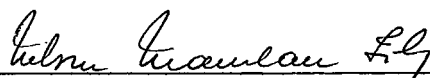


CONTRIBUIÇÃO ÀS APLICAÇÕES DOS MODELOS DE RECOBRIMENTO
E PARTICIONAMENTO EM PROGRAMAÇÃO INTEIRA

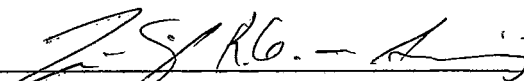
Geraldo Galdino de Paula Junior

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 MESTRE EM CIÊNCIAS (M.Sc.).

Aprovada por:



Prof. Nelson Maculan Filho
(Presidente)



Prof. João Lizardo R.H. de Araújo



Prof. Demétrio Alonso Ribeiro

RIO DE JANEIRO, RJ - BRASIL
MARÇO DE 1978

PAULA JUNIOR, GERALDO GALDINO DE

Contribuição às Aplicações dos Modelos de Recobrimento e Particionamento em Programação Inteira [Rio de Janeiro] 1978.

IX, 160p. 29,7cm (COPPE-UFRJ, M.Sc. Engenharia de Sistemas e Computação, 1978)

Tese - Univ. Fed. Rio de Janeiro. Prog. Engenharia de Sistemas e Computação

1. Otimização 2. Pesquisa Operacional I. COPPE/UFRJ II. Título(Série).

A Maria do Carmo

Ao Dudu

A meus pais e irmãs

Agradecimentos

Várias pessoas e instituições colaboraram para a realização deste trabalho. A elas agradecemos e em especial

ao Professor Nelson Maculan, pelo seu incentivo constante e pela orientação eficiente, colocando a nossa disposição seu amplo cabedal de experiência e conhecimentos técnicos;

ao Professor Michel Minoux, da Ecole Nationale Supérieure de Techniques Avancées, de Paris, com quem tivemos a oportunidade de discutir importantes aspectos deste trabalho, quando de sua passagem pela COPPE;

à Universidade Federal de Viçosa, que proporcionou suporte financeiro para a realização desta pesquisa;

aos Professores João Lizardo R. H. de Araújo e Demétrio Alonso Ribeiro, pela participação da banca de Tese;

e à Coordenação dos Programas de Pós-graduação de Engenharia da Universidade Federal do Rio de Janeiro, pela oportunidade do Mestrado em Engenharia de Sistemas e Computação.

Sumário

Este trabalho propõe um código para a solução automática de problemas combinatórios em Programação Inteira bivalente. O algoritmo de Enumeração Implícita é utilizado na forma proposta por GEOFFRION²², como simplificação do algoritmo de BALAS². Restrições substitutas ou Filtros, são implementados para acelerar o processo de enumeração de pseudo-soluções parciais. Os problemas de determinação dos conjuntos Recobrimento e Particionamento mínimos e suas formas particulares dentro da teoria dos grafos não orientados são tratados aqui, e como aplicação, é proposta uma metodologia para a solução do problema de alocação ótima ou pelo menos viável, de tripulações em rotas aéreas de uma companhia de aviação.

Summary

This work proposes a code for automatic solution of combinatorial problems in bivalent Integer Programming. The Implicit Enumeration algorithm is utilized in the form proposed by GEOFFRION²², as a simplification of the BALAS² algorithm. Filters or surrogate constraints are implemented to accelerate the partial pseudo-solutions enumeration process. The set-covering and set-partitioning problems and its particular versions within non-oriented graph theory, are treated here. As an application, a methodology is proposed for the solution of airline crew scheduling problem.

I N D I C E

CAPÍTULOS	PÁGINAS
Capítulo I - Modelos Combinatórios: Suas Áreas de Aplicação, Histórico, Dificuldade Computacional e Plano de Trabalho.....	1
1.1 - Introdução.....	1
1.2 - Histórico.....	2
1.3 - Dificuldades da Área e Política para Vencê-las..	3
1.4 - Plano de Trabalho.....	4
Capítulo II - O Processo de Enumeração das Pseudo-Soluções.....	5
2.1 - Introdução.....	5
2.2 - Definição do Problema.....	5
2.3 - Procedimento para Enumerar as Soluções de (P)...	7
2.4 - Critério para Construção do Melhor Complemento de uma Pseudo-Solução Parcial.....	11
2.5 - O Algoritmo.....	13
2.6 - Aceleração do Algoritmo, Filtros ou Restrições Substitutas.....	21
2.7 - Algumas Observações Adicionais.....	30
Capítulo III - Os Modelos de Recobrimento e Particionamento.....	32
3.1 - Introdução.....	32
3.2 - Conjunto Recobrimento e Conjunto Particionamento	32
3.3 - Definição dos Modelos.....	33
3.4 - Conversão de um (PCP) em um (PCR).....	37
3.5 - Reduções (Eliminação de Linhas e Colunas da Matriz dos Modelos de Recobrimento e Particionamento).....	42

CAPÍTULOS	PÁGINAS
3.6 - Os Modelos de Recobrimento e Particionamento e suas Formas Contínuas.....	44
3.7 - Algoritmo para a Solução dos Problemas de Recobrimento e Particionamento.....	45
Capítulo IV - O Emparelhamento Máximo e Recobrimento Mínimo em Grafos Não Orientados.....	47
4.1 - Introdução.....	47
4.2 - Conceitos Básicos.....	47
4.3 - Emparelhamento em Grafos Bipartidos.....	51
4.4 - O Problema do Recobrimento em um Grafo.....	53
4.5 - Relações entre Emparelhamentos e Recobrimentos em um Grafo Não Orientado.....	54
4.6 - Observações Adicionais.....	56
Capítulo V - Código para a Solução dos Problemas de Programação Inteira Bivalente, de Determinação dos Conjuntos Recobrimento e Particionamento Mínimos, Emparelhamento Máximo e Recobrimento Mínimo em um Grafo Não Orientado.....	57
5.1 - Introdução.....	57
5.2 - A Linguagem Utilizada.....	57
5.3 - Aspectos Básicos do Programa.....	58
5.4 - Descrição Sucinta das Principais Rotinas do Programa e suas Principais Variáveis Locais ou Passadas como Parâmetros.....	58
5.5 - Conclusões.....	67
5.6 - Como Utilizar o Programa.....	67
Capítulo VI - Aplicação: Catalogação e Alocação de Tripulações em Linhas (Rotas) Aéreas.....	70
6.1 - Introdução.....	70

CAPÍTULOS	PÁGINAS
6.2 - Aspectos Gerais e Formulação do Problema.....	70
6.3 - O Gerador da Matriz das Restrições (Matriz Catálogo).....	72
6.4 - A Questão dos Custos.....	73
6.5 - As Reduções.....	74
6.6 - Atribuição (Alocação) Individual de Tripulantes às Tripulações.....	74
6.7 - Um Tratamento Alternativo da Modelagem.....	75
6.8 - Uma Ilustração Numérica.....	76
Apêndice.....	81
Bibliografia.....	154

CAPÍTULO I

MODELOS COMBINATÓRIOS: SUAS ÁREAS DE APLICAÇÃO, HISTÓRICO, DIFICULDADE
COMPUTACIONAL E PLANO DE TRABALHO.

1.1 - INTRODUÇÃO.

Na literatura corrente sobre Programação Matemática e em particular sobre Programação Inteira, encontramos fartas referências ao fato de que o potencial de aplicação desta área tem crescido enormemente e nas duas últimas décadas numerosos avanços teóricos foram registrados. Como resultado destes avanços temos hoje uma vasta coleção de métodos e algoritmos que aliados ao desenvolvimento da exatidão, velocidade e sofisticação dos sistemas de computadores digitais, prometem ser de grande valia na solução de importantes problemas práticos que estão surgindo.

Neste trabalho tentaremos fazer ligeiras contribuições às aplicações de resultados de natureza combinatória, colocando à disposição um código para solução dos problemas de Programação Inteira bivalente, de determinação dos conjuntos recobrimento e particionamento mínimos, emparelhamento máximo e recobrimento mínimo em grafos não orientados.

Desde que as circunstâncias forçaram uma opção pela sofisticação tecnológica, vários problemas de natureza combinatória surgiram e continuam aparecendo e os identificamos frequentemente na construção de grandes redes de transmissão de informações ou de dados para processamento automático, no setor de transporte urbano, na alocação de tripulações e aviões das companhias aéreas, na administração do espaço físico em escolas e universidades, no setor político, na construção de circuitos eletrônicos impressos, na recuperação de informações de grandes arquivos em centros de processamento de dados, na seleção de projetos sujeitos a uma restrição or-

çamentária, etc. Com a necessidade das soluções desses problemas, está surgindo uma área de estudos que poderíamos denominar de "otimização combinatória" e é nesse campo que pretendemos desenvolver algumas experiências.

1.2 - HISTÓRICO.

Naturalmente foram muitos os pesquisadores que trabalharam com processos enumerativos de busca de soluções de problemas de Programação Inteira, que promoveram progressos no estudo de situações especialmente estruturadas, como os problemas dos conjuntos recobrimento e particionamento, suas relações e seu enfoque particular dentro da teoria dos grafos. Mas para fins do nosso trabalho e seguindo a ordem em que abordaremos os fatos, faremos um breve histórico, relacionando somente os nomes dos principais contribuintes.

Praticamente, o primeiro trabalho a estabelecer valiosos critérios no sentido de otimizar processos de enumeração e a receber ampla divulgação nos setores especializados, foi o de LAND e DOIG em 1960, resumidamente referenciado pelo nome de "Branch and Bound". Posteriormente, em 1963, BALAS introduziu as bases da enumeração implícita bivalente, aperfeiçoada por ele mesmo em 1965, quando o processo chamou a atenção dos especialistas de Pesquisa Operacional. Seguindo a esta fase, GLOVER, a partir de 1965, introduziu sensíveis progressos, mostrando a possibilidade de serem usadas restrições especiais que funcionariam como filtros, promovendo a aceleração do processo. GEOFFRION, 1967 e 1969, mostrou um interessante processo de gerar esses filtros a partir de soluções produzidas por variáveis duais relativas às restrições do problema proposto, juntamente com uma ligeira modificação do conceito de "poder" desses filtros no sentido de acelerar a convergência, inicialmente proposto por GLOVER. Daí em diante e por algum tempo, BALAS, GLOVER e GEOFFRION se alternaram e outros pesquisadores como SPIELBERG, LEMKE e SALKIN, introduziram refinamentos valiosos na forma dos

testes existentes. GARFINKEL e NEMHAUSER, promoveram estudos e algoritmos especiais para a solução dos problemas de determinação dos conjuntos recobrimento e particionamento mínimos. Nesse sentido podemos citar mais uma vez a participação de BALAS que proporcionou, juntamente com PADBERG, resultados interessantes nesse campo. A equivalência entre o emparelhamento máximo e o recobrimento mínimo em um grafo não orientado, foi construída por NORMAN e RABIN em 1959. Um recente trabalho de BALAS e SAMUELSSON, 1977, apresentou um novo algoritmo para os problemas de emparelhamento e recobrimento em grafos não orientados. Outros pesquisadores, como GREENBERG, BELLMORE, SPITZER, BALINSKI e QUANDT, se ocuparam de aplicações, resolvendo importantes problemas práticos, utilizando os recursos combinatórios mencionados.

1.3 - DIFICULDADES DA ÁREA E POLÍTICA PARA VENCÊ-LAS.

Diferentemente ao caso dos programas contínuos (Programação Linear), não há esperanças de, no futuro próximo, podermos resolver todos os problemas de Programação Inteira por um único algoritmo. Por outro lado, a dificuldade computacional observada nas técnicas de Programação Inteira de caráter geral, conduziu à formulação e desenvolvimento de métodos especiais para solução de problemas inteiros que possuam estruturas particulares. Isto é, algoritmos especiais são continuamente produzidos utilizando a estrutura particular de uma família de problemas. Uma outra possibilidade, é projetar algoritmos complexos, capazes de analisar, avaliar e decidir que tipo de procedimento poderia ser aplicado na solução do problema particular que se apresentasse. Por exemplo, existem algoritmos para solução do problema de determinação do conjunto recobrimento mínimo, que trabalham melhor com matrizes de alta densidade e outros, ao contrário, são mais eficazes no trato com matrizes de baixa densidade. Aceitamos a idéia de que qualquer estudo experimental dentro deste campo, deva ser levado a efeito com uma polí

tica (que pode ser uma dessas), bem definida em mente, o que tenderia minimizar dificuldades de ordem global.

1.4 - PLANO DE TRABALHO.

Iniciaremos com um capítulo sobre enumeração parcial que deverá se constituir no miolo de todo o processo que produzirá na sua última forma, a ferramenta com que poderemos resolver o problema (aplicação) proposto no CAPÍTULO VI. Em seguida cuidaremos de uma parte dedicada à questão da determinação dos conjuntos recobrimento e particionamento mínimos (na sua forma prática, o modelo de particionamento se assemelha ao de recobrimento com restrições de igualdade). Aí, no CAPÍTULO III, estará o esqueleto da aplicação que pretendemos recomendar. Segue o CAPÍTULO IV, onde relacionamos as idéias particulares, daquelas do CAPÍTULO III, dentro da Teoria dos Grafos, i. e., trataremos com o emparelhamento máximo e recobrimento mínimo em grafos não orientados. No CAPÍTULO V, descrevemos a construção do código, objeto central de nossas experiências e finalmente, no CAPÍTULO VI, ilustramos a aplicabilidade do programa, recomendando uma aplicação referente à alocação de tripulações em rotas aéreas para uma companhia de aviação.

CAPÍTULO II

O PROCESSO DE ENUMERAÇÃO DAS PSEUDO-SOLUÇÕES.

2.1 - INTRODUÇÃO.

Vamos coleccionar e ordenar os resultados necessários para a solução de um problema de Programação Inteira bivalente, pelo método de enumeração parcial ou enumeração implícita, como é conhecido na literatura corrente.

2.2 - DEFINIÇÃO DO PROBLEMA.

Seja z uma função

$$z : \{0,1\}^n \rightarrow K \subset \mathbb{R} \quad (\text{II.1})$$

$$z(x) = \sum_{j \in N} c_j x_j, \quad N = \{1, \dots, n\} \quad (\text{II.2})$$

Sobre z queremos resolver o seguinte problema:

$$(P) \quad \text{minimizar } z = \sum_{j \in N} c_j x_j \quad (\text{II.3})$$

$$\text{sujeito a } b_i + \sum_{j \in N} a_{ij} x_j \geq 0, \quad i \in M = \{1, \dots, m\} \quad (\text{II.4})$$

$$x_j \in \{0, 1\} \quad j \in N. \quad (\text{II.5})$$

Algumas vezes preferiremos uma forma mais compacta:

$$\text{minimizar } z = cx \quad (\text{II.6})$$

$$\text{sujeito a } b + Ax \geq \underline{0}, \quad (\text{II.7})$$

onde c é o vetor de n dimensões (c_1, \dots, c_n) , b e $\underline{0}$, respectivamente os vetores de m dimensões $(b_1, \dots, b_m)^t$ e $(0, \dots, 0)^t$. $x = (x_1, \dots, x_n)^t$ é um vetor binário de n dimensões e $A = (a_{ij})$, é uma matriz $m \times n$. Qualquer vetor binário x de n dimensões, será chamado uma "pseudo-solução" de (P). Qualquer pseudo-solução que satisfizer (II.4), será chamada de solução viável de (P) e a solução viável de (P) que minimizar z entre todas as soluções viáveis, implícita ou explicitamente enumeradas, será chamada de solução ótima de (P).

Suporemos sempre que $c \geq 0$, sem que isto constitua uma restrição à generalidade, uma vez que, ocorrendo $c_j < 0$, transformaremos x_j em $1 - x_j'$. No decorrer da apresentação das idéias, ficará clara a necessidade de mantermos sempre esta situação.

Também aproveitamos a possibilidade de podermos desenvolver um número na sua correspondente forma (expressão) de base 2, para estendermos nosso algoritmo na tarefa de solucionar qualquer problema de Programação Inteira Limitado, i.e., qualquer problema de Programação Inteira onde cada variável $x_k \geq 0$, possui uma cota superior u_k . De fato, se x_k é uma variável inteira qualquer, possuindo uma cota superior u_k , i.e., $x_k \leq u_k$, podemos escrever

$$x_k = \sum_{p=0}^K 2^p y_{kp}, \quad y_{kp} \in \{0,1\} \quad (\text{II.8})$$

onde K é o menor número determinado de tal forma a satisfazer $2^{K+1} - 1 \geq u_k$. O inconveniente desta representação, é a grande quantidade de variáveis que surgem para grandes valores de u_k , tornando a solução do problema impraticável dentro de um tempo computacionalmente viável.

2.3 - PROCEDIMENTO PARA ENUMERAR PARCIALMENTE AS SOLUÇÕES DE (P).

Para formalizar melhor as idéias envolvidas neste procedimento, precisamos de alguns termos (ou expressões), cujas definições daremos a gora. Começemos por "pseudo-solução parcial".

Uma pseudo-solução parcial, é uma atribuição de valores binários a um subconjunto das n variáveis.

Uma variável à qual não se atribuiu valor, será chamada de "variável livre" (trata-se de uma variável livre para receber uma atribuição binária no momento em que os testes do algoritmo determinarem).

Representaremos uma pseudo-solução parcial por W , conjunto dos índices das variáveis que compõem a pseudo-solução parcial, de tal forma que se $j \in W$, $x_j=1$ e se $-j \in W$, $x_j=0$. Para ilustrar, se

teremos $n=6$ e $W=\{2,4,-6\}$,
 $x_2=1$, $x_4=1$, $x_6=0$ e as variáveis livres re-

lativamente a esta pseudo-solução parcial, serão x_1 , x_3 e x_5 .

Dada uma pseudo-solução parcial W , tal que $\text{card}(W)=k$, existem $n-k$ variáveis livres. Atribuindo valores 0-1 a essas $n-k$ variáveis livres, podemos formar 2^{n-k} vetores de $n-k$ dimensões, que chamaremos de "vetores livres". Tomando o pequeno exemplo dado, temos os seguintes vetores li-

vres referentes a W:

$$(0,0,0), (0,1,0), (0,1,1), (0,0,1), (1,0,0), (1,1,0), (1,1,1), (1,0,1)^*$$

Com isso podemos definir o complemento de uma pseudo-solução parcial.

Um "complemento" ou "remate" de uma pseudo-solução parcial W, é uma pseudo-solução definida pelos valores das variáveis cujos índices estão em W, juntamente com os valores das variáveis especificados num dos vetores livres. Mais uma vez, se $n=6$ e $W=\{2,4,-6\}$, o remate da pseudo-solução parcial $(?,1,?,1,?,0)$, é a pseudo-solução $(1,1,0,1,1,0)$, formada com o vetor livre (*) dado acima.

O complemento nulo de uma pseudo-solução parcial, é uma pseudo-solução formada com o vetor livre que tem todos os elementos iguais a zero.

O número de complementos de uma pseudo-solução parcial é naturalmente o mesmo de vetores livres. Portanto, uma pseudo-solução parcial W, tal que $\text{card}(W)=k$, tem 2^{n-k} complementos ou remates. W é o complemento de si mesma quando não existem variáveis livres. Neste caso, $\text{card}(W)=n$.

Na descrição do procedimento para enumerar as pseudo-soluções parciais de (P), as variações dos verbos "podar" e "sondar", terão significados tecnicamente precisos:

O ato de "podar" uma pseudo-solução parcial (P), executado pelo algoritmo, corresponde à ação de:

i) enumerar a pseudo-solução parcial e enumerar implicitamente todos os seus complementos, se a pseudo-solução parcial não tem complemento viável.

ii) caso contrário, enumerar o melhor complemento viável da pseudo-solução parcial e enumerar implicitamente todos os seus outros complementos.

O algoritmo "sonda" uma pseudo-solução parcial quando ela é viável (ou tem complemento viável), verificando se seu melhor complemento viável produz uma cota superior sobre a solução ótima melhor (menor) do que

aquela conhecida até aqui. Em seguida poda a referida pseudo-solução parcial.

Faremos referência ao "um" como complemento lógico do "zero" e vice-versa, o que certamente não será confundido com complemento de uma pseudo-solução parcial.

O número de todos os -vetores binários- que definimos como pseudo-soluções de (P), é 2^n , e portanto, por um processo exaustivo de enumeração, poderíamos encontrar a solução ótima de (P), quando ela existisse. Entretanto, qualquer busca exaustiva da solução de (P), seria por demais dispendiosa, principalmente em termos de tempo, para os valores de n que surgem nas aplicações práticas. A idéia da enumeração parcial ou implícita, proposta por BALAS², surgiu no sentido de estabelecer um algoritmo munido de critérios que permitissem podar todas as pseudo-soluções parciais sem percorrer exaustivamente a árvore gerada por elas. Esses critérios dão seqüência a um mecanismo dinâmico de avanço e retorno sobre os ramos da árvore em cujos nós estão as pseudo-soluções.

O algoritmo estabelece uma poda passando ao complemento lógico do último elemento ainda não complementado na pseudo-solução parcial podada e liberando as variáveis cujos índices figuram na pseudo-solução parcial podada, imediatamente à direita dele.

Em resumo, enumeração implícita é um processo de solução baseado na geração de um grafo tipo árvore, cujo nós são depositários das pseudo-soluções parciais e baseado em critérios que permitem enumerar implícita ou explicitamente todas as pseudo-soluções possíveis.

W é um conjunto ordenado, no sentido de que a ordem dos seus elementos, reflete a ordem em que foram gerados, i. e., reflete a seqüência natural definida pelos mecanismos de avanço e retorno do processo enumerativo. Quando um índice de W é logicamente complementado, ele é também sublinhado para indicar que o outro valor de sua variável já foi considerado. Algumas vezes abreviaremos por "complemento lógico", aquilo que na verdade é o complemento lógico sublinhado.

Quando uma pseudo-solução parcial é produzida, o algoritmo

verifica se ela é viável. Se for, verifica ainda (sonda) se seu melhor complemento viável fornece uma cota superior sobre a solução ótima, melhor do que aquela conhecida até aqui. Se isto acontece, a pseudo-solução parcial é considerada sondada, seus complementos são excluídos de futuras considerações, excessão feita de seu melhor complemento viável que é armazenado, juntamente com a cota, para futuras averiguações (futuras comparações). Se a cota produzida não é melhor do que a existente, o algoritmo simplesmente poda a pseudo-solução parcial em questão. Se a pseudo-solução parcial não é viável, o algoritmo tenta vencer a inviabilidade, buscando uma pseudo-solução parcial descendente desta, da seguinte forma: coleciona as variáveis livres que fixadas ajudariam na obtenção de uma cota superior melhor do que a existente e cooperariam para vencer a inviabilidade. Se não existem variáveis livres com essas características, o algoritmo poda a pseudo-solução parcial. Caso contrário, ele elege dentre as variáveis com as características acima, aquela que minimiza a quantidade total de inviabilidade existente. O índice desta variável é acrescentado àqueles da pseudo-solução parcial considerada, formando com eles uma nova pseudo-solução parcial descendente daquela.

Representamos por (W^k) a seqüência das pseudo-soluções parciais. O algoritmo inicia o processo de enumeração com $W^0 = \phi$. O processo terminaria nesse ponto se o algoritmo podasse W^0 , pois $\text{card}(W^0)=0$ e os complementos de W^0 , em número de 2^n , seriam implicitamente enumerados. Caso contrário, o algoritmo obtém W^1 , como descendente de W^0 , pelo aumento desta com a eleição de uma variável livre, a cada passo tentando podar (ainda que depois de sondar), a presente pseudo-solução parcial. Prosseguindo assim, até que na p-ésima pseudo-solução parcial, W^p é sondada. O melhor complemento de W^p , se este produzir uma cota superior sobre a solução ótima melhor do que a cota conhecida até então, é guardado como uma solução candidata.

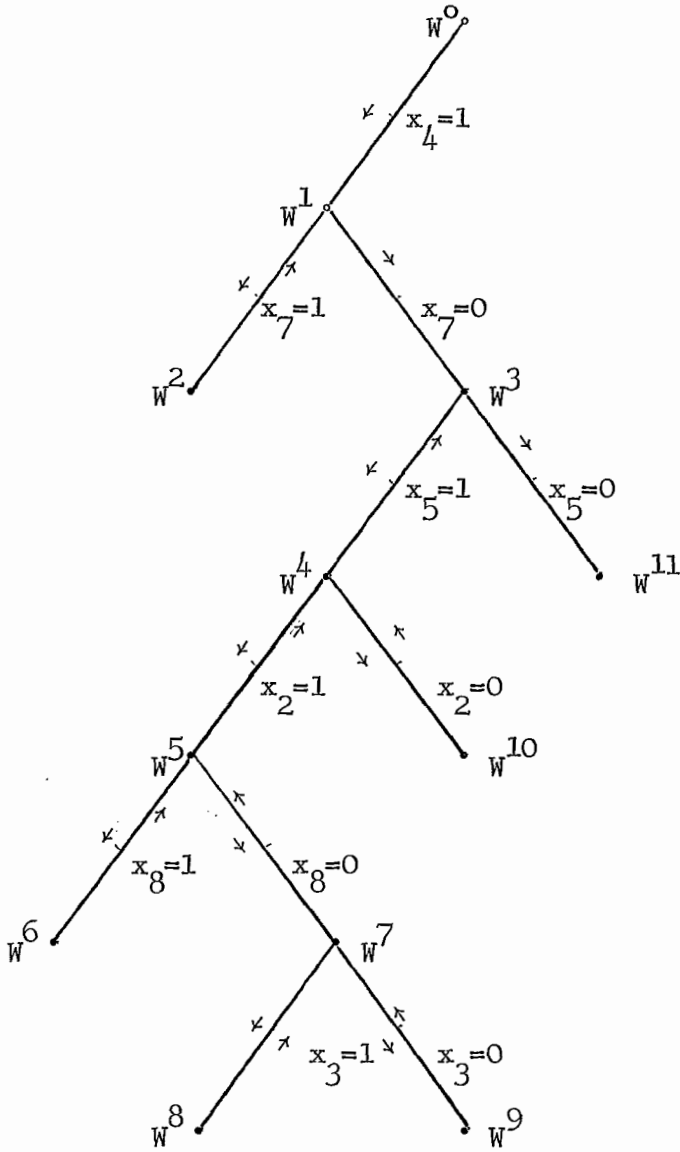
Dois membros sucessivos, W^p e W^{p+1} , da seqüência (W^k) são distintos ou porque $W^p \subset W^{p+1}$, que acontece quando W^{p+1} é descendente de W^p ou porque possuem elementos logicamente complementares. Este fato faz da seqüência (W^k) , uma seqüência não redundante, no sentido de que ela não dupli

ca nenhuma pseudo-solução parcial previamente podada.

A fim de visualizar melhor o que foi dito acima, a figura II.1, mostra parte de uma árvore gerada pelas pseudo-soluções parciais de um problema de Programação Inteira bivalente. Lá estão representados doze elementos da seqüência (W^k) . Através destes elementos podemos observar a lei que rege a formação da seqüência de pseudo-soluções parciais. Passamos de um membro para outro da seqüência, seja pela complementação lógica do último elemento ainda não complementado, seja pelo acréscimo de mais um elemento. No primeiro caso, abandonamos todos os elementos imediatamente à direita do último elemento complementado. Este fato corresponde a um retorno nos ramos da árvore. Quando a passagem para o próximo membro da seqüência, se faz através do acréscimo de mais um elemento, o algoritmo está provocando um avanço nos ramos da árvore (criando um descendente). Observamos que a diferença entre W^1 e W^2 , está no elemento a mais que W^2 possui (W^2 é descendente de W^1) e a diferença entre W^2 e W^3 , está no elemento logicamente complementado que W^3 possui.

2.4 - CRITÉRIO PARA CONSTRUÇÃO DO MELHOR COMPLEMENTO DE UMA PSEUDO-SOLUÇÃO PARCIAL.

Se para uma pseudo-solução parcial W , for construído um complemento, atribuindo a cada variável livre x_j o valor 0 ou 1, conforme $c_j \geq 0$ ou $c_j < 0$, respectivamente e lembrando que nosso problema é de minimização, estaremos diante de um complemento da pseudo-solução parcial, que se for viável, será o melhor complemento viável em questão. Como os custos de (P) são todos positivos ou temos a possibilidade de obtê-los assim, pela transformação de variáveis já explicada no início, observamos que o complemento nulo de uma pseudo-solução parcial, quando for viável, será o melhor complemento viável nessas condições. Assim, se o complemento nulo viável de uma pseudo-solução parcial não produzir uma cota superior sobre a solução



$$W^0 = \phi$$

$$W^1 = \{4\}$$

$$W^2 = \{4, 7\}$$

$$W^3 = \{4, \underline{-7}\}$$

$$W^4 = \{4, \underline{-7}, 5\}$$

$$W^5 = \{4, \underline{-7}, 5, 2\}$$

$$W^6 = \{4, \underline{-7}, 5, 2, 8\}$$

$$W^7 = \{4, \underline{-7}, 5, 2, \underline{-8}\}$$

$$W^8 = \{4, \underline{-7}, 5, 2, \underline{-8}, 3\}$$

$$W^9 = \{4, \underline{-7}, 5, 2, \underline{-8}, \underline{-3}\}$$

$$W^{10} = \{4, \underline{-7}, 5, \underline{-2}\}$$

$$W^{11} = \{4, \underline{-7}, \underline{-5}\}$$

⋮

Figura II.1

Parte de uma árvore de pseudo-soluções parciais de um problema de Programação Inteira bivalente

ótima melhor do que a atual, o algoritmo não tem necessidade de testar os outros complementos e o que ele tem a fazer, é podar a pseudo-solução parcial em questão.

Com relação ao complemento nulo de cada pseudo-solução parcial, o algoritmo tem duas coisas a fazer:

- i. Testar sua viabilidade.
- ii. Se for viável, verificar (sondar) se a cota superior sobre a solução ótima, gerada por este complemento, é melhor do que a última cota armazenada como candidata ao ótimo de (P)..

Obviamente, a segunda condição será testada se a primeira se cumprir e do cumprimento da segunda condição, resultará o armazenamento do referido complemento nulo como solução candidata ao ótimo. Do cumprimento ou não da condição (ii), o próximo passo que o algoritmo executa, é a poda da atual pseudo-solução parcial e o não cumprimento da condição (i), faz com que o algoritmo tente estabelecer uma pseudo-solução parcial descendente da atual.

2.5 - O ALGORITMO.

As restrições do nosso problema, têm a forma

$$b_i + \sum_{j \in N} a_{ij} x_j \geq 0, \quad i \in M. \quad (\text{II.9})$$

Chamando de y_i , $\forall i \in M$, a variável de folga, as restrições assumem o aspecto

$$b_i + \sum_{j \in N} a_{ij} x_j - y_i = 0, \quad i \in M, \quad (\text{II.10})$$

