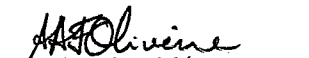
INSTALAÇÃO DE POSTOS DE ATENDIMENTO E VENDA DE INSUMOS NUMA COOPERATIVA AGRÍCOLA: UMA APLICAÇÃO DO PROBLEMA DA MOCHILA 0-1

Paulo Roberto de Castro Villela

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M. Sc.)

Aprovada por:

Claudio Thomas Bornstein
Presidente



Antonio Alberto Fernandes de Oliveira

Valdecir Antoninho Dalpasquale

RIO DE JANEIRO, RJ - BRASIL
OUTUBRO DE 1983

AGRADECIMENTOS

Ao se findar um trabalho como esse nos lembramos imediatamente daqueles que participaram diretamente do mesmo. Se refletirmos mais um pouco veremos que demos mais um passo na caminhada pela vida, e neste contexto muitos são aqueles que também participaram desse trabalho. A todos agradeço pela contribuição que me deram.

Em particular, agradeço aos professores e funcionários do Programa de Sistemas e Computação da COPPE/UFRJ pela atenção e dedicação; aos meus colegas no curso de mestrado pela amizade; aos colegas da Faculdade de Engenharia da UFJF pelo apoio; aos participantes da banca de tese pelas críticas a essa obra; à Rosana e à Ana pelo excelente trabalho datilográfico; à COTREFAL e à ACARPA pelos dados fornecidos; ao CENTREINAR pelo apoio recebido; à Rita, minha esposa, pela compreensão; ao Artur por ser meu filho; e aos meus pais por me criarem.

Em especial gostaria de agradecer ao meu orientador, Prof. Cláudio Thomas Bornstein, pela dedicação que mostrou ter e principalmente pela amizade e honestidade sempre presente em todos os momentos de nossa convivência.

RESUMO

Apresentamos uma modelagem para instalação de postos de atendimento e venda de insumos (sementes, fertilizantes, etc.) à agricultores de uma cooperativa, onde são analisados: 1) o dimensionamento de tais postos, e 2) a escolha de quais destes deverão ser construidos considerando a limitação de recursos financeiros e objetivando otimizar algum critério fixado. O segundo item desta análise conduziu ao Problema da Mochila 0-1, que é então estudado em detalhe, e para o qual desenvolve-se e implementa-se um algoritmo "branch-and-bound". São mostrados testes computacionais que comprovam a eficiência deste algoritmo.

ABSTRACT

We present a model for the establishment of service stations for a cooperative of agricultural production. First we analyse the dimensions of the service stations. Then we choose which of them may be constructed, given a limited amount of finantial recourse, so that given criteria are optimized. The second part of the analysis deals with the 0 - 1 Knapsack problem. We examine the problem with some detail and develop a branch-and-bound algorithm to solve it. Computational results are presented that show the efficiency of the algorithm.

ÍNDICE

APÊNDICE A

SOLUÇÃO ÓTIMA DE UM PM01 RELAXADO

APÊNDICE B

ALGUNS LIMITANTES SUPERIORES DO VALOR

ÓTIMO DE UM PM01

APÊNDICE C

LISTAGEM DO PROGRAMA (EM BASIC) PARA SE

DETERMINAR OS POSTOS A SEREM INSTALADOS

CAPÍTULO 1

INTRODUÇÃO

Quando começamos a trabalhar nesta tese, a idéia era aplicar técnicas de localização, desenvolvidas na COPPE, a uma situação concreta de armazenamento de grãos.

Dentro deste objetivo surgiu então a oportunidade de fazer um trabalho com o CENTREINAR (órgão resultante de convênio entre a Universidade Federal de Viçosa e a CIBRAZEM, e sediado em Viçosa, MG) que vinha desenvolvendo um novo silo secador utilizando energia solar como fonte térmica - ver Sinicio-Roa |[29]|. Dos contatos com o CENTREINAR ficou evidenciado o interesse em se estudar um sistema de armazenagem para uma cooperativa agrícola, visando ou não a utilização dos silos secadores a energia solar.

A partir deste contato com o CENTREINAR decidiu-se fazer o estudo para uma cooperativa situada no oeste do estado do Paraná e que se encontrava num bom grau de desenvolvimento agrícola, tecnológico e comercial e que portanto teria todas as qualidades para participar de um estudo desse nível. Tratava-se da Cooperativa Três Fronteiras (COTREFAL) sediada em Medianeira, PR.

Antes de visitarmos a COTREFAL para expor a idéia a seus diretores, conversamos, em Curitiba, com técnicos da ACARPA que nos adiantaram alguns dados sobre a região em que se localiza a COTREFAL, salientando que

boa parte de suas terras seriam inundadas pelo lago formado pela barragem da Usina Hidrelétrica de Itaipu.

Na COTREFAL constatamos que a mesma estava às voltas com o problema do alagamento das terras, o que faria diminuir sua produção agrícola tornando portanto menos premente a ampliação do seu sistema de armazenamento de grãos. Além disso, a sua principal prioridade, no momento, era a industrialização da produção de grãos fabricando óleos vegetais. Estes fatores nos levaram a abandonar a idéia original.

Entretanto, foi-nos apresentado um outro problema, relativo à instalação de novos postos de atendimento e venda de insumos (fertilizantes, sementes, etc) agrícolas. Naquela época a COTREFAL contava com 8 desses postos já instalados em diversas regiões e pretendia dobrar o número destes. A finalidade principal destes novos postos era melhorar o atendimento aos associados à Cooperativa bem como conseguir novas associações de produtores da região.

Tratava-se sem dúvida de um problema de localização. Onde instalar tais postos visando otimizar algum critério pré-estabelecido ?

Como, no entanto, uma série de condicionantes do problema já tinham sido bem definidas restou pouco espaço para uma otimização. As localidades onde seriam instalados os novos postos já estavam definidas segundo interesses e critérios aprendidos na prática do dia-a-dia. As regiões às quais cada posto serviria também já estavam delimitadas. Restava dimensio

nar cada posto para que atendesse satisfatoriamente sua região de influência. Outro problema real é que os recursos financeiros disponíveis não eram suficientes para construir de uma vez todos os 8 postos pretendidos (ver seções 2.3.3 e 2.3.4).

O problema do dimensionamento (ver seção 2.3.2) dos postos foi feito de duas maneiras. A primeira de acordo com a praxe da Cooperativa, e a segunda dimensionando-os para suportar a demanda nos períodos críticos (de piques).

A otimização do problema levando em conta restrições devido a limitações de recursos foi feita utilizando-se técnicas de programação matemática, estudada no capítulo 3. Para se poder escolher quais dos 8 postos seriam instalados foi preciso estabelecer critérios. Foram analizados 3 critérios: 1) Maximizar o número de produtores associados à COTREFAL que passariam a ter um posto mais próximo da sua propriedade; 2) maximizar o número de produtores não-associados à COTREFAL que passariam a ter um posto mais próximo da sua propriedade; e 3) maximizar o lucro (sobra líquida) obtido com a venda dos insumos. A comparação das soluções obtidas (seção 2.6) nos mostra o efeito de se privilegiar compromissos ora sociais ora comerciais. A análise crítica de tais resultados é deixada aos leitores desse trabalho. O segundo critério, especificamente, vai de encontro ao interesse imediato da COTREFAL de contrabalançar a perda de associados, que tiveram suas terras alagadas, com a entrada de novos associados, pois o critério em questão premia uma atitude promocional qual seja a instalação de posto perto da propriedade de um não-associado.

A modelagem do problema é feita no capítulo 2, onde também são dados detalhes acerca da determinação dos parâmetros do modelo. Como resultado dessa modelagem chegou-se ao Problema da Mochila 0-1 (PM01) que é examinado em detalhe no capítulo 3.

Evidentemente o interesse em estudar o PM01 extrapola, em muito, a necessidade de resolver o problema matemático resultante da modelagem feita no capítulo 2. A estrutura de um Problema da Mochila é freqüentemente encontrada em casos práticos e além disso tem importância fundamental em algoritmos de programação inteira onde aparece como um subproblema.

Dos métodos utilizados para solucionar o PM01, os do tipo "branch and bound" tem se mostrado os mais promissores. Na seção 3.2 apresentamos um exemplo numérico de PM01 com 3 variáveis onde visualizamos graficamente a solução do mesmo usando a técnica "branch and bound". Em 3.3 tentamos mostrar os procedimentos básicos para resolver um PM01 através da técnica "branch and bound", bem como apresentamos condições de convergência e otimalidade dos mesmos. Em 3.4 analisa-se diversos algoritmos publicados, dando alguns detalhes. Esse breve "survey" será útil na compreensão do algoritmo que desenvolvemos (seção 3.5) e cujos testes computacionais são apresentados em 3.6.

São ainda apresentados 3 apêndices: O primeiro justifica o cálculo da solução ótima de um PM01 relaxado, isto é, sem as restrições de integralidade das variáveis; o segundo analisa as diversas maneiras para se calcular limitantes superiores do valor ótimo de um PM01 e que inclui uma melhoria que desenvolvemos; e o último contém a listagem da implementação

em Basic do algoritmo desenvolvido em 3.5.

Todos os algoritmos tratados nesta tese conduzem à solução exata do PMO1. Existem, entretanto, atualmente algoritmos aproximativos, ver |[21]| e |[28]|, que conduzem a soluções boas mas não necessariamente ótimas em tempo polinomial. É fato conhecido que o PMO1 é NP-completo.

O objetivo do capítulo 3 foi desenvolver um algoritmo eficiente. Para medir a eficiência rodamos baterias de testes padronizados encontrados na literatura, utilizando o algoritmo aqui desenvolvido e o de Horowitz-Sahni |[14]|, que é freqüentemente usado por outros autores como comparação. Os testes computacionais foram feitos em microcomputador Schumec M100 em Basic Interpretado, sendo computados o número médio de operações elementares (adições, comparações, atribuições, multiplicações e buscas em vetor) para cada tipo de problema. Os resultados obtidos foram bastante satisfatórios como se acha apresentado em 3.6.

CAPÍTULO 2

INSTALAÇÃO DE POSTOS DE ATENDIMENTO E VENDA DE INSUMOS NUMA COOPERATIVA AGRÍCOLA

2.1 - INTRODUÇÃO

Situada no extremo oeste do estado do Paraná, a Cooperativa Três Fronteiras (COTREFAL) deseja aumentar o seu número de associados. Com esse aumento espera contrabalançar os associados perdidos com o alagamento de parte de suas terras pelo lago da Hidrelétrica de Itaipu e também atender a um aumento na demanda de grãos ocasionado pelo processo de industrialização que por ora passa a COTREFAL.

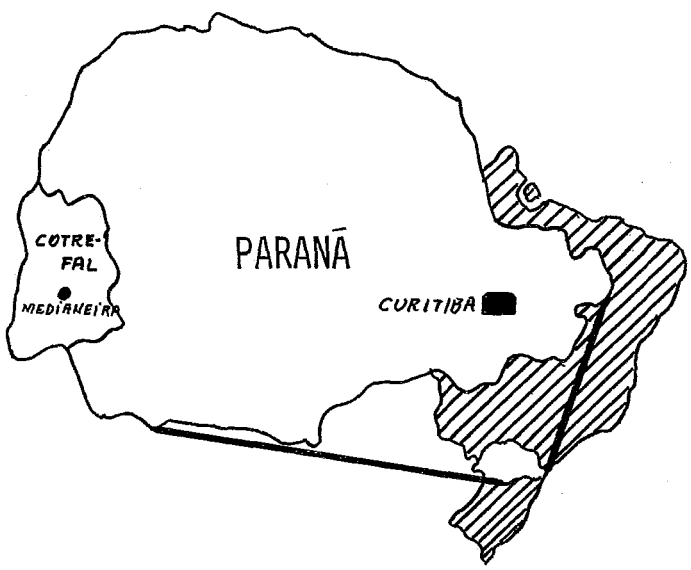


Fig. 2.1 - Situação Geográfica da COTREFAL no Paraná.

Para conseguir tal intento a COTREFAL instalará "Postos de Atendimento e Vendas de Insumos Agrícolas" (aos quais chamaremos de "POSTOS")

em localidades que até então não dispõem dessa comodidade.

Os Postos além de venderem aos agricultores (associados ou não à COTREFAL) insumos agrícolas (fertilizantes, sementes, defensivos, etc.) também terão um agrônomo residente e com isso seria oferecido um serviço de assistência completo ao agricultor sem que ele precise se deslocar longas distâncias.

Desta forma pretende-se beneficiar o associado da COTREFAL e também atrair novos associados que sem dúvida virão, face às facilidades que lhes serão oferecidas pela COTREFAL.

A COTREFAL definiu previamente 8 possíveis localidades onde se poderão instalar Postos. Mas apenas alguns dos 8 possíveis Postos serão instalados, isto porque os recursos financeiros disponíveis (próprios e empréstimos bancários) são limitados.

O objeto desse estudo é decidir quais dos 8 Postos deverão ser instalados, dentro das limitações financeiras, de tal forma que se otimize um determinado objetivo.

Estudaremos 3 objetivos separadamente:

1º Objetivo: Aumentar ao máximo o número de associados que terão um Posto mais perto (do que atualmente tem) de sua propriedade;

2º Objetivo: Aumentar ao máximo o número de não-associados que terão um Posto mais perto (do que atualmente tem) de sua propriedade;

3º Objetivo: Aumentar ao máximo os rendimentos (sobra líquida) auferidos com as vendas dos Postos.

Esses objetivos foram selecionados de modo a mostrar 3 aspectos diversos para os quais os Postos poderão ser usados. O primeiro objetivo valoriza o associado, o segundo leva em conta o objetivo da COTREFAL de atrair novos associados e o terceiro mostrará os efeitos de se considerar a "maximização do lucro" como meta.

Um ponto importante desse estudo é a comparação das soluções obtidas levando-se em conta cada um desses objetivos.

Atualmente a COTREFAL já tem instalado 8 Postos. Esses Postos não entrarão neste estudo, a não ser para mostrar as características de funcionamento do atual sistema de Postos, que certamente serão parecidas com as do novo sistema.

2.2 - CONSIDERAÇÕES INICIAIS

2.2.1- Técnica utilizada na solução do problema

O problema que resolveremos será solucionado através de técni

cas de Programação Matemática, mais especificamente através do "Problema da Mochila 0-1".

Os algoritmos utilizados na resolução do "Problema da Mochila 0-1" são objeto de estudo no Capítulo 3 desta tese.

2.2.2 - Data base para tomada de preços e coeficientes

Os preços bem como os coeficientes utilizados neste estudo tem por base dados colhidos junto à COTREFAL em setembro de 1981.

2.2.3 - Metodologia da coleta de dados

Os dados em que se baseou este estudo foram fornecidos diretamente pela COTREFAL e refletem de forma relativamente adequada a atual conjuntura observada na região bem como levam em conta as mudanças que se efetuarão com o alagamento da terra pela Hidrelétrica de Itaipu.

Estes dados não foram obtidos através de pesquisa de campo, arquivos, etc.

Deve-se lembrar que uma pesquisa de campo além de onerosa e demorada, pode levar a distorções e sempre deve ser acompanhada de um bom estudo crítico.

O que se fez, foi colher dados de forma indireta junto a pessoas que vivem o dia-a-dia da COTREFAL e a conhecem de forma relativamente global e precisa.

Desta maneira foi possível prever alterações na região levando em conta a inundação de grande parte de terras cultiváveis pelo lago da Hidrelétrica de Itaipu.

2.2.4 - Descrição de um Posto

Um Posto de Atendimento e Venda de Insumos Agrícolas é composto de um armazém para guarda do estoque de insumos, a serem vendidos aos produtores a medida que estes forem solicitando, e um escritório para administração e atendimento dos clientes.

De acordo com desejo da COTREFAL cada Posto deverá ser capaz de estocar um mínimo de 300 t e um máximo de 500 t de insumos. Além disso considera-se como sendo de 20 anos a vida útil de cada Posto.

Considera-se que cada tonelada de insumos requer 1 m^2 de área construída no Posto. Desta forma a área de um Posto deverá ficar entre 300 m^2 e 500 m^2.

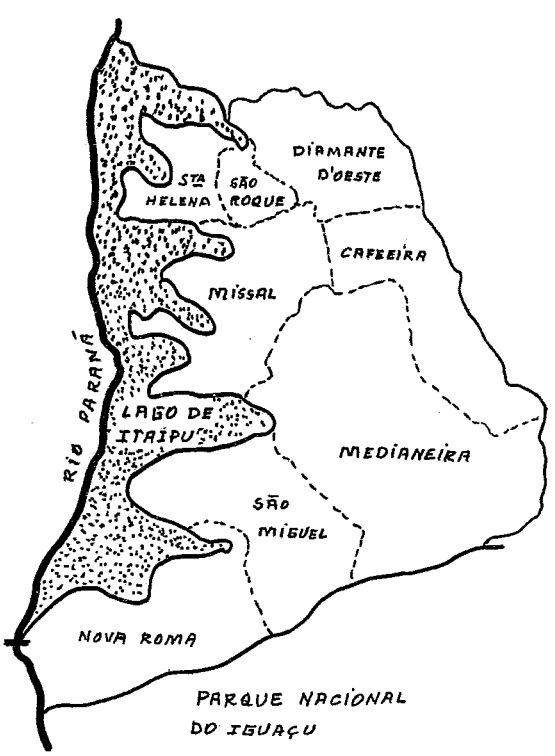
O custo unitário (Cr\$/m^2) de construção varia conforme a área construída do Posto segundo a seguinte equação:

$$c = 18.000 - 10\,a \quad \text{para} \quad 300 \leqslant a \leqslant 500 \qquad (2.1)$$

onde a é área (m^2) a ser construída e c é o custo unitário ($Cr\$/m^2$) da construção.

2.2.5 - Estrutura atual da armazenagem de insumos

Atualmente existem já instalados na COTREFAL 8 Postos de Atendimento e Venda de Insumos agrícolas, são eles: Postos de Nova Roma, São Miguel do Iguaçu, Medianeira, Cafeeira, Missal, São Roque, Diamante d'Oeste e Santa Helena.



CAPACIDADE INSTALADA

1. Nova Roma	2.160t
2. São Miguel	720t
3. Medianeira	19.800t
4. Cafeeira	2.700t
5. Missal	1.260t
6. Diamante d'Oeste ...	780t
7. São Roque	60t
8. Santa Helena	3.600t

Fig. 2.2 - Postos atualmente instalados na COTREFAL (Localização e respectiva região de influência)

No mapa da Fig. 2.2 tem-se a localização geográfica desses 8 Postos já instalados, sua capacidade estática de armazenagem de insumos e também sua respectiva região de influência conforme dados da COTREFAL. A região de influência de um dado Posto separa geograficamente os agricultores que compram daqueles que não compram no Posto.

Vários são os motivos que levam os agricultores a comprarem num ou noutro Posto, citamos por exemplo:

a) distância geográfica da propriedade ao Posto;
b) topografia e condições de tráfego da via de acesso entre a propriedade e o Posto;
c) comodidades oferecidas (banco, diversão, casas comerciais, etc.) na localidade onde se encontra o Posto.

A delimitação de cada região foi obtida observando-se os atuais hábitos dos agricultores.

2.3 - BASES PARA A FORMULAÇÃO MATEMÁTICA DO ESTUDO

2.3.1 - Localização dos 8 novos Postos

A COTREFAL selecionou 8 localidades que potencialmente comportam (econômica e socialmente falando) a instalação de um novo Posto de Atendimento e Venda de Insumos Agrícolas. Os critérios que a COTREFAL utilizou

para escolha dessas 8 localidades, potencialmente promissoras para a instalação de um Posto, não são objeto desse estudo.

Da mesma forma como acontece com os atuais Postos, onde cada um tem sua região de influência, um novo Posto terá a sua região de influência.

A delimitação geográfica dessas regiões de influência são bem mais complexas do que no caso dos Postos já instalados, pois para estes basta pesquisar o comportamento atual do agricultor, ao passo que para aqueles o processo foi totalmente empírico. Assim o pessoal da COTREFAL delimitou cada uma das 8 regiões de influência correspondentes a cada um dos 8 novos Postos potencialmente promissores.

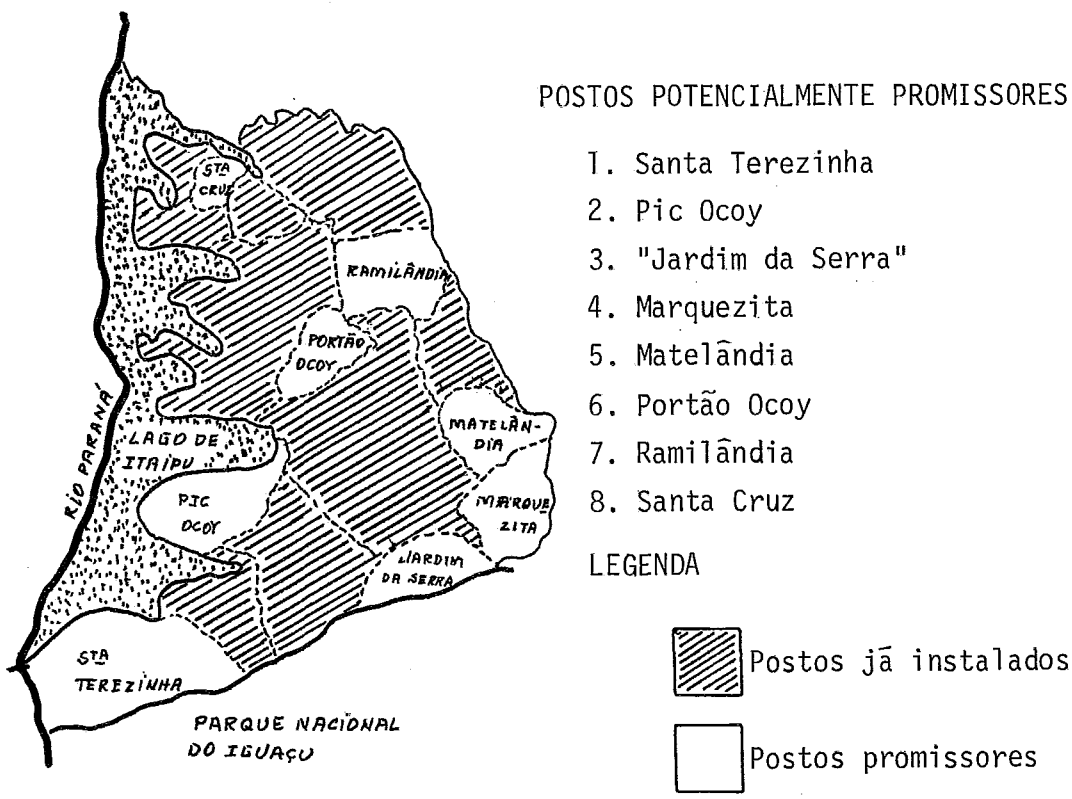


Fig. 2.3 - Localização e respectivas regiões de influências dos 8 Postos potencialmente promissores.

Estamos considerando, durante o estudo, que caso um Posto não seja efetivamente instalado, os agricultores de sua região de influência farão suas compras de insumos no Posto até agora utilizado (veja Tabela 2.1). Cabe esclarecer que os atuais Postos dão conta dos níveis atuais de demanda de insumos.

Desta forma a não instalação de um novo Posto não sobrecarregará um outro novo Posto que efetivamente venha ser instalado. Este fato simplificará bastante o estudo no momento em que formos dimensionar os novos Postos.

Região	Segunda Opção
1	Nova Roma
2	São Miguel
3	Medianeira
4	Medianeira
5	Medianeira e Cafeeira
6	Medianeira
7	Cafeeira
8	Santa Helena

Tab. 2.1 - Segunda opção na compra de insumos para os agricultores das regiões que não tiverem seus Postos instalados efetivamente.

2.3.2 - Dimensionamento dos 8 novos Postos

2.3.2.1 - Introdução

Cada um dos 8 novos Postos gerará ao redor de si uma região de influência. Sabendo-se as necessidades de insumos dos agricultores de cada uma dessas regiões, pode-se prever qual a quantidade de insumos a ser estocada em cada um dos novos Postos.

De posse dessa demanda será possível calcular a capacidade que deverá ter cada um dos novos Postos. Essa capacidade nos permitirá calcular o custo de construção de cada Posto.

Como os recursos financeiros são limitados, torna-se importante analisar alternativas de armazenamento que minimizem a capacidade estática de cada novo Posto, visando diminuir o custo de construção.

Uma das alternativas consiste em dimensionar cada Posto para estocar toda a necessidade anual de insumos a serem utilizados nas culturas de milho e soja. A outra alternativa leva em conta o comportamento diário de consumo de insumos, o que define a dimensão dos novos Postos de acordo com os piques diários de consumo.

2.3.2.2 - Dimensionamento pela demanda anual de insumos nas culturas de milho e soja

2.3.2.2.1 - Sazonalidade das culturas:

Na COTREFAL são plantadas basicamente 3 culturas: soja, milho e trigo. Soja e milho são semeados simultaneamente e o trigo em outra época do ano. Assim as terras são aproveitadas duas vezes anualmente.

Essa sazonalidade do cultivo faz com que a demanda por insumos agrícolas não seja uniformemente distribuida durante o ano. É, portanto, interessante aproveitar-se desse fato no dimensionamento dos Postos.

2.3.2.2.2 - Forma de reabastecimento dos estoques de insumos:

A COTREFAL reabastece seus atuais Postos anualmente com uma quantidade de insumos igual a demanda total (para milho, soja e trigo) anual.

Como as economias de escala para este caso são desprezíveis consideramos mais conveniente que ao invés de armazenar uma quantidade de insumos que satisfaça a demanda total anual referente às culturas de soja, milho e trigo, consideramos isoladamente dois períodos. O primeiro período implica no atendimento da demanda de insumos nas culturas de soja e milho enquanto no segundo período consideramos somente atendimento à demanda de insumos para o trigo.

Para o dimensionamento dos Postos entretanto, vemos que na prática basta considerarmos o período mais crítico, isto é, aquele onde a demanda de insumos é maior. É ele que será determinante no dimensionamento do Posto. No nosso estudo mostraremos que este período é aquele que atende às culturas de milho e soja.

2.3.2.2.3 - Previsão da demanda futura de insumos para cada cultura e por região:

Para se calcular as necessidades anuais de insumos para cada cultura e em cada região de influência dos 8 novos Postos, faz-se uma previsão a partir de dados reais fornecidos pela COTREFAL. Essa previsão da demanda futura se baseia na esperança (da COTREFAL) de que os níveis de venda de insumos aumentarão de 50%, em relação aos atuais níveis de venda aos associados, nas regiões onde efetivamente se instalar um Posto.

Portanto faremos primeiramente a previsão do que atualmente se vende na COTREFAL para os associados provenientes das regiões de influência dos novos Postos. Para fazer isto nos baseamos no "índice de fidelidade" (percentagem dos associados da COTREFAL que estão participando realmente da cooperativa) que é de 86%. Assim consideramos que 86% dos associados compram insumos na COTREFAL.

Consideremos então, para cada região de influência dos 8 novos Postos (veja Tabela 2.2), o número de associados (que de fato compram na

cooperativa), sua área média (ha) cultivada anualmente, a área total cultivada anualmente de cada cultura.

Região	Número de Associados	Área Média (ha) anual por associado	Área Total Anual (ha)		
			Milho	Soja	Trigo
1	155	30	930	3720	2325
2	103	24	741	1730	989
3	232	24	1114	4454	1948
4	30	23	276	414	34
5	34	17	289	289	58
6	26	22	343	228	29
7	30	18	378	162	16
8	103	23	355	2013	1184

Tab. 2.2 - Área total anual cultivada para cada tipo de cultura em cada região.

Tomando por base os dados da Tabela 2.3, peso dos insumos consumidos por unidade de área, para cada tipo de cultura, poderemos calcular a previsão da quantidade (em toneladas) de insumos anualmente comprados na COTREFAL em cada região e para cada tipo de cultura.

Cultura	Relação peso/área (kg/ha)
Soja	182,2
Milho	108,0
Trigo	216,0

Tab. 2.3 - Relação entre peso de insumos aplicados em cada tipo de cultura e a área cultivada.

Calcula-se então a previsão da quantidade (em toneladas) anual futura de insumos a serem comprados nos novos Postos da COTREFAL, discriminado por tipo de cultura (Tabela 2.4).

Região	Consumo atual (t)			Consumo futuro (t)		
	Milho	Soja	Trigo	Milho	Soja	Trigo
1	100	678	502	150	1017	753
2	80	315	214	120	473	320
3	120	812	421	180	1217	631
4	30	75	7	45	113	11
5	31	53	13	47	79	19
6	37	42	6	56	62	9
7	41	30	3	61	44	5
8	38	367	256	58	550	384

Tab. 2.4 - Consumo atual e futuro (50% acima do atual) de insumos em cada uma das culturas e por região.

2.3.2.2.4 - Dimensionamento dos novos Postos pelo critério da demanda anual

Conforme mostra a Tabela 2.4, as culturas de milho e soja é que requererão uma maior capacidade de estocagem de insumos dos Postos. Como, segundo a COTREFAL, cada tonelada de armazenagem estática requer 1 m^2 de área construída de Posto e considerando os limites de 300 m^2 e 500 m^2, definidos pela COTREFAL, teremos (Tabela 2.5) as seguintes previsões para as dimensões dos Postos:

Região	Consumo futuro (t) para milho e soja	Dimensão do Posto (m^2)
1	1168	500
2	593	500
3	1397	500
4	158	300
5	126	300
6	118	300
7	105	300
8	608	500

Tab. 2.5 - Consumo anual futuro de insumos nas culturas de milho e soja e dimensões calculadas dos Postos.

2.3.2.3 - Dimensionamento pelo comportamento do consumo diário de insumos

O critério do dimensionamento "pela demanda anual de insumos pa

ra milho e soja" não leva em conta como se distribui no tempo as retiradas (pelos compradores) dos insumos dos Postos. Essas retiradas são aleatórias existindo certas épocas de piques.

Se tivermos uma previsão desses piques (P_i), para cada região (i), e também do tempo (T) que COTREFAL levaria para recarregar o estoque de um Posto com uma quantidade igual ao pique, poderemos dimensionar cada Posto para estocar uma quantidade de insumos igual ao produto (P_i x T) e não teríamos, certamente, problemas de fornecimento de insumos aos agricultores.

Foi pedida à COTREFAL que estimasse (Tabela 2.6) esses piques para cada região.

Região (i)	Piques de consumo (t) (P_i)
1	162
2	90
3	180
4	75
5	75
6	45
7	45
8	60

Tab. 2.6 - Previsão dos piques de consumo diário de insumos em cada um dos Postos das 8 regiões.

Tendo em vista que o tempo de recarregamento (T) fornecido pela COTREFAL foi de 2 dias (T = 2), temos na Tabela 2.7 a capacidade que deveria possuir cada Posto segundo esse critério, bem como a dimensão (em m^2) que cada Posto deverá ter (considerando a equivalência 1 t ≡ 1 m^2 e os limites de 300 a 500 m^2 para a área dos Postos).

Região	Capacidade (t) calculada ($2 \times P_i$)	Dimensão (m^2)
1	324	324
2	180	300
3	360	360
4	150	300
5	150	300
6	90	300
7	90	300
8	120	300

Tab. 2.7 - Capacidade que os Postos de cada uma das 8 regiões deveriam apresentar e as dimensões que realmente terão face as limitações de área construída.

2.3.2.4 - Comparação dos dois critérios

O dimensionamento dos Postos através do critério da demanda anual de milho e soja tem a seu favor o fato de que se baseia numa praxe da COTREFAL, isto é, projetar para estocar toda a demanda (milho, soja e trigo) anual. Por outro lado isto conduz a Postos maiores do que o necessário quando comparamos com o dimensionamento através do critério dos piques. Em

bora esperássemos que esse segundo critério apresentasse uma redução maior nas dimensões calculadas para os Postos tal não ocorreu porque a limitação na área construída (300 a 500 m^2) contribuiu para assemelhar os resultados apresentados pelos dois critérios.

2.3.3 - Custo de construção dos postos

O custo de construção unitário ($Cr\$/m^2$) foi fornecido pela COTREFAL e é estimado pela equação (2.1):

$$c = 18.000 - 10\,a \quad p/ \quad 300 \leqslant a \leqslant 500 \qquad (2.1)$$

onde a é área (em m^2) construída e c é o custo unitário (em $Cr\$/m^2$).

Com este dado o custo de construção de cada Posto foi calculado e é apresentado na Tabela 2.8; como temos dois critérios de dimensionamento, há então dois custos para cada Posto.

Região	Custo de construção (Cr$)	
	Critério demanda anual	Critério dos piques
1	6.500.000,00	4.782.240,00
2	6.500.000,00	4.500.000,00
3	6.500.000,00	5.184.000,00
4	4.500.000,00	4.500.000,00
5	4.500.000,00	4.500.000,00
6	4.500.000,00	4.500.000,00
7	4.500.000,00	4.500.000,00
8	6.500.000,00	4.500.000,00
TOTAL	44.000.000,00	36.966.240,00

Tab. 2.8 - Custo total de construção dos Postos em cada região, segundo cada um dos dois critérios de dimensionamento.

2.3.4 - Recursos financeiros disponíveis

Os recursos financeiros com os quais conta a COTREFAL para construir os Postos são limitados. A COTREFAL dispõe ela própria de Cr$ 5.000.000,00 e é capaz de captar em empréstimos bancários mais Cr$ 30.000.000,00.

Portanto os recursos financeiros disponíveis são da ordem de Cr$ 35.000.000,00.

Consultando a última linha da Tabela 2.8 vê-se que esses recursos não dão para construir todos os 8 novos Postos potencialmente promissores, considerando qualquer um dos critérios de armazenagem.

2.3.5 - Sobra líquida gerada com a venda de insumos

A venda de insumos propicia à COTREFAL um percentual de "sobra líquida" que varia de 11% (no caso de venda para associado) a 18% (no caso de venda para não associado) sobre o preço (do produto) ao consumidor.

Entende-se por sobra líquida o preço de venda menos os custos diretos:

Sobra líquida = Preço de venda - custos diretos

Nos custos diretos inclui-se: preço de fábrica do produto, custo de fretes, seguros, perdas, despesas operacionais de manutenção dos Postos. Não se inclui nos custos diretos nenhuma parcela relativa à amortização do capital empregado na construção dos Postos.

O pagamento da construção deverá ser efetuado com recursos oriundos da sobra líquida obtida na venda dos insumos nos novos Postos que efetivamente se instalarem. É necessário então constatarmos se essa geração de recursos é capaz de cobrir o custo do financiamento da construção dos Postos.

2.3.5.1. Cálculo da sobra líquida gerada em cada Posto

Adotaremos que a sobra líquida gerada em cada Posto é de 11% sobre o preço de venda dos insumos. Para se estimar o montante (em Cr$) da venda de insumos nos novos Postos utilizaremos os dados da Tabela 2.4 (Demanda anual de insumos - em toneladas) e tomando-se o custo dos insumos por peso (Tabela 2.9), podemos calcular a sobra líquida gerada anualmente em cada um dos 8 Postos (Tabela 2.10).

Cultura	Relação preço/peso (Cr$/kg)
Milho	42,50
Soja	54,67
Trigo	75,56

Tab. 2.9 - Relação entre o peso dos insumos aplicados em cada tipo de cultura e o peso dos mesmos.

Região	Sobra líquida anual (1.000 Cr$) - Níveis atuais			
	Milho	Soja	Trigo	Total
1	467,5	4077,3	4172,4	8.717
2	374,0	1894,3	1778,7	4.047
3	561,0	4883,1	3499,2	8.943
4	140,3	451,0	58,2	649
5	144,9	318,7	108,1	572
6	173,0	252,6	49,9	475
7	191,7	180,4	24,9	397
8	177,7	2207,0	2127,0	4.512
TOTAL	—	—	—	28.312

Tab. 2.10 - Sobra líquida anualmente gerada pela venda de insumos em cada Posto (níveis atuais).

2.3.5.2 - Recursos gerados anualmente para pagamento do financiamento

Supõe-se que cada Posto que for efetivamente instalado aumentará em 50%, o nível de vendas na sua região, em relação ao atual nível de vendas para os agricultores daquela região.

Admitiremos então que esse aumento de 50% nas vendas, e consequentemente na sobra líquida gerada, é que pagará os custos do financiamento da construção. Na Tabela 2.11 estão calculados os recursos adicionais gerados que pagarão o financiamento, isto é, 50% dos atuais níveis de so

bra líquida gerada anualmente pela venda de insumos.

Região	Recursos adicionais gerados (1.000 Cr$)
1	4.358
2	2.024
3	4.472
4	325
5	286
6	238
7	199
8	2.256
TOTAL	14.156

Tab. 2.11 - Recursos adicionais gerados anualmente provenientes da sobra líquida de venda de insumos.

2.3.5.3 - Custo anual do financiamento da construção dos Postos

Evidentemente que devemos considerar nesse estudo a taxa de juros pela qual o financiamento será feito. Acontece porém que devemos considerar também o aumento inflacionário da geração de recursos através da venda de insumos. Além disso, ao contrário das parcelas do financiamento, essa geração de recursos segue um fluxo de caixa imprevisível pois não sabemos a quantidade mensal de recursos que será gerada.

Optamos então por uma simplificação do problema, qual seja a

de considerar nulas as taxas de juros e inflação.

Desta maneira o custo anual da construção de cada Posto fica sendo seu custo total dividido por 20, que é a vida útil de cada Posto. Na Tabela 2.12, mostra-se o custo anual do financiamento de cada Posto.

Região	Custo anual de construção (1000 Cr$)	
	Critério da demanda anual	Critério dos piques
1	325	239
2	325	225
3	325	259
4	225	225
5	225	225
6	225	225
7	225	225
8	325	225
	2.200	1.848

Tab. 2.12 - Custo anual do financiamento da construção de cada um dos 8 novos Postos.

2.3.5.4 - Comparação custo da construção x geração de recursos

Comparando os recursos gerados (Tabela 2.11) em cada região com os custos anuais de construção dos Postos (Tabela 2.12) verifica-se que todos Postos são auto financiáveis (exceto o Posto da região 7). Apesar da

construção do posto na região 7 não ser rentável, ele será considerado no nosso estudo face a um critério social de atendimento desta região. Isto é possível pois comparando-se os custos totais com os recursos totais adicionais gerados, verificamos que o sistema é globalmente auto-financiável.

2.3.6 - Número de associados e não-associados por região

A Tabela 2.13 dá o número de associados e não-associados por região. Estes dados serão necessários na definição dos coeficientes das funções objetivos dos problemas a serem analisados no próximo item.

Região	Associados	Não-associados
1	180	1 300
2	120	200
3	270	400
4	35	165
5	40	300
6	30	90
7	35	500
8	120	130

Tab. 2.13 - Número de agricultores associados e não-associados à COTREFAL por região.

2.4 - FORMULAÇÃO MATEMÁTICA

Esse estudo visa fazer uma comparação entre as soluções obtidas dos seguintes problemas:

Problema P_{A_1}

Selecionar dentre 8 Postos potencialmente promissores aqueles cujo custo total de construção (calculado pelo critério da demanda anual de insumos nas culturas de milho e soja) não ultrapasse Cr$ 35 milhões e que maximize o número de associados que passarão a ter um Posto mais perto de sua propriedade.

Matematicamente:

$$\text{Max } z = 180x_1 + 120x_2 + 270x_3 + 35x_4 + 40x_5 + 30x_6 + 35x_7 + 120x_8$$

sujeito a

$$6{,}5x_1 + 6{,}5x_2 + 6{,}5x_3 + 4{,}5x_4 + 4{,}5x_5 + 4{,}5x_6 + 4{,}5x_7 + 6{,}5x_8 \leq 35{,}0$$

$$x_i \in \{0,1\} \ \forall \ i = 1, \ldots, 8$$

onde: a) $x_i = 0$ significa que o Posto i não será instalado;

b) $x_i = 1$ significa que o Posto i será instalado;

c) os coeficientes da função objetivo são os números de associados, conforme dados da Tabela 2.13; e

d) os coeficientes da restrição são os custos de construção (em milhões Cr$) pelo critério da demanda anual, conforme dados da Tabela 2.8.

Problema P_{A_2}

Selecionar dentre 8 Postos potencialmente promissores aqueles cujo custo total de construção (calculado pelo critério dos piques diários) não ultrapasse Cr$ 35 milhões e que maximize o número de associados que passarão a ter um Posto mais perto de sua propriedade.

Matematicamente:

$$\text{Max } z = 180x_1 + 120x_2 + 270x_3 + 35x_4 + 40x_5 + 30x_6 + 35x_7 + 120x_8$$

sujeito a

$$4{,}8x_1 + 4{,}5x_2 + 5{,}2x_3 + 4{,}5x_4 + 4{,}5x_5 + 4{,}5x_6 + 4{,}5x_7 + 4{,}5x_8 \leq 35{,}0$$

$$x_i \in \{0, 1\} \; \forall \; i = 1, \ldots, 8$$

onde: a) $x_i = 0$ significa Posto i não será instalado;

b) $x_i = 1$ significa Posto i será instalado;

c) os coeficientes da função objetivo são os números de associados, conforme dados da Tabela 2.13; e

d) os coeficientes da restrição são os custos de construção (em milhões Cr$) pelo critério dos piques, conforme dados da Tabela 2.8.

Problema P_{B_1}

Selecionar dentre 8 Postos potencialmente promissores aqueles cujo custo total de construção (calculado pelo critério da demanda anual de insumos nas culturas de milho e soja) não ultrapasse Cr$ 35 milhões e que maximize o número de não-associados que passarão a ter um Posto mais

perto de sua propriedade.

Matematicamente:

$$\text{Max } z = 1300x_1 + 200x_2 + 400x_3 + 165x_4 + 300x_5 + 90x_6 + 500x_7 + 130x_8$$

sujeito a

$$6{,}5x_1 + 6{,}5x_2 + 6{,}5x_3 + 4{,}5x_4 + 4{,}5x_5 + 4{,}5x_6 + 4{,}5x_7 + 6{,}5x_8 \leq 35{,}0$$

$$x_i \in \{0, 1\} \; \forall \; i = 1, \ldots, 8$$

onde: a) $x_i = 0$ significa Posto i não será instalado;

b) $x_i = 1$ significa Posto i será instalado;

c) os coeficientes da função objetivo são o número de não associados, conforme Tabela 2.13; e

d) os coeficientes da restrição são os custos de construção (em milhões Cr$) pelo critério da demanda anual, conforme dados da Tabela 2.8.

Problema P_{B_2}

Selecionar dentre 8 Postos potencialmente promissores aqueles cujo custo total de construção (calculado pelo critério dos piques diários) não ultrapasse Cr$ 35 milhões e que maximize o número de não associados que passarão a ter um Posto mais perto de sua propriedade.

Matematicamente:

$$\text{Max } z = 1300x_1 + 200x_2 + 400x_3 + 165x_4 + 300x_5 + 90x_6 + 500x_7 + 130x_8$$

sujeito a

$$4,8x_1 + 4,5x_2 + 5,2x_3 + 4,5x_4 + 4,5x_5 + 4,5x_6 + 4,5x_7 + 4,5x_8 \leq 35,0$$

$$x_i \in \{0, 1\} \; \forall \; i = 1, \ldots, 8$$

onde: a) $x_i = 0$ significa Posto i não será instalado;

b) $x_i = 1$ significa Posto i será instalado;

c) os coeficientes da função objetivo são os números de não associados, conforme Tabela 2.13; e

d) os coeficientes da restrição são os custos de construção (em milhões Cr$) pelo critério dos piques, conforme dados da Tabela 2.8.

Problema P_{C_1}

Selecionar dentre 8 Postos potencialmente promissores aqueles cujo custo total de construção (calculado pelo critério da demanda anual de insumos nas culturas de milho e soja) não ultrapasse Cr$ 35 milhões e que maximize a sobra líquida adicional gerada pela venda de insumos nos Postos que efetivamente forem instalados.

Matematicamente:

$$\text{Max } z = 4,36x_1 + 2,02x_2 + 4,48x_3 + 0,33x_4 + 0,29x_5 + 0,24x_6 + 0,20x_7 + 2,26x_8$$

sujeito a

$$6,5x_1 + 6,5x_2 + 6,5x_3 + 4,5x_4 + 4,5x_5 + 4,5x_6 + 4,5x_7 + 6,5x_8 \leq 35,0$$

$$x_i \in \{0, 1\} \; \forall \; i = 1, \ldots, 8$$

onde: a) $x_i = 0$ significa Posto i não será instalado;

b) $x_i = 1$ significa Posto i será instalado;

c) os coeficientes da função objetivo são as sobras líquidas (em milhões Cr$) adicionais (50% do nível atual), conforme Tabela 2.11; e

d) os coeficientes da restrição são os custos de construção (em milhões Cr$) pelo critério da demanda anual, conforme dados da Tabela 2.8.

Problema P_{C_2}

Selecionar dentre 8 Postos potencialmente promissores aqueles cujo custo total de construção (calculado pelo critério dos piques diários) não ultrapasse Cr$ 35 milhões e que maximize a sobra líquida adicional gerada pela venda de insumos nos Postos que efetivamente forem instalados.

Matematicamente:

$$\text{Max } x = 4{,}36x_1 + 2{,}02x_2 + 4{,}48x_3 + 0{,}33x_4 + 0{,}29x_5 + 0{,}24x_6 + 0{,}20x_7 + 2{,}26\,x_8$$

sujeito a

$$4{,}8x_1 + 4{,}5x_2 + 5{,}2x_3 + 4{,}5x_4 + 4{,}5x_5 + 4{,}5x_6 + 4{,}5x_7 + 4{,}5x_8 \leq 35{,}0$$

$$x_i \in \{0, 1\} \;\forall\; i = 1, \ldots, 8$$

onde: a) $x_i = 0$ significa Posto i não será instalado;

b) $x_i = 1$ significa Posto i será instalado;

c) os coeficientes da função objetivo são as sobras líquidas (em milhões Cr$) adicionais (50% do nível atual), conforme Tabela 2.11; e

d) os coeficientes da restrição são os custos de construção (em milhões Cr$) pelo critério dos piques, conforme dados da Tabela 2.8.

2.5. SOLUÇÕES OBTIDAS

Cada um dos 6 problemas foi resolvido utilizando-se algoritmos desenvolvidos para o Problema da Mochila 0-1 e foram encontradas as seguintes soluções:

Problema P_{A_1}

Solução: $x^* = (1,1,1,1,1,0,0,1)^T$; $z^* = 767$

Problema P_{A_2}

Solução: $x^* = (1,1,1,1,1,0,1,1)^T$; $z^* = 802$

Problema P_{B_1}

Solução: $x^* = (1,1,1,1,1,0,1,0)^T$; $z^* = 2865$

Problema P_{B_2}

Solução: $x^* = (1,1,1,1,1,0,1,1)^T$; $z^* = 2995$

Problema P_{C_1}

Solução: $x^* = (1,1,1,1,1,0,0,1)^T$; $z^* = 13,74$

Problema P_{C_2}

Solução: $x^* = (1,1,1,1,1,1,0,1)^T$; $z^* = 13,98$

2.6 - CONCLUSÕES

2.6.1 - Resumo das soluções obtidas

Para facilitar esta análise das soluções obtidas para os problemas, construimos a Tabela 2.14.

Postos (Região.Localização)	Problemas					
	P_{A_1}	P_{A_2}	P_{B_1}	P_{B_2}	P_{C_1}	P_{C_2}
1. Santa Terezinha	S	S	S	S	S	S
2. Pic Ocoy	S	S	S	S	S	S
3. "Jardim da Serra"	S	S	S	S	S	S
4. Marquezita	S	S	S	S	S	S
5. Matelândia	S	S	S	S	S	S
6. Portão Ocoy	N	N	N	N	N	S
7. Ramilândia	N	S	S	S	N	N
8. Santa Cruz	S	S	N	S	S	S

Tab. 2.14 - Postos que serão efetivamente construídos segundo as soluções obtidas para os problemas propostos (S = Será construído, N = Não será construído).

A Tabela 2.15 resume o que quer dizer cada um dos problemas analisados.

Custo de Construção	Objetivos		
	A. Maximizar nº associados beneficiados	B. Maximizar nº não-associados beneficiados	C. Maximizar sobra líquida
1. Critério da demanda anual de insumos nas culturas de milho e soja	P_{A_1}	P_{B_1}	P_{C_1}
2. Critério dos piques diários de consumo de insumos	P_{A_2}	P_{B_2}	P_{C_2}

Tab. 2.15 - Critérios de cálculo de custo de construção e objetivos que foram pesquisados em cada problema proposto.

2.6.2 - Conclusões

Analisando as soluções obtidas constata-se que conforme o objetivo a ser maximizado, obtêm-se soluções diferentes. Isto pode ser visto comparando entre si as soluções de problemas com mesmo critério de cálculo de custo de construção, isto é, P_{A_1} , P_{B_1} e P_{C_1} ou P_{A_2}, P_{B_2} e P_{C_2} .

Também verifica-se que comparando-se problemas com o mesmo objetivo mas diferentes critérios de cálculo de custo de construção, isto é, P_{A_1} com P_{A_2} , ou P_{B_1} com P_{B_2} , ou P_{C_1} com P_{C_2} , obtêm-se a instalação de mais um Posto quando se passa do critério da demanda anual para o critério dos piques. Isto acontece porque os custos calculados pelo critério dos piques são menores.

As soluções aqui obtidas se assemelham bastante porque os recursos financeiros disponíveis quase são suficientes para construir todos os 8 Postos potencialmente promissores. Numa situação onde os recursos financeiros são mais "apertados" obteria-se certamente soluções bem diferentes para objetivos e critérios de cálculo de construção como esses aqui empregados.

Deve-se salientar que na modelagem procuramos nos ater ao máximo à metodologia vigente na cooperativa. O simples fato de se procurar otimizar matematicamente a aplicação dos recursos financeiros já constitui um grande avanço em relação ao que constatamos. De qualquer forma fica para um trabalho futuro introduzir conceitos novos, notadamente no que diz respeito à localização e dimensionamento dos Postos.

CAPÍTULO 3

O PROBLEMA DA MOCHILA 0-1

3.1 - INTRODUÇÃO

Suponhamos que se deseje encher uma mochila com n objetos cada um pesando p_i e possuindo um valor relativo v_i de tal forma que não se ultrapasse um determinado peso total M e que se maximize o valor total dos objetos que encherão a mochila.

Matematicamente o Problema da Mochila 0-1 (PM01), como é conhecido o problema exposto acima, pode ser expresso por:

$$\boxed{P_0} \qquad \text{Max} \sum_{i=1}^{n} v_i x_i \tag{3.1}$$

$$\text{sujeito a} \sum_{i=1}^{n} p_i x_i \leqslant M \tag{3.2}$$

$$0 \leqslant x_i \leqslant 1 \qquad i = 1, \ldots, n \tag{3.3}$$

$$x_i \text{ inteiro} \qquad i = 1, \ldots, n \tag{3.4}$$

Algumas considerações sobre as constantes (p_i, v_i e M) são feitas:

$$M, v_i \text{ e } p_i \text{ são inteiros positivos} \tag{3.5}$$

$$p_i \leqslant M, \forall i = 1, \ldots, n \tag{3.6}$$

$$\sum_{i=1}^{n} p_i > M \qquad (3.7)$$

Se algum p_i ou v_i ou M for fracionário, violando a condição (3.5), isto pode ser contornado multiplicando-se (3.1) ou (3.2) por um fator conveniente. E ainda, se tais constantes são negativas Glover [12] apresenta uma maneira de contornar o problema.

A condição (3.6) assegura que não há objeto que possa ser excluido, a priori, da mochila por possuir peso maior que o peso total permissível da mochila.

Se a condição (3.7) é violada isto significa que a solução ótima do PMO1 será trivial, isto é, todos os objetos deverão ser colocados na mochila. Evita-se este tipo de solução.

Além disso estaremos sempre considerando que as variáveis do PMO1 estão ordenadas não crescentemente segundo a razão valor/peso (v_j/p_j), isto é:

$$v_1/p_1 \geq v_2/p_2 \geq \ldots \geq v_n/p_n \qquad (3.8)$$

A justificativa para tal se deve ao fato de que a solução ótima de um PMO1 relaxado, isto é, um PMO1 sem a restrição de integralidade (3.4) (ver Apêndice A), é obtida fazendo igual a 1 aquelas variáveis que apresentam maior razão valor/peso, sem que se ultrapasse o peso total M da mochi

la; a próxima variável, segundo a ordenação não crescente das razões valor/peso, assume um valor fracionário, o bastante para completar o peso ainda não preenchido da mochila; as demais variáveis assumem o valor 0. Desta forma, ordenando as variáveis segundo a relação (3.8), achar a solução de um PMO1 relaxado fica bastante trivial, e como precisaremos, nos algoritmos aqui mostrados e desenvolvidos, resolver freqüentemente um PMO1 relaxado, tal ordenação facilitará bastante essa tarefa.

O que nos motivou estudar o problema da mochila foi a modelagem do problema de Instalação de Postos de Atendimento e Venda de Insumos para a Cooperativa Três Fronteiras - COTREFAL. Kolesar |[16]| cita diversas aplicações onde aparece o PMO1 como modelo. Entre tais aplicações podemos citar:

1) Operações de carregamento de navios;
2) Investimentos de capital;
3) Problema do corte;
4) Confiabilidade de redes; etc.

Além disso o PMO1 surge como um subproblema em certos algoritmos de programação inteira.

A necessidade de se obter métodos eficientes de solucionar o PMO1 fica justificada por essa gama de aplicações que o mesmo possui.

O Problema da Mochila 0-1 é NP-completo |[27]|, isto é, se se conhecer um algoritmo polinomial que o solucione então será possível construir

algoritmos polinomiais para uma variedade bem grande de problemas NP-completos. Nenhum problema NP-completo pode ser resolvido por algum algoritmo polinomial até hoje conhecido.

Diversas são as maneiras de se resolver o PM01, por exemplo, pode-se encará-lo como um problema genérico de programação inteira e resolvê-lo como tal utilizando-se o método dos planos de corte de Gomory, ou o algoritmo especializado para variáveis 0-1 de Balas, ou qualquer outro |[11]|.

Alguns autores, veja Kolesar |[16]|, resolveram o PM01 como um problema de caminho mínimo numa rede. Quando o peso total (M) da mochila é pequeno (comparado com $\sum_{i=1}^{n} p_i$), os algoritmos baseados em programação dinâmica, como os de Toth |[31]|, Horowitz-Sahni |[14]|, resolvem eficientemente o problema.

Entretanto, a maioria dos trabalhos publicados exploram a técnica "branch-and-bound", que produz algoritmos cuja eficiência computacional (em termos de tempo de execução, memória utilizada e flexibilidade) tem sido a melhor que se obteve até hoje, veja Suhl |[30]|.

Neste trabalho nos deteremos na análise de algoritmos tipo "branch-and-bound". Proporemos também um algoritmo "branch-and-bound" que incorpora algumas melhorias introduzidas até hoje e sugerimos uma pequena contribuição por nós desenvolvida.

O restante deste capítulo, dedicado ao PM01, será dividido em

6 seções e 3 apêndices.

Na seção seguinte apresentaremos um exemplo numérico de um PMO1 com 3 variáveis. Resolveremos este exemplo através de um esquema "branch-and-bound" no qual poderemos visualizar geometricamente os caminhos percorridos até se achar a solução ótima.

Na seção 3.3 tentamos inferir os principais procedimentos que devem apresentar um algoritmo "branch-and-bound" para solucionar o PMO1, sem especificar nenhum algoritmo em particular.

Na seção 3.4 faremos um histórico do desenvolvimento dos algoritmos "branch-and-bound" para solucionar o PMO1, desde o primeiro deles, o de Kolesar |16|, até os mais recentes. Nessa descrição não nos preocuparemos com detalhes mas sim com as melhorias que cada um diz ter introduzido.

Na seção 3.5 apresentamos o algoritmo "branch-and-bound" por nós desenvolvido.

E na última seção faremos uma análise computacional deste algoritmo comparando-o com o "branch-and-bound" de Horowitz-Sahni |14|, que constitui o algoritmo de partida e de referência para boa parte dos atuais algoritmos existentes.

Além dessas 6 seções, desenvolveremos três apêndices. No Apên

dice A mostramos como calcular a solução ótima de um PM01 relaxado, isto é, sem as restrições de integralidade (3.4). No Apêndice B damos alguns exemplos de como calcular limitantes superiores do valor ótimo de um PM01, e sugerimos um novo método que conduz a limitantes melhores que os até então existentes. E no Apêndice C apresentamos a listagem do algoritmo por nós desenvolvido.

3.2 - UM EXEMPLO NUMÉRICO DO PM01

3.2.1 - Introdução

Com a finalidade de dar uma visão global da aplicação de um algoritmo tipo "branch-and-bound" na solução do Problema da Mochila 0-1 resolveremos um exemplo com 3 variáveis. Além de nos mostrar os aspectos mais gerais e importantes dos "branch-and-bound", isto nos permitirá a visualização gráfica do método, fortalecendo assim nossa intuição para melhor aplicar e explorar as vantagens que tal método oferece.

Após o desenvolvimento do "branch-and-bound" na solução do exemplo, passaremos a uma apresentação mais formal e mais geral de um algoritmo "branch-and-bound" para resolver o Problema da Mochila 0-1

3.2.2. - Exemplo

Resolver o seguinte Problema da Mochila 0-1:

P_o

$$\text{Max } z = 5x_1 + 6x_2 + 7x_3 \tag{3.9}$$

$$\text{tal que } x_1 + 2x_2 + 3x_3 \leq 4 \tag{3.10}$$

$$0 \leq x_i \leq 1 \quad \forall i = 1, 2, 3 \tag{3.11}$$

$$x_i \text{ inteiro} \quad \forall i = 1, 2, 3 \tag{3.12}$$

Solução

Se no problema P_o abandonarmos as restrições de integralidade (3.12) formaremos um novo problema que denominaremos de P_o relaxado e que denotaremos por $\overline{P_o}$:

$\overline{P_o}$

$$\text{Max } z = 5x_1 + 6x_2 + 7x_3 \tag{3.9}$$

$$\text{tal que } x_1 + 2x_2 + 3x_3 \leq 4 \tag{3.10}$$

$$0 \leq x_i \leq 1 \quad \forall i = 1, 2, 3 \tag{3.11}$$

Todas as soluções viáveis de P_o são também viáveis de $\overline{P_o}$, porém a recíproca não é verdadeira, isto é, nem todas as soluções viáveis de $\overline{P_o}$ são viáveis de P_o. Na figura 3.1 mostramos a região de soluções viáveis de $\overline{P_o}$ enquanto que na figura 3.2 mostramos as soluções viáveis de P_o.

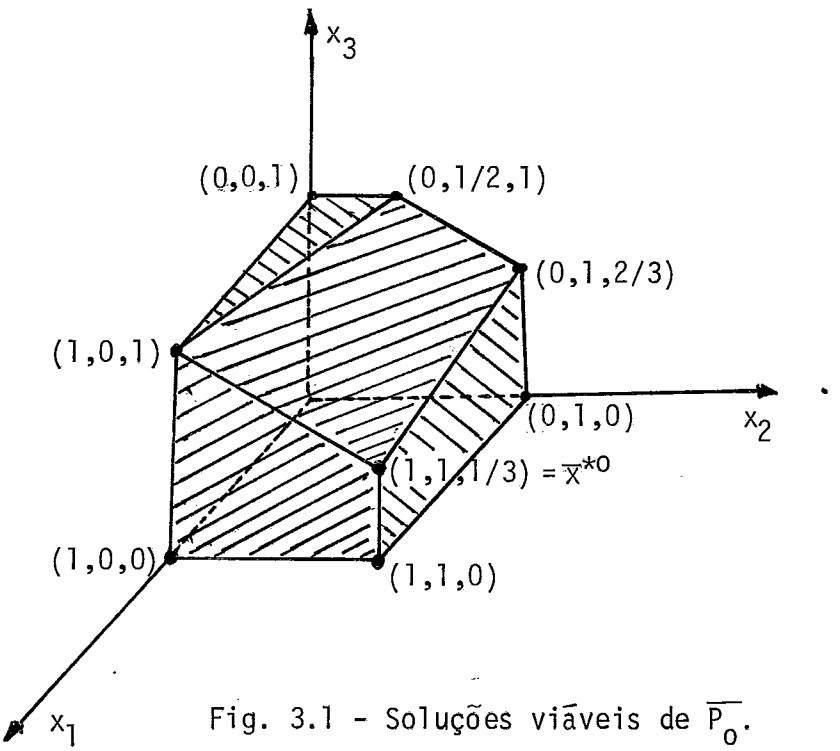


Fig. 3.1 - Soluções viáveis de $\overline{P_o}$.

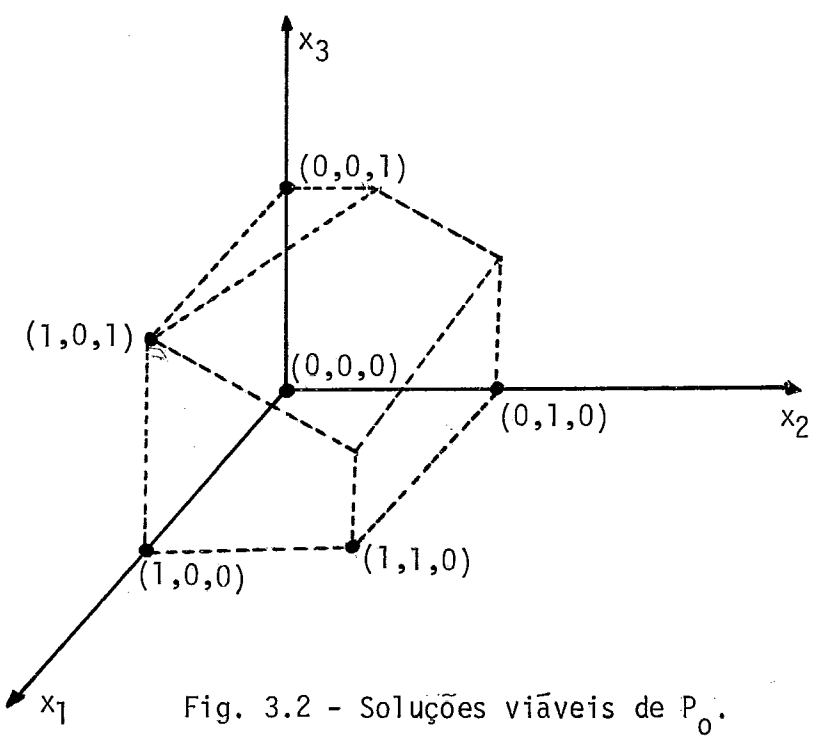


Fig. 3.2 - Soluções viáveis de P_o.

O problema $\overline{P_o}$ é um problema de programação linear (PPL) e portanto a sua solução ótima ($\overline{x}^{*o}$) estará num dos vértices da região de soluções viáveis de $\overline{P_o}$. Podemos verificar então que $\overline{x}^{*o} = (1, 1, 1/3)^T$ cu

jo valor é $\overline{z}^{*o} = 13{,}33$.

Se a solução ótima de $\overline{P_o}$ $(\overline{x}^{*o})$ fosse inteira então ela seria a solução ótima de P_o (x^{*o}) também, e terminaria assim a resolução do problema proposto.

Mas mesmo não sendo inteira a solução $\overline{x}^{*o}$, tomaremos a solução correspondente ao vértice vizinho de $\overline{x}^{*o}$, isto é, $(1,1,0)^T$ que corresponde a anular a variável fracionária da solução $\overline{x}^{*o}$. Tal solução, (1,1,0), é viável de P_o e passaremos a chamá-la de melhor solução viável (de P_o) até então encontrada e a denotaremos por $x^L = (1,1,0)^T$ e seu valor $z^L = 11$.

A solução x^L fornece um limitante inferior (z^L) para o valor (z^{*o}) da solução ótima (x^{*o}) de P_o:

$$z^L \leqslant z^{*o} \tag{3.13}$$

Nosso próximo passo na busca da solução ótima de P_o é expandir P_o, isto é, particionar a região de soluções viáveis de P_o em duas e promover a busca separadamente nessas duas regiões. Uma maneira simples de particionar a região de soluções viáveis de P_o é fixar uma das variáveis em 0 e em 1. Assim, fazendo $x_1 = 0$ geramos o problema P_1 e fazendo $x_1 = 1$ geramos P_2. Essa expansão (branching) de P_o em P_1 e P_2 é costumeiramente representada (veja Fig. 3.3) na forma de uma árvore. As variáveis não fixadas (x_2 e x_3) chamaremos de variáveis livres.

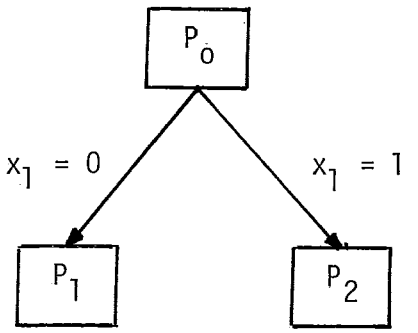


Fig. 3.3 - Árvore do "branch-and-bound".

P_1

$$\text{Max } z = 5x_1 + 6x_2 + 7x_3 \tag{3.9}$$

$$\text{tal que } x_1 + 2x_2 + 3x_3 \leqslant 4 \tag{3.10}$$

$$x_1 = 0\ ,\quad 0 \leqslant x_i \leqslant 1 \quad \forall i = 2, 3 \tag{3.14}$$

$$x_i \text{ inteiro} \quad \forall i = 1, 2, 3 \tag{3.12}$$

P_2

$$\text{Max } z = 5x_1 + 6x_2 + 7x_3 \tag{3.9}$$

$$\text{tal que } x_1 + 2x_2 + 3x_3 \leqslant 4 \tag{3.10}$$

$$x_1 = 1\ ,\quad 0 \leqslant x_i \leqslant 1 \quad \forall i = 2, 3 \tag{3.15}$$

$$x_i \text{ inteiro} \quad \forall i = 1, 2, 3 \tag{3.12}$$

Nas figuras 3.4 e 3.5 mostramos graficamente as regiões de soluções viáveis de P_1 e P_2 resultantes da divisão da região de soluções viáveis de P_o.

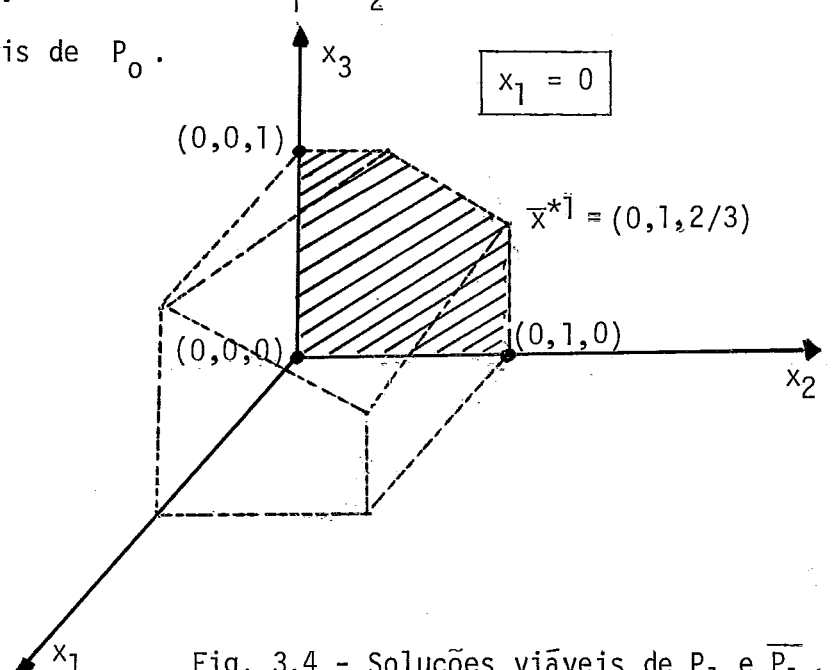


Fig. 3.4 - Soluções viáveis de P_1 e $\overline{P_1}$.

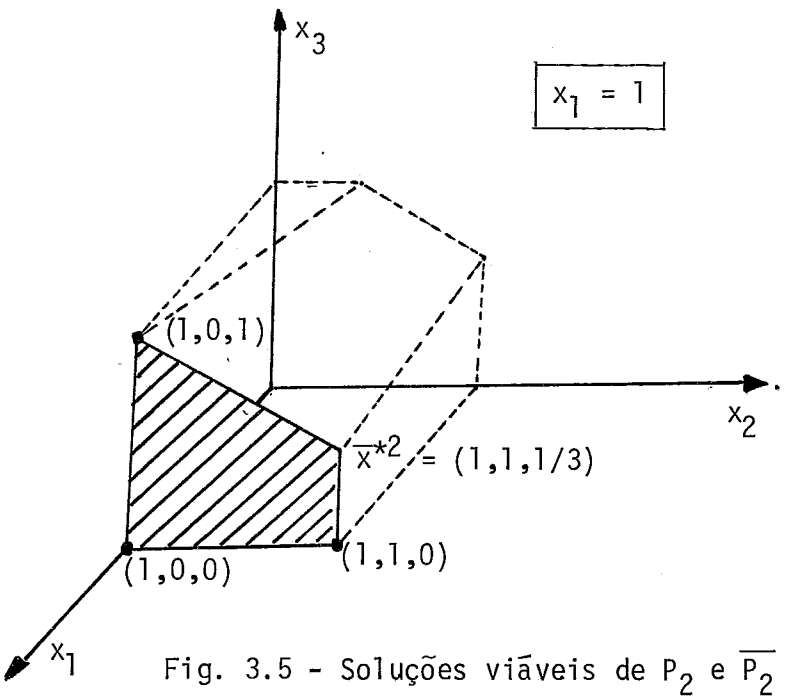


Fig. 3.5 - Soluções viáveis de P_2 e $\overline{P_2}$.

Pode-se observar que cada solução viável de P_0 ou é solução viável de P_1 ou de P_2, portanto a solução ótima de P_0 ou será a solução ótima de P_1 ou de P_2.

Relaxando os problemas P_1 e P_2 obtemos $\overline{P_1}$ e $\overline{P_2}$ cujas regiões de soluções viáveis são as áreas hachuradas nas figuras 3.4 e 3.5, respectivamente. A solução desses dois PPL's nos dá:

$$\text{Para } \overline{P_1}: \quad \bar{x}^{*1} = (0,\ 1,\ 2/3)^T, \quad \bar{z}^{*1} = 10{,}66 \tag{3.16}$$

$$\text{Para } \overline{P_2}: \quad \bar{x}^{*2} = (1,\ 1,\ 1/3)^T, \quad \bar{z}^{*2} = 13{,}33 \tag{3.17}$$

Da teoria de Programação linear sabemos que as soluções ótimas de P_1 e P_2 são tais que seus valores ótimos (z^{*1} e z^{*2}, respectivamente) não ultrapassam os valores ótimos de $\overline{P_1}$ e $\overline{P_2}$, respectivamente:

$$z^{*1} \leq 10,66 \qquad (3.18)$$

$$z^{*2} \leq 13,33 \qquad (3.19)$$

Portanto, o valor da solução ótima dos problemas relaxados constituem um limitante superior para o valor da solução ótima dos problemas não-relaxados.

Pela relação (3.13) sabemos que a solução ótima (z^{*0}) de P_0 é tal que seu valor é melhor que z^L, que no momento vale 11, logo $z^{*0} \geq 11$.

A relação (3.18) nos diz que a solução ótima de P_1 tem valor menor ou igual a 10,66, logo essa solução não pode ser ótima de P_0 pois esta é maior que 11.

Portanto sabemos que a solução ótima de P_0 não pode estar entre as soluções viáveis de P_1, isto é, provamos a não-otimalidade (para P_0) das soluções viáveis de P_1. Desta forma as soluções viáveis de P_1 serão abandonadas na busca da solução ótima de P_0, isto é, o problema P_1 não mais será expandido.

Fazendo esse mesmo raciocínio em relação ao problema P_2 não conseguiremos provar a não-otimalidade das soluções viáveis de P_2, isto porque $\overline{z}^{*2} = 13,33$ é maior que $z^L = 11$.

A comparação do limitante superior (z^{*i}) do valor ótimo de cada problema (P_i) com o valor da melhor solução até então encontrada (z^L)

nos permite podar a árvore representativa do "branch-and-bound", isto é, se a árvore é podada no nó correspondente ao problema P_i, este não será mais expandido; (o conjunto de soluções viáveis de P_i não será mais particionado). Essa poda só ocorrerá se o limitante superior (z^*i) do valor ótimo de P_i for menor que o valor (z^L) da melhor solução até então encontrada (x^L).

A solução ótima de P_0 é a solução ótima de P_2. Como a solução ótima de $\overline{P_2}$ é não inteira, particionaremos a região de soluções viáveis de P_2 em duas (fixando x_2 em 0 e em 1) e promoveremos a busca da solução ótima nessas duas regiões, o que constitui, na verdade, na solução dos problemas P_3 e P_4 gerados a partir de P_2 fazendo $x_2 = 0$ e $x_2 = 1$, respectivamente.

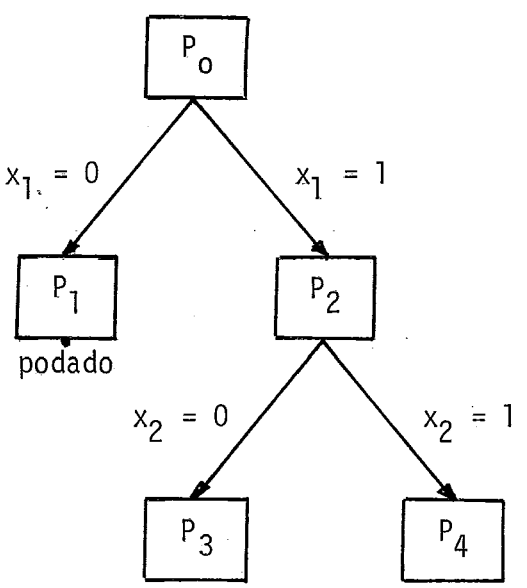


Fig. 3.6 - Árvore do "branch-and-bound"

P_3

$$\text{Max} \quad z = 5x_1 + 6x_2 + 7x_3 \tag{3.9}$$

$$\text{tal que} \quad x_1 + 2x_2 + 3x_3 \leqslant 4 \tag{3.10}$$

$$x_1 = 1 \ , \quad x_2 = 0 \ , \quad 0 \leqslant x_3 \leqslant 1 \tag{3.20}$$

$$x_i \text{ inteiro} \quad \forall i = 1, 2, 3 \tag{3.12}$$

P_4

$$\text{Max} \quad z = 5x_1 + 6x_2 + 7x_3 \tag{3.9}$$

$$\text{tal que} \quad x_1 + 2x_2 + 3x_3 \leq 4 \tag{3.10}$$

$$x_1 = 1 \ , \quad x_2 = 1 \ , \quad 0 \leq x_3 \leq 1 \tag{3.21}$$

$$x_i \text{ inteiro} \quad \forall i = 1, 2, 3 \tag{3.12}$$

Nas figuras 3.7 e 3.8 mostramos graficamente as regiões de soluções viáveis de P_3 e P_4, respectivamente.

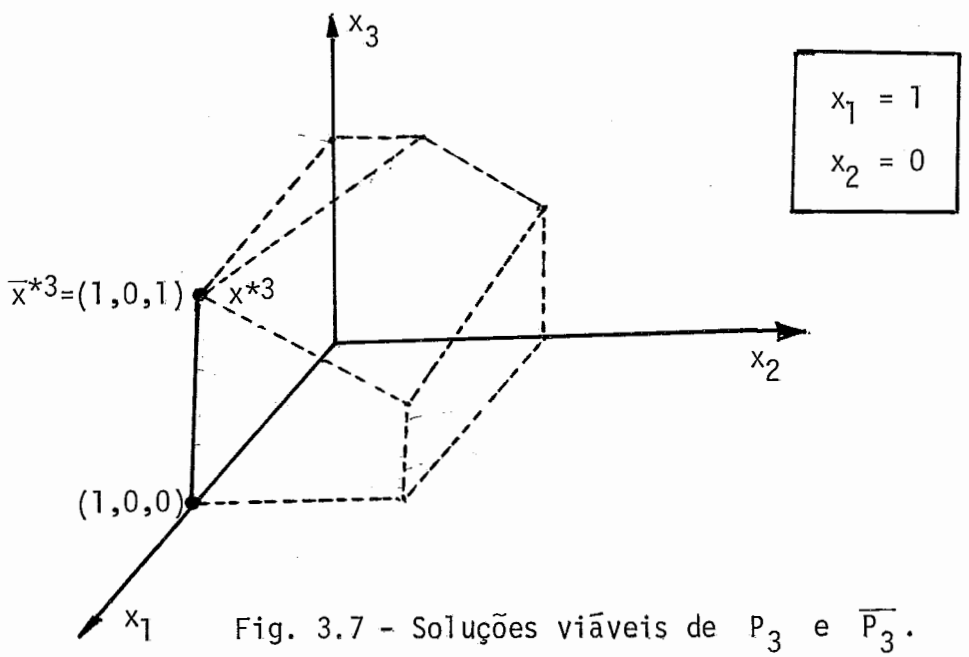


Fig. 3.7 - Soluções viáveis de P_3 e $\overline{P_3}$.

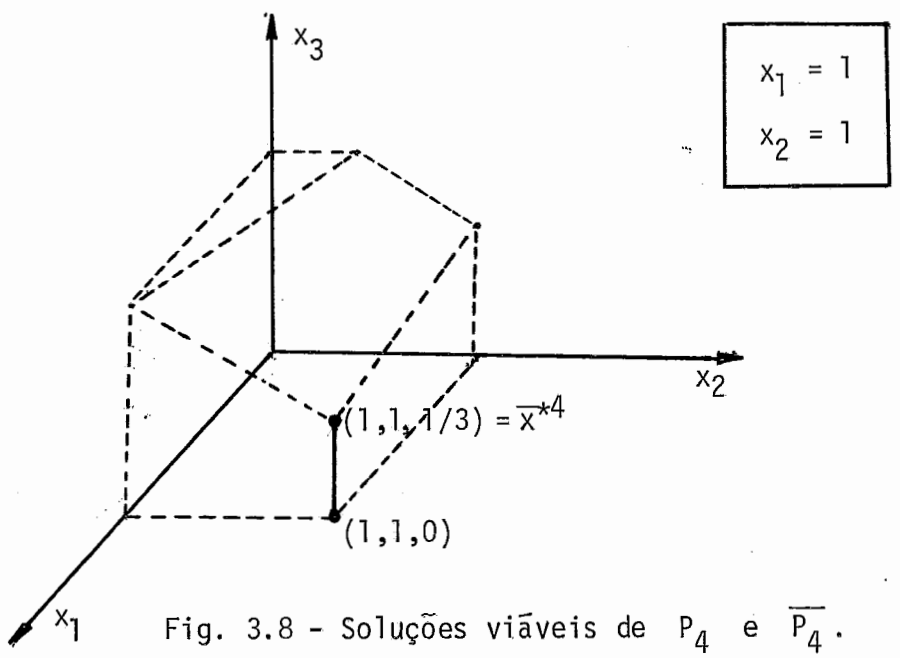


Fig. 3.8 - Soluções viáveis de P_4 e $\overline{P_4}$.

Novamente observa-se que as soluções viáveis de P_2 ou são soluções viáveis de P_3 ou de P_4.

Relaxando os problemas P_3 e P_4 obteremos os PPL's $\overline{P_3}$ e $\overline{P_4}$ cujas regiões de soluções viáveis são as linhas cheias nas figuras 3.7 e 3.8, respectivamente. Resolvendo-os teremos:

$$\text{Para } \overline{P_3}: \ \overline{x}^{*3} = (1, 0, 1)^T, \ \overline{z}^{*3} = 12 \qquad (3.22)$$

$$\text{Para } P_4: \ \overline{x}^{*4} = (1, 1, 1/3)^T, \ \overline{z}^{*4} = 13{,}33 \qquad (3.23)$$

Tomando por base a solução do problema $\overline{P_3}$ vemos que ela é inteira e portanto a solução ótima de P_3 será $x^{*3} = (1, 0, 1)^T$ cujo valor é $z^{*3} = 12$. Como esta solução é melhor que x^L, atualizaremos a melhor solução viável até então encontrada, isto é, $x^L = (1, 0, 1)^T$ e $z^L = 12$.

Notemos também que não precisaremos mais expandir P_3 pois achamos a solução ótima dele. Portanto, uma outra situação onde se deve podar a árvore é quando se obtém uma solução inteira para o problema relaxado, pois neste caso tal solução é a solução do problema não relaxado.

Tomando por base a solução de $\overline{P_4}$ vemos que a sua solução é não inteira e que o valor da solução ótima de P_4 é limitada superiormente em 13,33 que é maior que o z^L (= 12) atual. Devemos portanto expandir P_4.

Fixando x_3, no problema P_4, em 0 e em 1 obtemos os problemas P_5 e P_6, respectivamente.

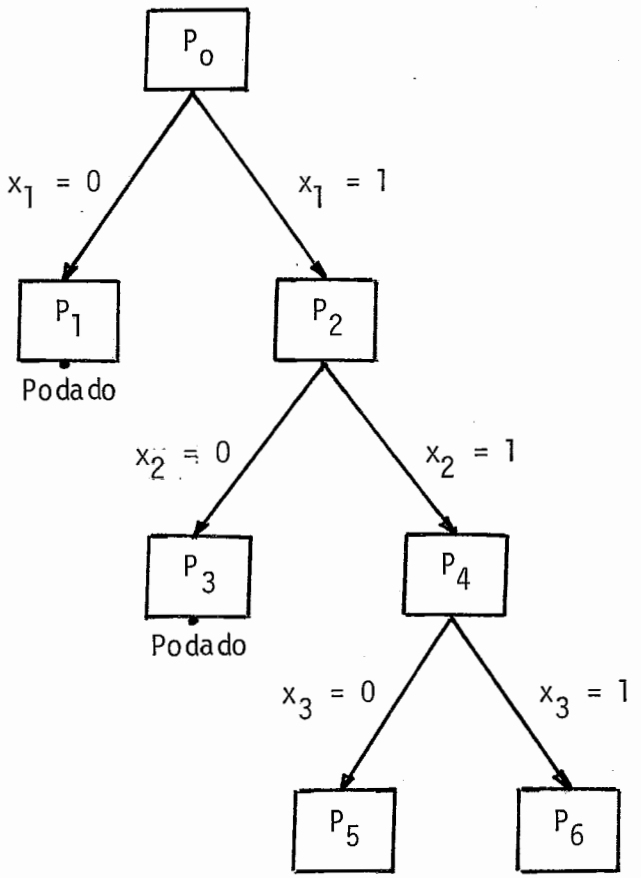


Fig. 3.9 - Árvore do "branch-and-bound"

P_5

$$\text{Max } z = 5x_1 + 6x_2 + 7x_3 \tag{3.9}$$

$$\text{tal que } x_1 + 2x_2 + 3x_3 \leqslant 4 \tag{3.10}$$

$$x_1 = 1\,,\ x_2 = 1\,,\ x_3 = 0 \tag{3.24}$$

$$x_i \text{ inteiro } \forall\ i = 1, 2, 3 \tag{3.12}$$

P_6

$$\text{Max } z = 5x_1 + 6x_2 + 7x_3 \tag{3.9}$$

$$\text{tal que } x_1 + 2x_2 + 3x_3 \leqslant 4 \tag{3.10}$$

$$x_1 = 1\,,\ x_2 = 1\,,\ x_3 = 1 \tag{3.25}$$

$$x_i \text{ inteiro } \forall\ i = 1, 2, 3 \tag{3.12}$$

A solução de P_5 é obviamente $x^{*5} = (1, 1, 0)^T$ cujo valor é

$z^{*5} = 11$. Como esta solução é pior que x^L então a solução ótima de P_0 não pode estar em P_5 e este é podado.

O problema P_6 não apresenta nenhuma solução viável e logo a solução ótima de P_0 não pode estar entre as soluções viáveis de P_6, logo P_6 é podado também. Portanto, o fato de um problema não apresentar nenhuma solução viável justifica podá-lo.

Não havendo mais nenhum problema a ser expandido, como nos mostra a figura 3.10, a solução ótima de P_0 é $x^{*0} = x^L = (1, 0, 1)^T$ de valor $z^{*0} = z^L = 12$.

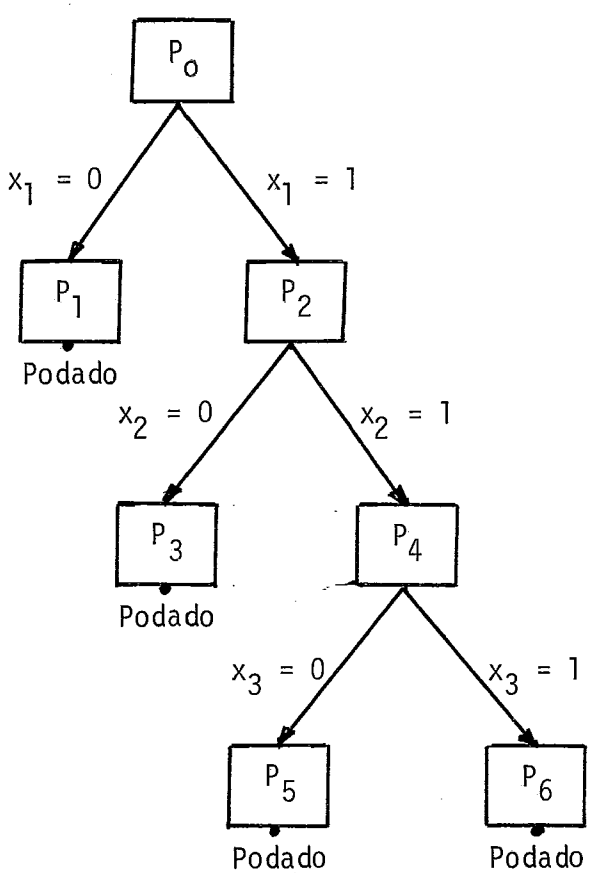


Fig. 3.10 - Árvore do "branch-and-bound" (Final).

Uma observação acerca deste exemplo é necessária. Se em determinado momento contássemos com mais de um problema possível de ser expandido, o que não ocorreu neste exemplo, escolheríamos um deles e o expandiríamos.

3.3 - PROCEDIMENTOS BÁSICOS DE UM "BRANCH-AND-BOUND" PARA SOLUÇÃO DO PMOI

3.3.1 - Introdução

Um algoritmo "branch-and-bound" para solução do PMOI consiste na utilização racional de dois tipos de procedimentos: particionamento e poda. Ver Breu-Burdet |03|.

É comum se associar ao desenvolvimento de um "branch-and-bound" uma árvore onde cada nó está associado a um problema. A raiz da árvore corresponde ao problema originalmente dado P_0 e que se quer solucionar. Alguns termos típicos de algoritmos "branch-and-bound" vem dessa associação com Teoria dos Grafos.

Nos algoritmos "branch-and-bound", primeiramente tenta-se achar a solução ótima x^{*0} de P_0 pesquisando de uma vez todo o conjunto S_0 de soluções viáveis de P_0. Se a tarefa torna-se difícil, particiona-se S_0 em subconjuntos S_i. Aí então a busca tem lugar em cada um desses subcon

juntos. Se em algum ou em alguns deles, novamente, a tarefa torna-se difícil, particiona-se aqueles onde isto ocorre. Na figura 3.11 mostramos, ilustrativamente, como se processa o particionamento sucessivo de S_0.

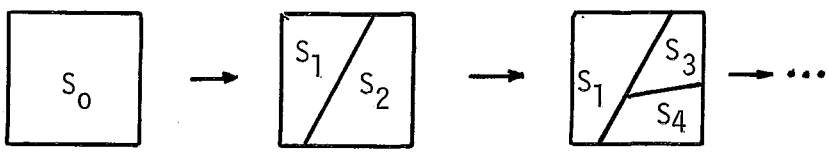


Fig. 3.11 - Particionamento sucessivo de S_0.

Buscar a solução ótima em um subconjunto S_i de S_0 significa resolver o problema P_i cujo conjunto de soluções viáveis é S_i e cuja função objetivo é a mesma de P_0. Evidentemente acharemos a solução ótima de P_i e não de P_0. A solução ótima de P_0 é a maior das soluções ótimas dos problemas P_i. Desta forma na figura 3.11, a solução ótima x^{*0} de P_0 será tal que:

$$z^{*0} = \max \{z^{*1}, z^{*2}\} = \max \{z^{*1}, z^{*3}, z^{*4}\} = \ldots \quad (3.26)$$

O particionamento, quando se trata de problemas com variáveis 0-1, se faz de maneira simples, basta fixar uma ou mais variáveis em 0 ou em 1.

Como a cada S_i se associa um específico problema P_i, e a cada P_i um específico nó na árvore do "branch-and-bound" é comum se confundir os termos particionar S_i, com expandir P_i, e problema P_i com nó P_i.

É fácil ver que utilizando-se somente a técnica do particiona

mento poderíamos achar a solução ótima de P_0, para isso bastaria formar uma partição de S_0 contendo apenas conjuntos unitários, neste caso a solução ótima de cada problema correspondente será o próprio elemento do conjunto; tirando o maior valor dessas soluções obteríamos a solução ótima x^{*o} de P_0. Evidentemente isto é impraticável, mesmo para problemas pequenos, pois o número máximo de tais conjuntos unitários é 2^n onde n é o número de variáveis de P_0.

O procedimento da poda é que então tentará determinar os subconjuntos S_i (Problemas P_i) que não serão mais particionados (expandidos), permitindo assim reduzir enormemente o número de problemas gerados no "branch-and-bound".

Embora a poda seja efetivamente o procecimento que diretamente parece governar a eficiência dos "branch-and-bound", não se deve esquecer que o particionamento determina uma estratégia de geração de subconjuntos S_i que poderão ser menos ou mais facilmente podados.

Um subconjunto S_i (ou problema P_i) deve ser podado, isto é, não deve ser mais particionado (expandido) quando uma das 3 situações acontece:

a) O subconjunto S_i é vazio

Isto acontece quando com a fixação de variáveis é violada a restrição (3.2) do PMO1, isto é, ultrapassado o peso total M da mochila.

b) Conhece-se a solução ótima x^{*i} de P_i

Ora se já conhecemos a solução x^{*i} de P_i não tem sentido particionar S_i para encontrar x^{*i}.

c) Prova-se que a solução ótima x^{*i} de P_i não pode ser solução ótima de P_0

Chamaremos esta prova de Teste de não-otimalidade. Este teste constitui-se do cálculo de um limitante superior (z^{u_i}) para o valor ótimo (z^{*i}) de P_i e de uma comparação de z^{u_i} com z^L (valor de uma solução conhecida de P_0). Se $z^{u_i} < z^L$ então a solução ótima x^{*i} não pode ser solução ótima de P_0 pois $z^{*i} \leq z^{u_i} < z^L \leq z^{*0}$, isto é, $z^{*i} < z^{*0}$. Este teste tem sentido prático pois z^{u_i} deve poder ser facilmente calculado (ver Apêndice B).

3.3.2 - Fluxograma geral

A figura 3.12 mostra um fluxograma bastante geral de um algoritmo "branch-and-bound" para solucionar o PMOI.

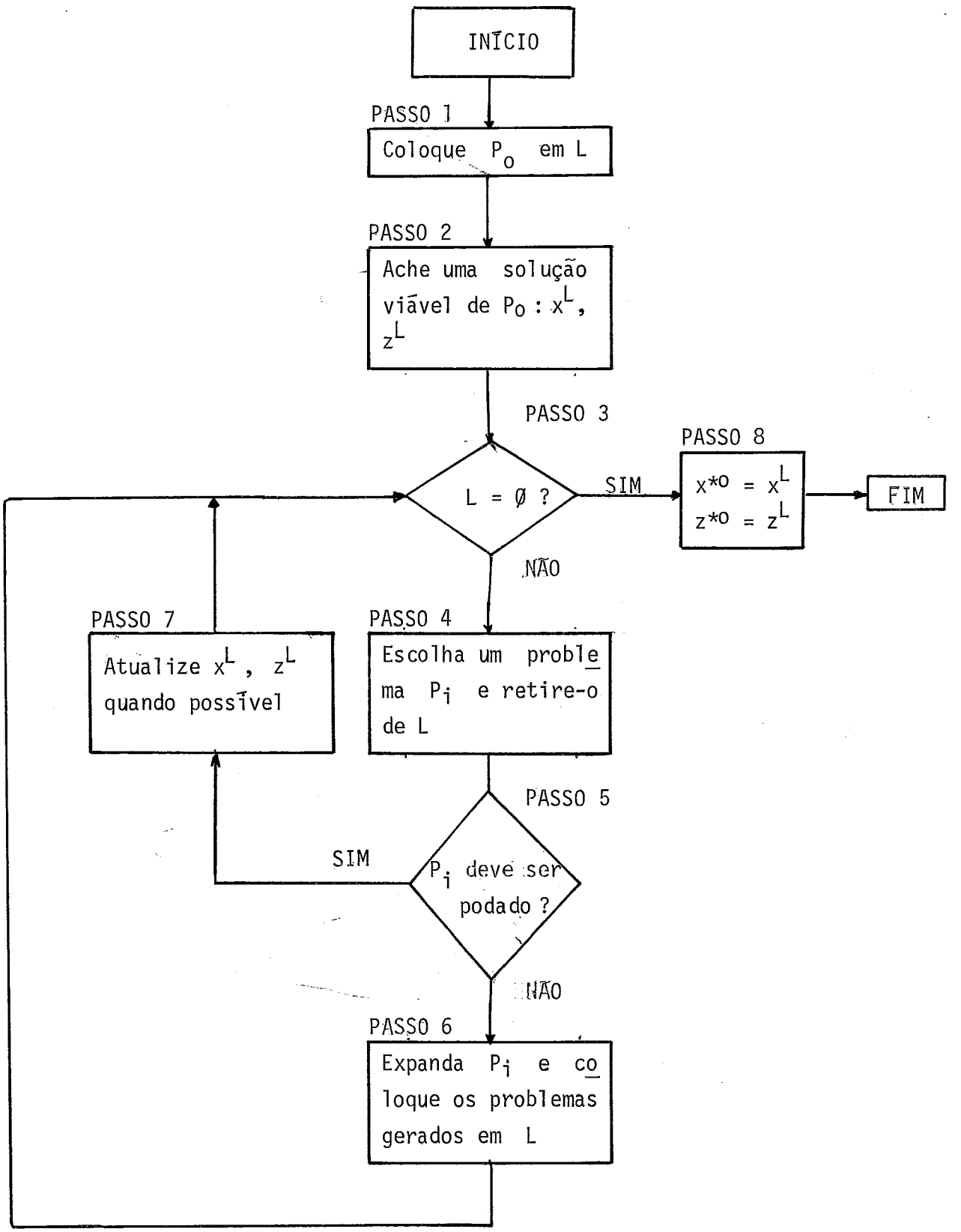


Fig. 3.12 - Fluxograma geral para resolver um PMO1.

Passo 1 - Inicialização da lista L de problemas a serem expandidos

A lista L simboliza o conjunto de todos os problemas que devem ser expandidos.

Inicialmente ela conterá apenas o problema original P_0.

Passo 2 - Inicialização de x^L, z^L

Deve-se inicializar com uma solução viável de P_0, tal solução é, portanto, a melhor solução até então encontrada designada por x^L cujo valor é z^L.

Uma solução viável trivial é fazer $x^L = (0, 0, \ldots, 0)^T$ e $z^L = 0$.

Passo 3 - Teste de finalização

O algoritmo entra na finalização (passo 8) quando não há mais nenhum problema a ser expandido.

Passo 4 - Escolha do problema P_i a ser expandido

Escolhe-se, segundo um determinado critério que varia de algoritmo para algoritmo, qual o problema a ser expandido. Retire-o da lista L.

Passo 5 - Testes de poda. P_i deve ser podado ?

Caso afirmativo, vá para o passo 7.

Verifica-se se P_i deve ou não ser expandido. Cada algoritmo faz alguns ou todos os testes de poda anteriormente citados.

Passo 6 - Expansão de P_i (Particionamento de S_i)

O particionamento de S_i é feito segundo os mais diversos critérios. Cada problema gerado é colocado em L. Vá para o passo 3.

Passo 7 - Atualização de x^L, z^L

Caso se conheça uma solução viável cujo valor é maior que z^L então atualiza-se x^L e z^L.

Passo 8 - Finalização

A solução ótima x^{*o} de P_o é então igual a x^L e $z^{*o} = z^L$. FIM.

3.3.3 - Exemplo

Para exemplificar essa seção, faremos um acompanhamento do exem

plo numérico desenvolvido na seção 3.2 com auxílio do fluxograma geral da seção anterior.

Passo 1: $L = \{P_0\}$

Passo 2: $x^L = (1, 1, 0)^T$ e $z^L = 11$

Passo 3: $L \neq \emptyset$

Passo 4: Problema escolhido: P_0. $L = \emptyset$

Passo 5: P_0 não deve ser podado (Os 3 testes de poda falham).

Passo 6: $P_1 = \{P_0 \mid x_1 = 0\}$. $P_2 = \{P_0 \mid x_1 = 1\}$. $L = \{P_1, P_2\}$

Passo 3: $L \neq \emptyset$

Passo 4: Problema escolhido: P_1. $L = \{P_2\}$

Passo 5: P_1 deve ser podado porque $z^{u1} = z^{*1} = 10{,}66 < z^L = 11$.

Passo 7: Sem efeito (Não se conhece nenhuma nova solução viável).

Passo 3: $L \neq \emptyset$

Passo 4: Problema escolhido: P_2. $L = \emptyset$

Passo 5: P_2 não deve ser podado (Os 3 testes de poda falham).

Passo 6: $P_3 = \{P_2 \mid x_2 = 0\}$. $P_4 = \{P_2 \mid x_2 = 1\}$. $L = \{P_3, P_4\}$

Passo 3: $L \neq \emptyset$

Passo 4: Problema escolhido: P_3. $L = \{P_4\}$

Passo 5: P_3 deve ser podado pois $x^{*3} = \bar{x}^{*3} = (1, 0, 1)^T$, $z^{*3} = 12$

Passo 7: Atualização de x^L e z^L: $z^L = 11 < 12 = z^{*3} \Rightarrow z^L = 12$, $x^L = (1,0,1)^T$

Passo 3: $L \neq \emptyset$

Passo 4: Problema escolhido: P_4. $L = \emptyset$

Passo 5: P_4 não deve ser podado (Os 3 testes de poda falham).

Passo 6: $P_5 = \{P_4 \mid x_3 = 0\}$. $P_6 = \{P_4 \mid x_3 = 1\}$. $L = \{P_5, P_6\}$

Passo 3: $L \neq \emptyset$

Passo 4: Problema escolhido: P_5 . $L = \{P_6\}$

Passo 5: P_5 deve ser podado pois $x^{*5} = (1, 1, 0)^T, z^{*5} = 11$

Passo 7: $z^L = 12 > 11 = z^{*5} \Longrightarrow x^L = (1, 0, 1)^T$, $z^L = 12$

Passo 3: $L \neq \emptyset$

Passo 4: Problema escolhido: P_6 . $L = \emptyset$

Passo 5: P_6 deve ser podado pois $S_6 = \emptyset$

Passo 7: Sem efeito (Não se conhece nenhuma nova solução viável)

Passo 3: $L = \emptyset$

Passo 8: $x^{*0} = (1, 0, 1)^T$, $z^{*0} = 12$. FIM

3.3.4 - Convergência e otimalidade

Finalizando devemos salientar que um algoritmo que siga o fluxograma da seção 3.3.2 convergirá se garantirmos que no passo 6 não geraremos subconjuntos de S_0 que já foram gerados anteriormente. Como só existe um número finito de subconjuntos S_i , distintos entre si, formados a partir do conjunto finito S_0 , a lista L não poderá crescer indefinidamente. Por outro lado, se o processo de expansão ou partição dos problemas (Passo 6) for aplicado um número suficiente de vezes, o algoritmo acabará fazendo com que no pior dos casos a lista L contenha somente problemas P_i cujos conjuntos S_i contêm um único elemento. Ora, tais problemas se

enquadram na situação (b) dos critérios de poda, pois a solução ótima x^{*i} (de P_i) é determinada trivialmente. Assim, eliminando estes problemas da lista, tornamos L vazio, o que leva ao término do procedimento.

Para provar a otimalidade temos que garantir que todo o conjunto S_0 está sendo pesquisado.

Uma maneira simples de garantir a convergência e a otimalidade é particionar cada S_i em subconjuntos disjuntos entre si e cuja união é igual a S_i, desta forma garantiremos que não geraremos subconjuntos anteriormente gerados e também que todo conjunto S_0 está sendo pesquisado.

3.4 - DESENVOLVIMENTO DOS ALGORITMOS "BRANCH-AND-BOUND" PARA SOLUÇÃO DO PM01

3.4.1 - Introdução

O primeiro algoritmo "branch-and-bound" para solução do PM01 foi apresentado por Kolesar [16] em 1967, ele tem a grande desvantagem de exigir muita memória computacional. Em 1970, Greemberg - Hegerich [13] apresentam um algoritmo que diminuiu bastante os requisitos de memória. Em 1974, Horowitz - Sahni [14] e Ahrens - Finke [01], em 1975, apresentam independentemente dois algoritmos bastante semelhantes e que vieram influenciar de forma marcante o desenvolvimento atual dos algoritmos "branch-and-bound"

para solução do PMO1.

Abordaremos nessa seção, de forma detalhada, os algoritmos de Kolesar, Greemberg - Hegerich e Horowitz - Sahni; além disso mencionaremos as principais características dos algoritmos mais recentes.

Também dedicaremos algumas considerações aos algoritmos de redução que permitem reduzir o número de variáveis de um dado PMO1 através da determinação, a priori, do valor que algumas dessas variáveis teriam na solução ótima do referido problema.

Para se ter uma idéia comparativa dos três algoritmos que apresentaremos a seguir, utilizaremos o fluxograma geral apresentado na seção 3.3. Sendo assim em cada algoritmo detalharemos cada passo do mesmo, de acordo com o fluxograma geral. Essa maneira de descrever tais algoritmos faz com que os mesmos tenham um aspecto descritivo diferente dos originalmente propostos mas não alteramos em nada a lógica dos mesmos.

3.4.2 - Algoritmo de Kolesar

3.4.2.1 - Descrição do algoritmo

O algoritmo apresentado por Kolesar em [16] tem a seguinte formulação:

Passo 1 - Inicialização da lista L de problemas a serem expandidos.

$L = \{P_0\}$. Além disso, deve-se calcular o limitante superior $z^{u_0} = \lfloor \overline{z}^{*0} \rfloor$ (maior inteiro menor que o valor ótimo de $\overline{P_0}$) do valor z^{*0} da solução ótima x^{*0} de P_0, armazenando-se z^{u_0} pois o mesmo será necessário no passo 4. Se a solução ótima $\overline{x}^{*0}$ de P_0 for inteira, esta também deverá ser armazenada pois será necessária nos passos 5 e 7.

Passo 2 - Inicialização da melhor solução x^L, z^L

Implicitamente, o algoritmo de Kolesar se inicia com a solução viável $x^L = (0, 0, \ldots, 0)^T$, $z^L = 0$.

Passo 3 - Teste de finalização

Se $L = \emptyset$ vá para o passo 8.

Passo 4 - Escolha do problema P_i a ser expandido

Deve-se escolher o problema P_i que apresentar o maior z^{u_i}. Faça então $L = L - \{P_i\}$

Passo 5 - O problema P_i deverá ser podado ?

Caso afirmativo vá para o passo 7. Um problema deverá ser poda

do se conhecemos a sua solução ótima x^{*i}, se $z^{ui} = \lfloor \overline{z}^{*i} \rfloor < z^L$ ou ainda se o conjunto de soluções viáveis de P_i é vazio.

Passo 6 - Expansão de P_i

O problema P_i é expandido em dois problemas fixando a variável livre de maior razão valor/peso em 0 e em 1. Isso garante que o conjunto S_i de soluções viáveis de P_i será particionado em dois conjuntos disjuntos cuja união é igual a S_i, logo essa regra de particionamento garante a convergência e otimalidade do algoritmo, como exposto na subseção 3.3.4. Além disso, para cada um dos dois problemas gerados deve-se calcular o limitante superior $z^{ui} = \lfloor \overline{z}^{*i} \rfloor$ (o maior inteiro menor que o valor ótimo do problema relaxado $\overline{P_i}$) para o valor de sua solução ótima. O valor desse limitante superior deve ser armazenado para ser usado no passo 5, e se a solução ótima $\overline{x}^{*i}$ do problema relaxado for inteira esta também deve ser armazenada, pois neste caso ela será solução ótima do problema gerado, isto é, $x^{*i} = \overline{x}^{*i}$, podendo ser usada posteriormente nos passos 5 e 7.

Se um dos problemas gerados for impossível então deve-se fazer negativo o limitante superior associado a ele.

Vá para o passo 3

Passo 7 - Atualização de x^L e z^L

Se é conhecida uma nova solução viável x^{*i} e se $z^{*i} > z^L$ en

tão faz-se: $z^L = z^{*i}$ e $x^L = x^{*i}$. Vá para o passo 3.

Passo 8 - Finalização

$x^{*o} = x^L$, $z^{*o} = z^L$. FIM.

3.4.2.2 - Aplicação numérica

Resolver o seguinte problema aplicando o algoritmo de Kolesar.

P_o

$$\text{Max } 60x_1 + 60x_2 + 40x_3 + 10x_4 + 20x_5 + 10x_6 + 3x_7 \tag{3.27}$$

$$\text{sujeito a } 30x_1 + 50x_2 + 40x_3 + 10x_4 + 40x_5 + 30x_6 + 10x_7 \leqslant 100 \tag{3.28}$$

$$0 \leqslant x_i \leqslant 1 \quad \forall i = 1, \ldots, 7 \tag{3.29}$$

$$x_i \text{ inteiro} \quad \forall i = 1, \ldots, 7 \tag{3.30}$$

Solução:

Passo 1: $L = \{P_o\}$. $z^{u_o} = \lfloor \bar{z}^{*o} \rfloor = 140$

Passo 2: $x^L = (0, 0, 0, 0, 0, 0, 0)^T$, $z^L = 0$

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_o. $L = \emptyset$

Passo 5: P_o não será podado.

Passo 6: $P_1 = \{P_o \mid x_1 = 0\}$. $P_2 = \{P_o \mid x_1 = 1\}$. $L = \{P_1, P_2\}$

$z^{u_1} = \lfloor \bar{z}^{*1} \rfloor = 110$. $z^{u_2} = \lfloor \bar{z}^{*2} \rfloor = 140$

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_2 $(z^{u2} > z^{u1})$. $L = \{P_1\}$

Passo 5: P_2 não será podado.

Passo 6: $P_3 = \{P_2 \mid x_2 = 0\}$. $P_4 = \{P_2 \mid x_2 = 1\}$. $L = \{P_1, P_3, P_4\}$

$z^{u3} = \lfloor \bar{z}^{*3} \rfloor = 120$. $z^{u4} = \lfloor \bar{z}^{*4} \rfloor = 140$

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_4. $L = \{P_1, P_3\}$

Passo 5: P_4 não será podado.

Passo 6: $P_5 = \{P_4 \mid x_3 = 0\}$. $P_6 = \{P_4 \mid x_3 = 1\}$.. $L = \{P_1, P_3, P_5, P_6\}$

$z^{u5} = \lfloor \bar{z}^{*5} \rfloor = 135$. $z^{u6} = -1$ pois $S_6 = \emptyset$.

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_5. $L = \{P_1, P_3, P_6\}$

Passo 5: P_5 não será podado.

Passo 6: $P_7 = \{P_5 \mid x_4 = 0\}$. $P_8 = \{P_5 \mid x_4 = 1\}$. $L = \{P_1, P_3, P_6, P_7, P_8\}$

$z^{u7} = \lfloor \bar{z}^{*7} \rfloor = 130$. $z^{u8} = \lfloor \bar{z}^{*8} \rfloor = 135$.

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_8. $L = \{P_1, P_3, P_6, P_7\}$

Passo 5: P_8 não será podado.

Passo 6: $P_9 = \{P_8 \mid x_5 = 0\}$. $P_{10} = \{P_8 \mid x_5 = 1\}$. $L = \{P_1, P_3, P_6, P_7, P_9, P_{10}\}$

$z^{u9} = \lfloor \bar{z}^{*9} \rfloor = 133$. $z^{u10} = -1$

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_9. $L = \{P_1, P_3, P_6, P_7, P_{10}\}$

Passo 5: P_9 não será podado.

Passo 6: $P_{11} = \{P_9 \mid x_6 = 0\}$. $P_{12} = \{P_9 \mid x_6 = 1\}$. $L = \{P_1, P_3, P_6, P_7, P_{10}, P_{11}, P_{12}\}$

$z^{u11} = 133$. $z^{u12} = -1$

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_{11}. $L = \{P_1, P_3, P_6, P_7, P_{10}, P_{12}\}$

Passo 5: P_{11} não será podado.

Passo 6: $P_{13} = \{P_{11} \mid x_7 = 0\}$. $P_{14} = \{P_{11} \mid x_7 = 1\}$. $L = \{P_1, P_3, P_6, P_7, P_{10}, P_{12}, P_{13}, P_{14}\}$

$z^{u13} = 130$. $z^{u14} = 133$ (sol. inteira $\overline{P_{14}}$)

$\overline{x}^{*14} = (1, 1, 0, 1, 0, 0, 1)^T$

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_{14}. $L = \{P_1, P_3, P_6, P_7, P_{10}, P_{12}, P_{13}\}$

Passo 5: P_{14} deverá ser podado: x^{*14} é conhecida.

Passo 7: $z^L = 133 = z^{*14}$, $x^L = (1, 1, 0, 1, 0, 0, 1)^T = x^{*14}$

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_{13}. $L = \{P_1, P_3, P_6, P_7, P_{10}, P_{12}\}$

Passo 5: P_{13} deverá ser podado pois $z^{u13} = 130 < z^L = 133$.

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_7. $L = \{P_1, P_3, P_6, P_{10}, P_{12}\}$

Passo 5: P_7 deverá ser podado pois $z^{u7} = 130 < z^L = 133$.

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_3. $L = \{P_1, P_6, P_{10}, P_{12}\}$

Passo 5: P_3 deverá ser podado pois $z^{u3} = 120 < z^L = 133$.

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_1. $L = \{P_6, P_{10}, P_{12}\}$

Passo 5: P_1 deverá ser podado pois $z^{u1} = 110 < z^L = 130$

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_6. $L = \{P_{10}, P_{12}\}$

Passo 5: P_6 deverá ser podado pois $z^{u6} = -1 < z^L = 133$

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_{10}. $L = \{P_{12}\}$

Passo 5: P_{10} deverá ser podado pois $z^{u10} = -1 < z^L = 133$

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: Escolhido P_{12}. $L = \emptyset$

Passo 5: P_{12} deverá ser podado pois $z^{u12} = -1 < z^L = 133$

Passo 7: Sem efeito

Passo 3: $L = \emptyset$

Passo 8: $x^{*o} = x^L = (1, 1, 0, 1, 0, 0, 1)^T$, $z^{*o} = z^L = 133$. FIM.

A árvore correspondente a este exemplo é dada na figura 3.13.

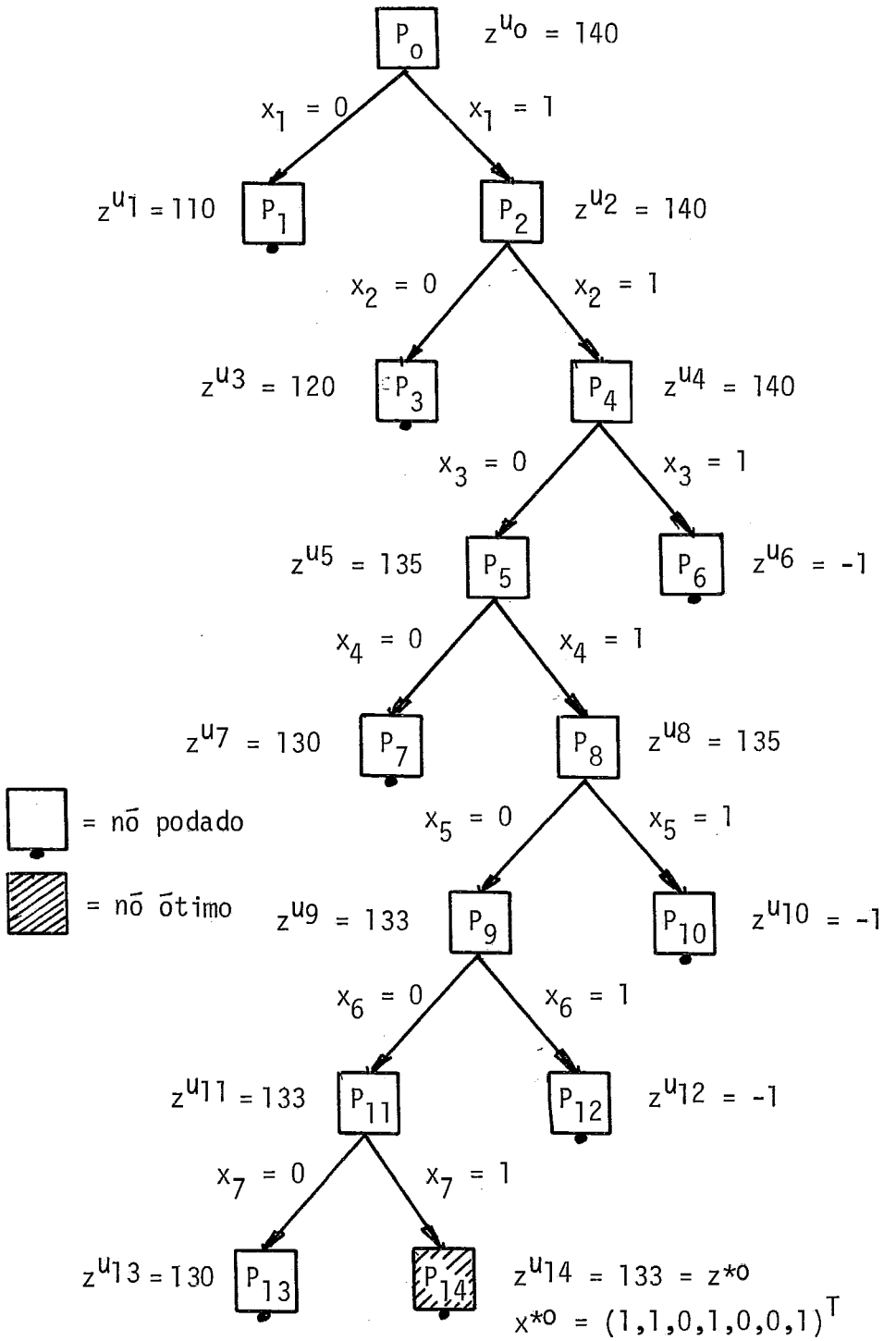


Fig. 3.13 - Árvore correspondente ao exemplo do algoritmo de Kolesar

3.4.2.3 - Observações

Na verdade o algoritmo original de Kolesar pula todos os passos

compreendidos entre as duas linhas tracejadas do exemplo, pois como podemos ver a poda de todos os problemas contidos em L, naquele trecho, é automática porque todos terão seus z^{ui} menores que z^{L} visto que este foi obtido do problema que possuia o maior limitante superior. Portanto no algoritmo originalmente proposto por Kolesar, o passo 7 não tem a ordem "vá para o passo 3", isto é, após o passo 7 é executado o passo 8.

O motivo de escrevermos uma modificação do algoritmo de Kolesar é mostrar como ele se encaixa no fluxograma geral da seção 3.3 e deixar bem claro que Kolesar executa várias podas, implicitamente, no final do algoritmo. Fica bastante óbvio então que em Kolesar a lista L se enche muito porque ele utiliza o procedimento da poda somente quando atinge a solução ótima.

3.4.3 - Algoritmo de Greenberg - Hegerich

3.4.3.1 - Descrição do algoritmo

Greenberg - Hegerich apresentam em |[13]| dois algoritmos "branch-and-bound" para solução do PMO1. Um deles é uma pequena melhoria no algoritmo de Kolesar e o outro é um algoritmo um pouco diferente que será apresentado aqui.

Passo 1 - Inicialização da lista L de problemas a serem expandidos

$L = \{P_0\}$, isto é, a lista L se inicializa com o problema P_0.

Passo 2 - Inicialização da melhor solução x^L, z^L

A solução ótima de $\overline{P_0}$, isto é, $\overline{x}^{*0}$ tem no máximo uma variável com valor fracionário (ver Apêndice A). A solução x^L é igual a $\overline{x}^{*0}$, exceto a sua variável fracionária que é anulada. O valor z^L é obviamente o valor da solução x^L .

Passo 3 - Teste de finalização

Se $L = \emptyset$ vá para o passo 8.

Passo 4 - Escolha do problema P_i a ser expandido

O problema escolhido é o último que entrou na lista L; trata-se portanto de uma escolha tipo LIFO ("Last in - First out"). Retire P_i da lista L.

Passo 5 - O problema P_i deverá ser podado ?

Caso afirmativo vá para o passo 7. São válidos aqui os três critérios de poda citados na seção 3.3. Neste passo é calculado o limite superior $z^{ui} = \lfloor \overline{z}^{*i} \rfloor$ do valor ótimo (z^{*i}) de P_i bem como $\overline{x}^{*i}$. Veja Apêndice A a respeito do cálculo da solução ótima de $\overline{P_i}$.

Passo 6 - Expansão de P_i

O problema P_i é expandido em dois problemas fixando a variável que tem valor fracionário em $\bar{x}^{*i}$, em 0 e em 1. O primeiro problema a ser colocado na lista L é aquele em que a variável foi fixada em 1, em seguida é colocado o outro. A fixação de uma única variável em 0 e em 1 garante que S_i foi particionado em dois subconjuntos disjuntos cuja união é igual a S_i, com isso fica garantida a convergência e otimalidade do algoritmo como exposto na subseção 3.3.4. Vá para o passo 3.

Passo 7 - Atualização de x^L, z^L

Se é conhecida uma nova solução viável x^{*i} (e isto acontece quando a solução $\bar{x}^{*i}$ é inteira) e se $z^{*i} > z^L$ então faz-se $z^L = z^{*i}$ e $x^L = x^{*i}$. Vá para o passo 3.

Passo 8 - Finalização

$x^{*0} = x^L$, $z^{*0} = z^L$. FIM.

3.4.3.2 - Aplicação numérica

Resolver o seguinte problema aplicando o algoritmo de Greenberg-Hegerich.

P_o Max $60x_1 + 60x_2 + 40x_3 + 10x_4 + 20x_5 + 10x_6 + 3x_7$ (3.27)

sujeito a $30x_1 + 50x_2 + 40x_3 + 10x_4 + 40x_5 + 30x_6 + 10x_7 \leq 100$ (3.28)

$0 \leq x_i \leq 1 \quad \forall i = 1, \ldots, 7$ (3.29)

x_i inteiro $\forall i = 1, \ldots, 7$ (3.30)

Solução:

Passo 1: $L = \{P_o\}$

Passo 2: $x^L = (1, 1, 0, 0, 0, 0, 0)^T$, $z^L = 120$

Passo 3: $L \neq \emptyset$

Passo 4: P_o é escolhido. $L = \emptyset$

Passo 5: $\bar{x}^{*o} = (1, 1, 1/2, 0, 0, 0, 0)^T$, $z^{uo} = 140$. P_o não deve ser podado.

Passo 6: $L = \{P_1, P_2\}$. $P_1 = \{P_o \mid x_3 = 1\}$. $P_2 = \{P_o \mid x_3 = 0\}$.

Passo 3: $L \neq \emptyset$

Passo 4: P_2 é escolhido. $L = \{P_1\}$

Passo 5: $\bar{x}^{*2} = (1, 1, 0, 1, 1/4, 0, 0)^T$, $z^{u2} = 135$. P_2 não deve ser podado.

Passo 6: $L = \{P_1, P_3, P_4\}$. $P_3 = \{P_2 \mid x_5 = 1\}$. $P_4 = \{P_2 \mid x_5 = 0\}$.

Passo 3: $L \neq \emptyset$

Passo 4: P_4 é escolhido. $L = \{P_1, P_3\}$

Passo 5: $\bar{x}^{*4} = (1, 1, 0, 1, 0, 1/3, 0)^T$, $z^{u4} = 133$. P_4 não deve ser podado.

Passo 6: $L = \{P_1, P_3, P_5, P_6\}$. $P_5 = \{P_4 \mid x_6 = 1\}$. $P_6 = \{P_4 \mid x_6 = 0\}$.

Passo 3: $L \neq \emptyset$

Passo 4: P_6 é escolhido. $L = \{P_1, P_3, P_5\}$

Passo 5: $\bar{x}^{*6} = (1, 1, 0, 1, 0, 0, 1)^T = x^{*6}$. P_6 é podado. $z^{*6} = 133$.

Passo 7: Como x^{*6} é conhecida e $z^{*6} = 133 > 120 = z^{L}$ então $z^{L} = 133$ e $x^{L} = x^{*6} = (1, 1, 0, 1, 0, 0, 1)^{T}$.

Passo 3: $L \neq \emptyset$

Passo 4: P_5 é escolhido. $L = \{P_1, P_3\}$

Passo 5: $\overline{x}^{*5} = (1, 4/5, 0, 0, 0, 1, 0)^{T}$, $z^{u5} = 118$. P_5 é podado.

Passo 7: Sem efeito.

Passo 3: $L \neq \emptyset$

Passo 4: P_3 é escolhido. $L = \{P_1\}$

Passo 5: $\overline{x}^{*3} = (1, 3/5, 0, 0, 1, 0, 0)^{T}$, $z^{u3} = 116$. P_3 é podado.

Passo 7: Sem efeito.

Passo 3: $L \neq \emptyset$

Passo 4: P_1 é escolhido. $L = \emptyset$

Passo 5: $\overline{x}^{*1} = (1, 3/5, 1, 0, 0, 0, 0)^{T}$, $z^{u1} = 136$. P_1 não deve ser podado

Passo 6: $L = \{P_7, P_8\}$. $P_7 = \{P_1 \mid x_2 = 1\}$. $P_8 = \{P_1 \mid x_2 = 0\}$.

Passo 3: $L \neq \emptyset$

Passo 4: P_8 é escolhido. $L = \{P_7\}$

Passo 5: $\overline{x}^{*8} = (1, 0, 1, 1, 1/2, 0, 0)^{T}$, $z^{u8} = 120$. P_8 é podado.

Passo 7: Sem efeito.

Passo 3: $L \neq \emptyset$

Passo 4: P_7 é escolhido. $L = \emptyset$

Passo 5: $\overline{x}^{*7} = (1/3, 1, 1, 0, 0, 0, 0)^{T}$, $z^{u7} = 120$. P_7 é podado.

Passo 7: Sem efeito

Passo 3: $L = \emptyset$

Passo 8: $x^{*o} = x^L = (1, 1, 0, 1, 0, 0, 1)^T$, $z^{*o} = z^L = 133$.

A árvore correspondente a este exemplo é dada na figura 3.14.

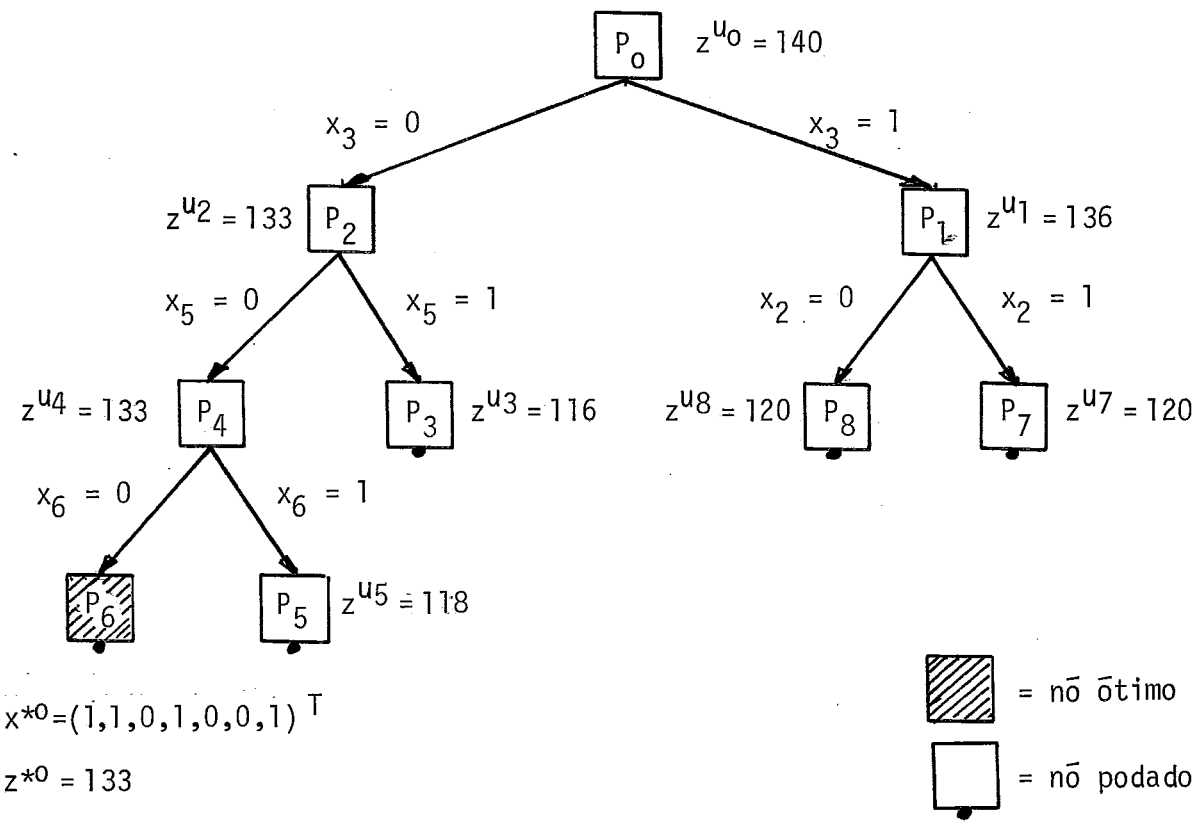


Fig. 3.14 - Árvore correspondente ao exemplo do algoritmo de Greemberg-Hegerich.

3.4.3.3 - Conclusões

Pode-se observar que a lista L, neste caso, tem o número máximo de problemas menor do que a lista L no algoritmo de Kolesar.

Deve-se ressaltar que o critério LIFO é o responsável pela

busca em profundidade do algoritmo de Greemberg - Hegerich, pois estaremos sempre expandindo o último nó gerado fazendo com que a árvore cresça mais no sentido longitudinal que transversal.

3.4.4 - Algoritmo de Horowitz - Sahni

3.4.4.1 - Descrição do algoritmo

Horowitz - Sahni propõe em [14] dois algoritmos para solução do PM01. Um deles utiliza a técnica de Programação Dinâmica, o outro do tipo "branch-and-bound" será apresentado a seguir.

Passo 1 - Inicialização da lista L de problemas a serem expandidos

A lista L inicializa com o problema P_0, isto é, $L = \{P_0\}$. Em P_0 todas as variáveis são consideradas livres.

Passo 2 - Inicialização da melhor solução x^L, z^L

Toma-se x^L como sendo igual a $\overline{x}^{*0}$, exceto a variável fracionária que é anulada. O valor z^L é obviamente o valor da solução x^L.

Passo 3 - Teste de finalização

Se $L = \emptyset$ vá para o passo 8.

Passo 4 - Escolha do problema P_i a ser expandido

O problema escolhido é o último que entrou na lista L, portanto trata-se de uma escolha do tipo LIFO ("Last in - First out"). Retire P_i da lista L.

Passo 5 - O problema P_i deverá ser podado ?

Caso afirmativo vá para o passo 7.

Como os testes de poda de P_i dependem do conhecimento da solução ótima $\overline{x}^{*i}$ do problema relaxado $\overline{P_i}$ e de seu valor $\overline{z}^{*i}$, examinemos primeiramente como calcular $\overline{x}^{*i}$ e $\overline{z}^{*i}$. A justificativa para tal procedimento pode ser vista no Apêndice A.

Sejam $x_1, x_2, \ldots, x_{k-1}$ as variáveis fixadas para o problema P_i em questão. Sejam, respectivamente, $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{k-1}$ os valores dessas variáveis fixadas.

Seja m o maior inteiro tal que:

$$\sum_{j=k}^{k+m-1} p_j \leq M - \sum_{j=1}^{k-1} p_j \hat{x}_j \qquad (3.31)$$

A solução ótima $\overline{x}^{*i}$ de $\overline{P_i}$ será então dada por:

$$\overline{x}_1^{*i} = \hat{x}_1, \ \ldots, \ \overline{x}_{k-1}^{*i} = \hat{x}_{k-1} \qquad (3.32)$$

$$\overline{x}_k^{*i} = \overline{x}_{k+2}^{*i} = \ldots = \overline{x}_{k+m-1}^{*i} = 1 , \qquad (3.33)$$

$$\overline{x}_{k+m}^{*i} = \frac{M - \sum_{j=1}^{k+m-1} p_j \overline{x}_j^{*i}}{p_{k+m}} , \quad \text{e} \qquad (3.34)$$

$$\overline{x}_{k+m+1}^{*i} = \ldots = \overline{x}_n^{*i} = 0 . \qquad (3.35)$$

O valor $\overline{z}^{*i}$ da solução ótima de P_i é dado por:

$$\overline{z}^{*i} = \sum_{j=1}^{n} v_j \overline{x}_j^{*i} \qquad (3.36)$$

O valor $\lfloor \overline{z}^{*i} \rfloor$ constitui um limitante superior (z^{ui}) para o valor ótimo (z^{*i}) de P_i (Ver Apêndice B).

Se $\overline{x}^{*i}$ for uma solução inteira então $x^{*i} = \overline{x}^{*i}$, isto é $\overline{x}^{*i}$ será também a solução ótima de P_i.

Determinada a solução ótima $\overline{x}^{*i}$ de $\overline{P_i}$ e seu valor $\overline{z}^{*i}$ estamos em condições de testar os dois critérios de poda deste algoritmo:

A) Teste de não-otimalidade:

Se $z^{ui} = \lfloor \overline{z}^{*i} \rfloor$ é menor ou igual a z^L (valor da melhor solução até então encontrada x^L), isto é, se $z^{ui} = \lfloor \overline{z}^{*i} \rfloor < z^L$ então P_i deve ser podado.

B) Se P_i tem todas as variáveis fixadas (em 0 ou 1) então P_i deve ser podado pois neste caso $\overline{P_i}$ terá também todas as variáveis fixadas e portanto a solução ótima $\overline{x}^{*i}$ de $\overline{P_i}$ é inteira e assim $x^{*i} = \overline{x}^{*i}$. No passo 6 veremos como são fixadas as variáveis.

Não se inclui aqui o primeiro teste geral de poda, como mostrado na seção 3.3, isto é, aquele que verifica se o conjunto S_i de soluções viáveis de P_i é vazio ou não, isto porque como veremos no passo 6, neste algoritmo, são gerados problemas P_i com pelo menos uma solução e portanto S_i nunca é vazio.

Se $\overline{x}^{*i}$ é inteiro mas em P_i nem todas as variáveis foram fixadas em 0 ou 1, então neste algoritmo não é feita a poda de P_i (o que, em princípio, poderia ter sido feito) pois desta forma evita-se testar em cada nó se $\overline{x}^{*i}$ é inteira ou não, e como este teste é na maioria das vezes negativo ($\overline{x}^{*i}$ não é inteira), economiza-se várias comparações no algoritmo.

Passo 6 - Expansão de P_i

Suporemos que no problema P_i temos $k-1$ variáveis fixadas

em 0 ou 1, isto é, $x_1, x_2, \ldots, x_{k-1}$ são variáveis fixas nos valores $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{k-1}$ e $x_k, x_{k+1}, \ldots, x_n$ são variáveis livres.

Seja $S = M - \sum_{j=1}^{k-1} p_j \hat{x}_j$ a "sobra" de peso ainda não preenchido da mochila no problema P_i. A sobra S nunca poderá ser negativa.

Para mostrar com precisão como são gerados os novos problemas $P_j, P_{j+1}, \ldots$ a partir de P_i utilizaremos um algoritmo (denominado "gerador") escrito numa linguagem semelhante ao ALGOL.

O problema P_{aux} é um problema auxiliar que é implicitamente gerado e expandido, isto é, ele tem uma existência efêmera, limitada a uma passagem do algoritmo gerador abaixo.

Algoritmo gerador

```
BEGIN
    q := 0 ;  P_aux := P_i ;  S := M - Σ_{j=1}^{k-1} p_j x̂_j ;
    WHILE  k+q ≤ n   AND  p_{k+q} ≤ S   DO
        BEGIN
            P_{j+q} := {P_aux | x̂_{k+q} = 0} ;  L = L U {P_{j+q}} ;
            P_aux := {P_aux | x̂_{k+q} = 1} ;  S = S - p_{k+q} ;
              q := q + 1
        END ;
    IF   k + q = n + 1   THEN
        BEGIN
            P_{j+q} := P_aux  ;  L := L U {P_{j+q}}
        END ;
    ELSE
        BEGIN
            P_{j+q} := {P_aux | x̂_{k+q} = 0} ;   L = L U {P_{j+q}}
        END ;
    m := q ;
    j := j + m + 1
END .
```

O problema P_i, após o término do algoritmo gerador, será expandido nos problemas P_j, P_{j+1}, ..., P_{j+m}. Na figura 3.16 apresentamos o trecho da árvore "branch-and-bound" resultante da expansão de P_i.

Graficamente podemos ver o processo de geração dos problemas P_j, P_{j+1}, ..., P_{j+m} da forma apresentada na figura 3.15.

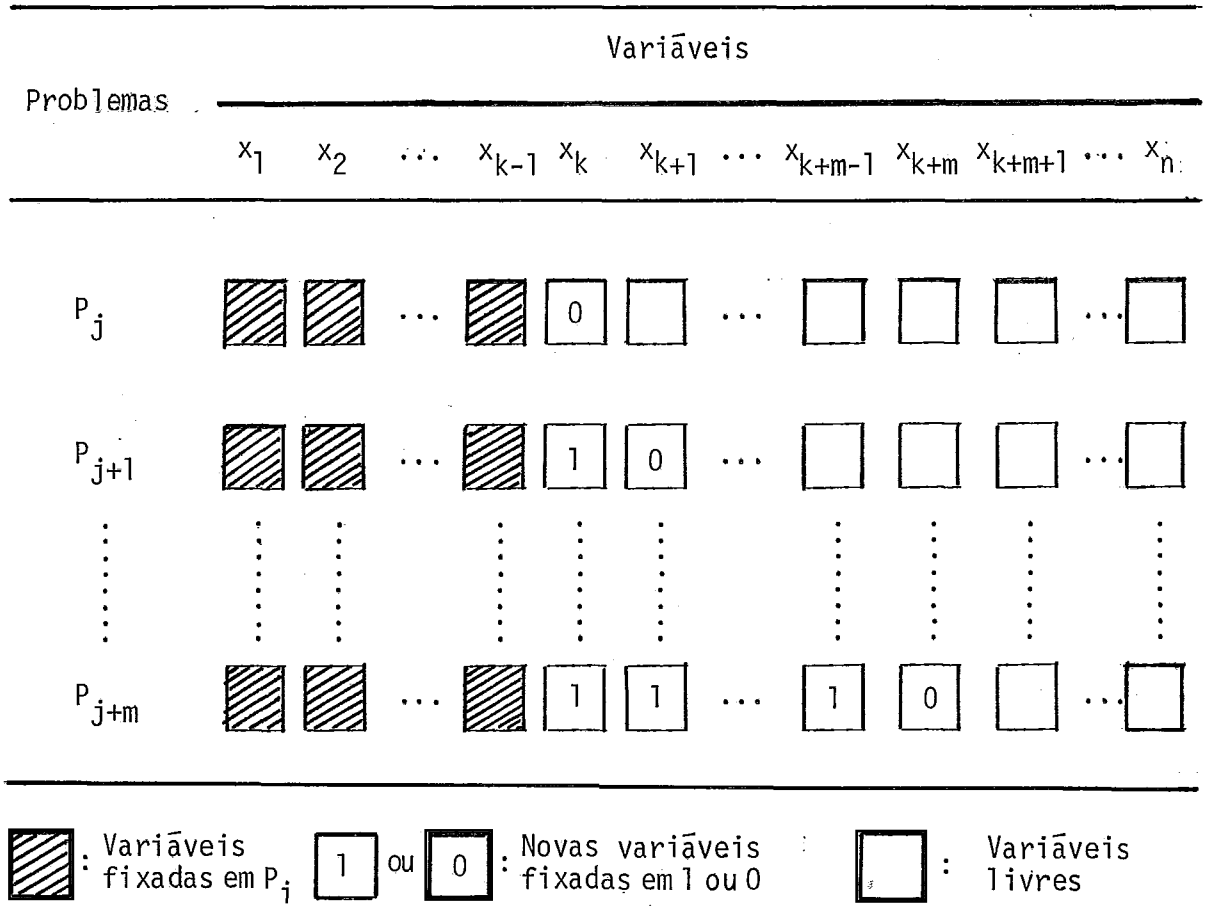


Fig. 3.15 - Fixação das variáveis nos problemas gerados na expansão de P_i.

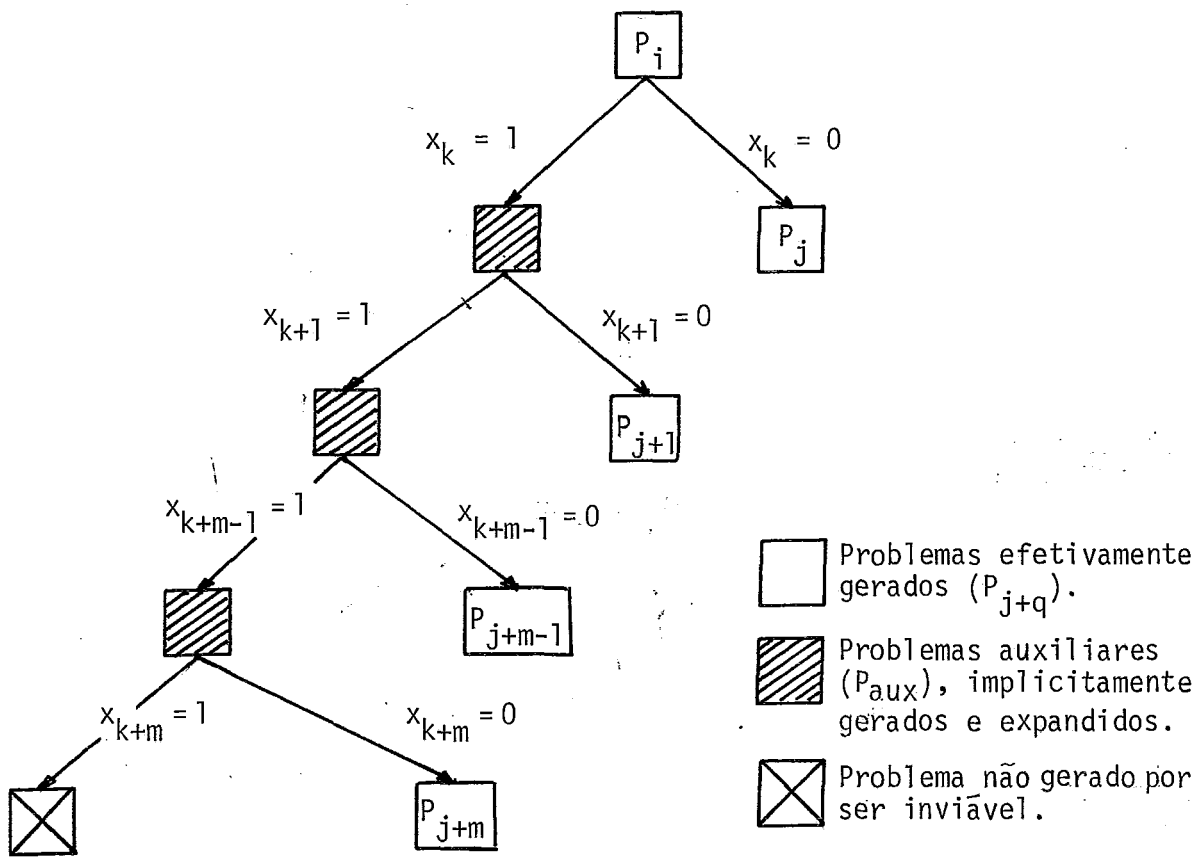


Fig. 3.16 - Trecho da árvore "branch-and-bound" representando a expansão do nó (problema) P_i, supondo $k+m \leqslant n$.

Os nós grafados por  na figura 3.16 apesar de não correspon derem explicitamente a nenhum problema gerado na expansão de P_i, são implicitamente gerados e expandidos. Por outro lado devido à condição (3.31) podemos garantir que o nó grafado por  na figura 3.16 dá origem a um problema inviável não sendo portanto gerado explicitamente.

Devido à condição (3.31) os nós P_j, P_{j+1}, ..., P_{j+m} são tais que o conjunto de soluções viáveis dos respectivos problemas é não vazio o que torna desnecessário, no algoritmo, o teste de poda segundo este critério.

Verifica-se também o grande número de nós gerados (implícita e explicitamente) em um único passo do algoritmo sendo um dos principais motivos para a grande eficiência computacional do mesmo.

É importante notar que o processo de geração dos problemas P_j, P_{j+1}, ..., P_{j+m} é tal que somente geramos problemas cujos conjuntos de soluções viáveis S_j, S_{j+1}, ..., S_{j+m} são disjuntos dois a dois pois o conjunto de variáveis fixadas para cada problema é distinto, e tais que $S_j \cup S_{j+1} \cup \ldots \cup S_{j+m} = S_i$, isto é, nenhuma solução viável de P_i é eliminada. Estes fatos garantem a convergência e a otimalidade do algoritmo conforme o raciocínio exposto na subseção 3.3.4.

Cabe ainda ressaltar que devido a ordem como os problemas gerados são colocados na lista L e tendo em vista o critério LIFO (passo 4) de escolha do problema a ser expandido, tais fatos garantem a busca em profundidade do algoritmo de Horowitz - Sahni.

Após o passo 6, vá para o passo 3.

Passo 7 - Atualização de x^L, z^L

Se é conhecida uma solução x^{*i} obtida após a fixação de todas as variáveis e se $z^{*i} > z^L$ então faz-se $x^L = x^{*i}$ e $z^L = z^{*i}$. Vá para o passo 3.

Passo 8 - Finalização

$x^{*o} = x^L$, $z^{*o} = z^L$. FIM.

3.4.4.2 - Aplicação numérica

Resolver o seguinte problema aplicando o algoritmo de Horowitz-Sahni.

P_o

$$\text{Max} \quad 60x_1 + 60x_2 + 40x_3 + 10x_4 + 20x_5 + 10x_6 + 3x_7 \tag{3.27}$$

$$\text{sujeito a} \quad 30x_1 + 50x_2 + 40x_3 + 10x_4 + 40x_5 + 30x_6 + 10x_7 \leqslant 100 \tag{3.28}$$

$$0 \leqslant x_i \leqslant 1 \qquad \forall i = 1, \ldots, 7 \tag{3.29}$$

$$x_i \text{ inteiro} \qquad \forall i = 1, \ldots, 7 \tag{3.30}$$

Solução

Passo 1: $L = \{P_o\}$

Passo 2: $x^L = (1, 1, 0, 0, 0, 0, 0)^T$, $z^L = 120$

Passo 3: $L \neq \emptyset$

Passo 4: P_o é escolhido. $L = \emptyset$

Passo 5: $\overline{x}^{*o} = (1, 1, 1/2, 0, 0, 0, 0)^T$, $z^{u_o} = 140$. P_o não deve ser podado.

Passo 6: $L = \{P_1, P_2, P_3\}$. $P_1 = \{P_o \mid x_1 = 0\}$. $P_2 = \{P_o \mid x_1 = 1, x_2 = 0\}$.

$P_3 = \{P_o \mid x_1 = x_2 = 1 \text{ e } x_3 = 0\}$.

Passo 3: $L \neq \emptyset$

Passo 4: P_3 é escolhido. $L = \{P_1, P_2\}$

Passo 5: $\overline{x}^{*3} = (1, 1, 0, 1, 1/4, 0, 0)^T$, $z^{u3} = 135$. P_3 não deve ser podado.

Passo 6: $L = \{P_1, P_2, P_4, P_5\}$. $P_4 = \{P_3 \mid x_4 = 0\}$. $P_5 = \{P_3 \mid x_4 = 1 \text{ e } x_5 = 0\}$

Passo 3: $L \neq \emptyset$

Passo 4: P_5 é escolhido. $L = \{P_1, P_2, P_4\}$

Passo 5: $\overline{x}^{*5} = (1, 1, 0, 1, 0, 1/3, 0)^T$, $z^{u5} = 133$. P_5 não deve ser podado.

Passo 6: $L = \{P_1, P_2, P_4, P_6\}$. $P_6 = \{P_5 \mid x_6 = 0\}$.

Passo 3: $L \neq \emptyset$

Passo 4: P_6 é escolhido. $L = \{P_1, P_2, P_4\}$

Passo 5: $\overline{x}^{*6} = (1, 1, 0, 1, 0, 0, 1)^T$, $z^{u6} = z^{*6} = 133$. P_6 deve ser podado.

Passo 6: $L = \{P_1, P_2, P_4, P_7, P_8\}$. $P_7 = \{P_6 \mid x_7 = 0\}$. $P_8 = \{P_6 \mid x_7 = 1\}$.

Passo 3: $L \neq \emptyset$

Passo 4: P_8 é escolhido. $L = \{P_1, P_2, P_4, P_7\}$

Passo 5: $x^{*8} = (1, 1, 0, 1, 0, 0, 1)^T$, $z^{*8} = 133$. P_8 é podado.

Passo 7: x^{*8} é conhecida através da fixação de todas as variáveis e

$z^{*8} = 133 > 120 = z^L$ logo $x^L = (1, 1, 0, 1, 0, 0, 1)^T$ e $z^L = 133$

Passo 3: $L \neq \emptyset$

Passo 4: P_7 é escolhido. $L = \{P_1, P_2, P_4\}$

Passo 5: $x^{*7} = (1, 1, 0, 1, 0, 0, 0)^T$, $z^{u7} = z^{*7} = 130$. P_7 é podado.

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: P_4 é escolhido. $L = \{P_1, P_2\}$

Passo 5: $\overline{x}^{*4} = (1, 1, 0, 0, 1/2, 0, 0)^T$, $z^{u4} = 130$. P_4 deve ser podado.

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: P_2 é escolhido. $L = \{P_1\}$

Passo 5: $\overline{x}^{*2} = (1, 0, 1, 1, 1/2, 0, 0)^T$, $z^{u2} = 120$. P_2 deve ser podado.

Passo 7: Sem efeito

Passo 3: $L \neq \emptyset$

Passo 4: P_1 é escolhido. $L = \emptyset$

Passo 5: $\overline{x}^{*1} = (0, 1, 1, 1, 0, 0, 0)^T$, $z^{u1} = 110$. P_1 deve ser podado.

Passo 7: Sem efeito

Passo 3: $L = \emptyset$

Passo 8: $x^{*o} = x^L = (1, 1, 0, 1, 0, 0, 1)^T$, $z^{*o} = z^L = 133$. FIM.

A árvore correspondente a este exemplo é dada na figura 3.17.

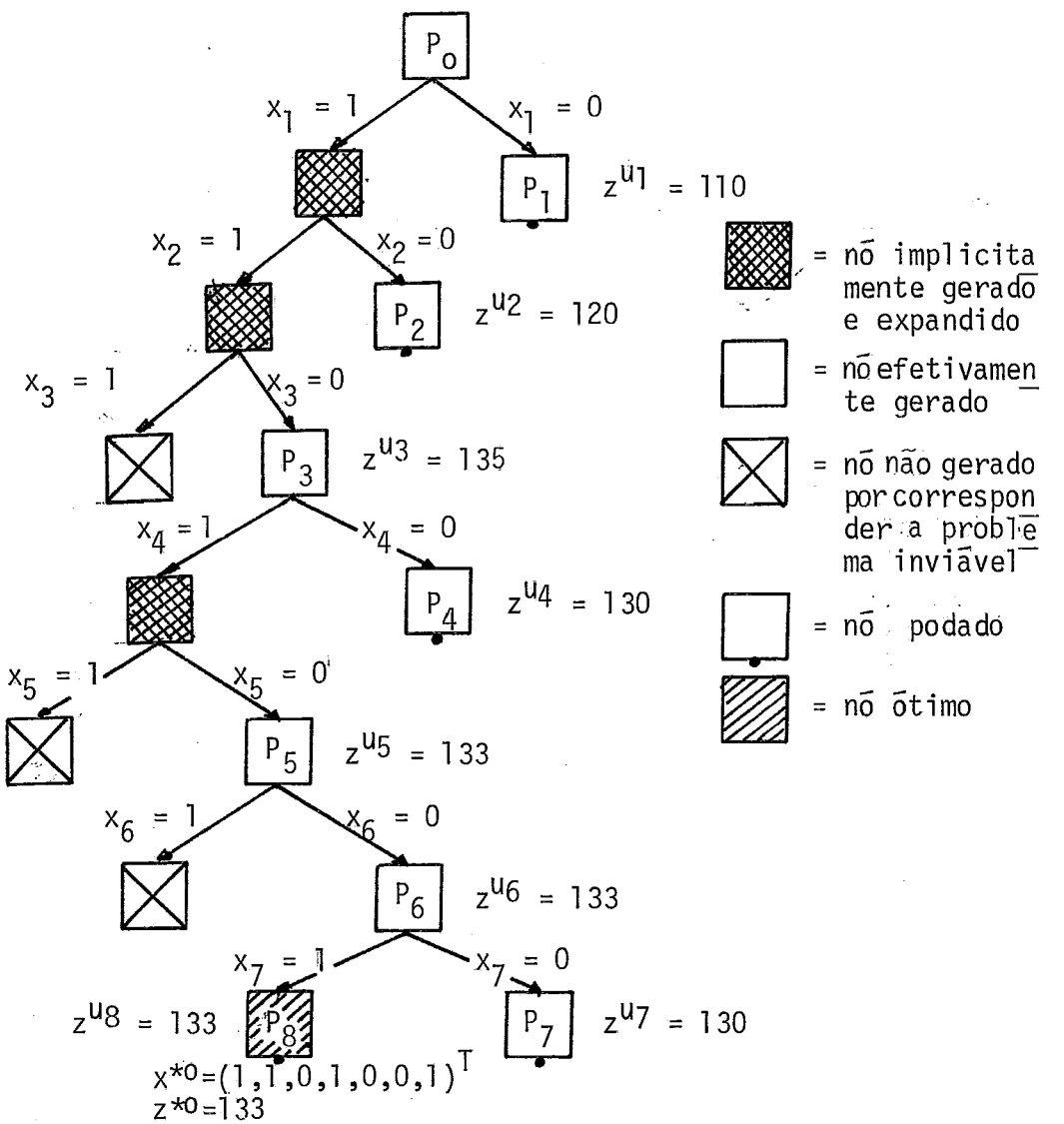


Fig. 3.17 - Árvore correspondente ao exemplo de algoritmo de Horowitz-Sahni.

3.4.5 - Algoritmos de Redução

3.4.5.1 - Introdução

Em 1973, Ingargiola-Korsh [15] publicaram um algoritmo que aplicado a um PM01, tem tempo de execução proporcional ao número de variáveis do problema, e é capaz de determinar, sem chegar à solução ótima, os valores que várias variáveis teriam nessa solução ótima. Desta forma consegue-se reduzir o número de variáveis para as quais não se conhece o valor na solução ótima. Essa redução pode ser bastante significativa e experiências computacionais têm demonstrado que, dependendo do tipo de problema, isso freqüentemente acontece.

O problema expurgado das variáveis reduzidas, isto é, das variáveis cujos valores na solução ótima foram determinados pelo algoritmo de redução, será denominado problema reduzido.

De uma maneira mais rigorosa, podemos dizer que um algoritmo de redução aplicado ao problema P_0 definido por:

$$P_0 \qquad \text{Max} \sum_{j=1}^{n} v_j x_j \tag{3.1}$$

$$\text{sujeito a} \sum_{j=1}^{n} p_j x_j \leqslant M \tag{3.2}$$

$$0 \leqslant x_j \leqslant 1 \quad j = 1, \ldots, n \tag{3.3}$$

$$x_j \text{ inteiro} \quad j = 1, \ldots, n \tag{3.4}$$

e valendo ainda as condições (3.5) a (3,8), determinará dois subconjuntos disjuntos J_1 e J_o do conjunto $J = \{1, 2, \ldots, n\}$ (conjunto dos índices das variáveis de P_o), de tal forma que se $j' \in J_o$ então $x_{j'}^{*o} = 0$ e se $j'' \in J_1$ então $x_{j''}^{*o} = 1$. O problema reduzido (P_R) será então definido por:

$$\boxed{P_R} \qquad \text{Max} \quad \sum_{j=1}^{n} v_j x_j \tag{3.1}$$

$$\text{sujeito a} \quad \sum_{j=1}^{n} p_j x_j < M \tag{3.2}$$

$$0 \leqslant x_j \leqslant 1 \qquad j = 1, \ldots, n \tag{3.3}$$

$$x_j \text{ inteiro} \qquad j = 1, \ldots, n \tag{3.4}$$

$$x_j = 0 \qquad j \in J_o \tag{3.37}$$

$$x_j = 1 \qquad j \in J_1 \tag{3.38}$$

Desta maneira, a solução ótima de P_R será a mesma de P_o.

Ao problema reduzido deve-se aplicar um algoritmo qualquer de solução do PMO1, pois obviamente P_R também é um PMO1, determinando assim a solução ótima de P_R e portanto de P_o.

De uma maneira geral, os algoritmos de redução são aplicados a um dado PMO1 antes de se aplicar um algoritmo de solução completa do PMO1. Alguns algoritmos no entanto, por exemplo Zoltners |[32]|, aplicam um algoritmo de redução como uma rotina interna do mesmo, sempre que detectam certas condições que indicam a possibilidade de reduzir variáveis.

3.4.5.2 - Descrição geral dos algoritmos de redução

Um algoritmo de redução gera problemas do tipo $P_i = \{P_o \mid x_j = p\}$ onde $p = 0$ ou $p = 1$. Para cada problema P_i gerado calcula-se um limitante superior (z^{Ui}) do seu valor ótimo (z^{*i}).

Supondo conhecida uma solução viável de P_o, x^L, de valor z^L, para cada problema P_i é feito então o teste de não otimalidade para verificar se a variável x_j pode ser fixada em p.

Se $z^{u_i} < z^L$ então $x_j^{*o} = 1 - p$, pois neste caso se x_j não pode ser fixada em p (porque daria uma solução de valor menor que z^L) só poderá assumir o valor $(1 - p)$, isto é, $x_j^{*o} = 1 - p$. Se $z^{u_i} \geqslant z^L$ nada poderemos afirmar sobre o valor de x_j^{*o}.

Cada vez que uma variável tiver seu valor na solução ótima (x^{*o}) determinado pode-se expurgá-la do problema original (P_o) o que dará origem a um problema reduzido (P_R). Ao problema reduzido se aplica novamente o algoritmo de redução. Esse procedimento deve-se repetir até não ser mais possível reduzir nenhuma variável. Outra maneira seria analisar todas as variáveis primeiro, verificar quais foram reduzidas e só aí então proceder à geração do problema reduzido. A primeira maneira tem um custo computacional mais elevado que a segunda sem que com isso se obtenha um aumento significativo do número de variáveis reduzidas. Portanto o comum é gerar o problema reduzido uma única vez, depois que todas as variáveis tiverem sido analisadas.

3.4.5.3 - Algoritmo de Ingargiola-Korsh

O que diferem os dois algoritmos de redução mais conhecidos, Ingargiola-Korsh [15] e Dembo-Hammer [07], é a maneira como se calcula o limitante superior z^{u_i} de cada problema $P_i = \{P_o \mid x_j = p\}$.

No algoritmo de Ingargiola-Korsh $z^{u_i} = \lfloor \overline{z}^{*i} \rfloor$, isto é, é o maior inteiro menor que o valor ótimo de $\overline{P_i}$, que é denominado limitante superior de Dantzig (ver Apêndice B).

Seja k o índice da variável de valor fracionário (admitindo solução não inteira para $\overline{P_o}$, se ocorresse solução inteira não teria sentido aplicar algoritmo de redução) na solução ótima $\overline{x}^{*o}$ de $\overline{P_o}$ (P_o relaxado), isto é:

$$\overline{x}_j^{*o} = 1 \qquad j = 1, 2, \ldots, k-1 \tag{3.39}$$

$$0 < \overline{x}_k^{*o} < 1 \tag{3.40}$$

$$\overline{x}_j^{*o} = 0 \qquad j = k+1, \ldots, n \tag{3.41}$$

São gerados dois tipos de problemas no algoritmo de Ingargiola-Korsh:

$$P_i = \{P_o \mid x_i = 0\} \qquad i = 1, \ldots, k \tag{3.42}$$

$$\text{e} \quad P_{i+1} = \{P_o \mid x_i = 1\} \qquad i = k, \ldots, n \tag{3.43}$$

Para cada um dos (n+1) problemas gerados é calculado o limitante superior de Dantzig.

Se $\lfloor \overline{z}^{*i} \rfloor < z^L$ então $x_i^{*o} = 1$, $i = 1, \ldots, k$

Se $\lfloor \overline{z}^{*i+1} \rfloor < z^L$ então $x_i^{*o} = 0$, $i = k, \ldots, n$

Se acontecer de um dado teste falhar, isto é, se $\lfloor \overline{z}^{*i} \rfloor \geq z^L$ ($i = 1, \ldots, n+1$) então se verifica se o valor da solução, obtida zerando-se a variável de valor fracionário (se esta não existir toma-se a própria solução) na solução $\overline{x}^{*i}$ de P_i , é maior que z^L. Se for, atualiza-se x^L e z^L com tal solução. Com isto vai se melhorando, sempre que possível, a solução x^L , z^L.

De nada adiantaria analisar problemas do tipo:

$$\{P_o \mid x_j = 1\} \qquad j = 1, \ldots, k-1 \tag{3.44}$$

$$\text{ou} \quad \{P_o \mid x_j = 0\} \qquad j = k+1, \ldots, n \tag{3.45}$$

pois em cada um destes casos o valor do limitante superior obtido será $\lfloor \bar{z}^{*o} \rfloor$ e como $\bar{z}^{*o} \geqslant z^{*o} \geqslant z^L$ nunca se conseguiria obter a redução através destes tipos de problemas.

3.4.5.3.1 - Exemplo

Aplicando-se o algoritmo de Ingargiola-Korsh ao problema dado a seguir, e tomando inicialmente a solução $x^L = (1,1,0,0,0,0,0)$, $z^L = 120$; obtém-se a redução de 3 das 7 variáveis: x_1, x_5 e x_6.

P_o

$$\text{Max} \quad 60x_1 + 60x_2 + 40x_3 + 10x_4 + 20x_5 + 10x_6 + 3x_7 \tag{3.27}$$

$$\text{Sujeito a} \quad 30x_1 + 50x_2 + 40x_3 + 10x_4 + 40x_5 + 30x_6 + 10x_7 \leqslant 100 \tag{3.28}$$

$$0 \leqslant x_j \leqslant 1 \qquad j = 1, \ldots, 7 \tag{3.29}$$

$$x_j \text{ inteiro} \qquad j = 1, \ldots, 7 \tag{3.30}$$

$K = 3$

Problema $P_1 = \{P_o \mid x_1 = 0\}$

$z^{u1} = \lfloor \bar{z}^{*1} \rfloor = 110 < z^L = 120 \implies \underline{x_1^{*o} = 1}$

Problema $P_2 = \{P_o \mid x_2 = 0\}$

$z^{u2} = \lfloor \bar{z}^{*2} \rfloor = 120 = z^L$

$\bar{x}^{*2} = (1,0,1,1,1/2,0,0) \implies (1,0,1,1,0,0,0) \implies z = 110 < z^L$

Problema $P_3 = \{P_o \mid x_3 = 0\}$

$z^{u3} = \lfloor \bar{z}^{*3} \rfloor = 135 < z^L = 130$

$\bar{x}^{*3} = (1,1,0,1,1/4,0,0) \Rightarrow (1,1,0,1,0,0,0) \Rightarrow z = 130 > z^L$

$z^L = 130 \quad , \quad x^L = (1,1,0,1,0,0,0)$

Problema $P_4 = \{P_o \mid x_3 = 1\}$

$z^{u4} = \lfloor \bar{z}^{*4} \rfloor = 136 > z^L = 130$

$\bar{x}^{*4} = (1,3/5,1,0,0,0,0) \Rightarrow (1,0,1,0,0,0,0) \Rightarrow z = 100 < z^L$

Problema $P_5 = \{P_o \mid x_4 = 1\}$

$z^{u5} = \lfloor \bar{z}^{*5} \rfloor = 136 > z^L = 130$

$\bar{x}^{*5} = (1,1,1/4,1,0,0,0) \Rightarrow (1,1,0,1,0,0,0) \Rightarrow z = 130 = z^L$

Problema $P_6 = \{P_o \mid x_5 = 1\}$

$z^{u6} = \lfloor \bar{z}^{*6} \rfloor = 116 < z^L = 130 \Rightarrow \underline{x_5^{*o} = 0}$

Problema $P_7 = \{P_o \mid x_6 = 1\}$

$z^{u7} = \lfloor \bar{z}^{*7} \rfloor = 118 < z^L = 130 \Rightarrow \underline{x_6^{*o} = 0}$

Problema $P_8 = \{P_o \mid x_7 = 1\}$

$z^{u8} = \lfloor \bar{z}^{*8} \rfloor = 133 > z^L = 130$

$\bar{x}^{*7} = (1,1,1/4,0,0,0,1) \Rightarrow (1,1,0,0,0,0,1) \Rightarrow z = 123 < z^L$

3.4.5.4 - Algoritmo de Dembo-Hammer

O algoritmo proposto em 1980 por Dembo-Hammer [07] para redução

do PMO1 utiliza um novo limitante superior $z^{u_i} = \lfloor \hat{z}^i \rfloor$ para cada problema P_i gerado. Esse novo limitante é pior que o de Dantzig utilizado por Ingargiola-Korsh, sendo de se esperar então que menos variáveis sejam reduzidas, mas em compensação será bem mais rápido.

Neste problema serão também gerados dois tipos de problemas:

$$P_i = \{P_o \mid x_i = 0\} \qquad i = 1, \ldots, k-1 \tag{3.46}$$

e

$$P_i = \{P_o \mid x_i = 1\} \qquad i = k+1, \ldots, n \tag{3.47}$$

onde k é o índice da variável de valor fracionário na solução ótima $(\overline{x}^{*o})$ de P_o.

A seguir mostraremos quem é e como se calcula o novo limitante superior $\lfloor \hat{z}^i \rfloor$ do problema P_i.

O problema $\overline{P_o}$ cujo valor ótimo é $\overline{z}^{*o}$, de solução ótima $\overline{x}^{*o}$ definida por (3.39), (3.40) e (3.41) é definido por:

$\boxed{\overline{P_o}}$

$$\text{Max } \overline{z}^o = \sum_{j=1}^{n} v_j x_j \tag{3.48}$$

Sujeito a

$$\sum_{j=1}^{n} p_j x_j \leqslant M \tag{3.2}$$

$$0 \leqslant x_j \leqslant 1 \qquad j = 1, \ldots, n \tag{3.3}$$

E o problema $\overline{P_i}$ cujo valor ótimo é $\overline{z}^{*i}$ é definido por:

$\overline{P_i}$ $\quad$ Max $\bar{z}^i = \sum_{j=1}^{n} v_j x_j$ (3.49)

Sujeito a $\quad \sum_{j=1}^{n} p_j x_j \leqslant M$ (3.2)

$0 \leqslant x_j \leqslant 1 \qquad j = 1, \ldots, n$ (3.3)

$$x_i = \begin{cases} 0 & \text{se } 1 \leqslant i \leqslant k-1 \\ 1 & \text{se } k+1 \leqslant i \leqslant n \end{cases} \quad (3.50)$$

No algoritmo de Ingargiola-Korsh o valor $\lfloor \bar{z}^{*i} \rfloor$ é tomado como limitante superior do valor ótimo z^{*i} de P_i.

Sabe-se que

$$\bar{z}^{*i} \leqslant z^{*o} \quad (3.51)$$

pois toda solução viável de $\overline{P_i}$ é também de $\overline{P_o}$ (a recíproca não é verdadeira) e a função objetivo de ambos tem os mesmos coeficientes v_j, $j = 1,\ldots,n$.

Dembo-Hammer propõe utilizar um novo limitante superior ($\lfloor \hat{z}^i \rfloor$) do valor ótimo de P_i tal que:

$$\bar{z}^{*i} \leqslant \hat{z}^i \leqslant \bar{z}^{*o} \quad (3.52)$$

Isto é, $\hat{z}^i$ não é tão bom quanto $\bar{z}^{*i}$ mas, obviamente, é melhor que $\bar{z}^{*o}$.

Tendo em vista (3.52) tem-se:

$$\overline{z}^{*i} = \overline{z}^{*o} - \Delta^i \quad \text{com} \quad \Delta^i \geq 0 \tag{3.53}$$

A idéia de Dembo-Hammer é fazer uma estimativa do valor de Δ^i. De posse de Δ^i_{est} (Δ^i estimado) define-se $\hat{z}^i$ como sendo

$$\hat{z}^i = \overline{z}^{*o} - \Delta^i_{est} \tag{3.54}$$

Logicamente Δ^i_{est} deve satisfazer a inequação:

$$\overline{z}^{*i} \leq \overline{z}^{*o} - \Delta^i_{est} \quad \text{com} \quad \Delta^i_{est} \geq 0 \tag{3.55}$$

Evidentemente deve-se esperar que o cálculo de Δ^i_{est} seja menos trabalhoso que calcular o valor exato de Δ^i, isto é, calcular $\overline{z}^{*o} - \overline{z}^{*i}$.

Dembo-Hammer baseiam-se na teoria de relaxação lagrangeana, veja Geoffrion [10], Maculan Filho [19], para calcular Δ^i_{est}. Tendo em vista a simplicidade do problema, o cálculo deste mesmo Δ^i_{est} pode ser enfocado de um modo mais particularizado e portanto não apelaremos para esta teoria geral.

Seja $\overline{x}^{*o}$ a solução ótima de $\overline{P_o}$, definida por:

$$\overline{x}^{*o}_j = 1 \qquad j = 1, \ldots, k-1 \tag{3.56}$$

$$\overline{x}^{*o}_j = (M - \sum_{i=1}^{k-1} p_i) / p_k \tag{3.57}$$

$$\overline{x}^{*o}_j = 0 \qquad j = k+1, \ldots, n \tag{3.58}$$

A equação (3.48) calculada na solução ótima é escrita como:

$$\bar{z}^{*o} = \sum_{j=1}^{k-1} v_j \times 1 + \frac{v_k}{p_k} \left(M - \sum_{j=1}^{k-1} p_j\right) + \sum_{j=k+1}^{n} v_j \times 0 \qquad (3.59)$$

Pelo Lema A.1 (veja Apêndice A), a solução ótima $\bar{x}^{*i}$ de $\overline{P_i}$ é tal que:

$$\sum_{j=1}^{n} p_j \bar{x}_j^{*i} = M \qquad (3.60)$$

Multiplicando a equação acima (3.60) por $(-v_k/p_k)$ e adicionando à função objetivo de $\overline{P_i}$ calculada no seu valor ótimo, tem-se:

$$\bar{z}^{*i} = \sum_{j=1}^{n} \left(v_j - \frac{v_k}{p_k} p_j\right) \bar{x}_j^{*i} + \frac{v_k}{p_k} M$$

ou

$$\bar{z}^{*i} = \sum_{j=1}^{k-1} \left(v_j - \frac{v_k}{p_k} p_j\right) \bar{x}_j^{*i} + 0 + \sum_{j=k+1}^{n} \left(v_j - \frac{v_k}{p_k} p_j\right) \bar{x}_j^{*i} + \frac{v_k}{p_k} M \qquad (3.61)$$

Subtraindo (3.61) de (3.59) tem-se:

$$\bar{z}^{*o} - \bar{z}^{*i} = \sum_{j=1}^{k-1} \left(v_j - \frac{v_k}{p_k} p_j\right)\left(1 - \bar{x}_j^{*i}\right) + 0 + \sum_{j=k+1}^{n} \left(v_j - \frac{v_k}{p_k} p_j\right)\left(0 - \bar{x}_j^{*i}\right) \qquad (3.62)$$

Como $\Delta^i = \bar{z}^{*o} - \bar{z}^{*i}$ tem-se que (3.62) é a expressão do valor

exato de Δ^i. Evidentemente não se dispõe dos $\overline{x}_j^{*i}$ para se calcular o valor exato de Δ^i.

Mas podemos, através de (3.62) estimar um valor mínimo para Δ^i e tomar esse $\Delta^i_{min} = \Delta^i_{est}$ e desta forma teremos calculado $\hat{z}^i$.

Analisemos o sinal dos coeficientes $(v_j - \frac{v_k}{p_k} p_j)$ na equação (3.62):

Para $j = k$ tem-se $v_j - \frac{v_k}{p_k} p_j = 0$ (3.62')

Para $j = 1, \ldots, k-1$ tem-se $v_j - \frac{v_k}{p_k} p_j \geqslant 0$ (3.62'')

Para $j = k+1, \ldots, n$ tem-se $v_j - \frac{v_k}{p_k} p_j \leqslant 0$ (3.62''')

isto porque $\frac{v_j}{p_j} \geqslant \frac{v_{j+1}}{p_{j+1}}$, $j = 1, \ldots, n-1$. (Condição 3.8 de P_o).

Logo (3.62) será reescrita como:

$$\Delta^i = \sum_{j=1}^{k-1} \overbrace{(v_j - \frac{v_k}{p_k} \cdot p_j)}^{\geqslant 0} (1 - \overline{x}^{*i}) - \sum_{j=k+1}^{n} \overbrace{(v_j - \frac{v_k}{p_k} p_j)}^{\leqslant 0} \overline{x}^{*i} \qquad (3.63)$$

Mas em cada problema $\overline{P_i}$ conhecemos o valor ótimo de pelo menos uma variável, que é o da variável fixada x_i. Logo $\overline{x}_i^{*i}$ é conhecido. A respeito das demais variáveis nada poderemos afirmar quanto aos seus valores ótimos.

Seja $\overline{x}_i^{*i}$ o valor da variável fixada no problema P_i, se

$i \leqslant k-1$ então $\bar{x}_i^{*i} = 0$, neste caso um limitante inferior para Δ^i valerá $(v_i - \frac{v_k}{p_k} p_i)$; se $i \geqslant k+1$ então $\bar{x}_i^{*i} = 1$, neste caso um limitante inferior para Δ^i valerá $(\frac{v_k}{p_k} p_i - v_i)$.

Tendo em vista (3.63) podemos afirmar que qualquer que seja o valor i teremos:

$$\Delta^i \geqslant \left| v_i - \frac{v_k}{p_k} p_i \right| \tag{3.64}$$

Logo podemos adotar para o valor de $\hat{z}^i$:

$$\hat{z}^i = \bar{z}^{*o} - \left| v_i - \frac{v_k}{p_k} p_i \right| \qquad i = 1, \ldots, n \tag{3.65}$$

Para cada problema do tipo (3.46) ou (3.47) é calculado o limitante superior $z^{u_i} = \lfloor \hat{z}^i \rfloor$. Se $z^{u_i} < z^L$ então $\bar{x}_i^{*o} = 1$ se $i = 1, \ldots, k-1$ ou $\bar{x}_i^{*o} = 1$ se $i = k+1, \ldots, n$.

Também neste algoritmo, de nada adiantaria examinar os problemas do tipo:

$$\{P_o \mid x_i = 1\} \qquad i = 1, \ldots, k \tag{3.66}$$

ou

$$\{P_o \mid x_i = 0\} \qquad i = k, \ldots, n \tag{3.67}$$

pois em ambos os casos o limitante inferior para Δ^i seria 0 e portanto $\hat{z}^i = \bar{z}^{*o}$. Logo $z^{u_i} = \hat{z}^i = \bar{z}^{*o} \geqslant z^L$.

3.4.5.4.1 - Exemplo

Aplicando o algoritmo de Dembo-Hammer ao mesmo exemplo de 3.4.5.3.1, tomando inicialmente $x^L = (1,1,0,0,0,0,0)$, $z^L = 120$, obtêm-se a redução de uma das 7 variáveis: x_1 .

$k = 3$, $\bar{z}^{*0} = 140$

$\frac{v_k}{p_k} = 1$

Problema $P_1 = \{ P_0 \mid x_1 = 0 \}$

$$z^{u1} = \lfloor \hat{z}^1 \rfloor = \lfloor 140 - | 60 - 30 | \rfloor = 110 < z^L = 120 \Rightarrow \underline{\bar{x}_1^{*0} = 1}$$

Problema $P_2 = \{P_0 \mid x_2 = 0\}$

$$z^{u2} = \lfloor \hat{z}^2 \rfloor = \lfloor 140 - |60 - 50| \rfloor = 130 > z^L = 120$$

Problema $P_3 = \{P_0 \mid x_4 = 1\}$

$$z^{u3} = \lfloor \hat{z}^3 \rfloor = \lfloor 140 - |10 - 10| \rfloor = 140 > z^L$$

Problema $P_4 = \{P_0 \mid x_5 = 1\}$

$$z^{u5} = \lfloor \hat{z}^5 \rfloor = \lfloor 140 - |20 - 40| \rfloor = 120 = z^L$$

Problema $P_5 = \{P_0 \mid x_6 = 1\}$

$$z^{u6} = \lfloor \hat{z}^6 \rfloor = \lfloor 140 - |10 - 30| \rfloor = 120 = z^L$$

Problema $P_6 = \{P_0 \mid x_7 = 1\}$

$$z^{u7} = \lfloor \hat{z}^7 \rfloor = \lfloor 140 - |3 - 10| \rfloor = 133 > 120 = z^L$$

3.4.6 - Algoritmos mais recentes

3.4.6.1 - Introdução

Após a divulgação do algoritmo de Horowitz-Sahni e também dos algoritmos de redução, descritos nas subseções anteriores, constata-se um desenvolvimento mais acentuado na busca de melhores algoritmos para solucionar o PMO1. A maioria desses algoritmos são do tipo "branch-and-bound", outros porém utilizam essa metodologia pelo menos em parte.

Descreveremos a seguir aspectos fundamentais de alguns desses algoritmos, sem descer a detalhes como fizemos para Kolesar, Greemberg-Hegerich e Horowitz-Sahni.

Salientamos que sempre ao mencionarmos o PMO1 P_o, tal como foi definido em (3.1) a (3.4), leva-se em consideração as condições (3.5) a (3.8).

3.4.6.2 - Algoritmo de Nauss

O algoritmo de Nauss [26], publicado em 1976, apresenta pequenas variações em relação ao de Horowitz-Sahni, como o próprio autor afirma, mas que reduz o tempo de execução em cerca de 1/3.

Nauss utiliza uma rotina de redução, tipo Dembo-Hammer, antes

de aplicar uma rotina de solução completa do PMO1. Nessa rotina Nauss introduz as seguintes alterações em relação ao algoritmo de Horowitz-Sahni:

1) No cálculo do limitante superior para um problema P_i, obtém-se também $\bar{x}^{*i}$, e se esta for inteira verifica-se se $\bar{z}^{*i} > z^L$, caso afirmativo atualiza-se x^L e z^L com os valores $\bar{x}^{*i}$ e $\bar{z}^{*i}$, respectivamente.

2) A expansão de um problema P_i se faz de acordo com o esquema exemplificado na figura 3.18.

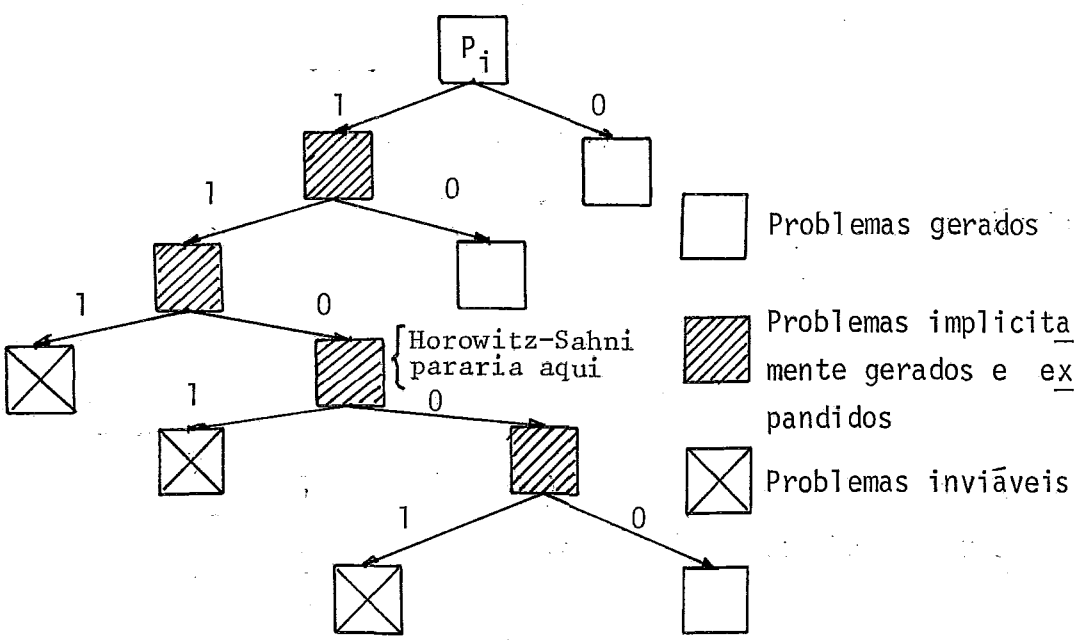


Fig. 3.18 - Expansão de um problema no algoritmo de Nauss.

Depois de uma série de valores de variáveis fixadas em 1 (como em Horowitz-Sahni) forma-se uma série de valores de variáveis fixadas em 0, enquanto uma fixação em 1 gera um problema inviável. A estratégia de Nauss é aprofundar mais a busca em cada ramo da árvore, gerando um maior número de nós implicitamente.

3.4.6.3 - Algoritmo de Suhl

Divulgado em 1977, o algoritmo de Suhl |30| também utiliza uma rotina de redução antes de processar completamente o PMO1. Essa rotina de redução utiliza os dois métodos que apresentamos. Primeiro se aplica o método de Dembo-Hammer, se a variável não é reduzida, então se aplica o método de Ingargiola-Korsh. Essa metodologia se justifica porque Dembo-Hammer é mais rápido mas mais fraco que Ingargiola-Korsh, portanto se aplica primeiro um teste rápido mas fraco e só se este falhar é que se aplica o teste mais forte, mas mais trabalhoso.

A expansão dos problemas (P_i) no algoritmo de Suhl é feita gerando-se dois problemas a partir de P_i fixando-se primeiramente a variável livre de P_i de menor índice em 1 e depois em 0.

O problema gerado com a variável fixada em 0 é colocado por último na lista e como o critério de escolha é LIFO ele é analisado primeiro. Isto faz com que, de uma maneira geral, os problemas a serem primeiro analisados tem as variáveis de maior razão valor/peso fixadas em 0, mas são justamente tais variáveis as que mais contribuem para elevar o valor do limitante superior, zerando-as obtém-se baixos valores de limitantes superiores e portanto maior a chance de serem podados os problemas. Esta estratégia tenta podar os ramos da árvore o mais prematuramente possível, só aprofundando a busca se ela se mostrar bastante promissora.

Essa maneira de gerar e colocar os problemas na lista, exige

que o algoritmo de Suhl apresente uma estrutura de dados bastante eficiente.

3.4.6.4 - Algoritmo de Martello-Toth

Também em 1977 é divulgado o algoritmo de Martello-Toth |[22]|, que segundo os autores superava em rapidez os mais eficientes algoritmos até então conhecidos.

Em 1978 é divulgada |[23]| uma implementação em Fortran desse mesmo algoritmo. Salientam que o desempenho do algoritmo para problemas com até 1000 variáveis é superior ao de Nauss e Horowitz-Sahni. Afirmam também que quando pré-processaram os problemas gerados com o algoritmo de redução de Ingargiola-Korsh, a eficiência computacional diminui consideravelmente em relação aos dois outros citados algoritmos.

A grande força do algoritmo de Martello-Toth está na maneira como ele evita expansões desnecessárias. Para exemplificar citaremos duas dessas estratégias:

1) No início do algoritmo é calculado um bom limitante superior do valor ótimo de P_0. Esse limitante é conhecido como sendo de Martello-Toth, veja |[22]|, |[24]| e também o Apêndice B. A cada nova solução x^L que se obtém compara-se z^L com esse limitante superior, se eles forem iguais, então a solução x^L é ótima.

2) Para cada variável x_{k-1} é procurado o valor do menor peso existente entre ela e a última variável (inclusive). Seja $p_{min,\,k-1}$ tal peso, isto é, $p_{min,\,k-1} = Min\ \{p_j \mid j = k, \ldots, n\}$. Seja um problema P_i onde as k-1 primeiras variáveis estão fixadas e seja $\bar{M}$ o peso ainda não ocupado da mochila em P_i. Se $p_{min,\,k-1} > \bar{M}$ então será desnecessário expandir P_i pois a solução ótima de P_i é conhecida bastando zerar as variáveis livres de P_i.

3.4.6.5 - Algoritmo de Zoltners

A principal característica do algoritmo de Zoltners [32], divulgado em 1978, é a aplicação de uma rotina de redução embutida dentro do algoritmo, que é chamado toda vez que se obtém um valor melhorado para solução x^L, z^L.

3.4.6.6 - Algoritmos que não exigem ordenação

Os algoritmos apresentados até aqui exigem que os dados sejam previamente ordenados de acordo com (3.8), isto é, segundo a razão não crescente valor/peso. A ordenação desses dados chega muitas vezes a se constituir numa parcela ponderável do tempo total gasto para resolver um PMO1. Segundo Martello-Toth [24] essa parcela chega a 32% para problemas com 50 variáveis, a 39% com 100 variáveis, e a 46% com 200 variáveis. Há, portanto, forte indício de que quanto maior o número de variáveis de um problema, maior será o tempo relativo gasto na ordenação dos dados.

Atualmente dois algoritmos, Balas-Zemmel [02] e Fayard-Plateau [08] estão na classe dos algoritmos que resolvem o PMO1 sem ordenação, isto é, sem exigir a validade da condição (3.8).

3.5. DESENVOLVIMENTO DE UM ALGORITMO "BRANCH-AND-BOUND" PARA SOLUÇÃO DO PMO1

3.5.1 - Introdução

Nesta seção apresentaremos o algoritmo KPVILL que determina uma solução ótima de um PMO1, tal como definido por (3.1) a (3.8). A aplicação de KPVILL pode ser precedida de um algoritmo de redução para se tentar diminuir o número de variáveis do PMO1 dado originalmente. Obrigatoriamente a aplicação de KPVILL deve ser precedida de um algoritmo de ordenação para arrumar as variáveis numa ordem não crescente das razões valor/peso, isto é, de acordo com a condição (3.8).

Com pequenas alterações KPVILL pode determinar não somente uma mas todas as soluções ótimas de um PMO1. Esta faculdade é importante quando se aplica KPVILL ao problema oriundo da modelagem feita no Capítulo 2, onde interessa saber qual a configuração de Postos a serem instalados. Havendo mais de uma configuração é interessante determiná-las pois isto facilitará a tomada de decisão pelo planejador, que poderia então optar entre mais de uma configuração ótima, e isto lhe permite levar em conta outros fatores, não considerados na modelagem, para tomada de decisão.

Segundo Breu-Burdet [03], um "branch-and-bound" pode ser visto sob duas óticas: tática e estratégica. A tática envolve informações locais de cada nó (problema), ao passo que a estratégica, lida com informações geradas em vários nós. Uma melhoria tática é aquela que aumenta a eficiência num único nó, isto é, os cálculos feitos para um determinado nó diminuem. Já uma melhoria estratégica tenta, em última análise, reduzir o número de nós gerados. Nem uma melhoria tática nem uma melhoria estratégica garantem, na maioria das vezes, isoladamente uma melhoria global do algoritmo. Uma melhoria tática pode ocasionar um aumento demasiado do número de nós gerados. Uma melhoria estratégica pode exigir mais cálculos em cada nó.

Na seção 3.4.4 apresentamos o algoritmo de Horowitz-Sahni [14] nos preocupando em mostrar a estrutura lógica do mesmo procurando evidenciar a estratégia e a tática utilizadas no mesmo. Essa abordagem permitiu uma comparação clara com outros algoritmos "branch-and-bound", o que seria pelo menos mais difícil se mostrássemos o algoritmo tal como originalmente proposto. Nesta seção mostraremos a estrutura de dados usada por Horowitz-Sahni capaz de representar os problemas gerados. A inclusão desta estrutura de dados no algoritmo da seção 3.4.4 reconstitui o algoritmo de Horowitz-Sahni tal como é apresentado em [14]. Denominaremos por HSAN esse algoritmo. A inclusão de uma estrutura de dados é necessária face ao interesse em implementar computacionalmente os algoritmos HSAN e KPVILL.

O algoritmo KPVILL tem uma estratégia igual a HSAN, bem como a mesma estrutura de dados. A novidade é que KPVILL introduz uma melhoria tática sem prejudicar a estratégia, isto é, sem aumentar o número de problemas gerados, o que se traduz numa melhoria global de KPVILL em

relação a HSAN, e se constatará com os testes computacionais mostrados na seção 3.6.

Essa melhoria tática consiste em diminuir o número de operações realizadas quando se calcula o limitante superior do valor ótimo de cada problema sem que com isso se obtenha um limitante superior pior que aquele sugerido por Horowitz-Sahni.

No Apêndice C apresentamos a listagem da implementação de KPVILL em Basic, inserida dentro de um programa que resolve especificamente o problema da determinação dos postos a serem instalados tal como apresentado na modelagem descrita no Capítulo 2. Esse programa é conversacional, evitando-se assim as dificuldades inerentes à formatação dos dados de entrada e à análise dos resultados de saída. O programa inclui além de KPVILL, uma rotina de ordenação (Shellsort, veja |[18]|) e uma rotina de redução baseada no algoritmo de Ingargiola-Korsh.

3.5.2 - Algoritmo HSAN

Inicialmente descreve-se a estrutura de dados proposta por Horowitz-Sahni para representar os problemas gerados. Logo a seguir é apresentado o algoritmo HSAN que nada mais é que a inclusão dessa estrutura de dados no algoritmo descrito na seção 3.4.4.

3.5.2.1 - Estrutura de dados

Um problema (P) qualquer gerado será representado por um vetor x e um apontador k. Denominaremos esta estrutura por: P = [x, k]. O apontador (k) armazena o índice da variável livre de menor índice no problema P, e o vetor x armazena os valores $\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{k-1}$ das variáveis fixadas no problema P.

Quando um problema tiver todas as variáveis fixadas convencio na-se que $k = n+1$, onde n é o número de variáveis do problema original P_o. Se um problema não tem nenhuma variável fixada então $k = 1$. Portanto tem-se que os possíveis valores para k são tais que $1 \leqslant k \leqslant n+1$.

Tendo em vista os valores que k pode assumir, o vetor x terá um comprimento fixo igual a n. mas somente as $(k-1)$ primeiras posições caracterizarão o problema, as demais conterão valores (denotados por *) que não têm significado algum para a definição do problema.

Na figura 3.20, o problema P é representado por:
P = [(1,1,0,*,*,...,*),4].

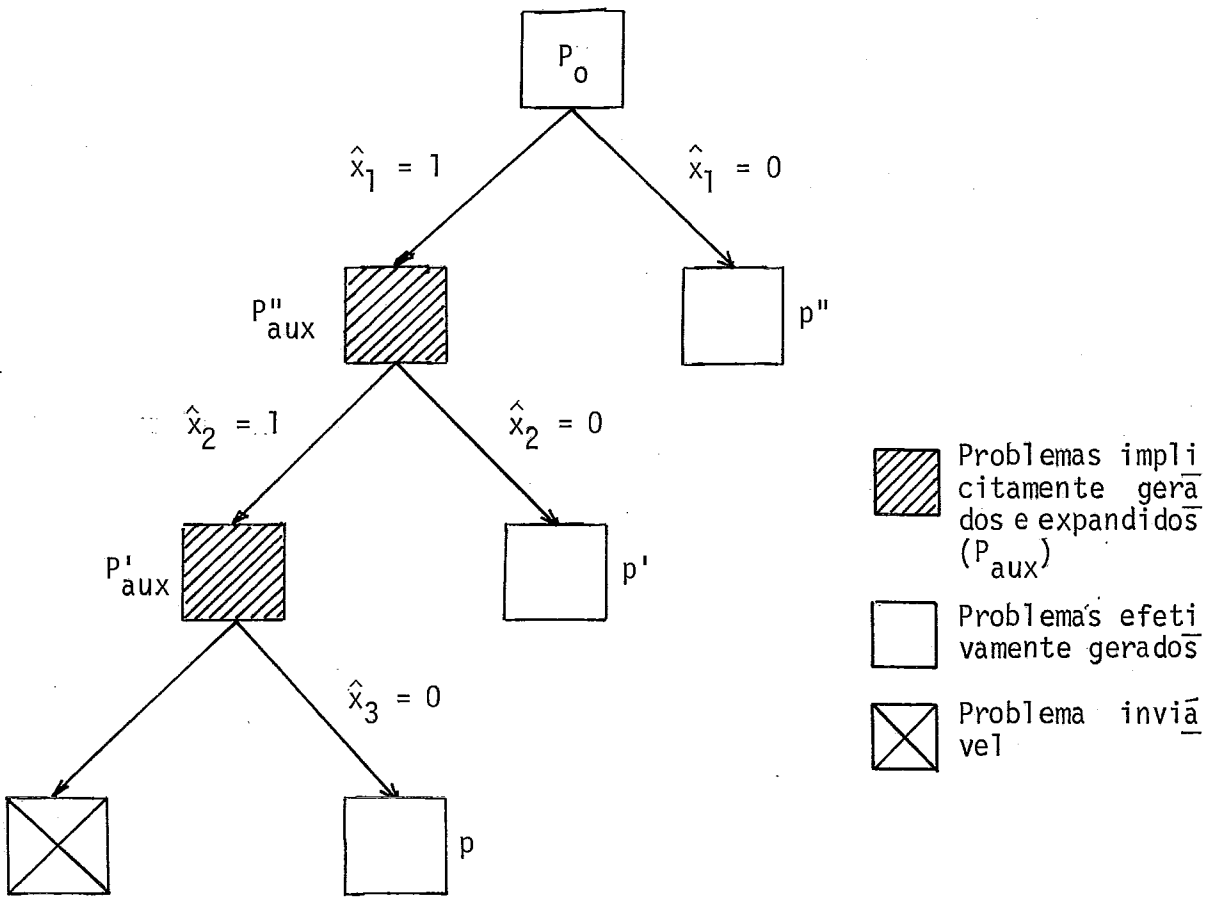


Fig. 3.20 - Árvore (trecho) "branch-and-bound" típica gerada pelo algoritmo de Horowitz-Sahni.

Ainda a título de exemplo:

1) O problema P_o será representado por $P_o = [x, 1]$, isto é, todas as variáveis são livres ($k = 1$) em P_o.

2) Um problema com todas as variáveis fixadas será representado por $[x, n+1]$.

3) Um problema onde somente a primeira variável está fixada em 1 é representado por $[(1,*,\ldots,*),2]$.

4) As estruturas $[x,0]$ ou $[x, n+2]$ não representam problema algum.

Uma maneira intuitiva de interpretar a estrutura de dados de um dado problema P é imaginar que o vetor x armazena o caminho na árvore "branch-and-bound" para ir de P_0 até P, e o apontador k indica quando parar nesta trilha.

De posse da estrutura $[x, k]$ de um problema P é sempre possível obter a estrutura de <u>todos</u> os problemas que foram gerados antes de P e na ordem que o precederam. Basta perceber que toda vez que há uma derivação (na árvore) para a esquerda (ver figura 3.20), isto é, $\hat{x}_i = 1$, sabemos que um problema é gerado derivando-se para a direita, isto é, fazendo-se $\hat{x}_i = 0$.

Por exemplo, na figura 3.20 os problemas que precederam $P = [(1, 1, 0, *, \ldots, *), 4]$ são (pela ordem):

$$P'_{aux} = [(1, 1, *, *, \ldots, *), 3]$$

$$P' = [(1, 0, *, *, \ldots, *), 3]$$

$$P''_{aux} = [(1, *, *, *, \ldots, *), 2]$$

$$P'' = [(0, *, *, *, \ldots, *), 2]$$

$$P_0 = [(*, *, *, *, \ldots, *), 1]$$

Este exemplo sugere que é possível recuperar a partir da estrutura de um problema qualquer P a própria lista L de problemas a serem expandidos (No exemplo citado L = {P', P"}) é inclusive na ordem em que foram colocados na mesma.

Desta maneira a definição dessa estrutura de dados para os problemas permite:

1) Caracterizar eficientemente qualquer problema gerado pelo algoritmo.

2) Eliminar a existência da lista L, pois a mesma pode ser recuperada a partir de qualquer problema.

3) Recuperar o último problema que teria sido colocado na lista L.

De posse da estrutura $[x, k]$ de um problema P, para se obter o último problema que teria entrado na lista L antes de P basta encontrar a variável $\hat{x}_i$ de mais alto índice no vetor x tal que $\hat{x}_i = 1$ onde $i < k$. Se não existir tal variável é porque a lista L estaria vazia. Se existir então o último problema que teria entrado na lista L antes de P é:

$$[(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{i-1}, 0, *, \ldots, *), i+1]$$

Esse procedimento, isto é, a busca do último problema que entrou na lista antes de P, é conhecido na literatura como retorno na trilha ("back-track").

Com a definição dessa estrutura de dados é necessário, no algoritmo, guardar somente o problema (P*) cabeça da árvore "branch-and-bound", isto é, aquele que é objeto de análise num dado instante da evolução do algoritmo, pois a partir de sua estrutura é possível obter o último problema que foi colocado na lista antes dele e conseqüentemente toda a lista L.

Para aumentar a eficiência do algoritmo, embora isto não seja necessário para caracterizar um dado problema, são também calculados (recursivamente) para o problema cabeça da árvore a sobra (S) de peso da mochila, isto é, $S = M - \sum_{j=1}^{k-1} p_j \hat{x}_j$ e o valor (Z) já atribuido, isto é, $Z = \sum_{j=1}^{k-1} v_j \hat{x}_j$. Desta forma além da estrutura $[x, k]$ serão também calculados a sobra S e o valor Z do problema cabeça da árvore.

3.5.2.2 - Descrição do algoritmo HSAN

De acordo com a estrutura lógica do algoritmo de Horowitz-Sahni tal como descrita na seção 3.4.4 e inserindo a estrutura de dados que acaba de ser definida descreveremos então o algoritmo HSAN. Utilizaremos para isto uma linguagem próxima do ALGOL destacando-se as seguintes exceções: 1) Textos grifados são comentários; 2) Textos entre chaves ({ }) representam procedimentos descritos genericamente.

Algoritmo HSAN

BEGIN

PASSO 1: Inicialização do problema P* (= P_0)

k := 1 ;
x := (0, 0, ..., 0);
S := M ;
Z := 0 ;

PASSO 2: Inicialização da melhor solução x^L, z^L

x^L := {$\bar{x}^{*0}$, zerando-se a variável de valor fracionário} ;
z^L := {valor da solução x^L };
GOTO PASSO 6 ;

PASSO 3: Teste de finalização

FOR i := k-1 STEP -1 UNTIL 1 DO

BEGIN

IF x_1 = 1 THEN GOTO PASSO 4

END ;

IF i = 0 THEN GOTO PASSO 8 ;

PASSO 4: Recuperação do último problema (P*) da lista

k := i + 1 ;
x_i := 0 ;
S := S + p_i ;
Z := Z - v_i ;

PASSO 5: O problema (P*) deve ser podado?

Todas as variáveis foram fixadas?

IF $k = n + 1$ THEN GOTO PASSO 7;

Cálculo do limitante superior z^u

$z^u := [z^*(\overline{P}^*)]$;

Teste de não otimalidade

IF $z^u \leqslant z^L$ THEN GOTO PASSO 3;

PASSO 6: Expansão do problema (P^*)

WHILE $k \leqslant n$ AND $p_k \leqslant S$ DO

BEGIN

$S := S - p_k$;
$Z := Z + v_k$;
$x_k := 1$;
$k := k + 1$

END;

Subpassos correspondentes à definição do problema P_{aux}.

IF $k \leqslant n$ THEN

BEGIN

$x_k := 0$;
$k := k + 1$

END;

GOTO PASSO 5;

PASSO 7: Atualização da melhor solução x^L, z^L

IF $z > z^L$ THEN

BEGIN

$z^L := Z$;

$X^L := x$

END ;

GOTO PASSO 3 ;

PASSO 8: Finalização

$x^* := x^L$;

$z^{*o} := z^L$

END .

Observações:

1. No passo 1, o problema $P_0 = [(0, 0, \ldots, 0), 1]$, S = M, Z = 0 é inicialmente o cabeça da árvore (P*).
2. Após o passo 2 vai-se direto ao passo 6 pois não teria sentido (nesse instante do algoritmo) executar os passos 3, 4, e 5 (como foi feito na seção 3.4.4).
3. O passo 3 verifica se há algum problema a ser expandido (L = Ø ?). Caso afirmativo ele fornece a variável x_i que vai ser fixada em 0 no passo 4. Caso negativo (i = 0) vai-se ao passo 8 de finalização.

4. No passo 4 é recuperada a estrutura e os parâmetros S e Z do último problema que teria entrado na lista L, que é o problema (P*) a ser analisado no passo 5.

5. No passo 5 são feitas algumas melhorias (em relação ao algoritmo na seção 3.4.4) quando é detectado uma das condições de poda. Assim se todas as variáveis estão fixadas (k = n + 1) vai-se ao passo 7 e neste atualiza-se (se possível) x^L e z^L. Se o teste de não otimalidade resulta $z^u \leqslant z^L$ vai-se direto ao passo 3 evitando-se passar pelo passo 7, 3 e 4.

6. No passo 6 é importante salientar que k, x, S e Z se referem sempre ao problema cabeça da árvore (P*). Notar também que, ao contrário do que se fazia na seção 3.4.4, não se toma nenhuma providência quanto aos problemas $\{P_{aux} \mid \hat{x}_k = 0\}$, isto porque os mesmos serão recuperados posteriormente no passo 4. Após o passo 6 vai-se direto ao passo 5 pois o problema que seria escolhido da lista é exatamente o cabeça da árvore (P*).

7. No passo 5, z^* ($\overline{P}^*$) é o valor ótimo do problema $P^*_{relaxado}$. O detalhamento de como calcular o limitante superior z^u pode ser visto no Apêndice B.

8. No passo 2 pode-se melhorar a solução inicial x^L, z^L (embora isso tenha se mostrado pouco prático), calculando-se uma "solução gulosa" de P_o tal como sugerido por Maculan Filho-Villares [20] ou Motta [25].

3.5.3 - Algoritmo KPVILL

No passo 5 do algoritmo HSAN fazemos o cálculo do limitante superior z^u. Veremos a seguir os detalhes necessários para se proceder a esse cálculo assim como uma inovação introduzida por KPVILL em relação a HSAN.

3.5.3.1 - Cálculo do limitante superior z^u

Seja um problema $P = [x, k]$, isto é:

$$\boxed{P} \qquad \text{Max } Z + \sum_{j=k}^{n} v_j x_j \tag{3.68}$$

$$\text{Sujeito a} \qquad \sum_{j=k}^{n} p_j x_j \leqslant S \tag{3.69}$$

$$0 \leqslant x_j \leqslant 1 \qquad j = k, \ldots, n \tag{3.70}$$

$$x_j \text{ inteiro} \qquad j = k, \ldots, n \tag{3,71}$$

onde $S = M - \sum_{j=1}^{k-1} p_j \hat{x}_j$ e $Z = \sum_{j=1}^{k-1} v_j \hat{x}_j$.

O limitante superior z^u do valor ótimo $z^*(P)$ de P é o limitante superior de Dantzig (ver Apêndice B) e é definido por $z^u = \lfloor z^*(\overline{P}) \rfloor$.

O cálculo de $z^*(\overline{P})$ é feito como se segue:

$$z^*(\bar{P}) = Z + \sum_{j=k}^{m-1} v_j + (S - \sum_{j=k}^{m-1} p_j) \frac{v_m}{p_m} \qquad (3.72)$$

onde m $(m \leqslant n)$ é o maior inteiro tal que $\sum_{j=k}^{m-1} p_j \leqslant S$.

A obtenção do valor de m no algoritmo HSAN é feita colocando-se um a um os objetos de peso p_k, p_{k+1}, ... na mochila sem que se ultrapasse a sobra S.

Analisemos uma maneira mais econômica de se calcular m satisfeitas certas condições. Neste novo método é colocado um bloco inteiro de objetos.

Seja F o maior inteiro tal que $\sum_{j=1}^{F-1} p_j \leqslant M$.

Sabe-se que

$$S = M - \sum_{j=1}^{k-1} p_j \hat{x}_j \qquad (3.73)$$

Quaisquer que sejam os valores $\hat{x}_j$ das variáveis fixadas em $[x, k]$ e se $k \leqslant F - 1$ tem-se:

$$\sum_{j=1}^{k-1} p_j \hat{x}_j + \sum_{j=k}^{F-1} p_j \leqslant M \qquad (3.74)$$

ou

$$\sum_{j=k}^{F-1} p_j \leqslant S \qquad (3.75)$$

Chamemos de:

$$S_k = \sum_{j=k}^{F-1} p_j \tag{3.76}$$

e de

$$Z_k = \sum_{j=k}^{F-1} v_j \tag{3.77}$$

Se $k \leqslant F - 1$, a expressão (3.72) de $z^*(\overline{P})$ pode ser escrita como:

$$z^*(\overline{P}) = z + \sum_{j=k}^{F-1} v_j + \sum_{j=F}^{m-1} v_j + \left(S - \sum_{j=k}^{F-1} p_j - \sum_{j=F}^{m-1} p_j\right) \frac{v_m}{p_m} \tag{3.78}$$

onde m é o maior inteiro tal que $\sum_{j=F}^{m-1} p_j \leqslant S - \sum_{j=k}^{F-1} p_j$. Se não existe m satisfazendo a esta condição então convenciona-se que $\sum_{j=F}^{m-1} v_j = \sum_{j=F}^{m-1} p_j = 0$.

Portanto, se $k \leqslant F - 1$ tem-se:

$$z^u = \left\lfloor Z + Z_k + \sum_{j=F}^{m-1} v_j + (S - S_k - \sum_{j=F}^{m-1} p_j) \frac{v_m}{p_m} \right\rfloor \tag{3.79}$$

onde m é o maior inteiro tal que $\sum_{j=F}^{m-1} p_j \leqslant S - S_k$.

A expressão (3.79) para z^u sugere que se $k \leqslant F - 1$, z^u seja calculado de outra forma:

1. Calcula-se uma única vez, no início do algoritmo, os valores

de S_k e Z_k para cada $k \leqslant F-1$.

2. Durante a execução do algoritmo (passo 5), toda vez que $k \leqslant F-1$ pode-se calcular o valor de m da seguinte maneira: Coloca-se de uma vez na mochila o bloco de objetos de peso S_k, em seguida coloca-se um a um os objetos p_F, p_{F+1}, ... sem que se ultrapasse o peso ainda não preenchido da mochila $S - S_k$.

3. O valor de z^u será então dado pela equação (3.79).

3.5.3.2 - Descrição do algoritmo KPVILL

O algoritmo KPVILL se diferencia do algoritmo HSAN pelo detalhamento do cálculo do limitante superior z^u, e pela inclusão de um novo passo (PASSO 0) que procede o cálculo dos blocos, isto é, dos valores S_k e Z_k, para $k \leqslant F-1$.

Algoritmo KPVILL

```
BEGIN
  PASSO 0: Formação dos blocos
      d := M;
      FOR  j := 1  STEP 1  UNTIL n  DO
          BEGIN
              IF  p_j > d  THEN GOTO AQUI ;
              d := d - p_j
          END ;
```

AQUI: F := j

IF F = 1 THEN GOTO PASSO 1

S_F := 0 ;

Z_F := 0 ;

FOR j := F - 1 STEP -1 UNTIL 1 DO

BEGIN

$S_j := S_{j+1} + p_j$;

$Z_j := Z_{j+1} + v_j$:

END ;

PASSO 1: Inicialização do problema P* (= P_0)

k := 1 ;

x := (0, 0, ..., 0) ;

S := M ;

Z := 0 ;

PASSO 2: Inicialização da melhor solução x^L, z^L

x^L := {$\bar{x}^{*0}$, zerando-se a variável de valor fracionário} ;

z^L := {valor da solução x^L } ;

GOTO PASSO 6 ;

PASSO 3: Teste de finalização

FOR i := k-1 STEP -1 UNTIL 1 DO

BEGIN

IF x_i = 1 THEN GOTO PASSO 4

END ;

IF i = 0 THEN GOTO PASSO 8 ;

PASSO 4: Recuperação do último problema (P*) da lista

$k := i + 1$;

$x_i := 0$;

$S := S + p_i$;

$Z := Z - v_i$;

PASSO 5: O problema (P*) deve ser podado?

Todas as variáveis foram fixadas?

IF $k = n + 1$ THEN GOTO PASSO 7 ;

Cálculo do limitante superior z^u

$d := S$;

$z^u := Z$;

$j := k$;

Colocação por bloco

IF $k < F$ THEN

BEGIN

$d := d - S_k$;

$z^u := z^u + Z_k$;

$j := F$

END ;

Colocação um a um

WHILE $p_j \leq d$ AND $j \leq n$ DO

BEGIN

$d := d - p_j$;

$z^u := z^u + v_j$;

$j := j + 1$

END ;

IF $j \leq n$ THEN $z^u = \lfloor z^u + d * v_j/p_j \rfloor$;

Teste de não otimalidade

IF $z^u \leqslant z^L$ THEN GOTO PASSO 3

PASSO 6: Expansão do problema (P*)

```
WHILE  k ⩽ n  AND  p_k ⩽ S  DO
    BEGIN
        S := S - p_k ;
        Z := Z + v_k ;
        x_k := 1 :
        k := k + 1
    END ;
IF  k ⩽ n        THEN
    BEGIN
        x_k := 0 ;
        k := k + 1
    END ;
GOTO  PASSO 5 ;
```

PASSO 7: Atualização da melhor solução x^L, z^L

```
IF  z > z^L   THEN
   BEGIN
       z^L := Z ;
       x^L := x
   END ;
GOTO  PASSO 3 ;
```

PASSO 8: Finalização

$x^{*o} := x^{L}$;

$z^{*o} := z^{L}$

END .

3.6 - TESTES COMPUTACIONAIS

3.6.1 - Introdução

Para verificar a eficiência de KPVILL em relação ao algoritmo HSAN rodamos baterias de problemas padronizados para os dois algoritmos.

Cada problema gerado aleatoriamente foi resolvido de 4 maneiras:

1) Aplicando KPVILL diretamente.
2) Aplicando HSAN diretamente.
3) Aplicando o algoritmo de redução de Ingargiola-Korsh e depois KPVILL.
4) Aplicando o algoritmo de redução de Ingargiola-Korsh e depois HSAN.

Em cada bateria de problemas de um mesmo tipo eram contados os números médios de adições, multiplicações, de comparações, de buscas em vetor e de atribuições e o número total médio dessas operações.

Os testes foram rodados em micro-computador Schumec M100.

Os problemas padronizados são indicados por Martello-Toth em |[24]| e são de 3 tipos:

1) Com dados não correlacionados

$$p_{min} \leq p_j \leq p_{max}$$

$$v_{min} \leq v_j \leq v_{max}$$

2) Com dados fracamente correlacionados

$$p_{min} \leq p_j \leq p_{max}$$

$$\max \{0, p_j - r\} \leq v_j \leq p_j + s$$

3) Com dados fortemente correlacionados

$$p_{min} \leq p_j \leq p_{max}$$

$$v_j = p_j + C$$

Os valores dos parâmetros usados foram:

$$p_{min} = v_{min} = 1$$

$$p_{max} = v_{max} = 100$$

$$r = s = c = 10$$

O peso total M da mochila foi calculado de duas maneiras:

$$M = \left\lfloor 0{,}8 \sum_{j=1}^{n} p_j \right\rfloor , \quad \text{ou}$$

$$M = \left\lfloor 0{,}5 \sum_{j=1}^{n} p_j \right\rfloor$$

Quanto mais correlacionados forem os dados mais difíceis se tornam os problemas (veja Chvátval [05]) e por esta razão nos problemas do tipo (1) geramos problemas com 50 e 100 variáveis, nos do tipo (2) com 25 e 50, e nos do tipo (3) com 10 e 20 variáveis.

As contagens do número de operações não incluem aquelas efetuadas nas rotinas de ordenação e redução.

O cálculo do limitante superior (z^u) no PASSO 5 do algoritmo HSAN foi efetuado colocando-se os objetos um a um na mochila. À exceção da introdução do PASSO 0 e da nova maneira de se calcular z^u em KPVILL., o código computacional usado foi idêntico para as implementações dos dois algoritmos.

3.6.2 - Resultados obtidos

A coluna (%) se refera a relação percentual do número de operações efetuadas em KPVILL / HSAN.

I. DADOS NÃO CORRELACIONADOS. 30 problemas. 50 variáveis. $M = 0,8 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	1052	3456	30	124	128	97
Mult.	58	60	97	9	10	90
Comp.	780	2494	31	136	137	99
Buscas	1224	3651	34	158	163	97
Atrib.	1093	3753	29	172	186	93
Total	4207	13414	32	599	624	96

II. DADOS NÃO CORRELACIONADOS. 30 casos. 100 variáveis. $M = 0,8 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	2132	12314	17	280	355	79
Mult.	124	127	98	23	24	96
Comp.	1605	8615	19	295	343	86
Buscas	2502	12726	20	352	429	82
Atrib.	2221	12931	17	375	481	78
Total	8584	46713	19	1325	1632	81

III. DADOS NÃO CORRELACIONADOS. 30 casos. 100 variáveis. $M = 0,5 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	2598	8477	31	932	1164	80
Mult.	261	263	99	96	97	99
Comp.	2840	6763	42	1105	1214	91
Buscas	3372	9267	36	1202	1436	84
Atrib.	3358	9649	35	1317	1627	81
Total	12429	34419	36	4652	5538	84

IV. DADOS NÃO CORRELACIONADOS. 30 casos. 50 variáveis. $M = 0,5 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	1157	2501	46	380	409	93
Mult.	103	105	98	36	37	97
Comp.	1167	2090	59	447	450	99
Buscas	1453	2824	52	489	521	94
Atrib.	1439	2979	48	538	596	90
Total	5319	10499	50	1890	2013	94

V. DADOS FRAC. CORRELACIONADOS. 30 casos. 25 variáveis. $M = 0,8 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	1497	2385	63	1366	2050	67
Mult.	93	94	99	85	86	99
Comp.	1415	2027	70	1324	1786	74
Buscas	1823	2720	67	1674	2364	71
Atrib.	1864	2932	64	1729	2561	68
Total	6692	10158	66	6178	8847	70

VI. DADOS FRAC. CORRELACIONADOS. 30 casos. 50 variáveis. $M = 0,8 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	1591	4089	39	1170	2476	47
Mult.	117	118	99	98	97	101
Comp.	1551	3268	48	1237	2122	58
Buscas	1991	4507	45	1482	2807	53
Atrib.	1922	4684	41	1496	2968	50
Total	7172	16666	43	5483	10470	52

VII. DADOS FRAC. CORRELACIONADOS. 30 casos. 25 variáveis $M = 0,5 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	1089	1430	76	952	1170	81
Mult.	116	117	99	104	105	99
Comp.	1322	1511	87	1201	1294	93
Buscas	1413	1764	80	1246	1494	83
Atrib.	1535	1988	77	1382	1677	83
Total	5475	6810	80	4885	5740	85

VIII. DADOS FRAC. CORRELACIONADOS. 30 casos. 50 variáveis. $M = 0,5 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	1755	3160	56	1461	2314	63
Mult.	215	216	100	185	186	100
Comp.	2265	3124	73	1960	2437	80
Buscas	2358	3783	62	1984	2849	70
Atrib.	2514	4126	61	2158	3151	68
Total	9107	14409	63	7748	10937	71

IX. DADOS FORT. CORRELACIONADOS. 30 casos. 10 variáveis. $M = 0,8 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	767	901	85	750	877	86
Mult.	20	21	95	19	20	95
Comp.	682	780	87	670	762	88
Buscas	938	1081	87	919	1054	87
Atrib.	964	1176	82	947	1149	82
Total	3371	3959	85	3305	3862	86

X. DADOS FORT. CORRELACIONADOS. 30 casos. 20 variáveis. $M = 0,8 \Sigma p_i$

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	2547	3179	80	2485	3030	82
Mult.	74	75	99	71	72	99
Comp.	2324	2725	85	2284	2618	87
Buscas	3139	3789	83	3068	3629	85
Atrib.	3209	4026	80	3150	3862	82
Total	11293	13794	82	11058	13211	84

XI. DADOS FORT. CORRELACIONADOS. 30 casos. 10 variáveis. M = 0,5 Σp_i

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	615	657	94	611	652	94
Mult.	36	37	97	36	38	95
Comp.	699	720	97	695	715	97
Buscas	806	860	94	802	852	94
Atrib.	836	925	90	832	919	91
Comp.	2992	3199	93	2976	3176	94

XII. DADOS FORT. CORRELACIONADOS. 30 casos. 20 variáveis. M = 0,5 Σp_i

	Sem Redução			Com Redução		
	KPVILL	HSAN	%	KPVILL	HSAN	%
Adições	3536	3897	90	3511	3848	91
Mult.	327	328	100	325	326	100
Comp.	4793	4781	100	4773	4742	100
Buscas	4706	5086	93	4676	5030	93
Atrib.	5288	5776	92	5262	5718	92
Total.	18650	19868	94	18547	19664	94

3.6.3 - Análise dos resultados

Um estudo dos dados contidos nas tabelas apresentadas nos permite concluir o seguinte:

1. Quanto maior o número de variáveis, maior a eficiência de KPVILL sobre HSAN.
2. A eficiência relativa de KPVILL sobre HSAN é mais acentuada quando não se usa redução.
3. Quanto maior o tamanho relativo da mochila ($M = 0{,}8 \Sigma p_i$ ou $M = 0{,}5 \Sigma p_i$), maior a eficiência de KPVILL.
4. Quanto mais correlacionados são os dados, mais difíceis (o número de operações é maior) são os problemas, tanto para KPVILL quanto para HSAN.
5. Para problemas com dados fortemente correlacionados a aplicação da redução não traz resultados satisfatórios, isto é, quase não se reduz os problemas. Daí a aplicação ou não da redução dar resultados quase idênticos, quer se use HSAN ou KPVILL.

De uma maneira geral KPVILL se mostrou mais eficiente que HSAN quando se mede o número de operações (adições, multiplicações, etc) efetuadas por ambos. A justificativa para esses fatos constatados através da análise das tabelas será objeto de outro trabalho, possivelmente uma parte da tese de doutorado do autor.

3.7. CONCLUSÃO FINAL

Obtém-se, ao final deste trabalho, um algoritmo eficiente para resolver o PMOI como uma contribuição ao desenvolvimento do assunto.

Além disso, foi feito um esforço para se tentar mostrar dentro de um mesmo padrão de raciocínio, os algoritmos existentes.

Com a implementação do programa conversacional esperamos ter fornecido uma solução completa para o problema da Instalação de Postos de Atendimento e Venda de Insumos, fornecendo desde os detalhes da modelagem até a solução computacional que pode ser implementada em qualquer microcomputador.

APÊNDICE A

SOLUÇÃO ÓTIMA DE UM PMO1 RELAXADO

A.1 - DEFINIÇÃO DO PROBLEMA

Quer se determinar a solução ótima $\overline{x}^*$, de valor $\overline{z}^*$, do seguinte PMO1 relaxado:

$\boxed{\overline{P}}$ $$\text{Max} \sum_{j=1}^{n} v_j x_j \tag{A.1}$$

$$\text{sujeito a} \sum_{j=1}^{n} p_j x_j \leqslant M \tag{A.2}$$

$$0 \leqslant x_j \leqslant 1 \quad \forall\, j = 1, \ldots, n \tag{A.3}$$

Consideremos válidas as seguintes condições:

$$v_j \text{ e } p_j \text{ são inteiros e positivos} \tag{A.4}$$

$$p_j \leqslant M \qquad \forall\, j = 1, \ldots, n \tag{A.5}$$

$$\sum_{j=1}^{n} p_j > M \tag{A.6}$$

$$\frac{v_1}{p_1} \geqslant \frac{v_2}{p_2} \geqslant \ldots \geqslant \frac{v_n}{p_n} \tag{A.7}$$

Em 1957, Dantzig |06| foi o primeiro a determinar a solução ótima deste problema.

A.2 - DETERMINAÇÃO DA SOLUÇÃO ÓTIMA

A.2.1 - Lema A.1

Se $\hat{x}$ é solução ótima de $\overline{P}$ então

$$\sum_{j=1}^{n} p_j \hat{x}_j = M \qquad (A.8)$$

Prova

Admitindo, por absurdo, que $\sum_{j=1}^{n} p_j \hat{x}_j < M$ então existirá pelo menos uma variável x_k em $\hat{x}$ que poderá crescer tendo em vista a condição (A.6).

Mas se é possível fazer crescer o valor $\hat{x}_k$, será possível fazer crescer o valor $\hat{z}$ de $\hat{x}$, logo $\hat{x}$ não pode ser solução ótima de $\overline{P}$.

Portanto $\sum_{j=1}^{n} p_j \hat{x}_j \geqslant M$, mas em vista de (A.2), tem-se que $\sum_{j=1}^{n} p_j \hat{x}_j = M$.

A.2.2 - Teorema A.2

A solução $\hat{x}$ é ótima para o problema $\overline{P}$ se:

$$\hat{x}_j = 1 \qquad j = 1, 2, \ldots, k-1 \tag{A.9}$$

$$\hat{x}_k = (M - \sum_{j=1}^{k-1} p_j) / p_k \tag{A.10}$$

$$\hat{x}_j = 0 \qquad j = k+1, \ldots, n \tag{A.11}$$

tal que

$$\sum_{j=1}^{n} p_j \hat{x}_j = M \tag{A.12}$$

$$0 \leqslant \hat{x}_k \leqslant 1 \tag{A.13}$$

Prova

A solução $\hat{x}$ é viável de $\overline{P}$ pois (A.12) garante que (A.2) não é violada, e (A.9), (A.11) e (A.13) garantem que (A.3) não é violada.

Seja $\tilde{x}$ uma solução ótima de $\overline{P}$ cujo valor é $\tilde{z}$, tal premissa é válida pois o conjunto de soluções viáveis de $\overline{P}$ é fechado e limitado logo $\overline{P}$ admite pelo menos uma solução ótima.

A solução $\tilde{x}$ pode ser escrita como:

$$\tilde{x}_j = \hat{x}_j + \Delta_j \qquad \forall j = 1, \ldots, n \tag{A.14}$$

onde Δ_j é um número real qualquer.

Provaremos que $\tilde{z} \leqslant \hat{z}$, onde $\hat{z}$ é o valor de $\hat{x}$.

Tendo em vista (A.9), (A.10) e (A.11), a solução $\tilde{x}$ pode ser escrita como:

$$\tilde{x}_j = 1 + \Delta_j \qquad j = 1, 2, \ldots, k-1 \quad \text{com} \quad \Delta_j \leqslant 0 \tag{A.15}$$

$$\tilde{x}_k = \hat{x}_k + \Delta_k \qquad \text{com} \quad \Delta_k \quad \text{qualquer} \tag{A.16}$$

$$\tilde{x}_j = 0 + \Delta_j \qquad j = k, \ldots, n \quad \text{com} \quad \Delta_j \geqslant 0 \tag{A.17}$$

Como $\tilde{x}$ é uma solução ótima de $\overline{P}$ então pelo Lema A.1 devemos ter $\sum_{j=1}^{n} p_j \tilde{x}_j = M$.

$$\sum_{j=1}^{n} p_j \tilde{x}_j = M \Longrightarrow \sum_{j=1}^{n} p_j (\hat{x}_j + \Delta_j) = M \Longrightarrow \sum_{j=1}^{n} p_j \hat{x}_j + \sum_{j=1}^{n} p_j \Delta_j = M$$

mas tendo em vista (A.12) tem-se

$$\sum_{j=1}^{n} p_j \Delta_j = 0 \tag{A.18}$$

O valor $\tilde{z}$ de $\tilde{x}$ pode ser escrito como

$$\tilde{z} = \sum_{j=1}^{n} v_j \tilde{x}_j = \sum_{j=1}^{n} v_j (\hat{x}_j + \Delta_j) = \sum_{j=1}^{n} v_j \hat{x}_j + \sum_{j=1}^{n} v_j \Delta_j = \hat{z} + \sum_{j=1}^{n} v_j \Delta_j$$

$$\tilde{z} = \hat{z} + \sum_{j=1}^{n} v_j \Delta_j \tag{A.19}$$

Analisemos o sinal de $\sum_{j=1}^{n} v_j \Delta_j$:

$$\sum_{j=1}^{n} v_j \Delta_j = \sum_{j=1}^{k-1} v_j \Delta_j + v_k \Delta_k + \sum_{j=k+1}^{n} v_j \Delta_j$$

Façamos duas hipóteses: $\Delta_k \leqslant 0$ ou $\Delta_k \geqslant 0$

1ª hipótese: $\Delta_k \leqslant 0$, neste caso

$$\sum_{j=1}^{n} v_j \Delta_j = \sum_{j=1}^{k} v_j \frac{p_j}{p_j} \Delta_j + \sum_{j=k+1}^{n} v_j \frac{p_j}{p_j} \Delta_j$$

$$\sum_{j=1}^{n} v_j \Delta_j = \sum_{j=1}^{k} \frac{v_j}{p_j} p_j \Delta_j + \sum_{j=k+1}^{n} \frac{v_j}{p_j} p_j \Delta_j$$

Tendo em vista (A.4), (A.7), (A.15) e (A.17)

$$\sum_{j=1}^{n} v_j \Delta_j \leqslant \sum_{j=1}^{k} \frac{v_k}{p_k} p_j \Delta_j + \sum_{j=k+1}^{n} \frac{v_{k+1}}{p_{k+1}} p_j \Delta_j$$

$$\sum_{j=1}^{n} v_j \Delta_j \leqslant \frac{v_k}{p_k} \underbrace{\sum_{j=1}^{k} p_j \Delta_j}_{\leqslant 0} + \frac{v_{k+1}}{p_{k+1}} \underbrace{\sum_{j=k+1}^{n} p_j \Delta_j}_{\geqslant 0}$$

Como $\frac{v_k}{p_k} \geqslant \frac{v_{k+1}}{p_{k+1}}$ e tendo em vista (A.18) tem-se:

$$\sum_{j=1}^{n} v_j\Delta_j \leqslant \frac{v_k}{p_k} \sum_{j=1}^{k} p_j\Delta_j + \frac{v_k}{p_k} \sum_{j=k+1}^{n} p_j\Delta_j$$

$$\sum_{j=1}^{n} v_j\Delta_j \leqslant \frac{v_k}{p_k} \sum_{j=1}^{n} p_j\Delta_j = 0$$

Logo $\sum_{j=1}^{n} v_j\Delta_j \leqslant 0$.

2ª hipótese: $\Delta_k \geqslant 0$, neste caso:

$$\sum_{j=1}^{n} v_j\Delta_j = \sum_{j=1}^{k-1} v_j \frac{p_j}{p_j} \Delta_j + \sum_{j=k}^{n} v_j \frac{p_j}{p_j} \Delta_j$$

$$\sum_{j=1}^{n} v_j\Delta_j = \sum_{j=1}^{k-1} \frac{v_j}{p_j} p_j\Delta_j + \sum_{j=k}^{n} \frac{v_j}{p_j} p_j\Delta_j$$

Tendo em vista (A.4), (A.7), (A.15) e (A.17)

$$\sum_{j=1}^{n} v_j\Delta_j \leqslant \sum_{j=1}^{k-1} \frac{v_{k-1}}{p_{k-1}} p_j\Delta_j + \sum_{j=k}^{n} \frac{v_k}{p_k} p_j\Delta_j$$

$$\sum_{j=1}^{n} v_j\Delta_j \leqslant \frac{v_{k-1}}{p_{k-1}} \underbrace{\sum_{j=1}^{k-1} p_j\Delta_j}_{\leqslant 0} + \frac{v_k}{p_k} \underbrace{\sum_{j=k}^{n} p_j\Delta_j}_{\geqslant 0}$$

Como $\frac{v_{k-1}}{p_{k-1}} \geqslant \frac{v_k}{p_k}$ e tendo em vista (A.18) tem-se:

$$\sum_{j=1}^{n} v_j\Delta_j \leqslant \frac{v_{k-1}}{p_{k-1}} \sum_{j=1}^{k=1} p_j\Delta_j + \frac{v_{k-1}}{p_{k-1}} \sum_{j=k}^{n} p_j\Delta_j$$

$$\sum_{j=1}^{n} v_j\Delta_j \leqslant \frac{v_{k-1}}{p_{k-1}} \sum_{j=1}^{n} p_j\Delta_j = 0$$

Logo $$\sum_{j=1}^{n} v_j\Delta_j \leqslant 0 .$$

Portanto em qualquer hipótese

$$\sum_{j=1}^{n} v_j\Delta_j \leqslant 0 \qquad (A.20)$$

E por conseguinte, tendo em vista (A.19), tem-se

$$\tilde{z} \leqslant \hat{z}$$

Mas como, por hipótese, $\tilde{z}$ é solução ótima, devemos ter:

$$\hat{z} \leqslant \tilde{z}$$

E assim

$$\hat{z} = \tilde{z}$$

Logo $\hat{z}$ é solução ótima de $\overline{P}$.

A.2.3 - Solução ótima de $\overline{P}$

Pelo teorema A.2 temos portanto que a solução ótima $\overline{x}^*$ do

problema $\bar{P}$ é dada por:

$$\bar{x}_j^* = 1 \qquad j = 1, 2, \ldots, k-1 \tag{A.21}$$

$$\bar{x}_k^* = (M - \sum_{j=1}^{k-1} p_j) / p_k \tag{A.22}$$

$$\bar{x}_j^* = 0 \qquad j = k+1, \ldots, n \tag{A.23}$$

tal que $\sum_{j=1}^{n} p_j \bar{x}_j^* = M$ e $0 \leqslant \bar{x}_k^* \leqslant 1$, e cujo valor ótimo $\bar{z}^*$ é dado por:

$$\bar{z}^* = \sum_{j=1}^{k-1} v_j + \frac{v_k}{p_k} (M - \sum_{j=1}^{k-1} p_j) \tag{A.24}$$

A.3 - APLICAÇÃO NUMÉRICA

Resolver o seguinte PMO1 relaxado:

$\boxed{\bar{P}}$ Max $15x_1 + 14x_2 + 14x_3 + 18x_4 + 17x_5 + 12x_6$

sujeito a $12x_1 + 14x_2 + 15x_3 + 24x_4 + 24x_5 + 17x_6 \leqslant 60$

$0 \leqslant x_j \leqslant 1 \qquad \forall j = 1, \ldots, 6$

Solução:

$\bar{x}_1^* = \bar{x}_2^* = \bar{x}_3^* = 1$

$\bar{x}_4^* = [60 - (12 + 14 + 15)]/24 = 19/24$

$$\bar{x}_5^* = \bar{x}_6^* = 0$$

$$\bar{x}^* = (1, 1, 1, 19/24, 0, 0)^T$$

$$\bar{z}^* = 57 + \frac{1}{4}$$

APÊNDICE B

ALGUNS LIMITANTES SUPERIORES DO VALOR ÓTIMO DE UM PMO1

B.1 - INTRODUÇÃO

Suponha o seguinte PMO1:

P

$$\text{Max } z = \sum_{i=1}^{n} v_i x_i \tag{B.1}$$

$$\text{sujeito a } \sum_{i=1}^{n} p_i x_i \leqslant M \tag{B.2}$$

$$0 \leqslant x_i \leqslant 1 \quad i = 1, \ldots, n \tag{B.3}$$

$$x_i \text{ inteiro} \quad i = 1, \ldots, n \tag{B.4}$$

onde v_i e p_i são inteiros positivos.

Consideraremos que as variáveis do problema P estão ordenadas segundo a ordem não-crescente da razão v_i/p_i, isto é:

$$\frac{v_1}{p_1} \geqslant \frac{v_2}{p_2} \geqslant \ldots \geqslant \frac{v_n}{p_n} \tag{B.5}$$

Para simplificar as definições dos limitantes superiores, consideraremos um objeto "fantasma" $n+1$ tal que $v_{n+1} = 0$ e $p_{n+1} = M$.

Chamemos de x^* a solução ótima de P e de z^* o valor dessa solução.

Denotaremos por $\overline{P}$ ao problema P relaxado, isto é, sem as res

trições de integralidade (B.4), por $\overline{x}^*$ a solução ótima de $\overline{P}$ e por $\overline{z}^*$ o seu valor.

Queremos determinar limitantes superiores, que denotaremos por z^u, de z^*, isto é, z^u deve ser tal que $z^* \leq z^u$. Tais limitantes superiores são de fundamental importância nos algoritmos "branch and bound" pois serão usados num dos possíveis critérios de poda da árvore gerada nesses algoritmos.

Constata-se que quanto melhor o limitante superior calculado, isto é, quanto menor o valor de z^u, maior será o esforço computacional para calculá-lo.

Veremos a seguir alguns limitantes superiores para o valor ótimo de um PM01. Destes, o mais usado nos algoritmos "branch and bound" é o de Dantzig que nada mais é que o maior inteiro menor que o valor ótimo do PM01 relaxado, cujo cálculo é visto no Apêndice A e foi apresentado pela primeira vez por Dantzig [06].

B.2 - LIMITANTE SUPERIOR DE DANTZIG (z^{u_D})

O relaxamento das condições de integralidade (B.4) sobre as variáveis de P nos conduz a um Problema de Programação Linear com variáveis limitadas, denotado por $\overline{P}$ ou problema P relaxado.

Segundo a teoria de Programação Linear, o valor ótimo ($\bar{z}^*$) de $\bar{P}$ é um limitante superior do valor ótimo (z^*) de P, isto porque todas as soluções viáveis de P são também de $\bar{P}$ (a recíproca não é verdadeira) e a função objetivo de P é a mesma de $\bar{P}$. Tendo em vista que os valores v_i são inteiros, então z^* também é inteiro e portanto podemos tomar o maior inteiro menor que $\bar{z}^*$, isto é, $\lfloor \bar{z}^* \rfloor$ como limitante superior de z^*.

O cálculo do valor ótimo $\bar{z}^*$ de $\bar{P}$ foi apresentado pela primeira vez por Dantzig |06| (veja Apêndice A) e desta forma o limitante superior (z^{u_D}) a ele associado é creditado a Dantzig e é dado por:

$$z^{u_D} = \sum_{i=1}^{k-1} v_i + \left\lfloor (M - \sum_{i=1}^{k-1} p_i)\, v_k / p_k \right\rfloor \qquad (B.6)$$

onde k é o maior índice tal que $\sum_{i=1}^{k-1} p_i \leq M$, e $\lfloor \alpha \rfloor$ significa o "maior inteiro menor que α".

B.3. LIMITANTE SUPERIOR DE MARTELLO-TOTH ($z^{u_{MT}}$)

Em 1977, Martello-Toth |22| apresentaram uma melhoria no limitante superior de Dantzig.

Seja k o índice calculado no limitante superior de Dantzig. A solução ótima x^* de P terá $x_k^* = 0$ ou $x_k^* = 1$.

A solução ótima de P com x_k fixado em 0 terá B_1 como li-

mitante superior de seu valor ótimo:

$$B_1 = \sum_{i=1}^{k-1} v_i + \left\lfloor (M - \sum_{i=1}^{k-1} p_i)\, v_{k+1}/p_{k+1} \right\rfloor \qquad (B.7)$$

que corresponde em alocar para a variável x_{k+1} todo o peso não alocado em x_k.

A solução ótima de P com x_k fixado em 1 terá B_2 como limitante superior de seu valor ótimo:

$$B_2 = \sum_{i=1}^{k-1} v_i + \left\lfloor v_k - (\sum_{i=1}^{k} p_i - M)\, v_{k-1}/p_{k-1} \right\rfloor \qquad (B.8)$$

que corresponde em "desalocar" de x_{k-1} o correspondente excesso de peso que foi alocado em x_k.

O limitante superior ($z^{u_{MT}}$) de Martello-Toth será então definido por:

$$z^{u_{MT}} = \max \{B_1, B_2\} \qquad (B.9)$$

B.3.1 - Teorema B_1

O limitante superior ($z^{u_{MT}}$) de Martello-Toth é melhor que o limitante superior (z^{u_D}) de Dantzig, isto é, $z^{u_{MT}} \leq z^{u_D}$.

Prova:

Provaremos que $B_1 \leq z^{u_D}$ e $B_2 \leq z^{u_D}$, pois sendo $z^{u_{MT}} = \max \{B_1, B_2\}$, teremos provado que $z^{u_{MT}} \leq z^{u_D}$.

1) Prova de que $B_1 \leq z^{u_D}$:

Como $\frac{v_{k+1}}{p_{k+1}} \leq \frac{v_k}{p_k}$ tem-se

$$\sum_{i=1}^{k-1} v_i + (M - \sum_{i=1}^{k-1} p_i) \frac{v_{k+1}}{p_{k+1}} \leq \sum_{i=1}^{k-1} v_i + (M - \sum_{i=1}^{k-1} p_i) \frac{v_k}{p_k}$$

Logo $B_1 \leq z^{u_D}$.

2) Prova de que $B_2 \leq z^{u_D}$:

$$\sum_{i=1}^{k-1} v_i + (M - \sum_{i=1}^{k-1} p_i) \frac{v_k}{p_k} =$$

$$= \sum_{i=1}^{k-1} v_i + v_k + (M - \sum_{i=1}^{k-1} p_i) \frac{v_k}{p_k} - p_k \frac{v_k}{p_k} =$$

$$= \sum_{i=1}^{k-1} v_i + v_k + \underbrace{(M - \sum_{i=1}^{k} p_i)}_{\leq 0} \frac{v_k}{p_k} \geq$$

$$\geq \sum_{i=1}^{k-1} v_i + v_k - (\sum_{i=1}^{k} p_i - M) \frac{v_{k-1}}{p_{k-1}}$$

Logo $B_2 \leq z^{u_D}$.

B.4 - LIMITANTE SUPERIOR DE HUDSON (z^{u_H})

Também em 1977, Hudson (veja em [24]) propõe uma melhoria para o limitante superior de Martello-Toth.

Hudson propõe que seja melhorado o limitante B_2, isto é, o limitante superior do valor ótimo do problema P com x_k fixado em 1. Denominaremos por B_3 esta melhoria de B_2.

Hudson estabelece que:

$$B_3 = \sum_{i=1}^{k'-1} v_i + v_k + \left\lfloor (M - p_k - \sum_{i=1}^{k'-1} p_i)\, v_{k'}/p_{k'} \right\rfloor \qquad (B.10)$$

onde k' é o maior índice tal que $\sum_{i=1}^{k'-1} p_i \leq M - p_k$.

Para uma comparação entre B_2 e B_3 podemos escrever:

$$B_2 = \sum_{i=1}^{k'-1} v_i + \sum_{i=k'}^{k-1} v_i + v_k - \left\lfloor (\sum_{i=1}^{k} p_i - M)\, v_{k-1}/p_{k-1} \right\rfloor$$

$$B_3 = \sum_{i=1}^{k'-1} v_i + v_k + \left\lfloor (M - p_k - \sum_{i=1}^{k'-1} p_i)\, v_{k'}/p_{k'} \right\rfloor$$

A diferença fundamental entre B_2 e B_3 é que em B_2 consideramos alocados os objetos k', $k'+1$, ..., $k-1$ ocasionando um excesso de peso da mochila. Este excesso é corrigido subtraindo a parcela $\left\lfloor (\sum_{i=1}^{k} p_i - M)\, v_{k-1}/p_{k-1} \right\rfloor$ onde está implícito que todo o excesso é atribui

do ao objeto de índice (k-1). Mas este é aquele dentre os (k-1) primeiros objetos, o que possui a menor relação valor/peso. Assim atribuindo todo o excesso ao objeto (k-1) estamos na verdade corrigindo, isto é, subtraindo pouco, o que gera um limitante superior maior que o necessário.

Já em B_3 a idéia é diferente. Como $x_k = 1$, resta na mochila um peso $M - p_k$. Para esta capacidade restante da mochila é aplicado o limitante superior de Dantzig, que equivale a alocar na mesma os objetos 1, 2, ..., k' até preenchê-la completamente. Vê-se que em B_3 não é alocado excesso de peso o que não gera a necessidade de correções especiais.

Desta forma o limitante superior (z^{u_H}) de Hudson é definido por:

$$z^{u_H} = \max \{B_1, B_3\} \qquad (B.11)$$

B.4.1 - Teorema B_2

O limitante superior (z^{u_H}) de Hudson é melhor que o limitante superior ($z^{u_{MT}}$) de Martello-Toth, isto é, $z^{u_H} \leqslant z^{u_{MT}}$.

Prova:

Como $z^{u_H} = \max \{B_1, B_3\}$ e $z^{u_{MT}} = \max \{B_1, B_2\}$, provaremos que $B_3 \leqslant B_2$ e assim teremos provado que $z^{u_H} \leqslant z^{u_{MT}}$.

Prova de que $B_3 < B_2$:

Por definição k é o maior inteiro tal que: $\sum_{i=1}^{k-1} p_i \leqslant M$.

Por absurdo, suponhamos que $k' \geqslant k$. Tem-se

$$\sum_{i=1}^{k-1} p_i \leqslant \sum_{i=1}^{k'-1} p_i \leqslant M - p_k \implies \sum_{i=1}^{k} p_i \leqslant M$$

o que contradiz a definição de k .

Faremos duas hipóteses:

1ª hipótese: $k' = k-1$

$$\sum_{i=1}^{k'-1} v_i + v_k + (M - p_k - \sum_{i=1}^{k'-1} p_i) \frac{v_{k'}}{p_{k'}} =$$

$$= \sum_{i=1}^{k-2} v_i + v_k + (M - p_k - \sum_{i=1}^{k-2} p_i) \frac{v_{k-1}}{p_{k-1}} =$$

$$= \sum_{i=1}^{k-2} v_i + v_{k-1} + v_k + (M - p_k - \sum_{i=1}^{k-2} p_i - p_{k-1}) \frac{v_{k-1}}{p_{k-1}} =$$

$$= \sum_{i=1}^{k} v_i + (M - \sum_{i=1}^{k} p_i) \frac{v_{k-1}}{p_{k-1}}$$

Logo $B_3 = B_2$.

2ª hipótese: $k' \leq k-2$

$$\sum_{i=1}^{k'-1} v_i + v_k + \underbrace{(M - p_k - \sum_{i=1}^{k'-1} p_i)}_{\geq 0} \frac{v_{k'}}{p_{k'}} =$$

$$= \sum_{i=1}^{k'-1} v_i + v_k + v_{k'} + \underbrace{(M - p_k - \sum_{i=1}^{k'-1} p_i - p_{k'})}_{\leq 0} \frac{v_{k'}}{p_{k'}} =$$

$$= \sum_{i=1}^{k'-1} v_i + v_k + v_{k'} + \underbrace{(M - p_k - \sum_{i=1}^{k'} p_i)}_{\leq 0} \frac{v_{k'}}{p_{k'}} =$$

$$= \sum_{i=1}^{k'} v_i + v_k - (\sum_{i=1}^{k'} p_i + p_k - M) \frac{v_{k'}}{p_{k'}} =$$

$$= \sum_{i=1}^{k-1} v_i + v_k - \sum_{i=k'+1}^{k-1} v_i - (\sum_{i=1}^{k} p_i - M - \sum_{i=k'+1}^{k-1} p_i) \frac{v_{k'}}{p_{k'}} =$$

$$= \sum_{i=1}^{k-1} v_i + v_k - \sum_{i=k'+1}^{k-1} p_i v_i / p_i - (\sum_{i=1}^{k} p_i - M - \sum_{i=k'+1}^{k-1} p_i) \frac{v_{k'}}{p_{k'}} \leq$$

$$\leq \sum_{i=1}^{k-1} v_i + v_k - \sum_{i=k'+1}^{k-1} p_i \frac{v_{k-1}}{p_{k-1}} - (\sum_{i=1}^{k} p_i - M - \sum_{i=k'+1}^{k-1} p_i) \frac{v_{k-1}}{p_{k-1}} =$$

$$= \sum_{i=1}^{k-1} v_i + v_k - (\sum_{i=1}^{k} p_i - M - \sum_{i=k'+1}^{k-1} p_i + \sum_{i=k'+1}^{k-1} p_i) \frac{v_{k-1}}{p_{k-1}} =$$

$$= \sum_{i=1}^{k-1} v_i + v_k - (\sum_{i=1}^{k} p_i - M) \frac{v_{k-1}}{p_{k-1}}$$

Logo $B_3 \leq B_2$

Em qualquer hipótese: $B_3 \leq B_2$.

B.5 - UMA MELHORIA NO LIMITANTE SUPERIOR DE HUDSON ($z^{u_{HM}}$)

Desenvolvemos a seguir uma melhoria do limitante superior B_1, isto é, do valor ótimo de P com x_k fixado em 0. Isto é, propomos um limitante superior B_4 melhor que B_1. Ao limitante superior

$$z^{u_{HM}} = \max \{B_4, B_3\} \tag{B.12}$$

chamaremos de "limitante superior de Hudson melhorado".

A idéia para se calcular B_4 é a mesma de Hudson quando calculou B_3, isto é, ao invés de alocar o peso não alocado em x_k todo em x_{k+1}, alocaremos em $x_{k+1}, x_{k+2}, \ldots, x_n$ respeitando os limites de peso correspondente a cada variável.

Desta forma B_4 será o limitante superior de Dantzig para o problema P com x_k fixado em 0.

$$B_4 = \sum_{i=1}^{k-1} v_i + \sum_{i=k+1}^{k''-1} v_i + \left\lfloor \left(M - \sum_{i=1}^{k-1} p_i - \sum_{i=k+1}^{k''-1} p_i\right) v_{k''}/p_{k''} \right\rfloor \tag{B.13}$$

onde k'' é o menor índice tal que $\sum_{i=k+1}^{k''} p_i \geq M - \sum_{i=1}^{k-1} p_i$. Suporemos que que se $k'' = k+1$ então $\sum_{i=k+1}^{k''-1} v_i = \sum_{i=k+1}^{k''-1} p_i = 0$.

B.5.1 - Teorema B_3

O limitante superior $z^{u_{HM}}$ é melhor que o limitante superior (z^{u_H}) de Hudson, isto é, $z^{u_{HM}} \leqslant z^{u_H}$.

Prova:

Como $z^{u_{HM}} = \max \{B_4, B_3\}$ e $z^{u_H} = \{B_1, B_3\}$, provaremos que $B_4 \leqslant B_1$ e assim teremos provado que $z^{u_{HM}} \leqslant z^{u_H}$.

Se $k'' = k + 1$ então $B_4 = B_1$.

. Prova de que $B_4 \leqslant B_1$, se $k'' > k + 1$:

$$\sum_{i=1}^{k-1} v_i + \sum_{i=k+1}^{k''-1} v_i + (M - \sum_{i=1}^{k-1} p_i - \sum_{i=k+1}^{k''-1} p_i) \frac{v_{k'}}{p_{k'}} =$$

$$= \sum_{i=1}^{k-1} v_i + \sum_{i=k+1}^{k''-1} p_i \frac{v_i}{p_i} + (M - \sum_{i=1}^{k-1} p_i - \sum_{i=k+1}^{k''-1} p_i) \frac{v_{k'}}{p_{k'}} \leqslant$$

$$\leqslant \sum_{i=1}^{k-1} v_i + \sum_{i=k+1}^{k''-1} p_i \frac{v_{k+1}}{p_{k+1}} + (M - \sum_{i=1}^{k-1} p_i - \sum_{i=k+1}^{k''-1} p_i) \frac{v_{k+1}}{p_{k+1}} =$$

$$= \sum_{i=1}^{k-1} v_i + (M - \sum_{i=1}^{k-1} p_i - \sum_{i=k+1}^{k''-1} p_i + \sum_{i=k+1}^{k''-1} p_i) \frac{v_{k+1}}{p_{k+1}} =$$

$$= \sum_{i=1}^{k-1} v_i + (M - \sum_{i=1}^{k-1} p_i) \frac{v_{k+1}}{p_{k+1}} = \sum_{i=1}^{k-1} v_i - (\sum_{i=1}^{k-1} p_i - M) \frac{v_{k+1}}{p_{k+1}}$$

Logo $B_4 \leqslant B_1$.

B.6 - APLICAÇÃO NUMÉRICA

Seja o PMO1:

$$\text{Max } z = 15x_1 + 14x_2 + 14x_3 + 18x_4 + 9x_5 + 3x_6 \qquad (B.14)$$

$$\text{sujeito a } 12x_1 + 14x_2 + 15x_3 + 24x_4 + 12x_5 + 6x_6 \leqslant 60 \qquad (B.15)$$

$$0 \leqslant x_i \leqslant 1 \qquad i = 1, \ldots, 6 \qquad (B.16)$$

$$x_i \text{ inteiro} \qquad i = 1, \ldots, 6 \qquad (B.17)$$

Determine os limitantes superiores z^{u_D}, $z^{u_{MT}}$, z^{u_H} e $z^{u_{HM}}$ para este problema.

Solução:

$$z^{u_D} = 15 + 14 + 14 + \lfloor 57/4 \rfloor = 57$$

$$z^{u_{MT}} = \max \{B_1, B_2\} = 57$$

$$B_1 = 15 + 14 + 14 + \lfloor 57/4 \rfloor = 57$$

$$B_2 = 15 + 14 + 14 + 18 - \lfloor 14/3 \rfloor = 56$$

$$z^{u_H} = \max \{B_1, B_3\} = 57$$

$$B_3 = 15 + 14 + 18 + \lfloor 28/3 \rfloor = 56$$

$$z^{u_{HM}} = \max \{B_4, B_3\} = 56$$

$$B_4 = 15 + 14 + 14 + 9 + 3 + \lfloor 0 \rfloor = 55$$

Portanto: $z^{u_{HM}} \leqslant z^{u_H} \leqslant z^{u_{MT}} \leqslant z^{u_D}$.

APÊNDICE C

LISTAGEM DO PROGRAMA (EM BASIC) PARA SE DETERMINAR OS POSTOS A SEREM INSTALADOS

C.1 - INTRODUÇÃO

O programa que apresentaamos a seguir foi escrito em Basic compatível com a maioria das versões de Basic disponíveis em microcomputadores nacionais. Em nenhum momento se apelou para comandos ou funções específicas de alguma versão mais requintada.

O programa está em linguagem conversacional e a razão de implementá-lo em Basic foi torná-lo acessível aos usuários que não disponham de sofisticados recursos computacionais.

O programa consiste de 3 partes principais: Uma rotina de ordenação (Shellsort), uma rotina de redução (baseada em Ingargiola-Korsh) e a implementação do algoritmo KPVILL.

Na implementação de KPVILL convêm frisar que foram feitas algumas melhorias no fluxo em relação ao algoritmo apresentado na seção 3.5.

C.2 - LISTAGEM DO PROGRAMA

```
5000 REM ***********************************************************
5010 REM *                                                         *
5020 REM *    INSTALACAO DE POSTOS DE ATENDIMENTO E VENDA DE       *
5030 REM *       DE INSUMOS PARA UMA COOPERATIVA AGRICOLA          *
5040 REM *                          <<<>>>                         *
5050 REM *       UMA APLICACAO DO PROBLEMA DA MOCHILA 0-1          *
5060 REM *                                                         *
5070 REM * COPPE-UFRJ-1983                        PAULO VILLELA    *
5080 REM ***********************************************************
5090 DIM IV(100),IP(100),V(100),P(100)
5100 DIM RAZ(101),O(100)
5110 DIM X(100),XL(100),PR(100),VR(100)
5120 DIM POSTO(100,10), NOMP$(50)
5130 SP=15 'SP=CONTROLE DE ESPACAMENTO
5140 FOR J=1 TO 24: PRINT: NEXT J
5150 PRINT TAB(SP) "*********************************"
5160 PRINT TAB(SP) "*                               *"
5170 PRINT TAB(SP) "*          INSTALACAO           *"
5180 PRINT TAB(SP) "* DE POSTOS DE ATENDIMENTO E *"
5190 PRINT TAB(SP) "*       VENDA DE INSUMOS        *"
5200 PRINT TAB(SP) "*         COTREFAL-PR           *"
5210 PRINT TAB(SP) "*             1983              *"
5220 PRINT TAB(SP) "*********************************"
5230 PRINT TAB(SP) "COPPE-UFRJ.PAULO VILLELA"
5240 PRINT
5250 PRINT "TECLE:"
5260 PRINT "      * PARA TERMINAR:"
5270 PRINT "              <CR>
5280 PRINT "      * PARA CONTINUAR:"
5290 PRINT "              <NUMERO DE POSTOS><CR>
5300 INPUT NT
5310 IF NT=0 THEN END
5320 N=NT: MT=0: RAZ(N+1)=0: NS=1
5330 PRINT: PRINT
5340 PRINT "ENTRE COM OS DADOS PARA CADA POSTO:"
5350 PRINT "   <NOME DA REGIAO>,<VALOR>,<CUSTO><CR>
5360 FOR J=1 TO N
5370     PRINT J;
5380     INPUT NOMP$(J),IV(J),IP(J)
5390     RAZ(J)=IV(J)/IP(J): O(J)=J
5400     XL(J)=0: X(J)=0: PR(J)=0: VR(J)=0
5410     NEXT J
5420 PRINT
5430 PRINT "ENTRE COM O CUSTO TOTAL:"
5440 INPUT "   <CUSTO TOTAL><CR>"; MT
```

```
6000 REM ***********************************************************
6010 REM                          ORDENACAO
6020 REM ***********************************************************
6030 I=1
6040 IF I<=N THEN I=I+1: GOTO 6040
6050 M=I-1
6060 M=INT(M/2): IF M<=0 GOTO 6170
6070 K=N-M
6080 FOR J=1 TO K: I=J+M
6090      I=I-M: IF I<=0 GOTO 6150
6100      L=I+M
6110      IL=O(L): II=O(I)
6120      IF RAZ(IL)<=RAZ(II) GOTO 6150
6130      T=O(I): O(I)=O(L): O(L)=T
6140      GOTO 6090
6150      NEXT J
6160 GOTO 6060
6170 FOR J=1 TO N
6180      T=O(J)
6190      V(J)=IV(T): P(J)=IP(T)
6200      RAZ(J)=V(J)/P(J)
6210      NEXT J
6220 REM
6230 REM
7000 REM ***********************************************************
7010 REM                          REDUCAO
7020 REM ***********************************************************
7030 K=1: NR=0: Z=0: M=MT
7040 IF P(K)>M GOTO 7070
7050 M=M-P(K): Z=Z+V(K): K=K+1
7060 IF K<=N GOTO 7040
7070 UD=Z+INT(M*RAZ(K))
7080 REF=K: ZL=Z: PES=M
7090 IF P(K)>PES GOTO 7110
7100 PES=PES-P(K): ZL=ZL+V(K)
7110 K=K+1: IF K<=N GOTO 7090
7120 REM
7130 REM ------------------------------------------------------------
7140 REM     VARIAVEIS X(1),....,X(REF)
7150 REM ------------------------------------------------------------
7160 J=REF
7170      PES=M: ZR=Z: I=REF+1: GOTO 7190
7180      PES=M+P(J): ZR=Z-V(J): I=REF
7190      IF I>N THEN ZU=ZR: GOTO 7240
7200      IF P(I)>PES GOTO 7230
7210      PES=PES-P(I): ZR=ZR+V(I)
7220      I=I+1: GOTO 7190
7230      ZU=ZR+RAZ(I)*PES
7240      IF ZU>=ZL GOTO 7260
7250      XL(J)=3: NR=NR+1: GOTO 7270
7260      IF ZR>ZL THEN ZL=ZR
7270      J=J-1
7280      IF J>0 GOTO 7180
7290 REM
```

```
7300 REM ------------------------------------------------------
7310 REM     VARIAVEIS X(REF),....,X(N)
7320 REM ------------------------------------------------------
7330 M=M-P(REF): Z=Z+V(REF)
7340 J=REF
7350      PES=M: ZR=Z: I=REF-1: GOTO 7370
7360      PES=M-P(J): ZR=Z+V(J): I=REF
7370      IF I=0 THEN ZU=ZR: GOTO 7420
7380      IF P(I)>=PES GOTO 7410
7390      PES=PES+P(I): ZR=ZR-V(I)
7400      I=I-1: GOTO 7370
7410      ZU=ZR+RAZ(I)*PES
7420      IF ZU>=ZL GOTO 7440
7430      XL(J)=2: NR=NR+1: GOTO 7450
7440      ZR=ZR-V(I): IF ZR>ZL THEN ZL=ZR
7450      J=J+1
7460      IF J<=N GOTO 7360
7470 REM
7480 REM ------------------------------------------------------
7490 REM          PROBLEMA REDUZIDO
7500 REM ------------------------------------------------------
7510 I=0: ZIN=0: N=NT
7520 FOR J=1 TO N
7530      IF XL(J)=0 GOTO 7590
7540      A=XL(J)-2
7550      T=O(J)
7560      FOR JJ=1 TO 10: POSTO(T,JJ)=A: NEXT JJ
7570      MT=MT-P(J)*A: ZIN=ZIN+V(J)*A
7580      GOTO 7620
7590      I=I+1
7600      V(I)=V(J): P(I)=P(J)
7610      RAZ(I)=RAZ(J): O(I)=O(J)
7620      NEXT J
7630 N=NT-NR
7640 REM
7650 REM TODAS AS VARIAVEIS FORAM
7660 REM REDUZIDAS?
7670 IF N=0 GOTO 9040
7680 REM
7690 REM
8000 REM ************************************************************
8010 REM                    SOLUCAO DO PM01
8020 REM ************************************************************
8030 REM
8040 REM ------------------------------------------------------
8050 REM              PASSO 1
8060 REM         FORMACAO DOS BLOCOS
8070 REM ------------------------------------------------------
8080 PES=MT: Z=ZIN
8090 FOR J=1 TO N
8100      IF P(J)>PES GOTO 8130
8110      PES=PES-P(J)
8120      NEXT J
```

```
8130 REF=J
8140 IF REF<=1 GOTO 8260
8150 VR(REF)=0: PR(REF)=0
8160 FOR J=REF-1 TO 1 STEP -1
8170      VR(J)=VR(J+1)+V(J)
8180      PR(J)=PR(J+1)+P(J)
8190      NEXT J
8200 REM
8210 REM ------------------------------------------------------
8220 REM                 PASSO 2
8230 REM             INICIALIZACAO
8240 REM ------------------------------------------------------
8250 REM ZL E' INICIALIZADA NA REDUCAO
8260 K=1: Z=ZIN: M=MT: NS=0
8270 FOR J=1 TO N: XL(J)=0: NEXT J
8280 GOTO 8500
8290 REM
8300 REM ------------------------------------------------------
8310 REM                 PASSO 3
8320 REM          TESTE DE FINALIZACAO
8330 REM ------------------------------------------------------
8340 K=K-1: IF K<=0 GOTO 9040
8350 IF X(K)=0 GOTO 8340
8360 REM
8370 REM ------------------------------------------------------
8380 REM                 PASSO 4
8390 REM           PROXIMO PROBLEMA
8400 REM ------------------------------------------------------
8410 M=M+P(K): Z=Z-V(K): X(K)=0: K=K+1
8420 REM
8430 REM ------------------------------------------------------
8440 REM                 PASSO 5
8450 REM    O PROBLEMA DEVE SER PODADO?
8460 REM ------------------------------------------------------
8470 REM (5.1) CALCULO DO LIM.SUPER. ZU
8480 REM
8490 REM COLOCACAO POR BLOCO:
8500 IF K>=REF GOTO 8550
8510 PES=M-PR(K): ZU=Z+VR(K)
8520 Q=REF: GOTO 8560
8530 REM
8540 REM COLOCACAO UM A UM:
8550 Q=K: ZU=Z: PES=M
8560 FOR J=Q TO N
8570      IF P(J)>PES GOTO 8610
8580      PES=PES-P(J): ZU=ZU+V(J)
8590      NEXT J
8600 GOTO 8650
8610 ZU=ZU+RAZ(J)*PES
8620 REM
8630 REM (5.2) TESTE DE NAO OTIMALIDADE
8640 REM
8650 IF ZU<ZL GOTO 8320
8660 REM
```

```
8670 REM ------------------------------------------------------
8680 REM                  PASSO 6
8690 REM          EXPANSAO DO PROBLEMA
8700 REM ------------------------------------------------------
8710 REM (6.1) PROBLEMAS AUXILIARES
8720 REM
8730 FOR J=K TO N
8740      IF P(J)>M THEN K=J: GOTO 8810
8750      M=M-P(J): Z=Z+V(J): X(J)=1
8760      NEXT J
8770 GOTO 8890
8780 REM
8790 REM (6.2) PROXIMO PROBLEMA
8800 REM
8810 X(K)=0: K=K+1
8820 IF K<N GOTO 8500
8830 IF K=N GOTO 8730
8840 REM
8850 REM ------------------------------------------------------
8860 REM                  PASSO 7
8870 REM ATUALIZACAO DA MELHOR SOLUCAO
8880 REM ------------------------------------------------------
8890 IF Z<ZL GOTO 8980
8900 IF Z=ZL THEN NS=NS+1: GOTO 8920
8910 NS=1: ZL=Z
8920 FOR J=1 TO N
8930      T=0(J): POSTO(T,NS)=X(J)
8940      NEXT J
8950 REM
8960 REM CASO EM QUE X(N)=1
8970 REM
8980 IF X(N)=1 THEN M=M+P(N): Z=Z-V(N)
8990 K=N: GOTO 8340
8991 REM
8992 REM
9000 REM ************************************************************************
9010 REM                  IMPRESSAO DA SOLUCAO OTIMA
9020 REM ************************************************************************
9030 REM
9040 PRINT: PRINT
9050 PRINT TAB(SP+7) "SOLUCAO OTIMA"
9060 PRINT TAB(SP) "-------------------------------------------------"
9070 PRINT TAB(SP) "REGIAO" TAB(SP+15) "DECISAO"
9080 PRINT TAB(SP) "-------------------------------------------------"
9090 FOR J=1 TO NT
9100      PRINT TAB(SP) NOMP$(J) TAB(SP+15);
9110      FOR JJ=1 TO NS
9120           IF POSTO(J,JJ)=0 GOTO 9140
9130           PRINT "S ";: GOTO 9150
9140           PRINT "N ";
9150           NEXT JJ: PRINT
9160      NEXT J
9170 PRINT TAB(SP) "-------------------------------------------------"
9180 PRINT TAB(SP) "VALOR OTIMO:" ZL
9190 STOP: GOTO 5140
```

BIBLIOGRAFIA

01. AHRENS, J. H. & FINKE, G. - Merging and Sorting Applied to the Zero-One Knapsack Problem. Operations Research, 23 : 1099-1109, 1975.

02. BALAS, E. & ZEMEL, E. - An Algorithm for Large Zero-One Knapsack Problems. Operations Research, 28 : 1130-1154, 1980.

03. BREU, R. & BURDET, C. A. - Branch and Bound Experiments in Zero-One Programming. Mathematical Programming Study, 2 : 1-50, 1970.

04. CABOT, A. V. - An Enumeration Algorithm for Knapsack Problems. Operations Research, 18 : 306-311, 1970.

05. CHVÁTVAL, V. - Hard Knapsack Problems. Operations Research, 28 : 1402-1411, 1980.

06. DANTZIG, G. B. - Discrete Variable Extremum Problems. Operations Research, 5 : 266-277, 1957.

07. DEMBO, R. S. & HAMMER, P. L. - A Reduction Algorithm for Knapsack Problems. Methods on Operations Research, 36 : 40-60, 1980.

08. FAYARD, D. & PLATEAU, G. - Resolution of the 0-1 Knapsack Problem: Comparison of Methods. Mathematical Programming, 8 : 272-307, 1975.

09. FAYARD, D. & PLATEAU, G. - An Algorithm for the Solution of the 0-1 Knapsack Problem (Algorithm 47). Computing, 28 : 269-287, 1982.

10. GEOFFRION, A. M. - Lagrangean Relaxation for Integer Programming. Mathematical Programming Study, 2 : 82-114, 1974.

11. GEOFFRION, A. M. & MARSTEN, R. E. - Integer Programming Algorithms: A Framework and a State-of-the Art Survey. Management Science, 18:405-491, 1972.

12. GLOVER, F. - A Multiphase Dual Algorithm for the Zero-One Integer Programming Problem. Operations Research, 13 : 879-919, 1965.

13. GREENBERG, H. & HEGERICH, R. L. - A Branch Search Algorithm for the Knapsack Problem. Management Science, 16 : 327-332, 1970.

14. HOROWITZ, E. & SAHNI, S. - Computing Partitions with Applications to the Knapsack Problem. Journal of the ACM, 21 : 277-292, 1974.

15. INGARGIOLA, G. P. & KORSH, J. F. - A Reduction Algorithm for the Zero-One Single Knapsack Problems. Management Science, 20 : 460-463, 1973.

16. KOLESAR, P. J. - A Branch and Bound Algorithm for the Knapsack Problem. Management Science, 13 : 723-735, 1967.

17. LAURIERE, M. - An Algorithm for the 0/1 Knapsack Problem. Mathematical Programming, 14 : 1-10, 1978.

18. LOESER, R. - Some Performance Tests of "quicksort" and Descendants. Communications of the ACM, 17 : 143-152, 1974.

19. MACULAN FILHO, N. - Relaxation Lograngienne: Le Problème du Knapsack 0-1. Publicação nº 435 do Département d'Informatique et de recherche opérationnelle da Universidade de Montreal, 1982, 18 p.

20. MACULAN FILHO, N & VILLARES, M. L. - Comentários Sobre um Algoritmo para a Solução Aproximada do Problema da Mochila. Pesquisa Operacional, 1 : 23-28, 1981.

21. MAGAZINE, M. J. & OGUZ, O. - A Fully Polynomial Approximation Algorithm for the 0-1 Knapsack Problem. European Journal of Operational Research, 8 : 270-273, 1981.

22. MARTELLO, S & TOTH, P. - An Upper Bound for the Zero-One Knapsack Problem and a Branch and Bound Algorithm. European Journal of Operational Research, 1 : 169-175, 1977.

23. MARTELLO, S & TOTH, P. - Algorith for the Solution of the 0-1 Single Knapsack Problem (Algorith 37). Computing, 21 : 81-86, 1978.

24. MARTELLO, S. & TOTH, P. - "The 0-1 Knapsack Problem" in Combinatorial Optimization. N. York, J. Wiley, 1979.

25. MOTTA, V. L. - Contribuição ao Estudo do Problema da Mochila. Tese de M. Sc. COPPE/UFRJ, 1981.

26. NAUSS, R. M. - An Efficient Algorithm for the 0-1 Knapsack Problem. Management Science, 23 : 27-31, 1976.

27. PAPADIMITRIOU, C. H. & STEIGLITZ, K. - Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, 1982.

28. SAHNI, S. - Approximate Algorithms for the 0/1 Knapsack Problem. Journal of the ACM, 22 : 115-124, 1975.

29. SINICIO, R. & ROA, G. - Secador Rural de Café, Cacau, Mandioca e outros Produtos Agropecuários com uso de Energia Solar. Série CENTREINAR nº 1, Viçosa, 1980.

30. SUHL, V. - An Algorithm and Efficient Data Structures for the Binary Knapsack Problem. European Journal of Operational Research, 2 : 420-428, 1978.

31. TOTH, P. - Dynamic Programming Algorithms for the Zero-One Knapsack Problem. Computing, 25 : 29-45, 1980.

32. ZOLTNERS, A. A. - A Direct Descent Binary Knapsack Algorithm. Journal of the ACM, 25 : 304, 311, 1978.