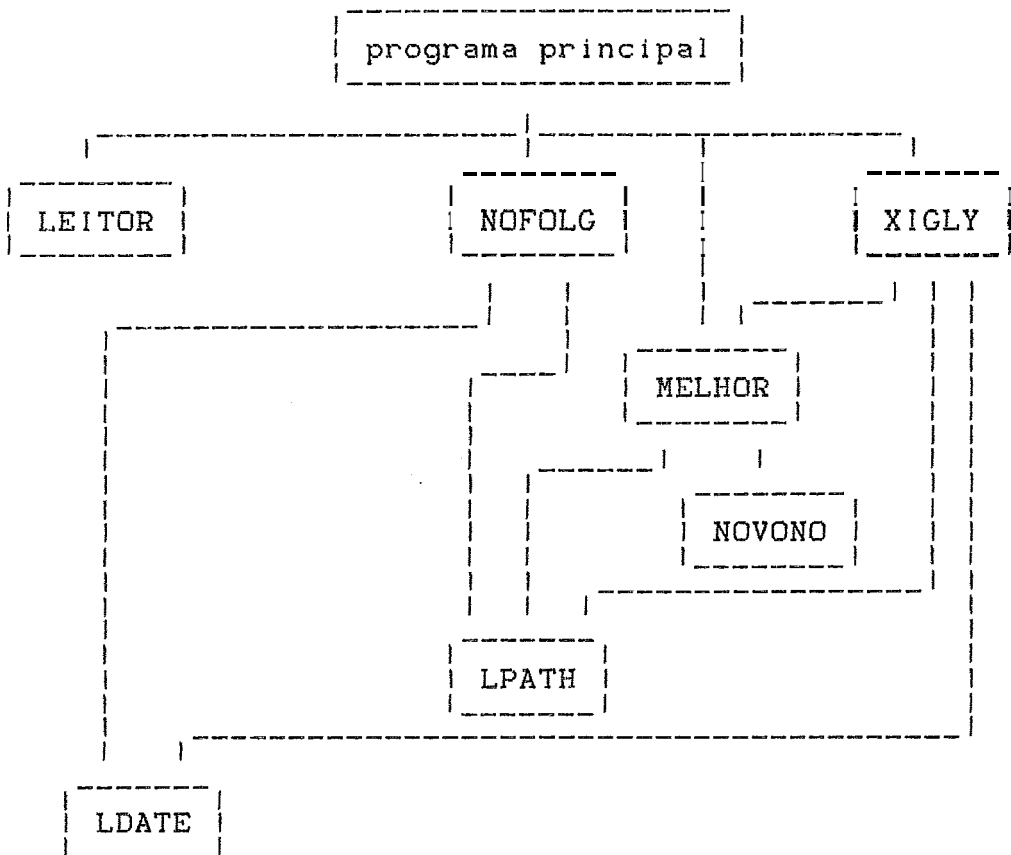


FIGURA IV.2 : Diagrama de Estrutura do Método de Compactação ao usar as folgas

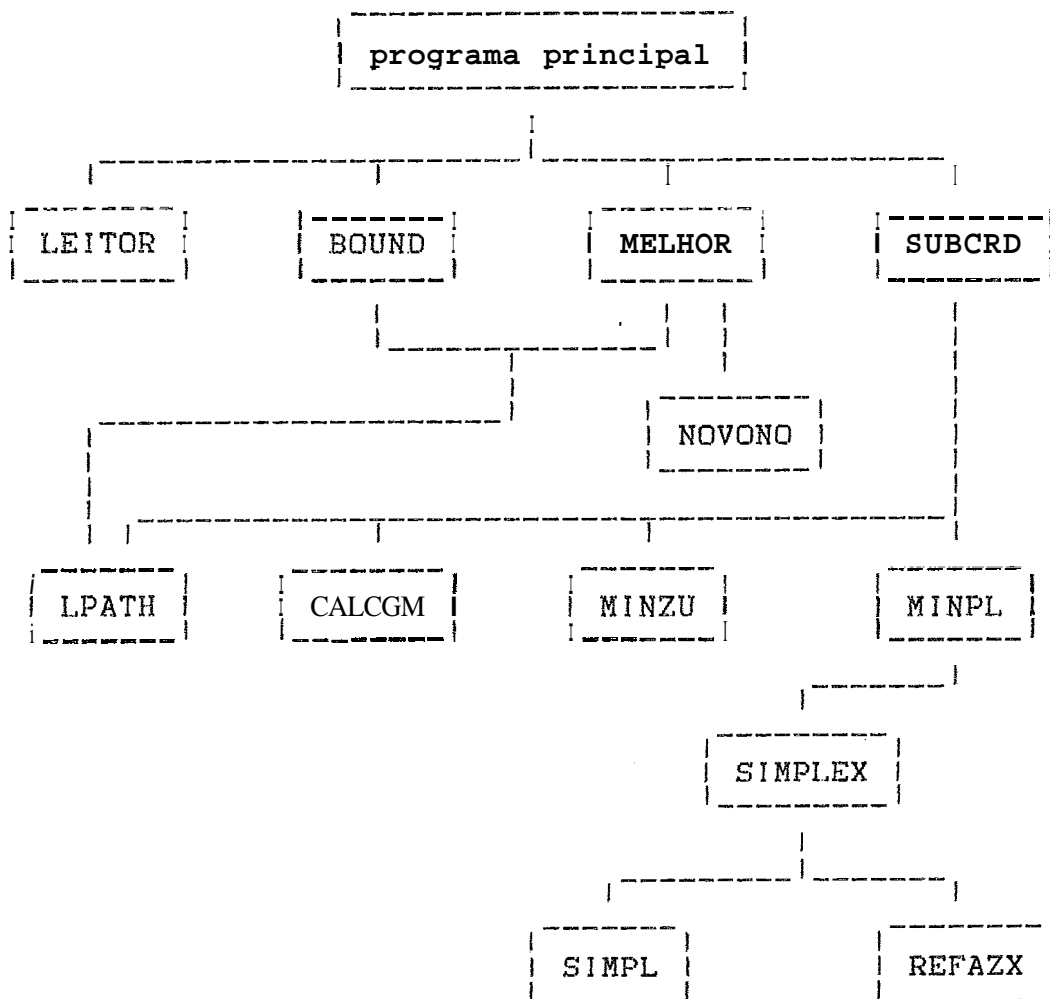


FIGURA IV.3 : Diagrama de Estrutura do Método do Sub-Gradiente

O programa principal apenas chama as subrotinas LEITOR , BOUND , MELHOR e SUBGRD . A seguir colocamos a lista de subrotinas usadas :

| | | | | | |
|--------|--------|--------|-------|---------|--------|
| LEITOR | BOUND | MELHOR | LPATH | NOVONO | SUBGRD |
| REFAZX | CALCGM | HINZU | MINPL | SIMPLEX | SIMPL |
| NOFOLG | XIGLY | LDATE | CALCM | | |

A seguir a lista das funções usadas :

| | | |
|-------|--------|--------|
| QNORM | CONSTE | LAMBDA |
|-------|--------|--------|

Colocamos a seguir , uma descrição sumária das subrotinas e funções , e de quais subrotinas são chamadas por cada uma delas :

Subrotinas :

LEITOR : Esta subrotina lê os dados do arquivo ARQXY . Ela usa a metade da dimensão das variáveis para X e a segunda metade para Y .

Não chama nenhuma subrotina .

BOUND : Obtem uma cota inferior para a compactação. Depois usando o algoritmo da seção II.6.2.1 , acha uma solução inicial que fornece uma cota superior para a compactação .

Chama a subrotina LPATH .

MELHOR : Tem como entrada de dados o vetor NO (vetor das variáveis 0-1) . Usando o algoritmo que descrevemos na seção II.6.1 , a subrotina devolve na vetor MO , o melhor vetor encontrado .

Chama as subrotinas LPATH e NOVONO .

LPATH : O nome desta subrotina é a abreviação de "Longest Path" . A subrotina LPATH usa duas vezes o algoritmo ACML descrito na seção II.5 . Uma vez o algoritmo ACML é usado para calcular os valores de X , e uma outra vez para calcular os de Y . Isto é feito para um dado vetor NO (vetor das variáveis 0-1) .

Não chama nenhuma subrotina .

NOVONO : Serve apenas para modificar o vetor NO , a partir de um NO dado . Modifica a coordenada NULTCO

(número da última coordenada já modificada) do vetor NO e a coordenada seguinte .

Não chama nenhuma subrotina .

SUBGRD : Usa o método do sub-gradiente para obter uma cota inferior para o problema de compactação .

Chama as subrotinas EPATH , CALCGM , MINZU e MIMPL .

Usa as funções CONSTE , LAMBDA e QNORM .

REFAZX : Após a execução da subrotina SIMPLEX , esta subrotina re-calcula as coordenadas do vetor X ou do vetor Y , que é a solução do problema de programação linear . Ou melhor , faz a ligação entre as subrotinas SUBGRD e SIMPLEX .

Não chama nenhuma subrotina .

CALCGM : Calcula as coordenadas dos vetores GAMAX e GAMAY, ou melhor , do vetor $W(X,Y,T)$ (veja na seção III.3 a proposição III.3), onde T é o vetor NO.

Não chama nenhuma subrotina .

MINZU : Resolve o problema III.24 , obtendo os valores de NO (na seção III.4 este vetor é denominado T).

Não chama nenhuma subrotina .

MINPL : Prepara as matrizes para resolver o problema III.23 ou o problema III.22 , usando o método simplex (ver subrotina SIMPLEX) .

Chama a subrotina SIMPLEX .

SIMPLEX: Inicializa o método simplex revisado . Esta subrotina é uma adaptação do programa do livro de MACULAN & PEREIRA [17] .

Chama as subrotinas SIMPL e REFAZX .

SIMPL : Aplica o método simplex revisado para problemas de programação linear . Adaptado do livro de **MACULAN & PEREIRA C173** .

Não chama nenhuma subrotina .

NOFOLG : Obtém uma solução inicial usando as folgas dos vetores **X** e **Y** calculadas nas subrotinas **LDATE** . Usa o algoritmo da seção 11.6.2.2 . Ela substitui a subrotina **BOUND** .

Chama as subrotinas **LPATH** e **LDATE** .

XIGLY : Após ter sido obtido um ótimo local usando a subrotina **MELHOR** , esta subrotina procura melhorar a compactação , tentando fazer $x_n x_n = y_n y_n$, ou se isto não for possível , os dois valores mais próximos . A subrotina usa as folgas para isto .

Chama as subrotinas **LPATH** , **LDATE** e **MELHOR** .

LDATE : Devolve no vetor **XL** a "ultima data" de **X** (veja seção 11.5.2) . Assim as folgas de **X** são calculadas fazendo a diferença $XL - X$.

Não chama nenhuma subrotina .

CALCM : Calcula as coordenadas dos vetores **Mx** e **My** . O algoritmo que calcula esses dois vetores , está na seção IV.4.2.1 .

Não chama nenhuma subrotina .

Funções :

QNORM : Calcula a quadrado da norma Euclidiana de um vetor dado (**GAMAX,GAMAY**) , ambos com **H** coordenadas .

CONSTE : Esta função calcula o valor da constante **C** da seção III.4 .

LAMBDA : Calcula o valor do passo do algoritmo da sub-gradiente (veja seção III.5) .

IV.3 Obtendo um Ótimo Local

Nesta seção vamos colocar os resultados que obtivemos, ao resolver um problema fictício com 120 variáveis 0-1 . Como já vimos na seção IV.1 , um problema deste tamanho tem mais de 10^{36} nós . Para cada nó , devemos resolver duas vezes o algoritmo ACML , que descrevemos na seção II.5 . Nu quadra 9V.1 , colocamos os dados deste problema .

| DADOS DO PROBLEMA : | | |
|---------------------------------|---|-----|
| Número de variáveis 0-1 | = | 120 |
| Número total de restrições em X | = | 373 |
| Número de coordenadas de X | = | 120 |
| Número total de restrições em Y | = | 414 |
| Número de coordenadas de Y | = | 119 |

QUADRO IV.1

Para cada nó usamos a subrotina LPATH , que calcula duas vezes o algoritmo ACML , uma para X e outra para Y . O tempo de execução desta subrotina ficou pequena . Nos testes que fizemos , sem considerarmos as variáveis 0-1 ficamos com 547 restrições em X e Y , e neste caso o tempo de CFU foi em geral , 0.0140 segundos no CYBER .

Se por outro lado considerarmos as variáveis 0-1 , o número de restrições aumenta para 667 , mas o tempo não cresce muito , ficando em 0.0150 segundos , em geral . No micro PC-XT/AT , o tempo total para esta subrotina ficou em cerca de 0.11 segundos .

No método de compactação se não usamos as variáveis 0-1 , o produto $x y$, funciona como uma cota inferior para área . Neste caso a solução é inviável , pois como dissemos as restrições das variáveis 0-1 não foram incluídas e podem ser desrespeitadas . Para este problema os valores obtidos estão no quadro IV.2 .

| | |
|-------------------------------|----------|
| UMA COTA INFERIOR : | |
| Menor valor possível para x | = 347 |
| Menor valor possível para y | = 378 |
| Cota inferior para a área | = 131166 |

QUADRO IV.2

Para resolver este problema , nós o atacamos de duas maneiras . A primeira é sem usar as folgas das variáveis , apenas fazendo variar a solução inicial . Os resultados desta primeira maneira , estão expostos na seção IV.3.1 . A segunda maneira , a qual colocamos os resultados na seção IV.3.2 , nós usamos as folgas das variáveis .

IV.3.1 Resultado Sem as Folgas

Primeiramente usamos a subrotina BOUND para procurar uma solução inicial. A melhor solução que encontramos neste caso e os tempos de execução das principais subrotinas, estão no quadro IV.3 .

| RESULTADO COM A SUBROTINA BOUND : | | | |
|--------------------------------------|------------|---|--------|
| Solução inicial : | Valor de x | = | 376 |
| | Valor de y | = | 398 |
| | Área | = | 149648 |
| Melhor solução : | Valor de x | = | 363 |
| | Valor de y | = | 394 |
| | Área | = | 143022 |
| Tempo de execução , em segundos , no | | | |
| CYBER : | BOUND | = | .271 |
| | MELHOR | = | 6.367 |
| | Total | = | 45.671 |
| Tempo total , em segundos , no | | | |
| PC-XT/AT : | BOUND | = | 1.38 |
| | MELHOR | = | 34.11 |
| | Total | = | 38.34 |

QUADRO IV.3

Com a intenção de obter uma solução melhor , do que a do quadro IV.3 , tentamos como solução inicial, fazer todas as coordenadas do vetor NO iguais a zero . Isto é, usamos inicialmente todas as restrições em X , e nenhuma em Y . Neste caso a subrotina BOUND continua no programa , mas ela não modifica a solução inicial . O resultado e os tem-

pos de execução estão no quadro IV.4 . Podemos ver que houve uma melhora do valor da área , que diminuiu de 143022 para 142784 . Porém o tempo da subrotina MELHOR , ficou 2.4 vezes maior do que quando utilizamos a subrotina BOUND .

| RESULTADO COM SOLUÇÃO INICIAL NO = 0 : | | | |
|--|------------|---|--------|
| ----- | | | |
| Solução inicial : | Valor de x | = | 435 |
| | Valor de y | = | 378 |
| | Área | = | 164430 |
| ----- | | | |
| Melhor solução : | Valor de x | = | 368 |
| | Valor de y | = | 388 |
| | Área | = | 142784 |
| ----- | | | |
| Tempo de execução , em segundos , no | | | |
| CYBER : | MELHOR | = | 15.301 |
| | Total | = | 85.131 |
| ----- | | | |

QUADRO IV.4

No caso anterior, procedemos usando somente as restrições em X . Vamos agora fazer o contrário , isto é , na tentativa de melhorar a solução encontrada , são usadas somente as restrições em Y . Ou seja , fazemos todas as coordenadas do vetor NO iguais a 1 , para esta solução inicial . O resultado bem como os tempos de execução estão no quadro IV.5 . Apesar do tempo de execução ter sido equivalente ao do caso anterior , a área não foi melhor do que a encontrada quando usamos a subrotina BOUND . Aliás esta foi a pior área que encontramos dentre as três soluções já obtidas .

| RESULTADO COM SOLUÇÃO INICIAL NO = 1 : | | | |
|--|------------|---|--------|
| Solução inicial : | Valor de x | = | 347 |
| | Valor de y | = | 465 |
| | Área | = | 161355 |
| Melhor solução : | Valor de x | = | 355 |
| | Valor de y | = | 407 |
| | Área | = | 144485 |
| Tempo de execução , em segundos , no | | | |
| CYBER : | MELHOR | = | 16.649 |
| | Total | = | 66.748 |

QUADRO IV.5

IV.3.2 Resultado Usando as Folgas

Não podemos garantir que as soluções encontradas até o momento, sejam ótimos globais. Veremos que estas são apenas ótimos locais. Como não sabemos que soluções iniciais devemos tentar para obter soluções melhores, resolvemos usar a sugestão de WATAMABE C291. Ele sugere que usemos as folgas das variáveis. Fazemos isto nas subrotinas NOFOLG e XIGLY.

Depois de usada a subrotina MELHOR, para obter um ótimo local, tentamos melhorar a solução com a subrotina XIGLY. Esta subrotina procura fazer $x = y$, ou se isto não for possível tenta tornar os valores das variáveis x e y

mais próximos . A subrotina **usa** as folgas para fazer isto . A idéia de porquê fazer isto , é geométrica . Olhando para a região viável dos pontos (x,y) (veja na figura IV.1) e para as hipérbolas $ao = x y$, ou ainda para as retas $zo = x + y$, podemos ver que qualquer uma das duas, hipérbole ou reta , que tem o menor valor de ao , é aquela que contém o ponto de coordenadas (a,b) . O ponto (a,b) é o vértice mais a esquerda e mais para baixo da região viável, que neste caso é um retângulo. Como o conjunto das soluções é discreto , a subrotina **XIGLY** não vai necessariamente , encontrar a melhor solução do problema . Assim essa subrotina é mais uma tentativa, de melhorar a solução do problema encontrada até o momento .

No quadro IV.6 colocamos o resultado obtido e os tempos de execução de algumas subrotinas , quando foram usadas as subrotinas **NOFOLG** e **XIGLY** . Podemos ver que esta é a melhor área que já obtivemos, mostrando que as soluções que encontramos até então eram apenas ótimos locais . Porém o tempo de execução cresceu bastante , em relação aos obtidos anteriormente .

Na seção IV.3.1 a melhor solução que encontramos foi fazendo , como solução inicial , todas as coordenadas do vetor **NO** iguais a **O** , aqui vamos fazer o mesmo . Para isto não usamos a subrotina **NOFOLG** , ou melhor ela será usada apenas de passagem sem alterar a solução inicial . A diferença desta seção para a seção IV.3.1 , é que aqui aplicamos depois da subrotina **MELHOR** a subrotina **XIGLY** . No quadro IV.7 colocamos os tempos e o resultado desta e-

xecução . A solução que encontramos , neste caso , foi pior do que a anterior quando usamos a subrotina NOFOLG . A solução foi a mesma encontrada sem as folgas , com a solução inicial igual , como era de se esperar . Quanto ao tempo de execução das subrotinas XIGLY e MELHOR , conjuntamente, foi praticamente o mesmo de quando usamos a subrotina NOFOLG .

| | | |
|--|------------------|---------|
| RESULTADO COM AS SUBROTINAS NOFOLG E XIGLY : | | |
| Solução inicial : | Valor de x = | 362 |
| | Valor de y = | 398 |
| | Área = | 144076 |
| Melhor solução da subrotina MELHOR : | | |
| | Valor de x = | 360 |
| | Valor de y = | 398 |
| | Área = | 143280 |
| Melhor solução da subrotina XIGLY : | | |
| | Valor de x = | 361 |
| | Valor de y = | 395 |
| | Área = | 142595 |
| Tempo de execução , em segundos , no CYBER : | | |
| | NOFOLG = | 27.330 |
| | MELHOR + XIGLY = | 20.738 |
| | Total = | 144.976 |

QUADRO IV.6

| | | | |
|--|------------------|--|--------|
| RESULTADO COM A SUBROTINA XIGLY E NO = 0 : | | | |
| Solução inicial : | Valor de x = | | 435 |
| | Valor de y = | | 378 |
| | Área = | | 164430 |
| Melhor solução da subrotina MELHOR : | | | |
| | Valor de x = | | 368 |
| | Valor de y = | | 388 |
| | Área = | | 143784 |
| Melhor solução da subrotina XIGLY : | | | |
| | Valor de x = | | 368 |
| | Valor de y = | | 388 |
| | Área = | | 142784 |
| Tempo de execução , em segundos , no CYBER : | | | |
| | MELHOR + XIGLY = | | 20.717 |
| | Total = | | 25.974 |

QUADRO IV.7

Da mesma forma que na seção anterior , onde tentamos melhorar a solução , usando a solução inicial com todas as coordenadas do vetor NO iguais a 1 , fizemos a mesmo usando as folgas das variáveis . No quadro IV.8 , colocamos o resultado . Este caso surpreendeu , pois apesar do tempo de execução das subrotinas MELHOR com a XIGLY , ter sido quase cinco vezes o tempo dos casos anteriores , obtivemos a mesma solução que encontramos usando a subrotina NOFOLG .

| | | | |
|--|------------------|--|---------|
| RESULTADO COM A SUBROTINA XIGLY E NO = 1 : | | | |
| Solução inicial : | Valor de x = | | 347 |
| | Valor de y = | | 465 |
| | Área = | | 161355 |
| Melhor solução da subrotina MELHOR : | | | |
| | Valor de x = | | 355 |
| | Valor de y = | | 407 |
| | Área = | | 144485 |
| Melhor solução da subrotina XIGLY : | | | |
| | Valor de x = | | 361 |
| | Valor de y = | | 395 |
| | Área = | | 142595 |
| Tempo de execução , em segundos , no CYBER : | | | |
| | MELHOR + XIGLY = | | 99.895 |
| | Total = | | 130.254 |

QUADRO IV.8

A surpresa vem de que a solução que já havíamos encontrado, usando somente a subrotina MELHOR , ter sido a pior dentre elas e com a subrotina XIGLY ser a melhor .

IV.4 Aplicando Relaxação Lagrangeana

Como já mencionamos na seção III.7 , quando usamos o método do subgradiente calculamos o semi-perímetro no lugar da área . No quadro IV.9 colocamos o resultado do problema

| RESULTADO COM A SUBROTINA BOUND E O SEMI-PERÍMETRO : | | | |
|--|------------------------|---|--------|
| Solução inicial : | Valor de x | = | 376 |
| | Valor de y | = | 398 |
| | Semi-perímetro | = | 774 |
| Melhor solução : | Valor de x | = | 368 |
| | Valor de y | = | 389 |
| | Semi-perímetro | = | 757 |
| | Área para esta solução | = | 143152 |
| Tempo de execução , em segundos , no CYBER : | | | |
| | BOUND | = | .239 |
| | MELHOR | = | 5.593 |
| | Total | = | 8.398 |

QUADRO IV.9

com 120 variáveis 0-1 que nos referimos na seção anterior, considerando o semi-perímetro e usando a subrotina BOUND para obter uma solução inicial .

Vamos agora calcular numericamente , o maior erro que poderemos cometer, quando usamos para este problema o semi-perímetro. Como não temos os valores de (x_0, y_0) e (x_1, y_1) , que são os ótimos do semi-perímetro e da área , respectivamente , vamos em primeiro lugar fazer uma aproximação grosseira para estes pontos . Podemos usar na equação III.30 , o ponto $(368, 389)$ nu lugar de (x_0, y_0) e $(347, 378)$ por (x_1, y_1) . Lembramos que $(a, b) = (347, 378)$ são as coorde-

nadas da cota inferior obtida para este problema (veja o quadro IV.2), e que a melhor solução que encontramos usando o semi-perímetro foi $(x_2, y_2) = (368, 389)$ (veja o quadro IV.9). Temos assim o erro absoluta

$$E < 368 \times 389 - 347 \times 378 = 11986 . \quad (\text{IV.1})$$

O erro percentual é obtido pelas equações III.34 e IV.1 ,

$$ER \times 100 < 11986 \times 100 \div 131166 < 9.2 \% .$$

Este cálculo do erro ficou muito distendido . Vamos calcular a erra usando E_1 e E_2 das equações III.31 e III.32 . Consideremos para efeito de cálculos que (x_2, y_2) é ótimo absoluto do prablema ao minimizar o semi-perímetro. Primeiramente devemos observar que $y_2 > x_2$ e logo o ponto (x_2, y_2) pertence à região da esquerda , usando a mesma denominação para as regiões da seção III.7 . Ainda mais , como $x_2 < b$ a região da direita é vazia , e assim devemos desconsiderar E_1 . Temos então que

$$E_2 = 143152 - 347 \times (757 - 347) = 882 . \quad (\text{IV.2})$$

Calculando o erro percentual , podemos usar as equações III.34 e IV.2 , obtendo

$$ER \times 100 < 882 \times 100 \div 131166 < 0.68 \% .$$

Se usássemos a área no lugar do semi-perímetro, quando aplicamos o método do sub-gradiente , o problema seria bem mais complicado . Não convém então usar a área , quando sabemos que o erro cometido , para este problema é inferior a 0.68 % .

Como o tempo de computação é muito grande para o método do sub-gradiente, usamos um problema menor do que aquele que vínhamos usando. Aplicamos este método em outros problemas também fictícios com no máximo 30 variáveis 0-1. Apesar destes problemas serem menores que o outro, um problema com 30 variáveis possui 2^{30} nós, que é um pouco maior que 10^9 . Os dados deste problema, estão no quadro IV.10.

| DADOS DO PROBLEMA COM 30 VARIÁVEIS 0-1 : | | |
|--|---|----|
| Número de variáveis 0-1 | = | 30 |
| Número total de restrições em X | = | 92 |
| Número de coordenadas de X | = | 31 |
| Número total de restrições em Y | = | 89 |
| Número de coordenadas de Y | = | 32 |

QUADRO IV.10

IV.4.1 A Cota Inferior com Um Único M

Nesta seção , colocamos os resultados que obtivemos quando aplicamos a alguns problemas , o método descrito nas seções III.4 e III.5 . Em todos estes problemas houve convergência , porém a valor da cota inferior não se alterou . Em todos eles o valor da cota inferior convergiu , mas para aquela cota inferior que já tínhamos obtido quando usamos uma das subrotinas , BOUND ou NOFOLG .

No quadro IV.11 colocamos a solução de um problema , que tem 10 variáveis 0-1 . Neste quadro estão os últimos valores encontrados , pelo método do sub-gradiente para as funções f_1 , f_2 e f_3 (veja as equações III.22 , III.23 e III.24 , respectivamente) , para a constante C e bem como para cota inferior . Colocamos os resultados quando resolvemos o problema com o valor de M grande , $M = 40$. Como os valores de x , y e da cota inferior devem ser números inteiros, os resultados do quadro IV.11 , podem ser arredondados para o menor inteiro maior ou igual que os valores encontrados .

No quadro IV.12 , colocamos o resultado de um problema com 20 variáveis 0-1 . O tipo de solução encontrada não difere muito daquela do quadro IV.11 . A convergência da cota inferior é para a mesma cota inferior obtida anteriormente .

| VALORES PARA O MÉTODO DO SUB-GRADIENTE COM M FIXO : | | | |
|---|---|-------------|--------|
| Dados do problema : | | | |
| Número de variáveis 0-1 | = | | 10 |
| Número total de restrições em X | = | | 36 |
| Número de coordenadas de X | - | | 15 |
| Número total de restrições em Y | = | | 34 |
| Número de coordenadas de Y | - | | 16 |
| Cota inferior com a subrotina BOUND : | | | |
| Valor de x | - | | 37 |
| Valor de y | - | | 38 |
| Semi-Perímetro | = | | 75 |
| Melhor solução : | | | |
| Valor de x | = | | 39 |
| Valor de y | = | | 45 |
| Semi-perímetro | = | | 84 |
| Resposta com o valor de M = 40 : | | | |
| Número de iterações | = | | 20 |
| Valor do último sk (*) | = | .00000125 | |
| Valor do último rk (*) | = | .001953 | |
| Valor da última f1 (**) | = | 36.99998003 | |
| Valor da última f2 (***) | = | 37.99999626 | |
| Valor da última f3 (****) | = | -.00024960 | |
| Valor do último C (*****) | = | -.00027706 | |
| Valor da última cota inferior | = | 74.99944963 | |
| | | | |
| Tempo de execução , em segundos , no CYBER : | | | |
| da subrotina SUBGRD | = | | 50.525 |
| | | | |
| * veja equação 1.21 : *** veja problema III.23 | | | |
| ** veja problema III.22 : **** " " III.24 | | | |
| | | | |
| ***** C é a constante da função £ na seção III.5 | | | |

| RESULTADO DO PROBLEMA COM 20 VARIÁVEIS 0-1 : | | |
|--|---|-------------------|
| Dados do problema : | | |
| Número de variáveis 0-1 | = | 20 |
| Número total de restrições em X | = | 62 |
| Número de coordenadas de X | = | 22 |
| Número total de restrições em Y | = | 57 |
| Número de coordenadas de Y | = | 23 |
| Cota inferior com a subrotina BOUND : | | |
| Valor de x | = | 66 |
| Valor de y | = | 80 |
| Semi-Perímetro | = | 146 |
| Melhor solução : | | |
| Valor de x | = | 70 |
| Valor de y | = | 80 |
| Semi-perímetro | = | 150 |
| Resposta com o valor de M = 40 : | | |
| Número de iterações | - | 40 |
| Valor do último sk | - | .0000000 |
| Valor do último rk | = | .0000002 |
| Valor da última f1 | = | 65.9999999 |
| Valor da última f2 | = | 80.0000000 |
| Valor da última f3 | - | -.0000002 |
| Valor do último C | = | -.0000000 |
| Valor da Última cota inferior | = | 145.9999998 |
| | | |
| Tempo de exec. (CYBER) da SURGRD | = | 309.727 s. |

QUADRO IV.12

Finalmente , nesta seção , apresentaremos o resultado do problema com 30 variáveis 0-1 cujos dados colocamos no quadro IV.10 . Neste problema nenhuma novidade foi acrescentada aos resultados que já havíamos exposto nos quadros anteriores , para problemas menores . O valor da cota inferior convergiu para a mesma cota que já havíamos obtido ,

usando apenas o algoritmo ACML , sem o método do sub-gradiante. Apesar do número de iterações ter sido elevado para o triplo da quantidade de variáveis 0-1 , isto é 90 , a cota inferior estacionou em 139.79267 desde a iteração de número 53 (como é uma cota inferior e esta é um número inteiro , arredondamos este valor para 140) . No quadro IV.13 colocamos o resultado deste problema .

| RESULTADO DO PROBLEMA COM 30 VARIÁVEIS 0-1 : | | |
|--|---------|-------------|
| Os dados do problema estão no quadro IV.10 | | |
| Cota inferior com a subrotina BOUND : | | |
| Valor de x | = | 59 |
| Valor de y | = | 81 |
| Semi-Perímetro | = | 140 |
| Melhor solução : | | |
| Valor de x | = | 65 |
| Valor de y | = | 85 |
| Semi-perímetro | = | 150 |
| Resposta com o valor de M = 40 : | | |
| Número de iterações | = | 90 |
| Valor do último sk | = | .0000000 |
| Valor do último rk | = | .0000000 |
| Valor da última f1 | = | 58.9977528 |
| Valor da última f2 | = | 81.0158617 |
| Valor da última f3 | = | .0000000 |
| Valor do último C | = | -.2209489 |
| Valor da última cota inferior | = | 139.7926656 |
| | | |
| Tempo de exec.(CYBER) da SUBGRD | = | 2367.378 s. |

QUADRO IV.13

IV.4.2 A Cota Inferior com o Valor de M Variável

Na seção anterior vimos que o método do sub-gradiente não foi muito proveitoso. A cota inferior, que estávamos tentando melhorar, foi a mesma que encontramos quando foi usado o algoritmo ACBL, a não ser para pequenos valores de M . Porém neste caso, não sabíamos até quando era possível diminuir o valor de M , sem modificar o problema. Pensando nisto é que resolvemos introduzir no problema, um valor de M para cada uma das restrições da tipo "ou". Com isso garantimos valores menores para M , com certeza de que as equações III.5 e III.6, sejam sempre satisfeitas.

Como teremos um valor de M para cada restrição do tipo "ou", denominaremos daqui em diante os novos valores de M por M_x e M_y . Para cada restrição do tipo "ou" em X , digamos a de número k , associamos um valor M_{xk} e para as equações tipo "ou" em Y , M_{yk} .

IV.4.2.1 Cálculo dos Valores de M_x e de M_y

Para calcular os valores de M_x e de M_y , fazemos primeiramente as coordenadas do vetor NO (o vetor das variáveis 0-1), iguais a 1. Isto é, consideramos todas as restrições da tipo "ou" em Y e não consideramos as em X . Usamos o algoritmo ACML, para este valor de NO , colocando o resultados nas vetores X e Y . Depois aplicamos outra vez o algoritmo ACML, porém agora com os valores das coordenadas do vetor NO todos iguais a 0. Neste caso estamos fazendo o oposto, usando as restrições do tipo "ou" em X e desconsiderando as em Y . O resultado é colocado nos vetores $XAUX$ e $YAUX$.

Nas coordenadas dos vetores X e $YAUX$ estão os maiores valores de X e Y , respectivamente e nas de $XAUX$ e Y os menores valores. Com isso calculamos, usando as restrições do tipo "ou" de número k ,

$$x_i - x_j + M_{xk} t_k \geq d_{ij} \quad e \quad (III.25)$$

$$y_s - y_t + M_{yk} (1 - t_k) \geq d_{st}, \quad (III.26)$$

os valores de M_x e M_y da seguinte forma:

$$M_{xk} = d_{ij} - x_i + x_{auxj} \quad e \quad (IV.3)$$

$$M_{yk} = d_{st} - y_{auxs} + y_t, \quad (IV.4)$$

onde x_i , x_{auxj} , y_{auxs} e y_t são respectivamente as coordenadas i , j , s e t dos vetores X , $XAUX$, $YAUX$ e Y .

IV.4.2.2 Resultados com M Variável

Ao adotarmos o critério de um valor de M para cada restrição houve uma melhoria, em relação à cota inferior obtida com o critério de um único M para todas as restrições. No caso de um único M , como já havíamos comentado, a cota inferior foi a mesma que encontramos com o algoritmo ACML, que é muito menos dispendioso de ser aplicado, em termos de tempo, do que o método do sub-gradiente.

Com o critério do M variável, em primeiro lugar, colocamos no quadro IV.14 o resultado que obtivemos com o problema de 10 variáveis 0-1, cujos dados estão no quadro IV.11. A melhor cota inferior que conseguimos, nesse problema até o momento é 75. Aplicando esse novo critério a cota foi ampliada para 76, já que consideramos o menor inteiro maior que a cota (uma vez que a cota é um número inteiro).

Adotamos uma regra parecida com aquela que foi usada por HELD, WOLFE e CROWDER [10], descrita na seção III.4. O que faremos foi manter rk (veja as seções III.4 e III.5) igual a 2 durante as 20 primeiras iterações, e depois dividimos os valores de rk a cada 5 iterações. Outras tentativas foram feitas, porém com essa regra, essa foi a melhor cota obtida.

| RESULTADO COM M VARIÁVEL E 10 VARIÁVEIS 0-1 : | | | |
|---|-------|---|------------|
| Os dados desse problema estão no quadro IV.11 . | | | |
| Cota inferior com a subrotina BOUND : | | | |
| Semi-Perímetro | | = | 75 |
| Melhor solução : Semi-perímetro = 84 | | | |
| Resposta com o valor de M variável : | | | |
| Número de iterações | | = | 70 |
| Número da primeira iteração com | | | |
| a cota inferior maior que 75 | | = | 20 |
| Valor do último sk | | = | .0000075 |
| Valor do último rk | | = | .0019531 |
| Valor da última fl | | = | 34.949176 |
| Valor da última f2 | | = | 38.384674 |
| Valor da última f3 | | = | -.004073 |
| Valor do último C | | = | 2.625246 |
| Valor da última cota inferior | | = | 75.9550231 |
| Valor da melhor cota inferior | | = | 75.955464 |
| | | | |
| Tempo de exec.(CYBER) da SUBGRD | | = | 208.749 s. |

QUADRO IV.14

Vamos apresentar agora o resultado , tom M variável, no problema com 30 variáveis 0-1 . Os dados desse problema estão quadro IV.10 . No quadro IV.13 colocamos o resultado para um única valor de M . Para M variável o resultado está no quadro IV.15 .

| | | |
|--|------------------|-------------|
| RESULTADO COM M VARIÁVEL E 30 VARIÁVEIS 0-1 : | | |
| Os dados desse problema estão no quadro IV.10 . | | |
| No quadra IV.13 está o resultado com um único M . | | |
| Cota inferior com a subrotina BOUND : | | |
| Semi-Perímetro | - | 140 |
| Melhor solução : | Semi-perímetro = | 150 |
| Resposta com o valor de M variável : | | |
| Número de iterações | = | 210 |
| Número da primeira iteração com a cota inferior maior que 140 | = | 87 |
| Número da primeira iteração com o valor da melhor cota | - | 172 |
| Valor do último sk | = | .0000000 |
| Valor do último rk | = | .0000000 |
| Valor da última f1 | = | 58.468038 |
| Valor da última f2 | = | 81.000000 |
| Valor da última f3 | = | -.095813 |
| Valor do último C | = | .751869 |
| Valor da última cota inferior | = | 140.124094 |
| Valor da melhor cota inferior | = | 140.124094 |
| | | |
| Tempo de exec. (CYBER) da SUBGRD | = | 7289.721 s. |
| Tempo médio de cada iteração | = | 34.71 s. |

QUADRO IV.15

Da mesma forma que no problema com 10 variáveis 0-1 , nesse também houve uma pequena melhora . A cota inferior , que com o algoritmo ACML foi 140 , nesse caso obtivemos 141 .

Adotamos também uma regra parecida com a de HELD , WOLFE e CROWDER [10] . O valor de rk ficou igual a 2

durante as 60 primeiras iterações , e depois foi dividido por dois a cada 5 iterações .

IV.4.2.3 Adotando Outra Regra

Observamos que durante a resolução de problemas que descrevemos na seção IV.4.2.2 , os valores das cotas inferiores em cada iteração , crescem em geral quando dividimos o valor de r_k . Porém essa divisão implica em valores menores para os elementos da sequência s_k . E pequenos valores de s_k implicam em ter pequenas alterações da cota inferior . Assim se por um lado , dividir r_k ao meio , como propuseram HELD , WOLFE e CROWDER [10] , aumenta a cota inferior , em cada divisão s_k diminui e a cota inferior cresce menos .

A forma que encontramos para que a cota inferior aumente e o valor de s_k não diminua tão rápido, é uma regra um pouco diferente da adotada por esses três autores . Deixamos r_k igual a 2 durante as p primeiras iterações , onde p é o número de variáveis 0-1 . Depois das p primeiras iterações , sempre que o problema obtiver uma cota inferior melhor em uma iteração , dividimos r_k por q_1 , onde

$$1 < q_1 < 2 \quad . \quad (IV.5)$$

No problema com 30 variáveis 0-1 , se em p iterações não houver melhora no valor da cota inferior , dividimos em cada iteração r_k por q_2 , onde

$$1 < q_2 < q_1 < 2 \quad . \quad (IV.6)$$

O critério de parada , é passar 2 p iterações sem que a cota inferior seja melhorada de um certo valor (em geral, adotamos para o problema de 10 variáveis 0-1 o valor 0.000001 e para o de 30 variáveis 0.0001) .

Notamos também que o maior gasto , em termos de tempo de computação , era resolver os dois métodos SIMPLEX em cada iteração , um para o problema de minimizar f_1 e o outro para minimizar f_2 . Para melhorar esse tempo observamos que ao resolver os problemas de f_1 e f_2 , as restrições são em todas iterações , dadas pelas equações III.1 e III.2 , respectivamente . Essas equações permanecem inalteradas durante o método . As alterações em cada iteração , são apenas nos valores de f_1 e f_2 . Assim em todas iterações , no final da primeira Fase do ÇIMPLEX as matrizes inversas e as variáveis básicas são sempre iguais , logicamente uma para X e outra para Y . Na segunda iteração guardamos os valores das duas inversas e das duas bases , já que na primeira fase usamos o algoritmo ACML no lugar do SIMPLEX . Com isso todas as primeiras fases dos algoritmos SIMPLEX , a partir da terceira iteração , foram executadas apenas uma vez . Como o maior tempo gasto em cada iteração ficava por conta da primeira fase , este foi na prática dividido por aproximadamente 10 .

Os problemas que resolvemos com 10 variáveis 0-1 , nessa seção , graças a esta redução de tempo foram resolvidos em um micro computador da linha PC-XT/AT e os de 30

variáveis em um CYEER 170/835 .

No problema com 10 variáveis conseguimos , com **essa** nova regra , uma cota inferior melhor do que a obtida anteriormente com a regra da seção IV.4.2.2 . O valor da cota aumentou de 74 para 77 . Usamos vários valores para q_1 , dada pela equação IV.5 . Notamos que se q_1 é "pequeno" (isso significando próxima de 1) , o algoritmo se perde,

| | |
|--|--------------|
| MELHOR COTA INFERIOR DO PROBLEMA DE 10 VARIÁVEIS : | |
| Os dados desse problema estão no quadro IV.11 . | |
| Cota inferior com a subrotina BOUND : | |
| Semi-Perímetro | = 75 |
| Melhor solução : Semi-perímetro | = 84 |
| Resposta com $q_1 = 1.174$: | |
| Primeira iteração com a cota maior que 75 : | |
| Iteração | = 15 |
| sk | = .00729979 |
| rk | = 1.05282800 |
| Primeira iteração com a cota maior que 76 : | |
| Iteração | = 31 |
| sk | = .00279590 |
| rk | = .47208160 |
| última iteração : | |
| Iteração | = 222 |
| sk | = .00000018 |
| rk | = .00002263 |
| Melhor cota | = 7'6.499820 |
| | |
| Tempo de exec.(PC-XT\AT) da SUDGRB | = 564.250 s. |
| Tempo médio de cada iteração | = 2.54 s. |

QUADRO IV.16

se por outro lado q_1 é "grande" (próximo de 2), a cota inferior obtida não é tão boa . A melhor cota que obtivemos foi com $q_1 = 1.174$, onde esse número foi encontrado experimentalmente. O resultado para esse valor de q_1 está no quadro IV.16 .

Essa regra aplicada ao problema com 30 variáveis 0-1, cujos dados estão no quadro IV.10 , não foi suficiente para ultrapassar a cota inferior obtida na seção IV.4.2.2 . Apesar da melhor cota que encontramos com essa nova regra ter

| | | |
|--|------------------|-------------|
| MELHOR COTA INFERIOR DO PROBLEMA DE 30 VARIÁVEIS : | | |
| Os dados desse problema estão no quadro IV.10 . | | |
| Cota inferior com a subrotina BOUND : | | |
| Semi-Perímetro | = | 140 |
| Melhor solução : | Semi-perímetro = | 150 |
| Resposta com $q_1 = 1.07$ e $q_2 = 1.03$: | | |
| Primeira iteração com a cota maior que 140 : | | |
| Iteração | = | 281 |
| sk | = | .00006277 |
| rk | = | .12755981 |
| última iteração : | | |
| Iteração | = | 1226 |
| sk | = | .00000005 |
| rk | = | .00005378 |
| Melhor cota | = | 140.191520 |
| | | |
| Tempo de exec.(CYBER) da SUBGRD | = | 2998.264 s. |
| Tempo médio de cada iteração | = | 2.45 s. |

QUADRO IV.17

sido superior , quando consideramos o menor inteiro maior que as cotas obtidas, as duas foram iguais a 141 . No quadro IV.17 colocamos o melhor resultado que achamos . Para esse resultado usamos os valores $q_1 = 1.07$ e $q_2 = 1.03$, onde q_1 e q_2 são como nas equações IV.14 e IV.15 , respectivamente .

IV.5 Resolvendo Problemas Reais

Nesta seção , estão os resultados da compactação dos layouts de dois circuitos que aparecem na literatura . Na tese do WATANABE E291 é feita a compactação desses dois circuitos , denominados Scell e T-Flip-Flop .

As restrições dos dois problemas para compactar tais circuitos , foram extraídas manualmente . A extração manual ocasionou um número menor de restrições do que as que foram encontradas por WATANABE C293 . Ao gerar restrições automaticamente aparecem restrições em excesso . Este número maior de restrições é devido a comparação de elementos distantes . Quando as restrições são geradas manualmente essas comparações de elementos distantes são facilmente descartadas . Por outro lado , só foi possível gerar restrições para problemas que não são muito grandes , devido a um número excessivo de escolhas para as variáveis 0-1 .

Também os pontos onde os fios devem ser quebradas , isto é , devem mudar de horizontais para verticais e vice-versa , foram escolhidos manualmente .

NO quadro IV.18 estão os dados do problema e o resultado da compactação do leiaute do circuito Scell .

| Scell | |
|---|--------|
| Dados da problema : | |
| Número de variáveis 0-1 | = 1 |
| Número total de restrições em X | = 16 |
| Número de coordenadas de X | = 12 |
| Número total de restrições em Y | = 18 |
| Número de coordenadas de Y | = 13 |
| Resultado da compactação : | |
| Valor da cota inferior para a área .. | = 432 |
| Melhor área encontrada | = 432 |
| Valor de X | = 18 |
| Valor de Y | = 24 |
| ***** A solução encontrada é ótima ***** | |
| Tempo de execução em segundos , no PC-XT/AT : | |
| BOUND | = 1.15 |
| MELHOR | = 1.70 |
| total | = 3.80 |

QUADRO IV.18

Nas figuras IV.4 e IV.5 estão o diagrama de varetas do Scell e seu leiaute compactado , respectivamente .

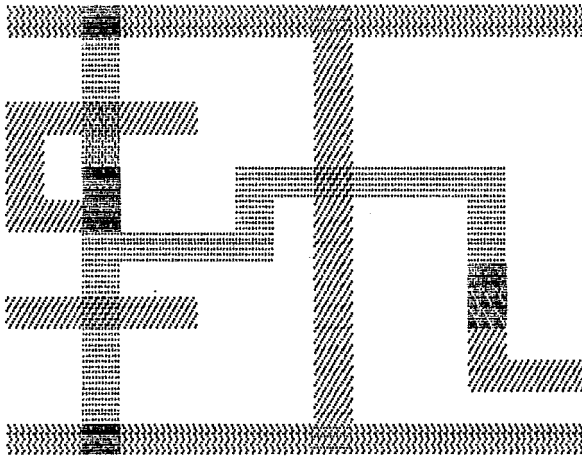


FIGURA IV.4 : Diagrama de varetas
do circuito SCCELL

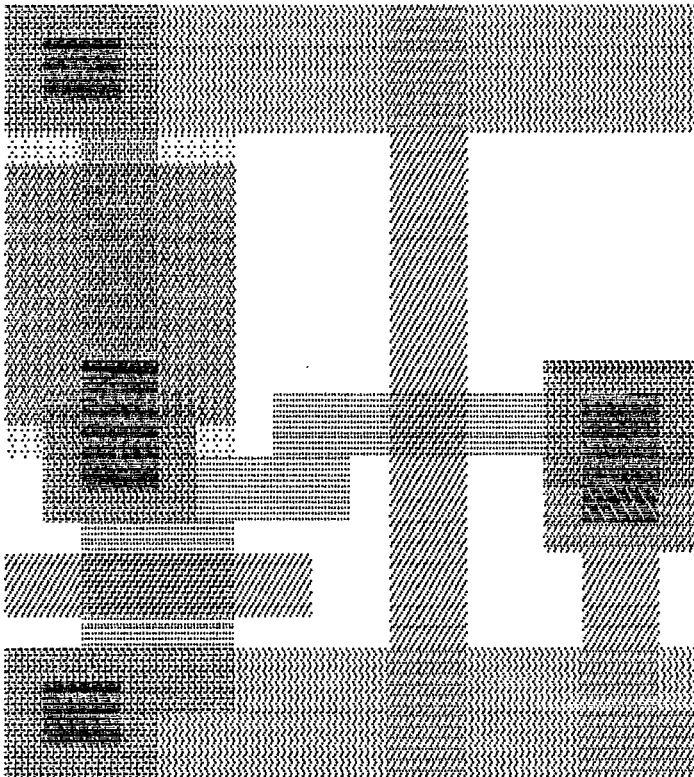


FIGURA IV.5 : Leiaute final do
circuito SCCELL

O resultado da compactação do leiaute do circuito integrado T-Flip-Flop foi um pouco melhor do que o que foi encontrado por WATANABE [29] . O valor da melhor área que obtivemos foi 1938 com o valor de X igual a 38 e o de Y igual a 51 . Enquanto que WATANARE C291 encontrou os valores 38 e 53 , **respectivamente** , fornecendo uma área

| T-Flip-Flop | |
|--|---------|
| Dados do problema : | |
| Número de variáveis 0-1 | = 25 |
| Número total de restrições em X | = 75 |
| Número de coordenadas de X | = 41 |
| Número total de restrições em Y | = 79 |
| Número de coordenadas de Y | = 39 |
| Resultado da compactação : | |
| Valor da cota inferior para a área .. | = 1862 |
| Melhor área encontrada | = 1938 |
| Valor de X | = 38 |
| Valor de Y | = 51 |
| Distância da solução ótima , Inferior a 4 % | |
| Tempo de execução em segundos , no PC-XT/AT : | |
| BOUND | = 2.31 |
| MELHOR | = 9.56 |
| total | = 14.78 |

de 2014 . Este resultado é 3.9 % maior .

No quadro IV.19 estão os dados da problema e o resultado da compactação .

Na figura IV.6 está o um diagrama de varetas para o circuito T-Flip-Flop e na Figura IV.7 o leiaute do circuito compactado .

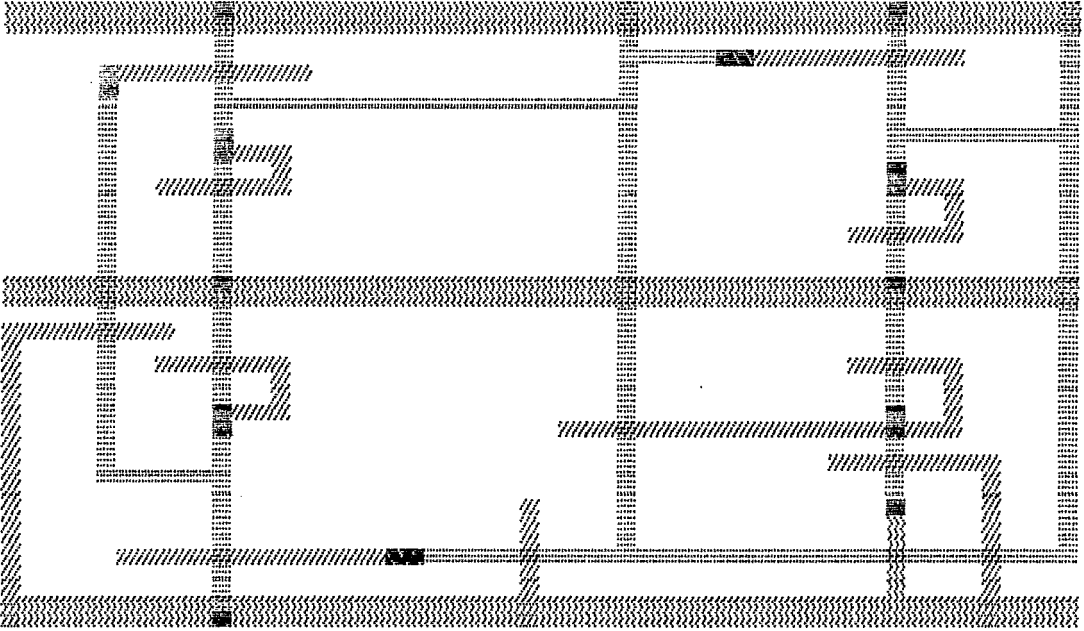


FIGURA IV.6 : Diagrama de varetas
do circuito TFF

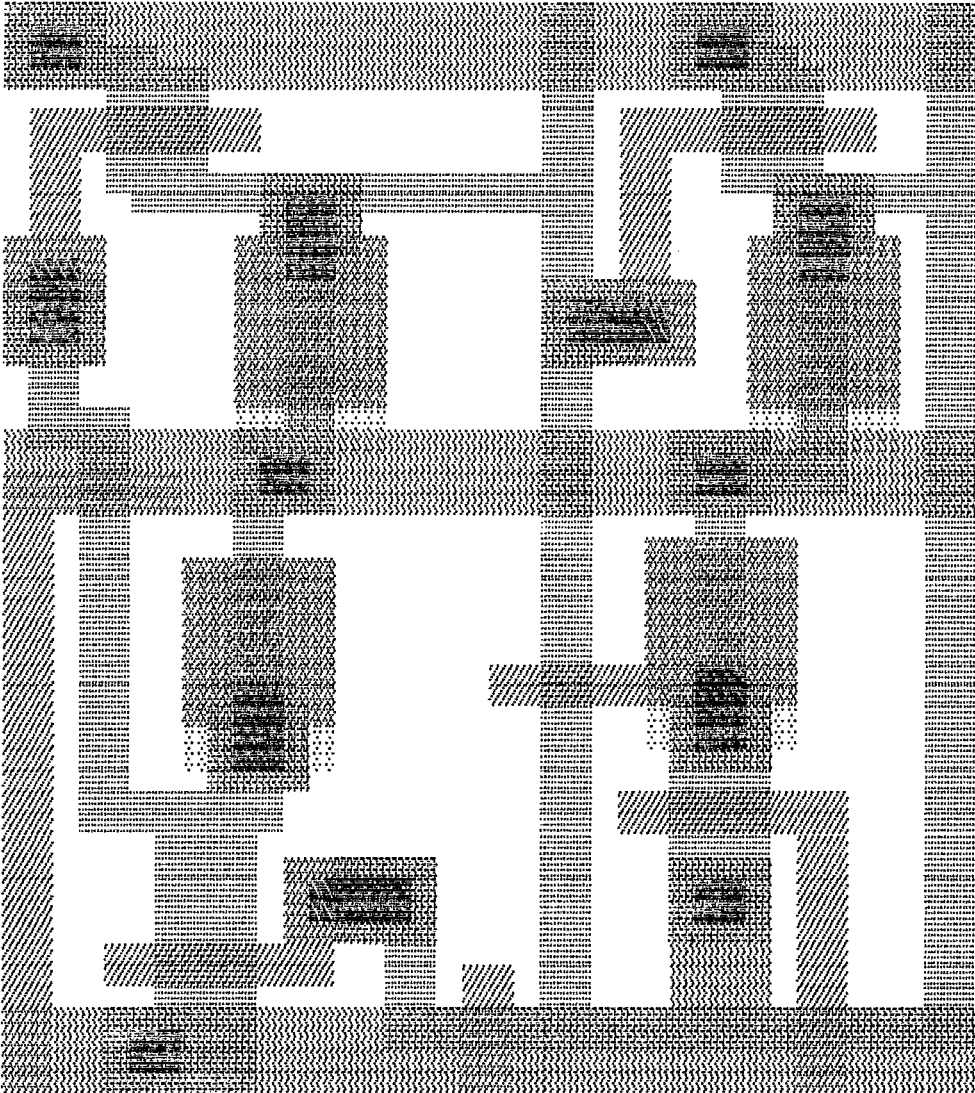


FIGURA IV.7 : Leiaute final do
circuito TFF

CAPÍTULO VCONCLUSÃO

No que se refere ao desenvolvimento da informática , podemos ressaltar como fatores relevantes o desenho e a fabricação dos circuitos integrados , com um aprimoramento cada dia maior . O número sempre maior de elementos em um mesmo "chip" , tem contribuído para um desempenho maior dos computadores , cada vez ocupando espaços menores . E para que isso seja possível , é bastante importante que se faça durante o projeto dos circuitos , a compactação do leiaute destes .

Como já mencionamos nos capítulos anteriores , tem-se empregado várias técnicas de fazer esta compactação . Essas técnicas podem ser vistas consultando-se a literatura em geral sobre esse assunto . A maioria das técnicas tem usado métodos uni-dimensionais . Uma novidade neste trabalho é o uso de um método bi-dimensional levando em conta as folgas das variáveis . Isto nos permitiu obter soluções melhores para os problemas que resolvemos na seção IV.3.2 . Nesse nosso primeiro trabalho usando as folgas , já foi possível ver que essas podem melhorar as soluções . Nos exemplos que resolvemos no capítulo IV as áreas dos leiautes dos circuitos ficaram menores ao usar as folgas .

Uma outra novidade para o problema de compactação, foi a utilização de relaxação lagrangeana. No princípio do trabalho parecia que a aplicação de relaxação lagrangeana, neste problema, não melhoraria em nada as cotas inferiores que já havíamos obtido. Porém quando empregamos um valor de M para cada restrição, conseguimos melhorar a cota inferior (ver seções III.6 e IV.4.2).

É interessante ressaltar ainda que foi possível resolver uma série de vezes cada exemplo, aplicando regras diferentes, graças a uma diminuição substancial no tempo de computação. Esta redução de tempo foi devida ao armazenamento de algumas matrizes, no final das primeiras fases nos métodos SIMPLEX. Notou-se que as primeiras fases do SIMPLEX em todas as iterações no método do sub-gradiente, são sempre as mesmas. Armazenando-se os resultados destas primeiras fases, uma para X e outra para Y , ao resolver o método SIMPLEX há necessidade de fazer as primeiras fases uma única vez. Como a fase que contribui com mais tempo é exatamente a primeira, houve uma sedução muito grande no tempo de computação.

No que se refere ao problema de otimização, a aplicação de valores variáveis para rk (ver seção IV.4.2.3), não só permitiu encontrar uma cota inferior melhor, como também melhorou a convergência do método.

Considerando o semi-perímetro no problema com 10 variáveis 0-1, obtivemos como melhor solução o valor 84. A cota inferior que encontramos somente com a subrotina

BOUND foi 75 (ver quadro IV.14) . O erro percentual E que podemos cometer , quando encontramos o valor 84 como um ótimo local e não como ótimo absoluto , com a subrotina MELHOR , é tal que .

$$E < (84 - 75) \div 75 \times 100 = 12 \% .$$

Se por outro lado usamos a cota inferior , achada com o método do sub-gradiente , que foi 77 (ver quadro IV.16) , vemos que o erro cometido deve ser um pouco menor

$$E < (84 - 77) \div 77 \times 100 < 9.1 \% .$$

Calculando este erro para o problema de 30 variáveis 0-1 , do qual encontramos como cotas inferiores os valores 140 com a subrotina BOUND e 141 com o método do sub-gradiente (ver quadro IV.171 , temos no primeiro caso um erro menor que 7.2 % e no segunda 6.4 % .

A decisão sobre que método deve ser usado , isto é , se é necessário usar as folgas ou o método do sub-gradiente , deve partir do projetista do circuito . A precisão do problema é um fator de peso nesta decisão .

Dois exemplos que estão propostos na tese de WATANABE [29] , também foram resolvidos aqui neste trabalho . São eles os circuitos SCELL e TFF (ver seção IV.55 . A contactação do layout de SCELL forneceu a mesma área que a obtida por WATANABE [29]. Para este exemplo foi encontrada a solução ótima . No caso do circuito TFF , a área obtida por WATANABE [29] foi pior do que a que encontramos aqui .

Como já dissemos , isso provavelmente se deve ao fato de termos gerado as restrições manualmente .

Um gerador automático de restrições pode criar restrições a mais . Por outro lado pode gerar restrições para **problemas maiores , a que fica impossível de ser feito manualmente** . Além disso , um gerador de restrições pode ter associado a ele um gerador automático de pontos de quebra ("jog points") . Estes pontos de quebra permitem a quebra dos fios do circuito , diminuindo sua área. Novos pontos de quebra introduzidos , por exemplo no circuito TFF , certamente permitiriam diminuir sua área. Estes dois geradores , de restrições e de pontos de quebra , ficam como sugestões para outros trabalhos .

É possível que com a continuação do estudo sobre a aplicação das folgas das variáveis no método de compactação venha melhorar ainda mais as soluções .

Quanto ao método do sub-gradiente , poderíamos propor para futuras pesquisas um estudo mais profundo na direção da variação dos valores de r_k em função da sequência z_k no sentido de melhorar sua convergência .

REFERÊNCIAS BIBLIOGRÁFICAS

- [O1] - AGMON, S., "The Relaxation Method for Linear Inequalities" , Canadian Journal of Mathematics 4 , p. 382-392 , 1954 .
- C023 - APOSTOL, T.M., "Calculus" , Ginn Blaisdell, Volume 11 , segunda edição , 1969.
- C033 - EVERETT, H. , "Generalized Lagrange Multiplier: Method for Solving Problems of Optimum Allocation of Resources" , Operations Research , vol. 11 , p. 399-417 , 1963 .
- [O4] - FISHER, M.L. , "The Lagrangian Relaxation Method for Solving Integer Programming Problems" , Management Science , vol. 27 , p. 1-18 , 1981 .
- [O5] - FISHER, M.L. , NORTHUP, W.D. and SHAPIRO, J.F. , "Using Duality to Solve Discrete Optimization Problems : Theory and Computational Experience" , Mathematical Programming Study 3 ; p. 56-94 , 1975 .
- E063 - GEOFFRION, A.M., "Lagrangian Relaxation and its Use in Integer Programming" , Mathematical Programming Study 2 , p. 82-114 , 1974 .
- C073 - GONDRAN, M. and MINOUX, M., "Graphs and Algorithms" , Wiley , 1984.

- C081 - GONZAGA,C. , "Busca de Caminhos em Grafos e Aplicações" , Anais da 1a Reunião de Matemática Aplicada , 1978.
- [091] - HELD,M.and KARP,R.M., "The Traveling Salesman Problem and Minimum Spanning Trees" , Operations research , vol. 18 , p. 1138-1162 , 1970 .
- [101] - HELD,M. , WOLFE,P. and CROWDER,H.P. , "Validation of Subgradient Optmization" , Mathematical Programming 6 , p. 62-88 , 1974 .
- [111] - HIRSCHFELD,H. , "Planejamento com PERT-CPM e Análise do Desempenho" , Atlas , 1982.
- [121] - KEDEM,G. and WATANABE,H. , "Graph-Optmization Techniques for IC Layout and Compaction" , IEEE Trans. on Computer Aided Design , Vol. CAD-3 , No.1 , p.12-20 , Januasy 1984 .
- L131 - KIRKPATRICK,S. , GELLAT,C.D.Jr. and VECCHI,M.P. , "Optmization by Simulated Annealing" , Science , vol. 220 , Mo.4598 , p.671-680 , May 1983 .
- C143 - LASDON,L.S., "Optimization Theory for Large Systems", Macmillan Series in Operations Reseach , 1970 .
- [15] - LIAO,Y.Z. and WONG,C.K. , "An Algorithm to Compact a VLSI Symbolic Layout with Mixed Constraints" , IEEE Trans. sn Computei- Aided Design of Circuits and Systems , Vol. CAD-2 No.2 , p.62-69 , April 1983 .

- [16] - MACULAN FILHO, N. , CAMPELLO, R.E. e LOPES, L.B.R. ,
"Relaxação Lagrangeana em Programação Inteira" ,
VII Congresso Nacional de Matemática Aplicada e
Computacional , Unicamp , 1984 .
- C173 - MACULAN FILHO, N. , PEREIRA, M.V.F. , "Programação
Linear" , São Paulo , Editora Atlas S.A. , 1980 .
- [18] - MEAD, C. and CONWAY, L., "Introduction to VLSI System",
Reading MA : Addison-Wesley , 1980.
- [19] - MLYNSKI, D.A. and SUNG, C. , "Layout Compaction" ,
Layout Design and Verification , T.Ohtsuki (Editor),
North Holland , p.199-235 , 1986.
- C201 - MOTZKIN, T. and SCHOENBERG, I.J. , "The Relaxation
Method for Linear Inequalities", Canadian Journal of
Mathematics h , p. 393-404 , 1954 .
- [21] - OTTEN, R.H.J.M. and LIER, M.C. , "Automatic IC-Layout:
The Geometry of The Islands" , Proceedings 1975 IEEE
International Symposium on Circuits and Systems ,
Newton , Mass. , p. 231-234 , 1975 .
- [22] - POLYAK, B.T., "Minimization of Unsmooth Functionals",
USSR Computational Mathematics and Mathematical
Physics 9 , p. 509-521 , 1969 .
- 6233 - ROCKAFFELAR, R.T. , "Convex Analysis" , Princeton
University Press , 1970 .

- C243 - SASTRY,S. and PARKER,A. , "The Complexity of Two-Dimensional Compaction of VLSI Layouts" , Proc. IEEE International Conference on Circuits and Computers , New York , p. 402-406 , 1982 .
- C253 - SCHLAG,M. , LIAO,Y.Z. and WONG,C.K. , "An Algorithm for Two-Dimensional Compaction of VLSI Layout" , INTEGRATION , VLSI Journal 1 , p. 179-209 , 1983 .
- C263 - SHAPIRO,J.F. , "Generalized Lagrange Multipliers in Integer Programming" , Operations research , vol. 19 p. 68-79 , 1971 .
- C271 - SHAPIRO,J.F. , "A Survey of Lagrangian Techniques for Discrete Optimization" , Annals of Discrete Mathematics 5 , p. 133-138 , 1979 .
- C281 - ULLMAN,J. , "Computational Aspects of VLSI" , Computers Science Press , 1984 .
- C295 - WATANABE,H. , "IC Layout Generation and Compaction Using Mathematical Optimization" , PH.D. Thesis , University of Rochester , N.Y. , 1984 .
- C303 - WILLIAMS,J.D. , "STICKS - A Graphical Compiler for High Level LSI Design" , Proc. AFIPS , Vol.47 , p.289-295 , 1978.
- C311 - WOLF,W. , MATHEWS,R. , NEWKIRK,J. and DUTTON,R. "Two-Dimensional Compaction Strategies" , Proc. IEEE International Conference on Computer Aided Design , p.90-91 , 1983 .

- [32] - WONG,C.K. , "An Optimal Two-Dimensional Compaction Scheme" , VLSI : Algorithms and Architectures , P.Bertolazzi and F.Luccio (Editores), North-Holland, p.205-220 , 1985 .

APÊNDICE

A.1 Arquivos para Entrada de Dados

Descreveremos nesta seção, os arquivos para entrada de dados do gerador de restrições descrito na seção A.3 .

Um dos arquivos contém a matriz de tecnologia , indicando as distâncias mínimas entre elementos e entre os grupos , e a largura mínima dos elementos .

O segundo arquivo contém a descrição dos elementos do circuito . Deve conter informações sobre o tipo de cada elemento e sua posição no circuito . As posições x_i e y_i devem estar na matriz $A(i,j)$. A coluna 1 e a linha 1 da matriz A devem ser nulas (não há elementos do circuito nas posições $(i,1)$ e $(j,1)$ da matriz A) .

Finalmente um terceiro arquivo deve conter a descrição e posicionamento das esquinas do circuito . Os tipos possíveis de esquinas são aqueles que vimos no capítulo II .

Devemos também ter conhecimento sobre os elementos que tem o mesmo potencial elétrico . Entre esses elementos não é necessário colocar nenhuma restrição para mantê-los distantes .

A.2 Arquivo Gerado

Este arquivo contém informações sobre as distâncias mínimas entre os elementos do circuito. Estas distâncias são as horizontais (X) e as verticais (Y). Como já vimos **essas** restrições são de dois tipos: "e" e "ou". Segue a descrição do formato do arquivo a ser gerado:

| linhas | conteúdo |
|---------------------------|--|
| 1 | número de restrições tipo "ou" , |
| 2 | número de : variáveis (nx) em X , restrições (mx) em X , variáveis (ny) em Y e restrições (my) em Y . |
| 3 até mx + 2 | restrições em X das tipos "e" e "ou" (estas linhas devem vir ordenadas) , |
| mx + 3 até mx + my + 2 | restrições em Y dos tipos "e" e "ou" (estas linhas devem vir ordenadas) . |

| Formato das linhas : | linha | 1 | formato | I4 |
|----------------------|-------|-------------|---------|-------------|
| | " | 2 | " | 4I4 |
| | " | 3 em diante | " | 2I4,F4.0,I4 |

Exemplos : 1) A restrição $X7 - X5 \geq 3$ tipo "e" deve estar escrita na parte do arquivo devido às restrições horizontais na seguinte forma :
bbb7bbb5bb3 .

2) Cada uma das componentes do par de restrições do tipo "ou" de número 20 , $X_{11} - X_9 \geq 4$ ou $Y_9 - Y_6 \geq 5$, deve ficar na parte do arquivo para X e para Y , respectivamente . E os formatos das linhas são:

```
bb11bbb9bb4.bb20
```

```
bbb9bbb6bb5.bb20
```

A.2.1 Método de Geração das Restrições Tipo "e" Horizontais

Gera as restrições tipo "e" para as distâncias mínimas horizontais X . A posição inicial dos elementos deve estar na matriz $A(i,j)$, $i = 1, \dots, n_x$ e $j = 1, \dots, n_y$, como foi explicado na seção A.1 .

A variável AUX indica se há elementos não comparáveis entre dois elementos comparáveis . Se os dois elementos comparáveis são "vizinhos" , isto é , não há nenhum outro entre eles , $AUX = 0$.

Algoritmo <Gera restrições "e" horizontais>

(X =: vetor das coordenadas horizontais ;
 x1 =: coordenadas de X ;
 A =: matriz com as entradas de dados ;
 i =: índice das linhas de A ;
 j =: índice das colunas de A ;
 AUX =: variável indicadora de vizinhança ;)

inicio

```

  AUX <- 0 ;
  J <- 2 ;
1: x1 <- 1 ;
  i <- 2 ;
  x2 <- i ;
2: se A(i,j) = 0
  então : vá para 3
  senão : x2 <- i
    se x1 = 1
      então : monte a restrição  $x_2 - x_1 \geq b$ 
              onde b é a largura de x2
              x1 <- x2
              vá para 3 ;
      senão : se x1 e x2 devem ser afas-
              tados um do outro ((ver na
              matriz de tecnologia))
              então : monte a restrição
                       $x_2 - x_1 \geq b$  , onde
                      b é a distância
                      mínima entre x1 e
                      x2
              se AUX = 0
                então :
                  x1 <- x2 ;
                senão :
                  x1 <- AUX
                  i <- x1
                  AUX <- 0 ;
    vá para 3 ;

```

senão : Se AUX = 0

então :

AUX <- X2 ;

3: i <- i + 1

Se i = nx + 1

então : monte a restrição para a largura do elemento
que está em A(i,j) : $x_2 - x_1 \geq$ (largura de
x1)

senão : vá para 2 ;

Se x1 é diferente de nx

então : Se AUX = 0

então : vá para 4 ;

senão : x1 <- AUX

AUX <- 0

i <- x1

vá para 3 ;

4: j <- j + 1

Se j é diferente de ny + 1

então : vá para 1 ;

Ordene as restrições em X ;

Limpe as restrições supérfluas ;

(Isto é : 1.Se houver duas restrições

$x_i - x_j \geq a$ e $x_i - x_j \geq b$

então : conserve apenas aquela que
tem maior valor entre a e b ;

2.Se houver três restrições do tipo

$x_i - x_j \geq a$, $x_j - x_k \geq b$ e

$x_i - x_k \geq c$ com $a + b > c$

então : elimine a restrição

$x_i - x_k \geq c$;)

fim

A.2.2 Método de Geração das Restrições Tipo "e" Verticais

Gera as restrições tipo "e" para as distâncias mínimas verticais Y . Esta é uma repetição do que foi feito na seção A.2.1 utilizando a matriz A , trocando-se X por Y e i por j , por essa razão omitiremos o algoritmo.

A.2.3 Método de Geração das Restrições Tipo "ou"

Gera as restrições tipo "ou" para as distâncias mínimas horizontais X ou verticais Y . Cada vez que são criadas novas restrições, isto deverá ocorrer aos pares. Um par é composto de uma restrição em X e outra em Y . Os pares de restrição são numerados e esta numeração deverá aparecer na arquivo de saída, tanto na parte de X quanto na de Y . Denominaremos os tipos de "esquinas" como na seção II.4, pelos seus números $k = 1, 2, 3, 4$. Rodemos ver que é suficiente comparar apenas, os tipos 1 com 3 e 2 com 4.

A seguir apresentamos o método de geração :

Algoritmo <Cera restrições "ou" >

```
(X =: vetor das coordenadas horizontais ;
 xi =: coordenadas de X ;
 y =. vetor das coordenadas verticais ;
 yi =: coordenadas de Y ;
 k , k2 =: tipos de "esquina" ;
 xk , xk2 =: posição horizontal de uma esquina do tipo k
           ou do tipo k2 , respectivamente ;
 yk , yk2 =: posição vertical de uma esquina do tipo k
           ou do tipo k2 , respectivamente ;)
```

inicio

```
k <- 1 ;
```

```
1: k2 <- k + 2 ;
```

```
2: Obtenha uma esquina do tipo k e uma do tipo k2 ;
```

```
Se não há mais pares de esquinas destes tipos
```

```
então : Se k = 2
```

```
então : vá para 4 ;
```

```
senão : k <- k + 1
```

```
vá para 3 ;
```

```
senão : Se k = 2
```

```
então : vá para 3 ;
```

```
(Comparação para k = 1)
```

```
Se xk2 > xk
```

```
então : Se Existem restrições para algum xi do tipo
```

```
 $x_i - x_k \geq a$  e  $x_{k2} - x_i \geq b$ 
```

```
então : vá para 2 :
```

```
senão : Se yk2 < yk
```

```
então : monte o par de
```

```
restrições
```

```
 $x_{k2} - x_k \geq c$  ou
```

```
 $y_{k2} - y_k \geq d$  ,
```

```
( e e d são obtidos na
matriz de tecnologia )
```


senão : Se Existem restrições para algum y_i do tipo $y_i - y_k \geq a$ e $y_{k2} - y_i \geq b$
então : vá para 2 ;
senão : monte o par de restrições $x_{k2} - x_k \geq c$ ou $y_{k2} - y_k \geq d$;

senão : Se $y_{k2} < y_k$

então : vá para 2 ;

senão : Se Existem restrições para algum y_i do tipo $y_i - y_k \geq a$ e $y_{k2} - y_i \geq b$

então : vá para 2 ;

senão : monte o par de restrições $x_{k2} - x_k \geq c$ ou $y_{k2} - y_k \geq d$;

vá para 2 ;

(Comparação para $k = 23$)

3: Se $x_{k2} > x_k$

então : Se Existem restrições para algum x_i do tipo $x_i - x_k \geq a$ e $x_{k2} - x_i \geq b$

então : vá para 2 ;

senão : Se $y_{k2} > y_k$

então : monte o par de restrições $x_{k2} - x_k \geq c$ ou $y_{k2} - y_k \geq d$;

(c e d são obtidos na matriz de tecnologia)

senão : Se Existem restrições para algum y_i do tipo $y_i - y_k \geq a$ e $y_{k2} - y_i \geq b$
então : vá para 2 ;

senão : monte o par
de restrições
 $x_k^2 - x_k \geq c$
ou
 $y_k^2 - y_k \geq d$;

senão : Se $y_k^2 < y_k$

então : vá para 2 ;

senão : Se Existem restrições para algum
 y_i do tipo $y_i - y_k \geq a$ e
 $y_k^2 - y_i \geq b$

então : vá para 2 ;

senão : monte o par de restri-
ções $x_k^2 - x_k \geq c$
ou $y_k^2 - y_k \geq d$;

4: Ordene as restrições em X e depois as em Y ;

Limpe as restrições supérfluas ,

isto é : 1. Se houver duas restrições

$x_i - x_j \geq a$ e $x_i - x_j \geq b$

então : conserve apenas aquela que
tem maior valor entre a e b :

⌈ É importante que observemos
o fato de que uma restrição
do tipo "ou" nunca pode eli-
minar uma do tipo "e" . }

⌈ Caso uma restrição do tipo
"ou" tenha sido eliminada ,
a restrição correspondente em
Y também deve ser eliminada

Faça o mesmo tipo de eliminação para a parte do arquivo Y , trocando x por y . I

2. Se houver três restrições do tipo

$$x_i - x_j \geq a \quad , \quad x_j - x_k \geq b \quad e$$

$$x_i - x_k \geq c \quad \text{com} \quad a + b > c$$

então : elimine a restrição

$$x_i - x_k \geq c ;$$

(As restrições $x_i - x_j \geq a$ e $x_j - x_k \geq b$ devem ser do tipo "e" , Da mesma forma que fizemos em 1 elimine a restrição correspondente em Y se esta for do tipo "ou" , e depois faça o mesmo para a parte do arquivo em Y .)

fim