

Deisemara Ferreira

**Um modelo de Dimensionamento de Lotes
aplicada à Indústria de Bebidas**



1910001077





**UM MODELO DE DIMENSIONAMENTO DE LOTES
APLICADO À INDÚSTRIA DE BEBIDAS**

Deisemara Ferreira

Dissertação de Mestrado
Pós-Graduação em Matemática Aplicada
MAP - 065



519
F383m

UM MODELO DE DIMENSIONAMENTO DE LOTES
APLICADO À INDÚSTRIA DE BEBIDAS

Deisemara Ferreira

Orientador: Profa. Dra. Maria do Socorro N. Rangel

Ferreira, Deisemara.

Um modelo de dimensionamento de lotes aplicado à indústria de bebidas/ Deisemara Ferreira. – São José do Rio Preto : [s.n.], 2002
119 f. : il. ; 30 cm.

Orientador: Maria do Socorro Nogueira Rangel.

Dissertação (mestrado) – Universidade Estadual Paulista. Instituto de Biociências, Letras e Ciências Exatas.

1. Programação inteira mista. 2. Planejamento da Produção. 3. Algoritmo híbrido. 4. Planos de Corte. 5. Pesquisa Operacional I. Rangel, Maria do Socorro. II. Universidade Estadual Paulista. Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU – 519.854

Agradecimentos

A Prof. Dra. Sílvia Rangel, pela orientação, incentivo e paciência ao longo deste trabalho.

Aos meus familiares que apoiaram este trabalho.

Aos amigos.

A FAPESP pela concessão da bolsa de estudos (Proc. nº 00/00023-0).

A todos os professores da FAPESP pela orientação.

A todos aqueles que me ajudaram e apoiaram ao longo deste trabalho, especialmente os meus pais e irmãos.

E a Deus, por ter me guiado por esta jornada tão importante.

Aos meus pais e avós .

E às minhas irmãs, Rosimara e Fernanda.

Dedico

Resumo

Neste trabalho, formulamos um modelo para um problema real de planejamento da produção de uma fábrica de refrigerantes. Para resolvê-lo propomos um algoritmo híbrido que combina o método de planos de corte e o algoritmo Branch and Bound.

Palavras-chave: Programação Inteira Mista, Planejamento da Produção, Algoritmos Híbridos, Método de Planos de Corte.



Sumário

Abstract

In this work, we formulate a Production Planning Model of a real soft drink company. To solve it we study a hybrid algorithm that combines Cutting Planes and the Branch and Bound algorithm.

Keywords: Mixed Integer Programming , Production Planning, Hybrid Algorithms, Cutting Plane.



Sumário

1 Modelos de Planejamento da Produção	3
1.1 Problemas de Dimensionamento de Lote	4
1.2 Problemas de Dimensionamento de Lote e Sequenciamento	11
2 Solução de Problemas de Otimização Inteira Mista	17
2.1 Complexidade Computacional	19
2.2 Método de Planos de Corte	22
2.2.1 Corte Fracionário de Gomory	27
2.2.2 Inequações Válidas para o Modelo de Lasdon-Terjung . . .	31
2.2.3 Inequações Supermodulares	32
2.3 Algoritmo Branch and Bound (B&B)	43
2.4 Métodos Heurísticos	50
3 Modelo de Planejamento da Produção para uma Fábrica de Refrigerantes (FR)	52
3.1 A Linha de produção	52
3.2 O planejamento da produção	56
3.3 Ferramentas para o Planejamento da Produção	63
4 Experimentos Computacionais	73
4.1 Geração de Dados	75



4.2	Solução do Modelo FR	77
4.3	O Uso de Inequações Supermodulares na Solução do Modelo FR .	94
4.3.1	O uso da Biblioteca de Subrotinas do CPLEX	94
4.3.2	O algoritmo SolFR	97
5	Considerações Finais	103
	Referências Bibliográficas	105



Introdução

A Pesquisa Operacional ganhou impulso principalmente durante a Segunda Grande Guerra, quando foi utilizada para resolução de problemas de operações militares: emprego eficiente de radares, canhões, táticas de bombardeio entre outras. Após a guerra as técnicas desenvolvidas, principalmente a forma de coleta de dados e informações para análise, passaram a ser utilizadas nas empresas em problemas de administração (Von Ellenrieder, 1971). Os modelos matemáticos também se tornaram uma nova ferramenta de trabalho nas empresas.

O uso da modelagem matemática favorece a inter-relação dos departamentos de uma empresa já que é necessário entender e coletar dados de vários setores. Permite também uma experimentação prévia das decisões a serem tomadas. No entanto, a utilização de modelos matemáticos nas indústrias passou por um processo de conquista onde foi, e ainda é necessário, provar a eficiência dos modelos no processo de decisão.

Com o desenvolvimento da pesquisa matemática e dos computadores, cada vez mais a modelagem matemática tem se mostrado uma boa ferramenta nos processos de decisão. Hoje é aplicada nos mais variados tipos de empresas e problemas tais como: indústria de bebidas, tabaco, transporte, localização (Clark, 2001) (Pattoloch, Schmidt, Kovalyov, 2001)(Williams, 1991).

Apesar deste avanço da ciência e da tecnologia ainda existem vários tipos de modelos que não possuem um método de solução capaz de fornecer uma solução ótima rapidamente. Este é o caso de alguns modelos de programação inteira mista que consideram tempos e custos de preparo de máquinas (tempo e custos de set up). Por isso, é interessante o estudo de métodos de solução que colaborem na obtenção de bons resultados para estes modelos.

Neste trabalho apresentamos um modelo de Programação Inteira Mista (PIM) para resolver um problema de uma fábrica de refrigerantes de São José do Rio Preto. O modelo desenvolvido envolve dimensionamento de lotes considerando



tempo e custos de preparo de máquinas (setup) e inicialização (startup), com restrições adicionais. Devido a complexidade do modelo obtido propomos sua solução através de um método híbrido que combina o método de planos de corte e o método Branch and Bound, conhecido como Branch and Cut (Padber e Rinald, 1991).

No capítulo 1 discutimos a modelagem de problemas clássicos de planejamento da produção encontrados na literatura.

Para solucionar estes e outros modelos de programação inteira mista existem vários métodos: métodos exatos como o algoritmo Branch and Bound, e o Método de Planos de Corte; e métodos de aproximação como é o caso das Heurísticas.

Estes métodos são explicados com detalhes no capítulo 2.

O capítulo 3 descreve o funcionamento da fábrica de refrigerantes visitada e o modelo matemático proposto (FR).

O método branch and cut desenvolvido, sua implementação, e testes computacionais realizados para testar a dificuldade do modelo formulado, e a eficiência do algoritmo proposto para sua solução, são apresentados no capítulo 4.

O último capítulo, capítulo 5, apresenta as conclusões obtidas com o trabalho e sugestões para trabalhos futuros.



Capítulo 1

Modelos de Planejamento da Produção

Um modelo matemático pode ser uma ferramenta útil na determinação do programa de produção de uma empresa. Para a construção do modelo, na maioria das vezes, é necessário levar em consideração diversos fatores que influenciam a produção tais como: capacidade da fábrica, se é necessário ou não preparar as máquinas, e em caso positivo qual o tempo de preparo e o custo, quantos itens serão produzidos, o horizonte de planejamento, os custos de produção, custos de armazenamento se houver excesso de produção (estoque); entre muitos outros fatores que podem surgir dependendo da empresa e do tipo de produto (Wagner, 1986).

Cada fator relevante para o problema implica em uma restrição e/ou variável no modelo.

Descreveremos neste capítulo modelos clássicos de dimensionamento de lote que incorporam alguns dos elementos descritos acima. Fazemos também uma breve discussão de modelos que combinam o dimensionamento de lote e o sequenciamento da produção.



1.1 Problemas de Dimensionamento de Lote

Consideremos um problema onde há a produção de um único item. Esta produção é realizada por uma única máquina. O objetivo da produção é suprir uma dada demanda d_t sobre um horizonte de planejamento de T períodos com o menor custo total possível.

O custo total é composto pelo custo de produção c_t e pelo custo de estoque h_t no período t ; Considerando x_t a quantidade a ser produzida no período t ; e_t o estoque (excedente de produção) no período t e, tendo em vista o objetivo de minimizar o custo total, a função objetivo do problema pode ser formulada como:

$$\text{Min } z = \sum_{t=1}^T (c_t x_t + h_t e_t).$$

A garantia de que a produção de cada período satisfaça a demanda é assegurada pela seguinte restrição de balanço e estoque:

$$x_t + e_{t-1} = d_t + e_t, \quad t = 1, \dots, T.$$

A produção deve ser não negativa. O estoque também será considerado não negativo pois, não estamos permitindo pedidos em atraso (backorders). Incluímos então ao modelo a restrição abaixo.

$$x_t \geq 0, \quad e_t \geq 0, \quad t = 1, \dots, T.$$



O modelo formulado é conhecido como modelo de dimensionamento de lote de um item não capacitado (PUNC):

$$\text{Min } z = \sum_{t=1}^T (c_t x_t + h_t e_t) \quad (1.1)$$

Sujeito à:

$$x_t + e_{t-1} = d_t + e_t, \quad t = 1, \dots, T; \quad (1.2)$$

$$x_t \geq 0, \quad e_t \geq 0, \quad t = 1, \dots, T. \quad (1.3)$$

Em muitos casos é necessário levar em consideração a capacidade da fábrica e o fato da máquina estar ou não preparada para a produção. Seja então C_t a capacidade de produção da fábrica em cada período t . Tendo em vista que a produção não pode ultrapassar esta capacidade é incluída ao modelo a restrição de capacidade:

$$x_t \leq C_t, \quad t = 1, \dots, T.$$

A função objetivo e as outras restrições permanecem inalteradas. Assim o novo modelo de produção de um item capacitado (PUC) é (Wolsey e Pochet, 1991):

$$\text{Min } z = \sum_{t=1}^T (c_t x_t + h_t e_t) \quad (1.4)$$

Sujeito à:

$$x_t + e_{t-1} = d_t + e_t, \quad t = 1, \dots, T; \quad (1.5)$$

$$x_t \leq C_t, \quad t = 1, \dots, T; \quad (1.6)$$

$$x_t \geq 0, \quad e_t \geq 0, \quad t = 1, \dots, T. \quad (1.7)$$



Se ao invés da capacidade da fábrica só houver a necessidade de garantir que só haverá produção se a máquina estiver preparada para tal, incorporamos ao modelo a restrição:

$$x_t \leq M y_t, \quad t = 1, \dots, T.$$

Onde, y_t é uma variável binária, chamada de variável de set-up, que é igual a 1 se a máquina estiver preparada e 0 caso contrário; e M é um número grande que evita a limitação da produção.

Com esta restrição, $0 \leq x_t \leq M$, poderá haver produção apenas se a máquina estiver preparada, $y_t = 1$.

Para preparar uma máquina e mantê-la preparada tem-se um custo fixo, que muitas vezes pode ser alto e por isso é necessário considerá-lo no custo total da produção (Constantino, 1996). Este custo (custo de set-up), c_t^s é incluído na função objetivo que minimiza o custo total.

Temos então o modelo de produção de um item não capacitado com set-up (PUNCS)(Wolsey e Pochet, 1991):

$$\text{Min } z = \sum_{t=1}^T (c_t x_t + h_t e_t + c_t^s y_t) \quad (1.8)$$

Sujeito à:

$$x_t + e_{t-1} = d_t + e_t, \quad t = 1, \dots, T; \quad (1.9)$$

$$x_t \leq M y_t, \quad t = 1, \dots, T; \quad (1.10)$$

$$x_t \geq 0, \quad e_t \geq 0, \quad y_t = 0/1, \quad t = 1, \dots, T. \quad (1.11)$$

Observando o modelo PUNCS percebe-se que se substituirmos M por C_t estaríamos considerando a capacidade e o preparo da máquina no mesmo modelo.

Com isto o modelo obtido seria um modelo de produção de um item capacitado com set-up (PUCS)(Wolsey e Pochet, 1991).

Se houver necessidade, pode se considerar o início da produção pois, algumas vezes a máquina, apesar de estar preparada, não começa a produzir imediatamente naquele período. Através da inclusão da variável de start-up z_t , que é 1 se há o início da produção no período t e 0 caso contrário, das restrições:

$$z_t \geq y_t - y_{t-1}, \quad t = 1, \dots, T;$$

$$z_t \leq y_t, \quad t = 1, \dots, T;$$

$$z_t \leq 1 - y_{t-1}, \quad t = 1, \dots, T;$$

e do custo de se iniciar a produção, c_t^z , que também pode ser incluído no custo total, formulamos um modelo de um item capacitado com custo de start-up (PUCSS)(Constantino, 1996).

$$\text{Min } z = \sum_{t=1}^T (c_t x_t + c_t^s y_t + c_t^z z_t + h_t e_t) \quad (1.12)$$

Sujeito à:

$$x_t + e_{t-1} = d_t + e_t, \quad t = 1, \dots, T; \quad (1.13)$$

$$x_t \leq C_t y_t, \quad t = 1, \dots, T; \quad (1.14)$$

$$z_t \geq y_t - y_{t-1}, \quad t = 1, \dots, T; \quad (1.15)$$

$$z_t \leq y_t, \quad t = 1, \dots, T; \quad (1.16)$$

$$z_t \leq 1 - y_{t-1}, \quad t = 1, \dots, T; \quad (1.17)$$

$$x_t \geq 0, e_t \geq 0, y_t \text{ e } z_t = 0/1, t = 1, \dots, T. \quad (1.18)$$

As três restrições inseridas garantem que a produção de um novo item será realizada apenas se a máquina for preparada naquele período, $y_t = 1$ e $y_{t-1} = 0$.

É possível modificar os modelos descritos acima para incorporar a produção de vários itens simultaneamente. Neste caso as variáveis passam a ser:

x_{it} é a quantidade produzida do item i no período t ;

e_{it} é a quantidade estocada do item i no período t ;

$$y_{it} \begin{cases} 1 & \text{se a máquina estiver preparada para produção do item } i \text{ no período } t. \\ 0 & \text{caso contrário.} \end{cases}$$

$$z_{it} \begin{cases} 1 & \text{se há o início da produção do item } i \text{ no período } t. \\ 0 & \text{caso contrário.} \end{cases}$$

E os custos:

c_{it} é o custo de produção do item i no período t ;

h_{it} é o custo de estoque do item i no período t ;

c_{it}^s é o custo de set-up do item i no período t ;

C_{it} é a capacidade de produção do item i no período t .

Há necessidade também de incorporar a produção dos vários itens nas restrições. Assim teremos restrições de balanço e estoque, capacidade e startup para cada item i e cada período t . O custo total a ser minimizado é obtido pela soma dos custos de todos os itens em todos os períodos.

Consideremos os seguintes modelos multi-item:



a) Modelo de Planejamento da Produção Multi-item Capacitado (PMC)(Staggemeier e Clark, 2001).

$$\text{Min } z = \sum_{i=1}^N \sum_{t=1}^T (c_{it}x_{it} + h_{it}e_{it}) \quad (1.19)$$

Sujeito à:

$$x_{it} + e_{i(t-1)} = d_{it} + e_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.20)$$

$$\sum_{i=1}^N b_i x_{it} \leq C_t, \quad t = 1, \dots, T; \quad (1.21)$$

$$x_{it} \geq 0, \quad e_{it} \geq 0, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.22)$$

onde b_i é a capacidade necessária para produzir uma unidade do produto i .

b) Modelo de Planejamento da Produção Multi-item Capacitado com custo de Setup(PMCS)(Trigeiro, Thomas e McClain, 1989).

$$\text{Min } z = \sum_{i=1}^N \sum_{t=1}^T (c_{it}x_{it} + h_{it}e_{it} + c_{it}^s y_{it}) \quad (1.23)$$

Sujeito à:

$$x_{it} + e_{i(t-1)} = d_{it} + e_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.24)$$

$$\sum_{i=1}^N b_i x_{it} \leq C_t, \quad t = 1, \dots, T; \quad (1.25)$$

$$x_{it} \leq M y_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.26)$$

$$x_{it} \geq 0, \quad e_{it} \geq 0, \quad t = 1, \dots, T, \quad i = 1, \dots, N. \quad (1.27)$$

c) Modelo de Planejamento da Produção Multi-item Capacitado com Custo de Set-up e Start-up (PMCSS) ou Modelo Cíclico de Produção Dinâmica Determinístico(Constantino, 1996).

$$\text{Min } z = \sum_{i=1}^N \sum_{t=1}^T (c_{it}x_{it} + c_{it}^s \cancel{y_{it}} z_{it} + c_{it}^z z_{it} + h_{it}e_{it}) \quad (1.28)$$

Sujeito à:

$$x_{it} + e_{i(t-1)} = d_{it} + e_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.29)$$

$$x_{it} \leq C_t y_{it}, \quad t = 1, \dots, T; \quad (1.30)$$

$$z_{it} \geq y_{it} - y_{i(t-1)}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.31)$$

$$z_{it} \leq y_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.32)$$

$$z_{it} \leq 1 - y_{i(t-1)}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.33)$$

$$\sum_{i=1}^T y_{it} \leq 1, \quad t = 1, \dots, T. \quad (1.34)$$

$$x_{it} \geq 0, \quad e_{it} \geq 0, \quad y_{it} \text{ e } z_{it} = 0/1, \quad t = 1, \dots, T, \quad i = 1, \dots, N. \quad (1.35)$$

Observe que a restrição

$$\sum_{i=1}^T y_{it} \leq 1, \quad t = 1, \dots, T.$$

garante que haja produção de apenas um item por período.

Nesta seção discutimos alguns modelos clássicos da literatura. No entanto, vários outros fatores podem ser incluídos para controlar a quantidade de insumos, o tempo disponível de produção, entre outros elementos do problema. Este é o caso do modelo de dimensionamento de lote discutido no capítulo 3.

A produção de itens diferentes e que precisam de um preparo especial das máquinas muitas vezes envolve o sequenciamento dos itens. Isto é, a ordem em que os itens são produzidos pode interferir na capacidade da produção. Os modelos multi-item discutidos nesta seção não consideram este elemento na definição dos lotes. A sequência da produção é estabelecida antes ou depois do dimensionamento dos lotes. Os modelos que consideraremos na seção 1.2 a seguir consideram o dimensionamento de lotes e o sequenciamento em um mesmo modelo.



1.2 Problemas de Dimensionamento de Lote e Sequenciamento

Na seção anterior descrevemos vários modelos de dimensionamento de lote. Entre eles apresentamos modelos onde há tempo e/ou custo de setup.

O setup aparece quando a produção de um item necessita de algum tipo de preparo: limpeza da máquina, ajuste, entre outros.

O tempo e/ou custo de setup pode depender do próximo item a ser produzido. Em uma empresa de refrigerante, por exemplo, a produção de um refrigerante diet após a produção de um refrigerante com açúcar necessita de uma limpeza mais detalhada da máquina, ou seja, há necessidade de um tempo de setup maior, do que se fosse produzida na ordem contrária, refrigerante diet e depois com açúcar. Outro exemplo que ilustra a relação do setup com a ordem da produção está na empresa de tintas. Produzir primeiro tonalidades diferentes de uma mesma cor, apenas tons de azul, exige menos setup do que produzir tons iguais de cores diferentes, azul claro, vermelho claro, depois azul escuro, em seguida vermelho escuro. Assim, percebemos como o custo e o tempo de setup podem depender da sequência em que os itens são produzidos (Toledo, 2001)(Potts e Wassenhove, 1992).

Além da sequência, a formação de estoque também contribui para a redução do setup pois, um lote grande utiliza o mesmo setup de um lote pequeno. É claro que existe uma quantidade adequada a ser produzida de forma que o custo de estoque seja o menor possível (Haase e Kimms, 2000). Baseado nisto, tem sido desenvolvidos modelos de produção que determinam simultaneamente o tamanho dos lotes e a sequência na qual os itens devem ser produzidos a fim de atender a demanda com custo de estoque e custo de setup mínimos.

Nesta seção apresentaremos alguns destes modelos que combinam dimensionamento de lote e sequenciamento da produção em um único modelo.



Consideremos a produção de vários itens realizada por uma única máquina. O objetivo da produção é suprir uma dada demanda d_t sobre um horizonte de planejamento de T períodos a um custo total mínimo. Porém, há uma capacidade C_t em cada período t que deve ser respeitada e a cada novo item a ser produzido é necessário um tempo para preparar a máquina que varia dependendo do item. Este preparo gera um custo c_{it}^s que juntamente com o custo de estoque h_{it} forma o custo total a ser minimizado.

Este problema é o mais simples no sentido de unir dimensionamento e sequenciamento da produção. É chamado de problema de sequenciamento e dimensionamento de lote discreto (PSDD), possui uma estrutura parecida com o modelo (PMCSS) formulado na seção anterior (Drexel e Kimms, 1997).

Este modelo considera intervalos de tempo menores (dias, horas,...) que PMCSS, e em cada intervalo pode ser produzido no máximo um item, o que é garantido pela restrição:

$$\sum_{i=1}^N y_{it} \leq 1, \quad t = 1, \dots, T.$$

A capacidade varia com os períodos e a parâmetro k_i indica quanto de capacidade o item x_{it} utiliza. O modelo, ao contrário de PMCSS, determina que se houver produção a capacidade seja totalmente utilizada, esta restrição é conhecida como "tudo-ou-nada":

$$k_{it}x_{it} = C_t y_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N.$$

A função objetivo minimiza o custo de mudança de item, ou seja, custo de iniciar a produção de um novo item c_{it}^z , e o custo de estoque h_{it} . O modelo completo (Drexel e Kimms, 1997) é:

$$\text{Min } z = \sum_{i=1}^N \sum_{t=1}^T (c_{it}^z z_{it} + h_{it} e_{it}) \quad (1.36)$$



Sujeito à:

$$x_{it} + e_{i(t-1)} = d_{it} + e_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.37)$$

$$k_{it}x_{it} = C_t y_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.38)$$

$$\sum_{i=1}^N y_{it} \leq 1, \quad t = 1, \dots, T; \quad (1.39)$$

$$z_{it} \geq y_{it} - y_{i(t-1)}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.40)$$

$$x_{it}, z_{it}, e_{it} \geq 0, y_{it} = 0/1, \quad t = 1, \dots, T, \quad i = 1, \dots, N. \quad (1.41)$$

Observemos que não há necessidade de definir z_{it} binária uma vez que o custo c_{it}^z positivo na função objetivo e a restrição (1.40) forçam esta condição.

Uma outra forma de modelar o problema é permitindo o uso parcial da capacidade, ou seja, substituindo a igualdade 1.38 por:

$$k_{it}x_{it} \leq C_t y_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N.$$

E assim temos o problema de dimensionamento de lote e setup contínuo (PDSC) (Drexler e Kimms, 1997).

$$\text{Min } z = \sum_{i=1}^N \sum_{t=1}^T (c_{it}^z z_{it} + h_{it} e_{it}) \quad (1.42)$$

Sujeito à:

$$x_{it} + e_{i(t-1)} = d_{it} + e_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.43)$$

$$k_{it}x_{it} \leq C_t y_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.44)$$

$$\sum_{i=1}^N y_{it} \leq 1, \quad t = 1, \dots, T; \quad (1.45)$$

$$z_{it} \geq y_{it} - y_{i(t-1)}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.46)$$

$$x_{it}, z_{it}, e_{it} \geq 0, y_{it} = 0/1, \quad t = 1, \dots, T, \quad i = 1, \dots, N. \quad (1.47)$$

Para aproveitarmos a capacidade que não é utilizada totalmente em um período podemos permitir a produção de mais um item por período. Para tanto substituímos a inequação de capacidade (1.44) do modelo PSDD pela inequação:

$$k_{it}x_{it} \leq C_t(y_{it} + y_{i(t-1)}), \quad t = 1, \dots, T, \quad i = 1, \dots, N.$$

Assim, permitimos a produção de um item, $i = 1$ por exemplo, que tenha a máquina preparada para sua produção no período $(t-1)$, $y_{1(t-1)} = 1$, e um item, $i = 2$, que tenha a máquina preparada para sua produção no período t , $y_{2t} = 1$. Pois,

$$k_{1t}x_{1t} \leq C_t(0 + 1), \quad t = 1, \dots, T;$$

$$k_{2t}x_{2t} \leq C_t(1 + 0), \quad t = 1, \dots, T.$$

Mantendo as demais restrições inalteradas temos então, o problema sequenciamento e dimensionamento de lote proporcional (PLSP)(Drexel e Kimms, 1997):

$$\text{Min } z = \sum_{i=1}^N \sum_{t=1}^T (c_{it}^z z_{it} + h_{it} e_{it}) \quad (1.48)$$

Sujeito à:

$$x_{it} + e_{i(t-1)} = d_{it} + e_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.49)$$

$$k_{it}x_{it} \leq C_t(y_{it} + y_{i(t-1)}), \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.50)$$

$$\sum_{i=1}^N y_{it} \leq 1, \quad t = 1, \dots, T; \quad (1.51)$$

$$z_{it} \geq y_{it} - y_{i(t-1)}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.52)$$

$$x_{it}, z_{it}, e_{it} \geq 0, y_{it} = 0/1, \quad t = 1, \dots, T, \quad i = 1, \dots, N. \quad (1.53)$$

Uma outra idéia que visa modelar os problemas de dimensionamento de lote incluindo o sequenciamento é a de dividir os períodos, definidos então de macro-períodos, em períodos menores, micro-períodos. O modelo obtido, que se segue, é conhecido como problema de sequenciamento e dimensionamento de lote geral (PSDG)(Meyr, 2000).

$$\text{Min } z = \sum_{s \in S_t} \sum_{i=1}^N \sum_{t=1}^T (c_{ij}^s z_{ijs}) + \sum_{i=1}^N \sum_{t=1}^T (h_{it} e_{it}) \quad (1.54)$$

Sujeito à:

$$\sum_{s \in S_t} x_{is} + e_{i(t-1)} = d_{it} + e_{it}, \quad t = 1, \dots, T, \quad i = 1, \dots, N; \quad (1.55)$$

$$\sum_{i=1}^N \sum_{s \in S_t} k_i x_{is} \leq C_t, \quad s \in S_t, \quad i = 1, \dots, N; \quad (1.56)$$

$$k_i x_{is} \leq C_t y_{is}, \quad s \in S_t, \quad i = 1, \dots, N; \quad (1.57)$$

$$\sum_{i=1}^N y_{is} = 1, \quad s \in S_t; \quad (1.58)$$

$$z_{ijs} \geq y_{i(s-1)} + y_{js} - 1, \quad s \in S_t, \quad i, j = 1, \dots, N; \quad (1.59)$$

$$x_{is} \geq m_i (y_{is} - y_{i(s-1)}), \quad s \in S_t, \quad i, j = 1, \dots, N; \quad (1.60)$$

$$x_{is}, z_{ijs}, e_{it} \geq 0, y_{is} = 0/1, \quad t = 1, \dots, T, \quad i = 1, \dots, N, s \in S_t. \quad (1.61)$$

Onde:



S_t = conjunto de micro-períodos s pertencentes ao macro-período t ;

z_{ijs} = é 1 se há troca de produção do item i para o item j no início do micro-período s ; e 0 caso contrário.

c_{ij}^s = é o custo de realizar a troca do item i para o item j .

m_i = produção mínima do produto i .

A restrição (1.55) é a restrição de balanço e estoque; a restrição (1.56) considera a capacidade; as restrições (1.57) e (1.58) permitem que haja produção apenas se a máquina estiver preparada para tal; a restrição (1.59) controla a mudança do item i para o item j na produção pois, $z_{ijs} = 1$ apenas se $y_{i(s-1)} = 1$ e $y_{js} = 1$ e a restrição (1.60) garante que se o setup de um novo item é realizado então há a produção de pelo menos m_i unidades deste item naquele micro período; por fim temos a garantia de não negatividade das variáveis. Observe que o número de micro-períodos, isto é o número de lotes, é determinado pelo usuário.

Podemos também considerar o tempo de preparo da mudança do item i para o j , st_{ij} , na restrição de capacidade. Isto equivale substituir a restrição (1.56) por:

$$\sum_{i=1}^N \sum_{s \in S_t} k_i x_{is} + \sum_{i=1}^N \sum_{j=1}^N \sum_{s \in S_t} st_{ij} z_{ijs} \leq C_t, \quad s \in S_t, \quad i = 1, \dots, N.$$

Temos então o modelo GLSP com tempo de setup (Meyr, 2000).

Muitos métodos são desenvolvidos para solução dos modelos apresentados. Todos buscam uma solução rápida e de boa qualidade, soluções ótimas se possível. No próximo capítulo descrevemos um pouco destes métodos e da teoria de complexidade que se desenvolveu para avaliá-los.



Capítulo 2

Solução de Problemas de Otimização Inteira Mista

Os modelos que descrevem problemas reais são modelos na maioria das vezes complexos, com várias variáveis e restrições. Resolvê-los e encontrar uma solução boa e rápida pode ser uma tarefa difícil.

Quando se aplica um método de solução em um problema linear testa-se direções factíveis que possam melhorar esta solução. Se não existe uma direção que melhore a solução obtida do problema linear sabemos que encontramos a solução ótima. No caso de problemas inteiros ou combinatoriais, em geral não se tem estas condições de otimalidade global. A solução destes problemas pode ser obtida através de enumeração implícita das soluções, isto é, uma vez obtida uma boa solução descarta-se uma parte das combinações possíveis por serem inviáveis ou por não terem possibilidade de superar o limite já obtido (Puccini e Pizzolato, 1987).

Várias técnicas de solução de modelos de programação inteira mista (PIM) são desenvolvidas para diminuir o tempo de solução através de bons limites (soluções factíveis). O uso de heurísticas, por exemplo, garante uma solução factível. Um algoritmo híbrido, que combine heurística e o método clássico branch and bound



pode diminuir o tempo de solução de um problema e dar uma solução ótima.

Neste capítulo descreveremos métodos clássicos de solução de PIM como o algoritmo Branch and Bound e o método de Planos de Corte; e métodos heurísticos. Faremos também uma breve discussão sobre complexidade computacional, que nos orienta na decisão se um modelo ou um algoritmo são bons ou não.

Supomos que o leitor esteja familiarizado com resolução de problemas de otimização linear. Em particular o método Simplex (Vanderbei, 2001) (Puccini e Pizzolato, 1987) e o método Dual Simplex (Bertsimas e Tsitsiklis, 1997) (Bertsekas, 1992).



2.1 Complexidade Computacional

Nesta seção discutimos a complexidade computacional de problemas de otimização, para isto é necessário então definirmos o que é um problema, um exemplar do modelo ou problema e um algoritmo.

Definição 2.1: Um problema é uma questão geral a ser respondida, normalmente com vários parâmetros, ou variáveis livres, cujos valores são deixados sem especificação. Um problema é definido por: uma descrição geral de todos seus parâmetros estabelecendo que propriedades a resposta ou solução deve satisfazer. Um exemplar de um problema é obtida especificando valores particulares para todos os parâmetros do problema. (Garey e Johnson, 1997)

Definição 2.2: Um algoritmo é um procedimento passo-a-passo para solução de problemas. Um algoritmo é dito resolver um problema se puder ser aplicado a qualquer exemplar do problema e garantir uma solução para este exemplar. (Garey e Johnson, 1997)

Diante da disponibilidade de vários algoritmos para resolução de um mesmo problema, surge a necessidade de se ter um critério de análise para a escolha do mais adequado.

O critério mais utilizado é o tempo de execução.

Este tempo se refere ao número de vezes que uma determinada operação é realizada.

Para medir o tempo de execução foi desenvolvida a função tempo $T(.)$ do algoritmo, que associa o tempo de execução ao tamanho dos dados de entrada (exemplar do problema).

Diante dos inúmeros exemplares que podem ser geradas criou-se três tipos de situações onde o comportamento do algoritmo pode ser estudado e a função tempo obtida:

i) Estudo do melhor caso, ou seja, o menor tempo que o algoritmo leva para executar todas possíveis entradas de tamanho n .



ii) Estudo do pior caso, que é o maior tempo sobre as entradas de tamanho n .

iii) Estudo do caso médio (ou esperado), é a média dos tempos de execução de todas entradas de tamanho n .

Apesar de ser possível fazer estas avaliações, a obtenção da função $T(\cdot)$ é difícil. Por isso, criou-se o conceito de complexidade assintótica. Isto é, ao invés de determinar $T(\cdot)$, determina-se uma função $F(\cdot)$ que limita o tempo do algoritmo.

Definição 2.3 :Sejam f e $g : N \rightarrow R$. Diz-se que f domina g assintoticamente quando $\exists$ constantes C e $n_0 > 0$ tais que :

$$g(n) \leq C f(n), \forall n \geq n_0.$$

Assim, se a função $F(\cdot)$ domina assintoticamente a função $T(\cdot)$ que define o tempo de execução de um algoritmo dizemos que este algoritmo é de ordem $O(F(\cdot))$. A notação O , (*big - O*) foi introduzida por Paul Bachmann em 1894.(Campello e Maculan, 1994)

Por exemplo, seja $g(n) = 3n^3 + 2n^2$. $g(n)$ é de ordem $O(n^3)$, pois, para $n_0 = 0$ e $C = 5$ tem-se $3n^3 + 2n^2 \leq 5n^3$ (Campello e Maculan, 1994).

Sabendo então como medir a eficiência de um algoritmo, podemos definir um algoritmo como sendo bom se o seu tempo de execução puder ser limitado por uma função polinomial e então diz-se que o algoritmo tem complexidade de tempo polinomial. Se a função obtida é exponencial o algoritmo é dito ter complexidade exponencial (Campello e Maculan, 1994).

Esse conceito de "bom" se deve ao fato de que sempre existe um N tal que a partir dele as funções exponenciais são maiores que as funções polinomiais.

A classificação não se restringe apenas aos algoritmos. Muitos problemas são dados como fáceis ou difíceis se podem ou não ser resolvidos por algoritmos com tempo polinomial. Para classificá-los existem as classes P, NP e NP-completo.

Um problema é dito pertencer á classe P se existe um algoritmo com tempo polinomial que o resolva.

Para definirmos a classe NP precisamos saber o que é um algoritmo não-



determinístico.

Definição 2.4 Um algoritmo é não determinístico se gera respostas diferentes para um mesmo exemplar, ou seja, há um desvio condicional que vai para dois lugares diferentes ao mesmo tempo.

Um problema é dito pertencer a classe NP, se é resolvido por um algoritmo não determinístico.

Observemos que os algoritmos que não possuem o desvio são casos especiais de algoritmos não determinísticos. Com isso podemos concluir que $P \subseteq NP$.

Se todo problema da classe NP puder ser transformado em tempo polinomial em um problema P' , que também está na classe NP, então dizemos que este problema é NP-completo. Caso não pertença à classe NP dizemos que é NP-difícil (Reeves, 1993).

Tendo em vista as dificuldades de resolução de certos modelos como o modelo PUCSS visto na seção 1.1, que é NP-difícil (Bitran e Yanasse, 1982), várias pesquisas são desenvolvidas afim de se encontrar métodos que dêem boas soluções no menor tempo possível para estes modelos.

Métodos como Branch and Bound (Fleischman, 1990), Heurísticas (Araújo e Arenales, 2000), Planos de Corte (Constantino, 1996), são adaptados e utilizados sozinhos ou em algoritmos híbridos (Belvaux e Wolsey, 2000) no propósito de melhorar o processo de solução de problemas que ainda não são resolvidos em tempo polinomial. As próximas seções fazem uma breve descrição destes algoritmos.



2.2 Método de Planos de Corte

Considere um modelo de otimização linear inteiro (PI) definido por:

$$\max Z = cx$$

Sujeito à:

$$Ax \leq b$$

$$x \in Z_+^n$$

O conjunto de pontos $S = \{x \in Z_+^n : Ax \leq b\}$ é chamado de região factível do modelo.

Se substituirmos a restrição $x \in Z_+^n$ por $x \in \mathbb{R}_+^n$ no conjunto S , obtemos um modelo linear:

$$\max Z = cx$$

Sujeito à:

$$Ax \leq b$$

$$x \in \mathbb{R}_+^n$$

Este novo problema é chamado de Relaxação Linear de PI, (RL).

Definição 2.5: Um poliedro $P \subseteq \mathbb{R}^n$ é o conjunto de pontos que satisfaz um número finito de inequações lineares, isto é, $P = \{x \in \mathbb{R}_+^n : Ax \leq b\}$, onde (A, b) é uma matriz $m \times (n + 1)$ (Nemhauser e Wolsey, 1988).

Para um problema de otimização inteiro a região factível são pontos, isto é, $S \subset Z_+^n$ onde $S = \{x \in Z_+^n : Ax \leq b\}$.

Definição 2.6: Dado um conjunto $S \subseteq \mathbb{R}^n$, um ponto $x \in \mathbb{R}^n$ é uma *combinação convexa* de pontos de S se existir um conjunto finito de pontos $\{x^i\}_{i=1}^t$ em S e um $\lambda \in \mathbb{R}^n$ com $\sum_{i=1}^t \lambda_i = 1$ e $x = \sum_{i=1}^t \lambda_i x^i$. O envoltório convexo



de S , denotado por $\text{conv}(S)$, é o conjunto de todos pontos que são combinações convexas dos pontos de S (Nemhauser e Wolsey, 1988).

Assim observamos que, ao contrário dos modelos lineares a região factível dos modelos inteiros não constituem um conjunto convexo.

Exemplo 2.1

Problema Inteiro

$$\text{Max } z = x + y$$

$$\text{sujeito à: } x + y \leq 3$$

$$x + 3y \leq 6$$

$$x, y \geq 0 \text{ e inteiro.}$$

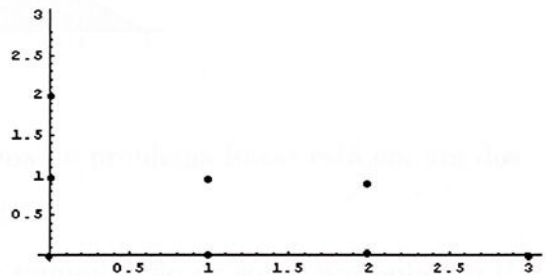


Fig. 2.1-Região Factível do Problema Inteiro

Problema Relaxado

$$\text{Max } z = x + y$$

$$\text{sujeito à: } x + y \leq 3$$

$$x + 3y \leq 6$$

$$y, x \geq 0$$

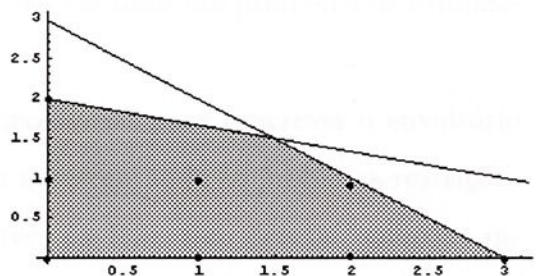


Fig. 2.2-Região Factível do Problema Relaxado

As inequações que definem o envoltório convexo deste problema são,
 $x \geq 0$, $y \geq 0$, $x + y \leq 3$ e $\frac{1}{2}x + y \leq 2$.

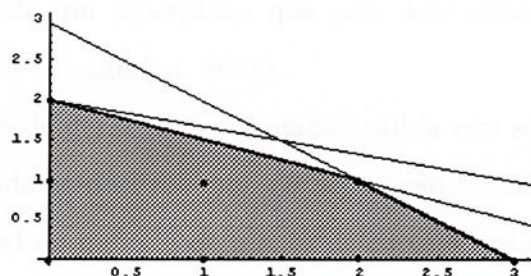


Fig. 2.3-Envoltório Convexo

Pelo teorema 2.1 abaixo, a solução ótima do problema linear está em um dos vértices (ponto extremo) da região factível.

Teorema 2.1: Considere o problema de minimização cx sobre um poliedro P . Suponha que P tem pelo menos um ponto extremo e que há uma solução ótima. Então, há uma solução ótima que é um ponto extremo de P (Bertsimas e Tsitsiklis, 1997).

Sendo assim se conhecermos as equações que formam o envoltório convexo de um problema inteiro poderemos resolvê-lo como se fosse um problema de otimização linear.

No entanto, o número de inequações necessárias para descrever o envoltório convexo pode ser alto. Uma alternativa é ir acrescentando ao problema restrições (inequações) que diminuam a região factível da R.L. sem "cortar" soluções inteiras, e ir se aproximando do envoltório convexo. Estas inequações são chamadas Inequações Válidas.

Definição 2.7: Uma inequação do tipo $a^T x \leq \alpha$, $a \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$, é chamada inequação válida em relação à S se $S \subseteq \{x \in \mathbb{R}^n / a^T x \leq \alpha\}$, ou seja, é uma inequação que é satisfeita por qualquer solução inteira factível (Ferreira e Wakabayashi, 1996).

Encontrar uma inequação válida implica encontrar um plano que separe um ponto infactível da região factível. O teorema 2.2 a seguir garante que isto é

sempre possível.

Teorema 2.2: Suponha P e $\tilde{P}$ sejam dois poliedros disjuntos e não vazios em $\mathbb{R}^n$. Então existe um hiperplano que gera dois semi-espacos, H e $\tilde{H}$, tais que $P \subset H$ e $\tilde{P} \subset \tilde{H}$ (Vanderbei, 2001).

O problema de gerar uma inequação válida que separa um ponto e um conjunto é conhecido como Problema de Separação.

A idéia geral do método de planos de corte é tratar o problema inteiro como um problema linear, ao qual são acrescentadas progressivamente inequações válidas (Planos de Corte) que cortam parte da região factível. Este corte é feito de forma a eliminar a solução ótima da relaxação linear, se esta for infactível para o problema inteiro.

A primeira aplicação deste método foi feita por Gomory na década de 50 para resolver problemas de programação linear inteira (PLI) (Wakabayashi e Ferreira, 1998). A idéia básica do método está descrita no algoritmo PC abaixo.

Algoritmo PC

Inicialização: *Resolva a relaxação linear do problema PLI.*

Passo 1: *Se a solução ótima for inteira pare, o problema está resolvido.*

Se for infactível pare pois o problema inteiro será infactível.

Se for fracionária passe para o passo 2.

Passo 2: *Gere uma inequação válida que elimina a solução ótima da relaxação linear. (Problema de Separação)*

Passo 3: *Acrescente o plano de Corte ao problema.*

Passo 4: *Reotimize o problema obtido no passo 3 e volte ao passo 1.*



No exemplo 2.1 a solução ótima da R.L. é $(\frac{3}{2}, \frac{3}{2})$. A inequação $\frac{1}{2}x + y \leq 2$ é um Plano de Corte que elimina este ponto ótimo não inteiro.

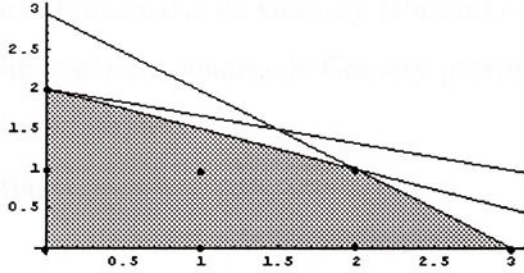


Fig. 2.4-Plano de Corte

Observe que este plano de corte faz parte da definição do envoltório convexo do problema e não está incluído no conjunto de restrições original.

As definições 2.8, 2.9 e 2.10 a seguir, ajudam a caracterizar melhor as inequações que definem o envoltório convexo de um PLI.

Definição 2.8: As inequações $\pi x \leq \pi_0$ e $\gamma x \leq \gamma_0$ são ditas equivalentes se $(\gamma, \gamma_0) = \lambda(\pi, \pi_0)$ para algum valor $\lambda > 0$. Se existir $\mu > 0$ tal que $\gamma \geq \mu\pi$ e $\gamma_0 \leq \mu\pi_0$, então $\{x \in \mathbb{R}_+^n : \gamma x \leq \gamma_0\} \subset \{x \in \mathbb{R}_+^n : \pi x \leq \pi_0\}$. Neste caso dizemos que $\gamma x \leq \gamma_0$ domina ou é mais forte que $\pi x \leq \pi_0$ ou que $\pi x \leq \pi_0$ é dominada por ou é mais fraca que $\gamma x \leq \gamma_0$.

Definição 2.9: Se (π, π_0) é uma inequação válida para P , e $F = \{x \in P : \pi x = \pi_0\}$, F é chamada uma face de P , e dizemos que (π, π_0) representa F . Uma face F é dita ser própria se $F \neq \emptyset$ e $F \neq P$ (Nemhauser e Wolsey, 1988).

Definição 2.10: Uma inequação válida maximal é uma inequação que não é dominada por nenhuma outra, ou seja, ela define uma face não vazia do $\text{conv}(S)$. (Nemhauser e Wolsey, 1988)

Nas subseções a seguir discutimos com mais detalhes procedimentos para a geração de planos de corte.

2.2.1 Corte Fracionário de Gomory

O Plano de Corte descrito a seguir foi proposto por Gomory na década de 50 e é chamado de Corte Fracionário de Gomory (Puccini e Pizzolato, 1987).

Para obter um corte fracionário de Gomory partimos da solução ótima da RL obtida pelo método Simplex.

Seja x_{Bi} variável básica escrita como;

$$x_{Bi} = \bar{b}_i - \sum_{j \in N} y_{ij} x_j, \quad i = 1, \dots, m. \quad (2.1)$$

Onde:

y_{ij} é o coeficiente das variáveis não básicas;

g_i é a parte fracionária de $\bar{b}_i$;

f_{ij} é a parte fracionária de y_{ij} ;

N conjunto dos índices das variáveis não básicas.

Considere que

$$\bar{b}_i = [\bar{b}_i] + g_i \text{ e } y_{ij} = [y_{ij}] + f_{ij} \quad (2.2)$$

Substituindo 2.2 na equação 2.1 resulta:

$$x_{Bi} = ([\bar{b}_i] - \sum_{j \in N} [y_{ij}] x_j) + (g_i - \sum_{j \in N} f_{ij} x_j). \quad (2.3)$$

Temos que $g_i < 1$, e que todos f_{ij} e x_j são maiores ou iguais a zero. Então $(g_i - \sum_{j \in N} f_{ij} x_j)$ é estritamente menor que 1. Queremos x_{Bi} inteira, então $(g_i - \sum_{j \in N} f_{ij} x_j)$ também deve ser inteira. Sendo este último termo menor que 1 e inteiro podemos supor:

$$(g_i - \sum_{j \in N} f_{ij} x_j) \leq 0. \quad (2.4)$$

Temos então:

$$-\sum_{j \in N} f_{ij} x_j \leq -g_i;$$

ou

$$\sum_{j \in N} f_{ij} x_j \geq g_i \quad (2.5)$$

A inequação 2.5 é chamada de Corte Fracionário de Gomory.

A seguir ilustramos o algoritmo PC (seção 2.4) onde no passo 2 é gerado um corte fracionário de Gomory e a reotimização do passo 4 é feita usando o método dual simplex.

Exemplo 2.3

$$\text{Max } z = 2x_1 + x_2$$

Sujeito á:

$$x_1 + x_2 \leq 5$$

$$-x_1 + x_2 \leq 0$$

$$6x_1 + 2x_2 \leq 21$$

$$x_1, x_2 \geq 0$$

A solução ótima da RL é:

$$x_2 = \frac{9}{4} - \frac{3}{2}x_3 + \frac{1}{4}x_5$$

$$x_4 = \frac{1}{2} + 2x_3 - \frac{1}{2}x_5$$

$$x_1 = \frac{11}{4} + \frac{1}{2}x_3 - \frac{1}{4}x_5$$

$$(\text{Max}) z = \frac{3}{4} - \frac{1}{2}x_3 - \frac{1}{4}x_5$$

Como x_1 , x_2 e x_4 não possuem valores inteiros podemos gerar Cortes de Gomory usando qualquer uma delas.

x_2 :

$$x_2 + \frac{3}{2}x_3 - \frac{1}{4}x_5 = \frac{9}{4}$$

$$\frac{3}{2} = \lfloor \frac{3}{2} \rfloor + f_{23} = 1 + \frac{1}{2}$$

$$-\frac{1}{4} = \lfloor -\frac{1}{4} \rfloor + f_{25} = -1 + \frac{3}{4}$$

$$\frac{9}{4} = \lfloor \frac{9}{4} \rfloor + \frac{1}{4} = 2 + \frac{1}{4}$$

$$\text{Corte gerado } \frac{1}{2}x_3 + \frac{3}{4}x_5 \geq \frac{1}{4}.$$

x_1 :

$$x_1 - \frac{1}{2}x_3 + \frac{1}{4}x_5 = \frac{11}{4}$$

$$-\frac{1}{2} = \lfloor -\frac{1}{2} \rfloor + f_{13} = -1 + \frac{1}{2}$$

$$\frac{1}{4} = \lfloor \frac{1}{4} \rfloor + f_{15} = 0 + \frac{1}{4}$$

$$\frac{11}{4} = \lfloor \frac{11}{4} \rfloor + \frac{3}{4} = 2 + \frac{3}{4}$$

Corte gerado $\frac{1}{2}x_3 + \frac{1}{4}x_5 \geq \frac{3}{4}$.

x_4 :

$$x_4 + \frac{1}{2}x_5 = \frac{1}{2}$$

$$1 = \lfloor 1 \rfloor + f_{44} = 1 + 0$$

$$\frac{1}{2} = \lfloor \frac{1}{2} \rfloor + f_{45} = 0 + \frac{1}{2}$$

Corte gerado $\frac{1}{2}x_5 \geq \frac{1}{2}$.

Vamos usar o plano de corte gerado pela variável básica x_1

$$\frac{1}{2}x_3 + \frac{1}{4}x_5 \geq \frac{3}{4}.$$

Coloquemos s_1 como variável de folga. $\frac{1}{2}x_3 + \frac{1}{4}x_5 - s_1 = \frac{3}{4}$, $s_1 \geq 0$

$$s_1 = \frac{1}{2}x_3 + \frac{1}{4}x_5 - \frac{3}{4}. \quad (2.6)$$

Acrescentamos a restrição 2.6 a RL do PI e usamos como nova variável básica.

A nova solução básica é primal infactível $s_1 < 0$. Como temos uma solução dual factível reotimizamos usando o dual-simplex. A nova solução ótima é:

$$x_2 = 0 + x_5 - 3s_1$$

$$x_4 = \frac{7}{2} - \frac{3}{2}x_5 + 4s_1$$

$$x_1 = \frac{7}{2} - \frac{1}{2}x_5 + s_1$$

$$x_3 = \frac{3}{2} - \frac{1}{2}x_5 - 2s_1$$

$$z = 7 - s_1$$

As linhas de x_1 , x_3 , x_4 geram o mesmo corte $x_5 \geq 1$. Acrescentando-o ao problema e reotimizando a solução ótima será:

$$x_2 = 1 - 3s_1 + 2s_2$$

$$x_4 = 2 + 4s_1 - 3s_2$$

$$x_1 = 3 + s_1 - s_2$$

$$x_3 = 1 + 2s_1 - s_2$$

$$x_5 = 1 + s_2$$

$$(Max)z = 7 - s_1$$

O problema inteiro está resolvido pois a solução da RL é inteira. A solução ótima do PI é então:

$$x_1^* = 3, x_2^* = 1, z^* = 7.$$

Observemos que se uma variável de folga/excesso s_i , associada ao plano de corte i reentrar na base o plano de corte i pode ser eliminado do problema, pois esta restrição é redundante. Este fato impõe um limite máximo no número de cortes de Gomory presentes no problema relaxado a cada iteração. No máximo n cortes são gerados, que é o número de variáveis. Qualquer combinação linear inteira das linhas com x_i fracionária, pode ser usado para gerar o plano de corte.

Observamos que os cortes fracionários de Gomory foram gerados a partir da solução ótima da relaxação linear. Um procedimento mais geral é gerar os planos de corte através do algoritmo Chvátal-Gomory (Algoritmo CG) que utiliza os dados da matriz de restrições descrito a seguir (Rangel, 1995).

Algoritmo CG

Passo 1: $\sum_{j \in N} (ua_j)x_j \leq ub$ para todo $u \geq 0$, $u \in \mathbb{R}_n$.

Passo 2: $\sum_{j \in N} \lfloor ua_j \rfloor x_j \leq ub$, desde $x \geq 0$

$$\text{implica } -\sum_{j \in N} ((ua_j) - \lfloor ua_j \rfloor)x_j \leq 0.$$

Passo 3: $\sum_{j \in N} \lfloor ua_j \rfloor x_j \leq \lfloor ub \rfloor$,

Chvátal desenvolveu este procedimento explicitamente em 1973. Porém Gomory em 1958-1963 o havia desenvolvido implicitamente (Rangel, 1995). Este procedimento foi utilizado por Vanderbeck e Wolsey (1992) para gerar inequações válidas para o modelo de Lasdon-Terjung, descrito na próxima seção.



2.2.2 Inequações Válidas para o Modelo de Lasdon-Terjung

Considere um problema onde as máquinas produzem a uma taxa b , um determinado item. A demanda deste item do período t , d_t , deve ser satisfeita pelo número de máquinas x_t trabalhando no período. Os custos envolvidos são o custo de estoque h_t e o custo de start-up f_t . Queremos determinar quantas máquinas estão ativas, quantas serão iniciadas no período t , z_t e qual será o estoque e_t ao final de cada período (Vanderbeck e Wolsey, 1992), (Lasdon, 1970).

Temos o seguinte modelo formulado:

$$\text{Min } z = \sum_{t=1}^T h_t e_t + \sum_{t=1}^T f_t z_t \quad (2.7)$$

Sujeito à:

$$e_{t-1} + b x_t = d_t + e_t, \quad t = 1, \dots, T; \quad (2.8)$$

$$z_t \geq x_t - x_{t-1}, \quad t = 1, \dots, T; \quad (2.9)$$

$$x_t, z_t, e_t \geq 0 \text{ e inteiras}, \quad t = 1, \dots, T; \quad (2.10)$$

$$e_0 = 0, \quad x_0 = 0, \quad e_T = k. \quad (2.11)$$

onde k é a quantidade de estoque desejada ao final do período de planejamento.

Proposição 2.3: Seja X a região factível do problema inteiro apresentado acima e $d_{p,q} = \sum_{t=p}^q d_t$. A inequação

$$e_t + \left\lceil \frac{b+1}{2} \right\rceil x_{t+1} + \left\lceil \frac{b-1}{2} \right\rceil x_{t+2} + z_{t+2} \geq \left\lceil \frac{d_{t+1,t+2}}{2} \right\rceil \quad (2.12)$$

é válida para o envoltório convexo de X , $\text{conv}(X)$ (Vanderbeck e Wolsey, 1992).

Prova:

A inequação 2.12 foi obtida através de uma iteração do algoritmo CG, considerando um subconjunto das restrições que definem o conjunto X .

Somando a equação (2.8), para os períodos $t+1$ e $t+2$ temos



$$\begin{array}{r} e_t + bx_{t+1} = d_{t+1} + e_{t+1} \\ + \\ e_{t+1} + bx_{t+2} = d_{t+2} + e_{t+2} \\ \hline e_t + e_{t+1} + b(x_{t+1} + x_{t+2}) = d_{t+1} + d_{t+2} + e_{t+1} + e_{t+2} \end{array}$$

Como $e_{t+2} \geq 0$ obtemos

$$e_t + b(x_{t+1} + x_{t+2}) \geq d_{t+1} + d_{t+2}. \quad (2.13)$$

Adicionando a inequação (2.9) para o período $t + 2$, $z_{t+2} \geq x_{t+2} - x_{t+1}$, a (2.13) e dividindo a inequação resultante por 2 obtemos

$$\left(\begin{array}{r} z_{t+2} \geq x_{t+2} - x_{t+1} \\ + \\ e_t + b(x_{t+1} + x_{t+2}) \geq d_{t+1} + d_{t+2} \end{array} \right) \frac{1}{2}$$

$$\frac{1}{2}e_t + \frac{b}{2}(x_{t+1} + x_{t+2}) + \frac{1}{2}z_{t+2} \geq \frac{d_{t+1} + d_{t+2}}{2} + \frac{x_{t+2}}{2} - \frac{x_{t+1}}{2}$$

Tomando o teto de todos os termos da inequação

$$\left\lceil \frac{1}{2} \right\rceil e_t + \left\lceil \frac{b+1}{2} \right\rceil x_{t+1} + \left\lceil \frac{b-1}{2} \right\rceil x_{t+2} + \left\lceil \frac{1}{2} \right\rceil z_{t+2} \geq \left\lceil \frac{d_{t+1} + d_{t+2}}{2} \right\rceil.$$

Obtemos

$$e_t + \left\lceil \frac{b+1}{2} \right\rceil x_{t+1} + \left\lceil \frac{b-1}{2} \right\rceil x_{t+2} + z_{t+2} \geq \left\lceil \frac{d_{t+1} + d_{t+2}}{2} \right\rceil.$$

◇

É possível provar ainda que sob certas condições a inequação válida 2.12 desenvolvida acima é uma faceta de $\text{conv}(X)$ (Vanderbeck e Wolsey, 1992).

Uma outra forma de gerar as inequações válidas para problemas PLI, é aproveitar a estrutura do modelo para desenvolvê-las. As Inequações Supermodulares discutidas a seguir são geradas a partir da estrutura do modelo

PUCSS,(seção 1.1).

2.2.3 Inequações Supermodulares

Considere o modelo de determinação de lote de um item capacitado com startup e setup (PUCSS) descrito na seção 1.1:



$$\text{Min } z = \sum_{t=1}^T (c_t x_t + c_t^s y_t + c_t^z + h_t e_t) \quad (2.14)$$

Sujeito à:

$$x_t + e_{t-1} = d_t + e_t, \quad t = 1, \dots, T. \quad (2.15)$$

$$x_t \leq C_t y_t, \quad t = 1, \dots, T. \quad (2.16)$$

$$z_t \geq y_t - y_{t-1}, \quad t = 1, \dots, T. \quad (2.17)$$

$$z_t \leq y_t, \quad t = 1, \dots, T; \quad (2.18)$$

$$z_t \leq 1 - y_{t-1}, \quad t = 1, \dots, T; \quad (2.19)$$

$$x_t \geq 0, \quad e_t \geq 0, \quad y_t \text{ e } z_t = 0/1, \quad t = 1, \dots, T. \quad (2.20)$$

Nesta subseção iremos descrever uma classe de inequações válidas, inequações supermodulares à esquerda, para o conjunto de pontos da região factível de PUCSS, que será denotado por X .

A idéia que motiva a definição destas inequações é o objetivo de encontrar um limite inferior $LB(x, y, z)$ para o estoque e_t do período t dependente de x, y, z .

As inequações Supermodulares à esquerda tem este nome porque as funções utilizadas para encontrar o limite são supermodulares¹ e o período ao qual x, y, z pertencem são períodos posteriores à e_j . Ou seja, faremos uma estimativa para o valor do estoque no período t em função da produção futura do item.

O papel do estoque é ajudar a suprir a demanda quando a produção não é suficiente. O estoque e_{k-1} de um período $(k-1)$ pode ser usado para suprir a demanda de vários períodos posteriores à $(k-1)$. Para isto é necessário que:

$$e_{k-1} \geq ((d_k - x_k) + (d_{k+1} - x_{k+1}) + \dots + (d_j - x_j)) \quad (2.21)$$

¹Definição: (Nemhauser e Wolsey, 1998) Suponha que N seja um conjunto finito e f seja uma função de valor real sobre um subconjunto de N . f é supermodular se

$-f(S) + (-f(U)) \geq (-f(S \cup U)) + (-f(S \cap U))$ para $S, U \subseteq N$.



Neste caso o produto estocado no período $(k-1)$ pode suprir, juntamente com a produção, a demanda dos períodos k até j .

Observando a inequação 2.16, vemos que se não ocorre setup no período t , a produção x_t é nula. Assim, podemos desconsiderar a produção nos períodos onde não ocorrem setup na expressão 2.21. Denotaremos este conjunto de períodos por S , onde $S \subseteq \{k, \dots, T\}$ para $k \in N = \{1, \dots, T\}$.

Considerando que a produção nos períodos $\{k, \dots, j\}$ é no máximo $\sum_{t \in \{k, \dots, j\} \setminus S} C_t$ para cada $j \in \{k, \dots, T\}$, o valor do estoque no período anterior, e_{k-1} , tem que ser maior ou igual à:

$$f_k(S) = \max_{j=k, \dots, T} \left\{ \left(D_{kj} - \sum_{t \in \{k, \dots, j\} \setminus S} C_t \right)^+ \right\},$$

onde D_{kj} é a soma das demandas d_t para $t \in \{k, \dots, j\}$.

Isto é, $f_k(S)$ é um limite inferior para e_{k-1} .

Abaixo enunciamos um lema que garante que se ocorre setup em algum período o valor de $f_k(S)$ será transformado em valor negativo ou nulo e o estoque não será limitado, $e_t \geq 0$.

Lema 2.1 Suponha $k \in N$, $S \subseteq \{k, \dots, T\}$, $u = \min\{t : t \in S\}$,

$v = \max\{t : t \in S\}$, o menor e o maior elemento do conjunto S .

Se $y_u + Z_{u+1,v} < 1$ então nenhum set-up ocorre em S . Onde $Z_{u+1,v} = \sum_{i=u+1}^v z_i$ e 0 se $u+1 > v$.

Demonstração:

$$y_j = y_u + y_{u+1} - y_u + y_{u+2} - y_{u+1} + \dots + y_j.$$

Considerando $z_i \geq y_i - y_{i-1}$ (restrição 2.17) podemos escrever,

$$y_u + y_{u+1} - y_u + y_{u+2} - y_{u+1} + \dots + y_j \leq y_u + \sum_{i=u+1}^j z_i.$$

Tendo que v é o maior elemento do conjunto S ,

$$y_u + \sum_{i=u+1}^j z_i \leq y_u + Z_{u+1,v} < 1.$$

Sendo y_j binária resulta que $y_j = 0$, ou seja, não ocorre setup em S .

◇

Assim, a função $f_k(s)$ e o lema 2.1 acima garantem que a inequação:



$$e_{k-1} \geq f_k(S)(1 - y_u - Z_{u+1,v}). \quad (2.22)$$

é válida para o modelo PUCSS.

Acompanhando o exemplo abaixo, retirado de (Constantino, 1996), percebemos que é possível obter inequações válidas mais fortes do que (3.21) se fizermos uma decomposição do valor de $f_k(S)$.

Exemplo 2.4 Considere um exemplar do problema de um único item PUCSS, cuja demanda, capacidade e estoque são na tabela 2.1.

Tabela 2.1: Dados de um exemplar do modelo PUCSS				
	período 2	período 3	período 4	período 5
capacidade	3	11	8	7
estoque	2	3	4	5
demanda	1	8	2	6

Suponhamos $k=2$ e que nenhum set-up ocorre no intervalo $[u,v]$, $2 \leq u \leq v \leq 5$.

Calculando f_2 para os possíveis conjuntos S , obtemos:

$$f_2\{(2,2)\} = 1, f_2\{(2,3)\} = 9, f_2\{(2,4)\} = 11, f_2\{(2,5)\} = 17, f_2\{(3,3)\} = 6, \\ f_2\{(3,4)\} = 8, f_2\{(3,5)\} = 14, f_2\{(4,4)\} = 0, f_2\{(4,5)\} = 3, f_2\{(5,5)\} = 0.$$

Tomemos $f_2\{(2,3)\}$, como exemplo, $f_2\{(2,3)\} = 9$ é limite inferior para e_1 .

$$e_1 \geq f_2\{(2,3)\} = 9 \quad (2.23)$$

Mas, $f_2\{(2,3)\} = 9$ pode ser decomposto como $1+6+9-1-6$.

Se observarmos, o primeiro termo da decomposição, 1, é limite inferior para e_1 se nenhum set-up ocorre no período 2; o segundo se nenhum set-up ocorre no período 3; e o terceiro se nenhum set-up ocorre nos períodos 2 e 3. Esta decomposição resulta na inequação:

$$1(1 - y_2) + 6(1 - y_3) + 2(1 - y_2 - z_3) \leq 9(1 - y_2 - z_3) \leq e_k. \quad (2.24)$$

Que é mais forte que (2.23) (ver definição 2.8 seção 2).



De uma maneira geral, queremos então decompor $f_k(S)$. Para obter esta decomposição, defina $\beta_{uv} \in \mathbb{R}$ para $k \leq u \leq v \leq T$ como:

$$\beta_{uv} = f_k([u, v]) + f_k([u + 1, v - 1]) - f_k([u + 1, v]) - f_k([u, v - 1]).$$

Assim, a expressão geral para (2.24) é:

$$e_k \geq \beta_{uv}(1 - y_u - Z_{u+1,v}).$$

Podemos definir f_k também em função de valores não negativos de β .

Lema 2.2: Se $\beta_{u,v} = f_k([u, v]) + f_k([u + 1, v - 1]) - f_k([u + 1, v]) - f_k([u, v - 1])$, então

- (i) $\beta_{u,v} \geq 0$, para $k \leq u \leq v \leq n$;
- (ii) $f_k([u, v]) = \sum_{u \leq \tilde{u} \leq \tilde{v} \leq v} \beta_{\tilde{u}, \tilde{v}}$, para $k \leq u \leq v \leq n$;

Teorema 2.3:(Constantino,1996) (Inequações Supermodulares á Esquerda)

Dado $k \in N$ e $\beta_{uv} = f_k([u, v]) + f_k([u + 1, v - 1]) - f_k([u + 1, v]) - f_k([u, v - 1])$, para $k \leq u \leq v \leq T$, suponha que F seja um conjunto de pares (u, v) tal que $F \subseteq \{(u, v) : k \leq u \leq v \leq T\}$.

Então a seguinte inequação é válida para X :

$$e_{k-1} \geq \sum_{(u,v) \in F} \beta_{u,v}(1 - y_u - Z_{u+1,v}). \quad (2.25)$$

Demonstração: Devemos demonstrar que qualquer ponto da região factível X satisfaz a inequação 2.24.

Tomemos um ponto $(x, y, z, e) \in X$ e $S = \{j \in \{k, \dots, n\} : y_j = 0\}$ é o conjunto dos períodos onde não ocorre setup.

Se $S = \emptyset$, ou seja, ocorre setup em todos os períodos, $y_j = 1$ para $k \leq j \leq n$, e sendo $\beta_{uv} \geq 0 \forall u, v$, o valor $\sum_{(u,v) \in F} \beta_{u,v}(1 - 1 - Z_{u+1,v})$ é negativo. Temos então:

$$e_{k-1} \geq 0 \geq - \sum_{(u,v) \in F} \beta_{u,v} Z_{u+1,v}.$$



Suponhamos $S \neq \emptyset$. Decompomos S em p subconjuntos disjuntos tal que $v_{i-1}+1 < u_i$ para $i=2, \dots, p$. Se um par (u, v) não satisfizer a relação $u_i \leq u \leq v \leq v_i$ para algum i então este par (u, v) não está contido em S e pelo menos um setup ocorre neste intervalo $\{u, u+1, \dots, v\}$. Então $1 - y_u - Z_{u+1,v} \leq 0$ pois, se fosse maior que zero pelo lema 2.1 não ocorreria setup neste intervalo. Daí,

$$\sum_{(u,v) \in F} \beta_{u,v}(1 - 1 - Z_{u+1,v}) \leq \sum_{i=1}^p \sum_{(u,v) \in F, u_i \leq u \leq v \leq v_i} \beta_{u,v}(1 - 1 - Z_{u+1,v})$$

Observe que no segundo membro tomamos $(u, v) \in F$ e tal que $u_i \leq u \leq v \leq v_i$. Tirando $(-Z_{u+1,v})$ temos

$$\begin{aligned} \sum_{i=1}^p \sum_{(u,v) \in F, u_i \leq u \leq v \leq v_i} \beta_{u,v}(-Z_{u+1,v}) &\leq \sum_{i=1}^p \sum_{(u,v) \in F, u_i \leq u \leq v \leq v_i} \beta_{u,v} \leq \\ &\leq \sum_{i=1}^p \sum_{u_i \leq u \leq v \leq v_i} \beta_{u,v}. \end{aligned}$$

Pelo lema 2.2 ii podemos substituir a soma $\sum_{u_i \leq u \leq v \leq v_i} \beta_{u,v}$ por $\sum_{i=1}^p f_k([u_i, v_i])$.

Sendo f_k supermodular, a interseção dos conjuntos vazia, e $f_k(\emptyset) = 0$ temos:

$$\sum_{i=1}^p f_k([u_i, v_i]) \leq f_k(\cup_{i=1}^p [u_i, v_i]) = f_k(S) \leq e_{k-1}.$$

◇

Até aqui construímos inequações válidas para o modelo PUCSS. Para utilizá-las como planos de corte o primeiro passo é resolver o problema de separação, ou seja, encontrar uma inequação que seja violada pelo ponto ótimo $(x, y, z, e) \in \mathbb{R}^{4T+1}$. Mais ainda, queremos que a diferença,

$$\sum_{(u,v) \in F} \beta_{u,v} \rho_{uv} - e_{k-1} \tag{2.26}$$

onde $\rho_{uv} = (1 - y_u - Z_{u+1,v})$, seja a maior possível.

O que determina tal inequação é k , período para o qual a inequação será gerada, e o conjunto F de intervalos. Os valores β_{uv} dependem de k e para cada $k \in N$ há um conjunto F que maximiza $\sum_{(u,v) \in F} \beta_{u,v} \rho_{uv}$.



O fato de f_k ser supermodular resulta, $\beta_{uv} \geq 0$. Pois se tomarmos $S=(u+1, v)$ e $U=(v, v-1)$ na definição de função supermodular podemos comprovar que

$$f_k(u, v) + f_k(u+1, v-1) \geq f_k(u+1, v) + f_k(u, v-1) \text{ e assim}$$

$$\beta_{uv} = f_k([u, v]) + f_k([u+1, v-1]) - f_k([u+1, v]) - f_k([u, v-1]) \geq 0.$$

Sabendo que $\beta_{uv} \geq 0$, F maximiza a expressão 2.26 se e somente se:

$$\{(u, v) : k \leq u \leq v \leq T \text{ e } \beta_{u,v} \rho_{uv} > 0\} \subseteq F \text{ e}$$

$$F \subseteq \{(u, v) : k \leq u \leq v \leq T \text{ e } \rho_{uv} \geq 0\}$$

Ou seja, F contém todos os pares tal que $\beta_{uv} \rho_{uv} > 0$ e ainda pode conter pares onde $\beta_{uv} \rho_{uv} = 0$.

Há necessidade de após gerar F , verificar se a inequação foi violada, já que podem ocorrer casos onde a expressão 2.26 não seja positiva.

Denotaremos $\gamma_{uv} = D_{kv} - C_{k,u-1}$, para $k \leq u, v \leq n$.

Para relacionar γ_{uv} com β_{uv} observemos que

$$\max_{j=k, \dots, T} \{(D_{kj} - \sum_{t \in \{k, \dots, j\} \setminus \{u, \dots, v\}} C_t)\} = D_{kv} - C_{k,u-1} \text{ para } k \leq u \leq v \leq n, \text{ e}$$

$$\gamma_{uv}^+ = f_k(\{u, \dots, v\}).$$

Desde que $d_j \leq C_j$, $\gamma_{uv} < 0$ para $u > v$ então $\gamma_{uv}^+ = 0$ para $u > v$.

Pode-se calcular γ_{uv} para $k \leq u \leq v \leq n$ recursivamente fazendo:

$$\gamma_{k,k-1} = 0;$$

$$\gamma_{uv} = \gamma_{u,v-1} + d_v;$$

$$\gamma_{uv} = \gamma_{u-1,v} - C_{u-1}.$$

O seguinte lema dá uma condição para β_{uv} ser positivo.

Lema 2.3:(Constantino, 1996) Suponha que γ_{uv} seja definido como $D_{kv} - C_{k,u-1}$. Se $\beta_{uv} > 0$, então $\gamma_{uv}^+ > 0$ e $\gamma_{u+1,v-1}^+ = 0$.

Para computar $\rho_{uv} = 1 - y_u - Z_{u+1,v}$ usaremos as definições:

$$\rho_{k,k-1} = 1 - y_k + z;$$



$$\rho_{uv} = \rho_{u,v-1} - Z_v;$$

$$\rho_{uv} = \rho_{u-1,v} + z_u - y_u + y_{u-1}.$$

Definido o período k para o qual será gerada a inequação, e conhecido o ponto (x, y, z, e) solução da RL, o algoritmo de separação pode ser definido como o algoritmo IISE descrito a seguir.

Algoritmo para Identificação de Inequações Supermodulares à Esquerda (IISE), (Constantino, 1996)

Inicialização: $F = \emptyset$; $(u, v) = (k, k-1)$; $\gamma_{uv} = 0$; $\rho_{uv} = 1 - y_u - z_{v+1}$;

Iteração principal: Para $(v < n)$ faça:

passo 1: $v = v + 1$;

passo 2: $\gamma_{uv} = \gamma_{u,v-1} + d_v$;

passo 3: $\rho_{uv} = \rho_{u,v-1} - z_v$;

passo 4: se $(\rho_{uv} > 0)$ $F = F \cup \{(u, v)\}$;

passo 5: enquanto $((u < v)$ e $\gamma_{uv} - C_u > 0)$ faça

a) $u = u + 1$

b) $\gamma_{uv} = \gamma_{u-1,v} - C_{u-1}$;

c) $\rho_{uv} = \rho_{u-1,v} + z_u - y_u + y_{u-1}$;

d) se $(\rho_{uv} > 0)$ $F = F \cup \{(u, v)\}$;

passo 6: Verifica se a inequação obtida é violada pela solução da RL.

Nos passos 4 e 5 d) calculamos ρ_{uv} e escolhemos os pares (u, v) tal que $\rho_{uv} > 0$.

Assim, garantimos $\beta_{uv} \rho_{uv} > 0 \subseteq F$.

Como d_v é sempre não negativo e $\gamma_{u,v-1} > 0$, no passo 2 teremos $\gamma_{uv} \geq 0$. O passo 5 garante que mesmo incrementando u teremos $\gamma_{uv} > 0$.

A seguir mostraremos um exemplo de aplicação do algoritmo para uma instância do problema PUCSS.

Exemplo 2.5 Tomemos $k=2$, $T=6$;

Capacidade = $\{3, 2, 5, 7, 8, 11\}$;



$$\text{Demanda} = \{2, 0, 3, 4, 5, 9\};$$

$$y = \{3, 0, 0, 0.625, 0.625, 0.5455\};$$

$$z = \{3, 0, 0, 0.625, 0, 0\};$$

$$e = \{7, 7, 4, 0, 0, 0\};$$

Inicialização: $F = \emptyset$, $(u, v) = (2, 1)$, $\gamma_{2,1} = 0$, $\rho_{2,1} = 1 - y_2 + z_2 = 1 - 0 + 0 = 1$.

Enquanto $(v < 6)$

(1 < 6)

Passo: 1) $v = 1 + 1$

Passo : 2) $\gamma_{2,2} = \gamma_{2,1} + d_2 = 0$

Passo : 3) $\rho_{2,2} = \rho_{2,1} - z_2 = 1$

Passo : 4) $\rho > 0 \Rightarrow F = \{(2, 2)\}$

Passo : 5) $(2 \not< 2)$. Volto para o iteração principal.

(2 < 6)

Passo: 1) $v = 2 + 1$

Passo : 2) $\gamma_{2,3} = \gamma_{2,2} + d_3 = 3$

Passo : 3) $\rho_{2,3} = \rho_{2,2} - z_3 = 1$

Passo : 4) $\rho_{2,3} > 0 \Rightarrow F = F \cup \{(2, 3)\}$

Passo : 5) $((2 < 3) \wedge \gamma_{2,3} - C_2 = 1 > 0)$

a) $u = 2 + 1$

b) $\gamma_{3,3} = \gamma_{2,3} - C_2 = 1$

c) $\rho_{3,3} = \rho_{2,3} + z_3 - y_3 + y_2 = 1$

d) $\rho_{3,3} > 0 \Rightarrow F = F \cup \{(3, 3)\}$

Passo : 5) $(3 \not< 3)$. Volto para o iteração principal.

(3<6)

Passo: 1) $v=3+1$ Passo : 2) $\gamma_{3,4} = \gamma_{3,3} + d_4 = 5$ Passo : 3) $\rho_{3,4} = \rho_{3,3} - z_4 = 0.375$ Passo : 4) $\rho_{3,4} > 0 \Rightarrow F = F \cup \{(3,4)\}$ Passo : 5) $((3 < 4)e\gamma_{3,4} - C_3 = 0)$. Volto para o iteração principal.

(4<6)

Passo: 1) $v=4+1$ Passo : 2) $\gamma_{3,5} = \gamma_{3,4} + d_5 = 10$ Passo : 3) $\rho_{3,5} = \rho_{3,4} - z_5 = 0.375$ Passo : 4) $\rho_{3,5} > 0 \Rightarrow F = F \cup \{(3,5)\}$ Passo : 5) $((3 < 5)e\gamma_{3,5} - C_3 = 5 > 0)$ a) $u=3+1$ b) $\gamma_{4,5} = \gamma_{3,5} - C_3 = 5$ c) $\rho_{4,5} = \rho_{3,5} + z_4 - y_4 + y_3 = 0.375$ d) $\rho_{4,5} > 0 \Rightarrow F = F \cup \{(4,5)\}$ Passo : 5) $4(4 < 5)e\gamma_{4,5} - C_4 = -2 < 0$. Volto para o iteração principal.

(5<6)

Passo: 1) $v=5+1$ Passo : 2) $\gamma_{4,6} = \gamma_{4,5} + d_6 = 14$ Passo : 3) $\rho_{4,6} = \rho_{4,5} - z_6 = 0.375$ Passo : 4) $\rho_{4,6} > 0 \Rightarrow F = F \cup \{(4,6)\}$ Passo : 5) $((4 < 6)e\gamma_{4,6} - C_4 = 7 > 0)$ a) $u=4+1$ 

$$b) \gamma_{5,6} = \gamma_{4,6} - C_4 = 7$$

$$c) \rho_{5,6} = \rho_{4,6} + z_5 - y_5 + y_4 = 0.375$$

$$d) \rho_{5,6} > 0 \Rightarrow F = F \cup \{(5, 6)\}$$

Passo : 5) $(5 < 6) e \gamma_{5,6} - C_5 = -1 < 0$. Volto para o iteração principal.

$(6 \not< 6)$.

Encontramos $F = \{ (2, 2); (2, 3); (3, 3); (3, 4); (3, 5); (4, 5); (4, 6); (5, 6) \}$

Os valores de β_{uv} associados são dados abaixo.

$$\beta_{22}=0; \beta_{23}=2; \beta_{33}=1; \beta_{34}=4;$$

$$\beta_{35}=0; \beta_{45}=5; \beta_{46}=2; \beta_{56}=7;$$

A inequação gerada é então,

$$e_1 \geq 0(1 - y_2) + 2(1 - y_2 - z_3) + 1(1 - y_3) + 4(1 - y_3 - z_4) + \\ 0(1 - y_3 - z_4 - z_5) + 5(1 - y_4 - z_5) + 2(1 - y_4 - z_5 - z_6) + 7(1 - y_5 - z_6)$$

Passo : 6) Vejamos se este conjunto F viola a inequação gerada.

$$\sum_{(u,v) \in F} \beta_{u,v}(1 - y_u - Z_{u+1,v}) = 9,75.$$

A inequação é violada pois,

$$e_{k-1} = e_1 = 7 \not\geq 9,75 = \sum_{(u,v) \in F} \beta_{u,v}(1 - y_u - Z_{u+1,v}).$$

E pode portanto ser incluída ao modelo diminuindo então a região factível.

•

É importante observar que se a inequação não fosse violada calcularíamos um novo conjunto F escolhendo outro valor para k .

No capítulo 4 discutimos a eficiência destas inequações na solução de um modelo de determinação de lotes de uma fábrica de refrigerantes.

2.3 Algoritmo Branch and Bound (B&B)

O algoritmo Branch and Bound original deve-se a Lang e Doig em 1960 e está na classe dos métodos de enumeração implícita (Rangel, 1995). O termo branch refere-se ao fato de que o método efetua partições no espaço das soluções. O termo bound ressalta que a prova da otimalidade da solução utiliza limites calculados ao longo da enumeração (Goldbarg e Luna, 2000). Assim, a idéia básica do método é procurar a solução ótima particionando o conjunto de soluções possíveis do problema de programação linear inteiro (PPLI) em subconjuntos (subproblemas) mais restritos. Estes subproblemas por sua vez, fornecerão limites que servirão para avaliar a necessidade de novas partições.

Este processo pode ser representado através de um esquema de árvore de decisão.

A primeira partição é feita a partir da solução da relaxação do problema inicial.

A solução da RL é um limite superior para o valor da função objetivo (FO) do PLI, considerando que este seja um problema de maximização. Se a solução não for inteira escolhemos uma das variáveis fracionárias x_k , e construímos dois novos problemas de PL pela introdução das restrições:

$x_k \leq \lfloor x_k \rfloor$ e $x_k \geq \lfloor x_k \rfloor + 1$, onde $\lfloor x_k \rfloor$ é o maior valor inteiro menor ou igual a x_k .

Resolvemos estes dois subproblemas e se necessário repetimos o processo. Podemos descrever o algoritmo numa iteração t do método, para um problema de maximização, através dos seguintes passos:



Algoritmo B&B

Inicialmente precisamos de um limite inferior para z_{pi} , por exemplo $z^* = -\infty$.

Chamemos de lista mestra (LM) a lista dos problemas.

Passo 1: Se a lista mestra está vazia pare. Se não selecione um subproblema P'_p e retire da lista.

Passo 2: Resolva a relaxação linear de P'_p . Se for infactível ou $z_p \leq z^*$, (onde z_p é a solução ótima da RL de P'_p), P'_p está sondado, volte ao passo 1. Se não vá para o passo 3.

Passo 3: Se a solução da RL de P'_p é inteira, isto é, satisfaz as restrições de inteiros, guarde a solução e atualiza o limite inferior; $z^* = z_p$. P'_p está sondado. Volte ao passo 1. Se não prossiga para o passo 4.

Passo 4: Escolha x_j , que não tenha valor inteiro na solução ótima de P'_p . Seja $x_j = \hat{b}_i$.

Crie dois novos subproblemas:

O primeiro é obtido acrescentando à P'_p a restrição $x_j \leq \lfloor \hat{b}_i \rfloor$.

O segundo acrescentamos $x_j \geq \lfloor \hat{b}_i \rfloor + 1$.

Acrescente estes dois novos subproblemas à lista mestra e volte ao passo 1.

Vamos ilustrar o algoritmo 1 através do seguinte exemplo.

Exemplo 2.2

$$\text{Max } z = 4x_1 + x_2 \quad (2.27)$$

Sujeito à:

$$7x_1 + 4x_2 \leq 13 \quad (2.28)$$

$$x_1, x_2 \geq 0, \text{ e inteiro.} \quad (2.29)$$

Inicialização: Temos $LM = \{P1\}$ limite inferior $z = -\infty$.

Iteração 1:

Passo 1: Seleccionamos $P1$.

Passo 2: Resolvemos a RL de $P1$. A solução da RL é



$$z_1 = 7.43, x_1 = \frac{13}{7}, x_2 = 0.$$

Passo 3: A solução não é inteira.

Passo 4: Escolhemos x_1 para particionar e acrescentamos à lista mestra os subproblemas P2 com a restrição $x_1 \leq 1$ e P3 com $x_1 \geq 2$.

Iteração 2:

Passo 1: Temos $LM = \{P2, P3\}$. Seleccionemos P3.

Passo 2: Resolvendo P3 vemos que é infactível.

Iteração 3:

Passo 1: Temos $LM = \{P2\}$. Seleccionemos P2.

Passo 2: Resolvendo a RL de P2 obtemos a solução:

$$z_2 = 5.5 < z^*, x_1 = 1, x_2 = 1.5.$$

Passo 3: A solução é fracionária.

Passo 4: Escolhemos x_2 para particionar e geramos então dois novos subproblemas P4 e P5 com $x_2 \leq 1$, $x_2 \geq 2$, respectivamente e os colocamos na lista mestra.

Iteração 4:

Passo 1: Temos $LM = \{P4, P5\}$. Seleccionemos P4.

Passo 2: Resolvendo a RL de P4 obtemos:

$$x_1 = x_2 = 1, z_4 = 5$$

Passo 3: A solução é inteira e portanto será guardada. O limite inferior deve ser atualizado $z^* = 5 = z_4$

Iteração 5:

Passo 1: $LM = \{P5\}$.

Passo 2: Resolvendo a RL de P5 obtemos:

$$x_1 = 0.71, x_2 = 2, z_5 = 4.84.$$

O subproblema é factível porém $z_5 = 4.84 < 5 = z^*$. O que significa que não obteremos solução melhor se prosseguirmos particionando então paramos.



Iteração 6:

Passo 1: A LM está vazia. Paramos o algoritmo.

-

Analisando as soluções obtidas vemos que o subproblema P4 fornece a solução ótima:

$$x_1 = x_2 = 1 \quad z_4 = 5$$

A representação da árvore de busca do modelo é dado na figura 2.6 abaixo:

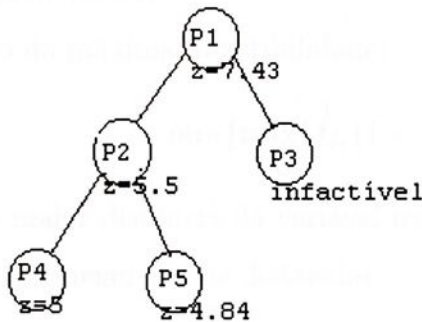


Fig. 2.6-Árvore de busca

A eficiência do método Branch and Bound depende da qualidade de seus limites, do critério de escolha das variáveis a serem particionadas, e do critério de seleção de subproblemas da LM. Há na literatura por exemplo (Nemhauser e Wolsey, 1988) vários critérios para estas escolhas.

Limitantes

- *Relaxação Linear (Nemhauser e Wolsey, 1988)*
- *Relaxação Lagrangeana (Fisher, 1981)*
- *Heurísticas (Reeves, 1993)*

Escolha da variável para partição (Nemhauser e Wolsey, 1988)• *Máxima infactibilidade*

Seja f_j a parte fracionária da variável x_j . O critério de escolha de variável de máxima infactibilidade escolhe a variável tal que:

$$\max_j \{\min\{f_j, (1 - f_j)\}\}$$

Ou seja, guarda a menor distância de x_j a um inteiro e escolhe a variável que tiver a maior destas distâncias.

• *Mínima infactibilidade*

Faz o inverso da máxima infactibilidade:

$$\min_j \{\max\{f_j, (1 - f_j)\}\}$$

Armazena a maior distância da variável fracionária a um inteiro e escolhe a variável que tiver a menor destas distâncias.

• *Ramificação forte (Strong Branch)*

Fixa as variáveis candidatas (fracionária) com valores diferentes. A que resultar melhor valor para o subproblema resolvido é escolhida.

Escolha do subproblema (Nemhauser e Wolsey, 1988)• *Busca em profundidade*

A busca em profundidade opta pelo subproblema que acabou de ser inserido na lista mestra, ou seja, se o nó resolvido não for podado o próximo nó escolhido será um de seus filhos. Percebemos então que a busca segue se aprofundando em um ramo na árvore.



A representação em árvore deste tipo de busca é do tipo:

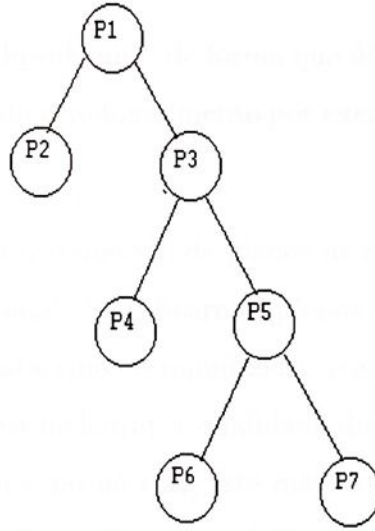


Fig. 2.7-Árvore de busca em profundidade

• *Busca em largura*

A busca em largura caminha na árvore resolvendo todos os nós de um nível para depois passar para o próximo. Podemos definir nível como sendo a menor distância, número de nós, do nó corrente ao nó raiz.

A árvore de busca neste caso é:

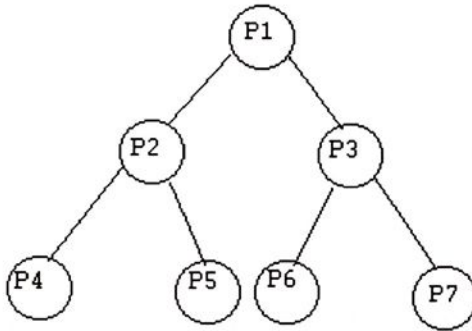


Fig. 2.8-Árvore de busca em largura

• *Melhor limite superior*

Supondo que juntamente com o subproblema guardemos a solução de sua relaxação, o próximo nó escolhido seria o que tem maior valor na relaxação (melhor

limite superior).

- *Melhor estimativa*

Pode se resolver os subproblemas de forma que dêem uma solução, não necessariamente factível, fazendo arredondamento por exemplo, e o que obtiver melhor valor é escolhido.

É possível ainda utilizar o método de planos de corte para acelerar a solução do método Branch and Bound. Se aplicarmos planos de corte em um nó da árvore do Branch and Bound, estaremos diminuindo a região factível do subproblema associado e assim podemos melhorar a qualidade do limite obtido. Se os planos de corte são gerados apenas no nó raiz, este método é conhecido como Cut and Branch. Temos ainda o método Branch and Cut (Crowder, Johnson e Padberg, 1983) (Padberg e Rinaldi, 1991) se geramos os planos de corte em outros nós da árvore.

Tendo em vista que as heurísticas podem fornecer limitantes para o método Branch and Bound, na próxima seção fazemos uma breve discussão sobre o tema.



2.4 Métodos Heurísticos

Métodos exatos para solução de problemas de programação linear inteira mista, nem sempre fornecem a solução ótima em tempo hábil, seja por causa de sua estrutura ou por sua dimensão. Muitas vezes, o que se busca é obter soluções factíveis para estes problemas em tempo razoável. Para este fim são aplicados os métodos heurísticos, que visam fornecer soluções factíveis boas, muitas vezes até bem próximas da solução ótima.

O termo heurística é derivado do grego *heuriskein*, que significa descobrir ou achar [Reeves, 1993]. Em otimização podemos definir uma heurística como sendo um método de solução de problemas de otimização que visa encontrar, com tempo e custo razoáveis, uma solução para o problemas, sendo que não há garantia de otimalidade para solução encontrada.

Uma possível classificação das heurísticas é (Goldbarg e Luna, 2000):

Heurísticas Clássicas:

Míopes ou Gulosas, Busca local, Particionamento.

Heurísticas Estocásticas:

Busca Tabu (Tabu Search), Simulated Annealing, GRASP.

Heurísticas Analógicas:

Redes Neurais, Computação Evolutiva.

O que justifica o uso de tais métodos é a rapidez de solução. A classe de modelos NP-difíceis, (seção 2.1), motiva cada vez mais o estudo de heurísticas. Modelos como PMCSS, que é NP-difícil (Bitran e Yanasse, 1982) tem sido resolvido com o uso de heurísticas (Araújo e Arenales, 2000).

Também encontramos aplicações das heurísticas Busca Tabu (Gopalakrishnan e outros, 2001), Simulated Annealing (Özdamar e Ali Bozyel, 2000) na solução de modelo de dimensionamento de lote com tempos de set up.

Além disso, métodos heurísticos podem ser utilizados para resolver o problema de separação em algoritmos de planos de corte. Vanderbeck e Wolsey (1992)



fazem uso de um algoritmo guloso na geração de uma família de inequações válidas para o modelo de produção de Lasdon-Terjung. As heurísticas também podem ser úteis na redução do número de nós da árvore do Branch and Bound (Constantino, 1996).

No capítulo 4 apresentamos resultados da solução de um modelo de determinação de lotes, onde os métodos Branch and Bound, Planos de Corte e Heurísticas são utilizados de maneira híbrida.

Capítulo 3

Modelo de Planejamento da Produção para uma Fábrica de Refrigerantes (FR)

O estudo de problemas reais é muito importante tanto para a solução do problema em si quanto para o amadurecimento da formulação de modelos matemáticos. Além disso, colabora para o enriquecimento da literatura sobre o assunto.

Neste capítulo, descrevemos as linhas de produção de uma fábrica de refrigerantes na região de São José do Rio Preto e propomos um modelo matemático para o dimensionamento de lote de uma destas linhas.

3.1 A Linha de produção

A Empresa Arco Íris fabrica água, pinga e 12 sabores de refrigerantes em diversos tamanhos. A unidade da empresa situada em São José do Rio Preto é responsável pela produção dos refrigerantes e vasilhames descartáveis. A fábrica possui 3 linhas de produção, das quais uma se destina à produção de refrigerantes em vasilhames de vidro.

A produção do refrigerante é realizada em várias etapas. Primeiro, é prepara-



do o xarope que produzirá o refrigerante. Este xarope é armazenado em tanques que são ligados, através de mangueiras especiais, à linha de produção. A fábrica possui 4 tanques grandes e 3 pequenos. Os tanques pequenos tem capacidade para 4 tachadas, onde um tachada são 1310 litros, e os grandes para 10 tachadas. Os tanques menores se destinam a sabores com demanda menor. Os tanques devem trabalhar com pelo menos meia capacidade para garantir a homogeneidade do xarope. A homogenização é o processo de mistura dos ingredientes do xarope. A hélice que mistura os ingredientes está abaixo do meio do tanque e deve estar quase toda coberta para que seja possível misturar os ingredientes e deixar o xarope homogêneo. Uma amostra do líquido do fundo e de cima são retiradas. Se estiverem iguais a hélice pode ser desligada. Um proporcionador mistura o xarope composto à água tratada na proporção de uma parte de xarope para cinco de água. Esta mistura recebe gás carbônico e se torna o refrigerante que vai para enchedora.

As máquinas são ajustadas inicialmente para produzir refrigerantes de um determinado sabor em um determinado tamanho. Se for necessário fabricar outro sabor e/ou tamanho é preciso parar a linha de produção e fazer os ajustes necessários nas máquinas para trocar o tamanho e/ou trocar o sabor do xarope.

Os insumos necessários para a produção são: xarope de diversos sabores, vasilhames, tampas, rótulos variados, e água gaseificada.

Cada unidade de produção, ou linha de produção, é constituída por uma esteira rolante e diversas máquinas alinhadas em série. As máquinas são utilizadas para esterilizar os vasilhames, encher as garrafas, fechar, rotular, codificar e empacotar os refrigerantes. Ao final do processo os refrigerantes são colocados nos paletes e estocados. O palete é um suporte de madeira feito de forma a ter espaços onde o garfo da empilhadeira se encaixa para poder movimentá-los. A operação da linha de produção é realizada na ordem descrita acima. Se for necessário a retirada de um vasilhame por algum problema, este só será retirado depois de ser rotulado. Existe apenas uma entrada de vasilhames. Uma representação da linha



de produção está ilustrada na Figura 1 a seguir.

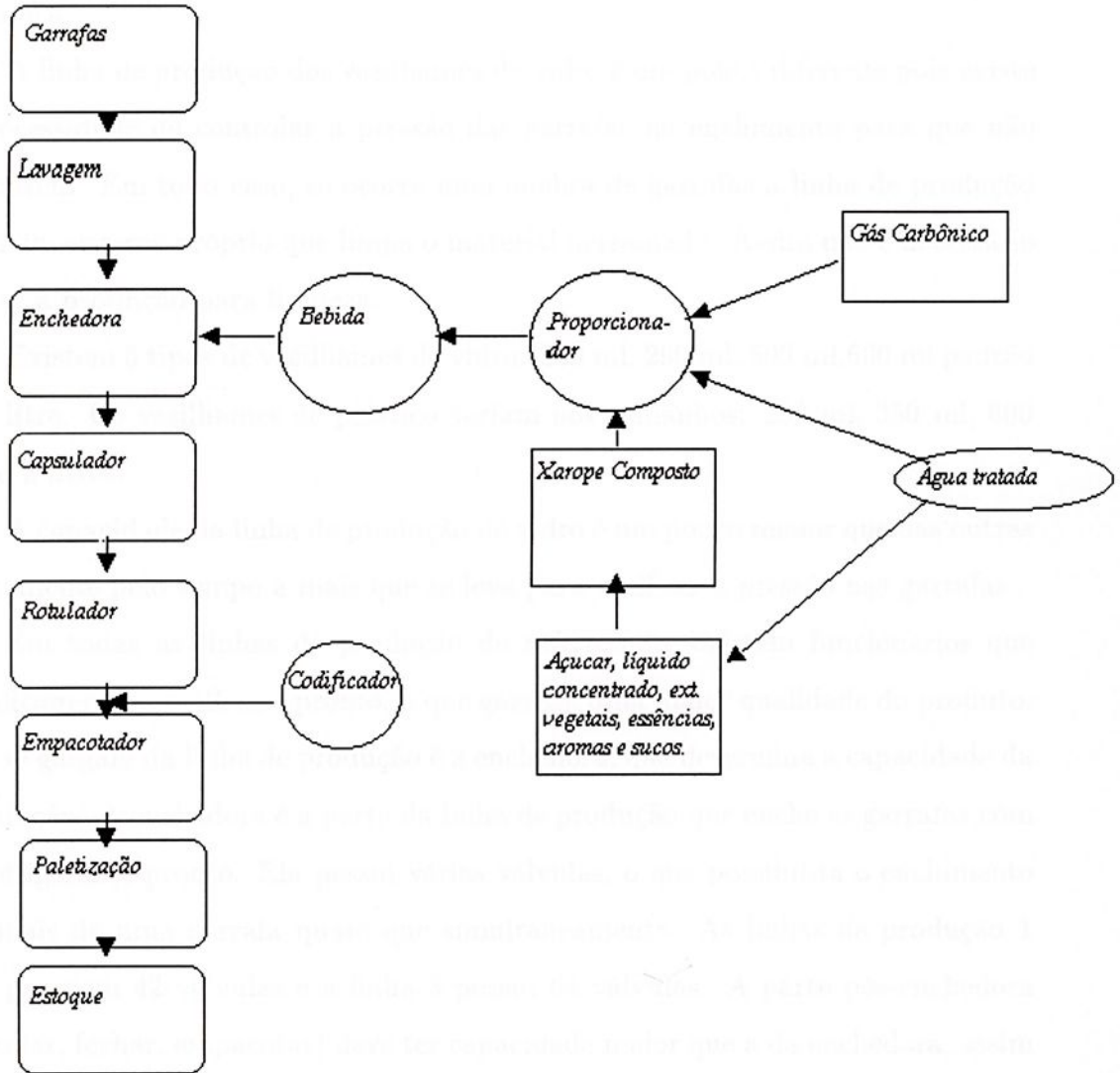


Fig. 3.1: Esquema de Produção da Fábrica de Refrigerantes

Os vasilhames de vidro são comprados e os de plástico são produzidos na própria fábrica em um galpão separado.

Para produzir as garrafas plásticas a matéria prima é sugada por um tubo e passa por processo de ressecamento, recebe a forma de um tubo de ensaio com o "boca" pronta e então vai para a forma. Existe um sistema elétrico interno que mantém a temperatura mais quente que a temperatura ambiente.

Os dois tipos de embalagem passam por um processo de lavagem antes de serem enchidos. Os vasilhames de vidro são lavados de forma especial com soda cáustica.

A linha de produção dos vasilhames de vidro é um pouco diferente pois existe a necessidade de controlar a pressão das garrafas no enchimento para que não estourem. Em todo caso, se ocorre uma quebra de garrafas a linha de produção tem um sistema próprio que limpa o material derramado. Assim não é necessário parar a produção para limpeza.

Existem 5 tipos de vasilhames de vidro: 200 ml, 290 ml, 600 ml, 600 ml padrão e 1 litro. Os vasilhames de plástico variam nos tamanhos: 250 ml, 350 ml, 600 ml e 2 litros.

A capacidade da linha de produção de vidro é um pouco menor que das outras justamente pelo tempo a mais que se leva para verificar a pressão nas garrafas.

Em todas as linhas de produção de refrigerante existem funcionários que verificam cada vasilhame pronto, o que garante uma maior qualidade do produto.

O gargalo da linha de produção é a enchedora, que determina a capacidade da produção. A enchedora é a parte da linha de produção que enche as garrafas com o refrigerante pronto. Ela possui várias válvulas, o que possibilita o enchimento de mais de uma garrafa quase que simultaneamente. As linhas de produção 1 e 2 possuem 42 válvulas e a linha 3 possui 64 válvulas. A parte pós-enchedora (rotular, fechar, empacotar) deve ter capacidade maior que a da enchedora, assim como a parte pré-enchedora (preparo do líquido), pois se a enchedora estiver trabalhando bem o restante da linha de produção deve acompanhá-la. Os tanques foram construídos para armazenar xarope suficiente para a enchedora trabalhar pelo menos 4 horas consecutivas.



3.2 O planejamento da produção

Discutimos a seguir o modelo proposto para o planejamento da produção da fábrica de refrigerantes (FR) descrito na seção anterior. Este modelo tenta retratar matematicamente o planejamento para uma linha de produção de vasilhames descartáveis.

O planejamento consiste em determinar a quantidade de refrigerante que será produzida de forma a satisfazer a demanda do mercado, não ultrapassando as capacidades da fábrica e com objetivo de minimizar os custos de produção, armazenamento e preparo de máquinas.

Definindo os índices:

i : sabor, $i=1,\dots,s$;

j : tamanho, $j=1,\dots,q$;

t : período, $t=1,\dots,n$;

Vamos supor conhecidas as quantidades:

b_{ij} = xarope necessário na produção de uma caixa de refrigerante sabor i
tamanho j ;

k_j = tempo de produção de refrigerante do tamanho j ;

o_i = tempo de preparo da máquina para produção do refrigerante de sabor i ;

f_j = tempo de preparo da máquina para produção do refrigerante de tamanho j ;

h_t = tempo disponível no período t ;

c_{ij}^p = custo de produção de uma caixa de refrigerante sabor i tamanho j ;

c_j^a = custo de armazenamento de uma caixa de refrigerante do tamanho j ;

c_j^{stam} = custo de preparo da máquina (custo de set up) para produção
do refrigerante de tamanho j ;

c_i^{ssab} = custo de preparo da máquina (custo de set up) para produção
do refrigerante de sabor i ;



d_{ijt} = demanda de refrigerante sabor i , tamanho j no período t ;

c_t = capacidade de produção no período t , medido em número de tachadas;

$\min$ = capacidade mínima do tanque.

$\max$ = capacidade máxima do tanque.

l = capacidade de estoque, em m^3 ;

a_j = espaço ocupado por uma caixa de refrigerantes do tamanho j , em m^3 ;

Observe que estamos considerando os tempos e custos de setup independentes da sequência de produção.

As variáveis de decisão do modelo são:

x_{ijt} = quantidade, em número de caixas, do refrigerante sabor i , tamanho j , produzida no período t ;

y_{it} = quantidade, em número de tachadas, de xarope do sabor i usado no período t ;

e_{ijt} = quantidade, em número de caixas, de refrigerante do sabor i tamanho j em estoque no período t ;

$$z_{it} = \begin{cases} 1 & \text{se a máquina estiver ajustada para a produção do sabor } i \text{ no período } t \\ 0 & \text{caso contrário;} \end{cases}$$

$$w_{jt} = \begin{cases} 1 & \text{se a máquina estiver ajustada para a produção do tamanho } j \text{ no período } t \\ 0 & \text{caso contrário;} \end{cases}$$

$$s_{it} = \begin{cases} 1 & \text{se há o início da produção do sabor } i \text{ no período } t \\ 0 & \text{caso contrário;} \end{cases}$$

$$v_{jt} = \begin{cases} 1 & \text{se há o início da produção do tamanho } j \text{ no período } t \\ 0 & \text{caso contrário;} \end{cases}$$

Função Objetivo

A função objetivo expressa o objetivo do modelo que é minimizar o custo total, que envolve o custo de produção, o custo de armazenamento e o custo de preparo de máquinas. O custo de preparo de máquina é considerado pois, há um custo fixo de manter uma máquina preparada.

A função objetivo é dada por:

$$\text{Min} \sum_{i=1}^s \sum_{j=1}^q \sum_{t=1}^n (c_{ij}^p x_{ijt} + c_j^a e_{ijt} + c_j^{stam} w_{jt} + c_i^{ssab} z_{it})$$

Restrições

- Quantidade de xarope disponível para a produção dos refrigerantes

Observamos que uma tachada são 1310 litros de xarope que produzem 7860 litros de refrigerante. Todo xarope preparado deve ser utilizado na produção, não havendo assim sobras. Então, a restrição abaixo garante que a quantidade necessária de xarope para a produção de cada sabor de refrigerante, $\sum_{j=1}^q b_{ij} x_{ijt}$, seja igual a quantidade disponível no período, $1310 y_{it}$.

$$\sum_{j=1}^q b_{ij} x_{ijt} = 1310 y_{it}, \quad i = 1, \dots, s; t = 1, \dots, n.$$

- Capacidade de produção medida em termos de tachadas de xarope

A capacidade de produção da fábrica é medida pelo número máximo de tachadas de xarope que as máquinas conseguem trabalhar por período. Sendo assim, a quantidade total de tachadas utilizadas na produção dos refrigerantes não deve ultrapassar a capacidade da fábrica, medida em número de tachadas.

$$\sum_{i=1}^s y_{it} \leq c_t, \quad t = 1, \dots, n.$$



- Tempo disponível

A fábrica tem disponível um número limitado de horas de trabalho por dia. Durante este período são produzidos vários tipos de refrigerante. Cada troca de sabor e/ou tamanho de refrigerante consome um tempo de preparo das máquinas. Portanto, a soma dos tempos de produção dos refrigerantes com os tempos de troca de sabor e tamanho de refrigerante devem ser menor ou igual o tempo disponível para produção .

$$\sum_{i=1}^s \sum_{j=1}^q k_j x_{ijt} + \sum_{i=1}^s o_i s_{it} + \sum_{j=1}^q f_j v_{jt} \leq h_t, \quad t = 1, \dots, n.$$

- Ajustes das máquinas para troca de sabor

Havendo a necessidade de produção do sabor i a máquina deve estar preparada para a produção deste. Logo, só pode haver produção se a máquina estiver pronta ($z_{it} = 1$). Para isto incluímos no modelo a restrição de ajuste de sabor.

$$x_{ijt} \leq M z_{it}, \quad i = 1, \dots, s; \quad j = 1, \dots, q; \quad t = 1, \dots, n.$$

- Ajustes das máquinas para troca de Tamanho

Para produzir um tamanho j também é necessário que a máquina esteja preparada para tal ($w_{jt}=1$). Então incluímos a restrição de ajuste de tamanho da máquina.

$$x_{ijt} \leq M w_{jt}, \quad i = 1, \dots, s; \quad j = 1, \dots, q; \quad t = 1, \dots, n.$$

- Balanço e Estoque

A empresa recebe os pedidos de compra de refrigerante, demanda, em cada período. Estes pedidos devem ser atendidos pela produção daquele período e/ou



pelo estoque do período anterior. Todo o excedente é novamente estocado e eventualmente usado em períodos posteriores.

A restrição que garante o balanço entre o atendimento da demanda e a quantidade estocada é:

$$e_{ijt-1} + x_{ijt} = d_{ijt} + e_{ijt}, \quad i = 1, \dots, s; \quad j = 1, \dots, q; \quad t = 1, \dots, n.$$

- Capacidade de Estoque

O armazenamento das caixas de refrigerante é feito em um galpão coberto, e de tamanho limitado. Por isso, incluímos ao modelo a restrição que garante que o espaço ocupado pelos refrigerantes a serem estocados não ultrapassará o espaço disponível.

$$\sum_{i=1}^s \sum_{j=1}^q a_j e_{ijt} \leq l, \quad t = 1, \dots, n.$$

- Capacidade dos Tanques

Os tanques que armazenam o xarope utilizado na produção dos refrigerantes tem uma capacidade máxima e mínima de armazenamento que deve ser respeitada. Por isso, incluímos no modelo a seguinte restrição.

$$\min \leq y_{it} \leq \max, \quad i = 1, \dots, s; \quad t = 1, \dots, n.$$

- Troca de Sabor

As três restrições a seguir garantem que se as máquinas foram preparadas no período t , e não estavam prontas no período anterior, $s_{it} = 1$ e $s_{i(t-1)} = 0$, haverá produção do sabor i naquele período. Com isto conseguimos contar o tempo de preparo das máquinas para o novo sabor que começará a ser produzido. Note que



a variável s_{it} está incluída na restrição de tempo.

$$s_{it} \geq z_{it} - z_{i(t-1)}, \quad i = 1, \dots, s; \quad t = 1, \dots, n;$$

$$s_{it} \leq z_{it}, \quad i = 1, \dots, s; \quad t = 1, \dots, n;$$

$$s_{it} \leq 1 - z_{i(t-1)}, \quad i = 1, \dots, s; \quad t = 1, \dots, n.$$

- Troca de Tamanho

Para considerar a troca de tamanho usamos as restrições similares às de troca de sabor descritas acima, considerando que um novo tamanho de refrigerante será produzido.

$$v_{jt} \geq w_{jt} - w_{j(t-1)}, \quad j = 1, \dots, q; \quad t = 1, \dots, n;$$

$$v_{jt} \leq w_{jt}, \quad j = 1, \dots, q; \quad t = 1, \dots, n;$$

$$v_{jt} \leq 1 - w_{j(t-1)}, \quad j = 1, \dots, q; \quad t = 1, \dots, n.$$

- Tipo de variáveis

Para definir as variáveis binárias e garantir não negatividade da produção, incluímos as seguintes restrições ao modelo.

$$x_{ijt} \geq 0, \quad e_{ijt} \geq 0, \quad z_{it} = 0/1, \quad w_{jt} = 0/1, \quad y_{it} \geq 0, \\ i = 1, \dots, s; \quad j = 1, \dots, q; \quad t = 1, \dots, n.$$



O modelo construído (FR) é dado a seguir.

$$\min \sum_{i=1}^s \sum_{j=1}^q \sum_{t=1}^n (c_{ij}^p x_{ijt} + c_j^a e_{ijt} + c_j^{stam} w_{jt} + c_i^{ssab} z_{it}) \quad (3.1)$$

Sujeito à:

$$e_{ijt-1} + x_{ijt} = d_{ijt} + e_{ijt}, \quad i = 1, \dots, s; j = 1, \dots, q; t = 1, \dots, n. \quad (3.2)$$

$$\sum_{j=1}^q b_{ij} x_{ijt} = 1310 y_{it}, \quad i = 1, \dots, s; t = 1, \dots, n. \quad (3.3)$$

$$\sum_{i=1}^s y_{it} \leq c_t, \quad t = 1, \dots, n. \quad (3.4)$$

$$\sum_{i=1}^s \sum_{j=1}^q k_j x_{ijt} + \sum_{i=1}^s o_i s_{it} + \sum_{j=1}^q f_j v_{jt} \leq h_t, \quad t = 1, \dots, n. \quad (3.5)$$

$$x_{ijt} \leq M z_{it}, \quad i = 1, \dots, s; j = 1, \dots, q; t = 1, \dots, n. \quad (3.6)$$

$$x_{ijt} \leq M w_{jt}, \quad i = 1, \dots, s; j = 1, \dots, q; t = 1, \dots, n. \quad (3.7)$$

$$s_{it} \geq z_{it} - z_{i(t-1)}, \quad i = 1, \dots, s; t = 1, \dots, n. \quad (3.8)$$

$$s_{it} \leq z_{it}, \quad i = 1, \dots, s; t = 1, \dots, n. \quad (3.9)$$

$$s_{it} \leq 1 - z_{i(t-1)}, \quad i = 1, \dots, s; t = 1, \dots, n. \quad (3.10)$$

$$v_{jt} \geq w_{it} - w_{j(t-1)}, \quad j = 1, \dots, q; t = 1, \dots, n. \quad (3.11)$$

$$v_{jt} \leq w_{jt}, \quad j = 1, \dots, q; t = 1, \dots, n. \quad (3.12)$$

$$v_{jt} \leq 1 - w_{j(t-1)}, \quad j = 1, \dots, q; t = 1, \dots, n. \quad (3.13)$$

$$\sum_{i=1}^s \sum_{j=1}^q a_j e_{ijt} \leq l, \quad t = 1, \dots, n. \quad (3.14)$$

$$\min \leq y_{it} \leq \max, \quad i = 1, \dots, s; t = 1, \dots, n. \quad (3.15)$$

$$x_{ijt} \geq 0, \quad e_{ijt} \geq 0, \quad z_{it} = 0/1, \quad w_{jt} = 0/1, \quad y_{it} \geq 0, \quad i = 1, \dots, s; j = 1, \dots, q; t = 1, \dots, n. \quad (3.16)$$



Como vimos no modelo descrito acima o processo de planejamento da produção é bastante complexo. Há vários fatores que o engenheiro de produção deve levar em consideração no momento de determinar a programação da produção: as quantidades de xarope e tempo de produção disponíveis, a demanda a ser suprida, a capacidade das máquinas, manutenção de uma quantidade razoável de refrigerante em estoque para conseguir suprir uma demanda inesperada. Além disso, existe um tempo e um espaço máximos para estocagem dos refrigerantes. Tudo isso deve ser retratado no modelo matemático para que sua resposta seja um plano de trabalho o mais completo possível.

O modelo FR foi definido considerando a produção de refrigerantes de s sabores, em q tamanhos e em n períodos, envolvendo $[n(2(s + q) + s(2q + 1))]$ variáveis das quais $[ns(2q + 1)]$ são não negativas e $[2n(s + q)]$ são binárias. O total de restrições é $[n(4s + 3(sq + 1 + q))]$. Estes números podem gerar instâncias grandes dependendo do número de sabores, tamanhos e períodos considerados.

Na próxima seção explicitamos algumas ferramentas que podem colaborar na geração das instâncias e discutimos a utilização do modelo matemático proposto como uma ferramenta para o planejamento da produção.

3.3 Ferramentas para o Planejamento da Produção

A seção 3.2 descreve a construção de um modelo de otimização que pode ser utilizado como ferramenta no processo de determinação dos lotes da produção. Esta ferramenta considera os fatores relevantes da linha de produção como capacidades, tempos, demanda entre outros. Alguns destes dados são fixos, por exemplo a capacidade da máquina, outros não como é o caso da demanda semanal. Para cada novo planejamento é necessário atualizar os dados não fixos. O uso de softwares de modelagem facilitam o processo de construção e solução de uma instância do modelo.

O software de modelagem faz a interface entre o usuário e o modelo matemáti-



co. Além disso, permite uma documentação do problema e ainda gera instâncias do modelo em vários formatos. Estes formatos se referem a forma específica de entrada de dados necessária para alguns softwares de otimização. Existem no mercado vários softwares de modelagem por exemplo, LINGO (Schrage, 2001), MPL (Maximal Software, 1998).

Neste trabalho fazemos o uso do MPL (Modeling Programming Language). Descrevemos a seguir seus principais comandos.

- i) TITLE - Comando inicial que indica o título do modelo;
- ii) INDEX - Declara os índices das variáveis do problema;
- iii) DATA - Neste campo são declarados os dados do problema, que podem ser fornecidos através de arquivos de dados, com o comando DATAFILE, podem ser digitados diretamente no modelo em construção; ou podem ainda ser extraídos de bancos de dados ou planilhas de cálculo.
- iv) DECISION VARIABLES - Declara as variáveis de decisão do modelo;
- v) MODEL - Inicia a descrição do modelo;
- vi) MAX ou MIN - Declara a função objetivo;
- vii) SUBJECT TO - Declara as restrições;
- viii) BOUND - Declara os limites das variáveis;
- xix) BINARY e INTEGER - Indica o tipo de variável, binária ou inteira. Se a variável for maior ou igual a zero não é necessário nenhum comando específico.
- x) END - Indica o fim do modelo.

Ao final de cada linha digitada no arquivo deve-se colocar ";". Os comentários podem ser inseridos entre chaves, {...}.

A seguir mostramos o arquivo FR.mpl com a descrição do modelo FR (descrito na seção 3.2 sem as restrições de limite dos tanques) feita de acordo com a estrutura do MPL.



{Arquivo FR.mpl}

TITLE

Fab_Ref

{Modelo de programação inteira mista que faz a determinação dos lotes para o planejamento da produção de uma fábrica de refrigerantes.}

INDEX

{São considerados os sabores, os tamanhos e o período no qual o refrigerante deve ser produzido.}

sabor:=1..2;

tam:=1..3;

per:=1..5;

DATA

{São declaradas a seguir as quantidades de xarope, tempos de produção, tempo de troca de produção de sabor e tamanho, tempo disponível de produção, custos de produção, armazenamento, e preparo das máquinas para troca de sabor e tamanho, demanda, volume das caixas em estoque, capacidades de estoque e produção. O comando DATAFILE indica que os dados estão nos arquivos com extensão ".dat"}

quant_b[sabor, tam]:=DATAFILE(quant_b.dat);

Tprod_k[tam]:=DATAFILE(Tprod_k.dat);

Ttrosab_o[sabor]:=DATAFILE(Ttrosab_o.dat);

Trotam_ff[tam]:=DATAFILE(Trotam_f.dat);

Tdisp_h[per]:=DATAFILE(Tdisp_h.dat);

cusprod[sabor, tam]:=DATAFILE(cusprod.dat);

cus_arm[tam]:=DATAFILE(cus_arm.dat);

cus_startS[sabor, per]:=DATAFILE(cus_startS.dat);

cus_startV[tam, per]:=DATAFILE(cus_startV.dat);

deman[sabor, tam, per]:=DATAFILE(deman.dat);

vol[tam]:=DATAFILE(vol.dat);

cap_estoq[per]:=DATAFILE(cap_estoq.dat);

cap_prod[per]:=DATAFILE(cap_prod.dat);

M2[tam, sabor, per]:=DATAFILE(M2.dat);

M1[tam, sabor, per]:=DATAFILE(M1.dat);



DECISION VARIABLES

{As variáveis de decisão do modelo que definem o tamanho dos lotes, as quantidades de xarope necessárias para produção, as quantidades de estoque e as que determinam o preparo das máquinas e o início da produção, são declaradas a seguir.}

$x[\text{sabor}, \text{tam}, \text{per}]$;

$y[\text{sabor}, \text{per}]$;

$z[\text{sabor}, \text{per}]$;

$w[\text{tam}, \text{per}]$;

$k[\text{sabor}, \text{tam}, \text{per}]$;

$s[\text{sabor}, \text{per}]$;

$v[\text{tam}, \text{per}]$;

MODEL

{Pretende-se minimizar o custo total, que envolve os custos de produção, armazenamento e início da produção.}

Min Custo = $SUM(\text{sabor}, \text{tam}, \text{per}: \text{cusprod} * x + \text{cus_arm} * k + \text{cus_start} S * z + \text{cus_start} V * w)$;

SUBJECT TO

{Todas as restrições, 3.2 -3.15 respectivamente, do modelo FR são declaradas a seguir.}

$BE[\text{sabor}, \text{tam}, \text{per}]: k[\text{sabor}, \text{tam}, \text{per}-1] + x = \text{deman} + k[\text{sabor}, \text{tam}, \text{per}]$;

$capx[\text{sabor}, \text{per}]: SUM(\text{tam}: \text{quant_b} * x) = y * 1310$;

$captac[\text{per}]: SUM(\text{sabor}: y) \leq \text{cap_prod}$;

$captem[\text{per}]: SUM(\text{sabor}, \text{tam}: T_{\text{prod_k}} * x) + SUM(\text{sabor}: T_{\text{trosab_o}} * s) + SUM(\text{tam}: T_{\text{rotam_f}} * v) \leq T_{\text{disp}}$

$Ts[\text{sabor}, \text{tam}, \text{per}]: x \leq M2 * z$;

$Tt[\text{sabor}, \text{tam}, \text{per}]: x \leq M1 * w$;

$s1St[\text{sabor}, \text{per}]: s \geq z - z[\text{per}-1]$;

$s2St[\text{sabor}, \text{per}]: s \leq z$;

$s3St[\text{sabor}, \text{per}]: s \leq 1 - z[\text{per}-1]$;

$t1St[\text{tam}, \text{per}]: v \geq w - w[\text{per}-1]$;

$t2St[\text{tam}, \text{per}]: v \leq w$;

$t3St[\text{tam}, \text{per}]: v \leq 1 - w[\text{per}-1]$;

$caparm[\text{per}]: SUM(\text{sabor}, \text{tam}: \text{vol}[\text{tam}] * (k[\text{sabor}, \text{tam}, \text{per}-1] + x - \text{deman})) \leq \text{cap_estoq}$;



BINARY

{É necessário indicar apenas as variáveis binárias.}

z;

w;

s;

v;

END

{Fim do modelo FR.}

Pelos índices indicados no modelo geraremos uma instância do problema com 2 sabores, 2 tamanhos diferentes de refrigerantes, e 5 períodos no horizonte de planejamento. Isto resulta em uma instância com 90 variáveis e 145 restrições que o MPL gera quase instantaneamente. Para gerar modelos de outras dimensões é necessário atualizar os dados e os índices.

Os softwares de modelagem permitem também o acesso direto a pacotes de otimização. Com isto podemos automatizar o uso do modelo matemático.

Abaixo ilustramos a interpretação da resposta do modelo FR.

Exemplo 3.1: Consideremos uma situação onde será planejada a produção de 2 sabores e 2 tamanhos diferentes de refrigerantes, em 5 períodos. Ou seja, 4 refrigerantes distintos A_{11} , B_{12} , C_{21} , D_{22} . Os dois números ao pé da letra indicam o sabor e o tamanho do refrigerante respectivamente.



A demanda de cada tipo de refrigerante em cada período e os custos de produção e estoque foram gerados aleatoriamente e são descritos nas tabelas 3.1-3.3 abaixo:

Tabela 3.1: Demanda dos refrigerantes					
Refrig.	Per.1	Per.2	Per.3	Per.4	Per.5
A_{11}	53	642	545	562	621
B_{12}	23	618	649	600	564
C_{21}	9	695	645	582	641
D_{22}	48	504	548	695	589

Tabela 3.2: Custos de produção	
Refrig.	Custo prod.
A_{11}	1,364
B_{12}	1,727
C_{21}	1,364
D_{22}	1,000

Tabela 3.3: Custos de Estoque	
Tamanho	Custo Est.
1	0.682
2	1.318

O software MPL foi usado para obter esta instância do problema no formato MPS (Ilog, 2000) para ser resolvida usando o software CPLEX (Ilog, 2000). A solução ótima obtida, com custo total de R\$16.279,1, indica que a produção dos lotes deve ser feita de acordo com os dados descritos na tabela 3.4 e o estoque no período é descrito na tabela 3.5. A quantidade de xarope necessária em cada período é dada na tabela 3.6.



Tabela 3.4: Determinação dos lotes

Refrig.	Per.1	Per.2	Per.3	Per.4	Per.5
A_{11t}	700	246,8	713,63	141,55	621
B_{12t}	35,71	605,2	649	600	564
C_{21t}	718,18	704,55	174,5	675,9	298,85
D_{22t}	48	504	548	695	589

Tabela 3.5: Estoque

Refrig.	Per.1	Per.2	Per.3	Per.4	Per.5
A_{11t}	647	251,8	420,45	0	0
B_{12t}	12,71	0	0	0	0
C_{21t}	709,182	718,727	248,21	242,15	0
D_{22t}	0	0	0	0	0

Tabela 3.6: Número de Tachadas de Xarope

Refrig.	Per.1	Per.2	Per.3	Per.4	Per.5
Sabor 1	4.6	4.7	7.8	4	6.9
Sabor 2	3.8	6.8	4.5	7.9	5.3

Observando o planejamento do primeiro período vemos que sua produção é de 700 caixas. Como a demanda exige apenas 53 caixas, as 647 caixas restante serão estocadas. No segundo período a produção, 246.8 caixas, não é suficiente para abastecer a demanda de 642, utilizamos então os itens estocados no primeiro período. Este estoque não é todo utilizado e pode portanto ajudar a suprir a demanda de períodos posteriores.

O terceiro período tem uma produção maior que sua demanda. Esta produção excedente aumentará o estoque já existente. A produção no quarto período é pequena e o estoque é todo utilizado. Assim, a produção do quinto período deve ser igual sua demanda.

Lembremos que todo este planejamento foi feito levando em consideração as



capacidades das máquinas, os tempos disponíveis de produção, o espaço disponível para armazenamento entre outros fatores já citados, e objetiva minimizar o custo total do processo. Assim, quando um estoque é muito maior que a produção significa que estocar é mais barato que produzir naquele momento. Produzir o refrigerante D_{22} na quantidade exata no período 5 para suprir a demanda por exemplo, é mais vantajoso do que manter este produto em estoque. Observe que o custo de estocar este item é maior que o custo de produção.

O custo de preparo de máquina também influencia na determinação dos lotes. As variáveis de setup, w_{jt} e z_{it} , assumiram valor 1 para todos os refrigerantes em todos os períodos. Isto porque há produção de todos os refrigerantes em todos os períodos o que significa que as máquinas devem estar preparadas para a produção. Assim, é natural também que as variáveis de startup, v_{jt} e s_{it} , que indicam novo preparo tenham valor 1 apenas no primeiro período, já que consideramos que a máquina permanece preparada para os próximos períodos.

Observe que a solução obtida acima não indica a ordem em que os refrigerantes devem ser produzidos. O planejamento da produção de itens que envolvem tempos de setup, deve incluir também o sequenciamento pois, a ordem da produção altera o tempo total de preparo das máquinas.

Observemos os tempos de preparo das máquinas quando há troca de refrigerantes no período 1 apresentados na tabela 3.7 a seguir.



Tabela 3.7: Tempos de preparo				
Refrig.	A_{111}	B_{121}	C_{211}	D_{221}
A_{11}	0	2.73	14.818	17.6
B_{12}	2.73	0	31,36	4.46
C_{21}	4.46	7.18	0	2.73
D_{22}	17.6	4.46	4.46	0

Se produzirmos a sequência de refrigerantes A_{11} , B_{12} , C_{21} , D_{22} , no período 1 teremos as trocas:

- do sabor/tamanho 11 para 12 (tempo 2,73); 12 para 21 (tempo 31,36) e 21 para 22 (tempo 2,73).

Esta sequência resulta no tempo total de preparo 36,82. Porém, se realizarmos a sequência:

- do sabor/tamanho 11 para 12 (tempo 2,73); 12 para 22 (tempo 4,46) e 22 para 21 (tempo 4,46);

teremos um tempo total de preparo de 11,65. Um terço do tempo da primeira sequência proposta.

Existem métodos para considerar esta diferença no tempo total de preparo devido ao sequenciamento da produção na definição do programa de produção. A primeira proposta consiste em fazer o planejamento da produção em dois estágios. Em um estágio é definido o dimensionamento dos lotes e no outro o sequenciamento da produção. Uma segunda proposta é combinar em um único modelo o dimensionamento dos lotes e o sequenciamento da produção conforme descrito na seção 1.2.(Drexel e Kimmms, 1997).

Neste trabalho supomos que o planejamento da produção é feito em dois estágios. O modelo proposto na seção 3.2 considera apenas o dimensionamento dos lotes. O sequenciamento dos itens pode ser feito em um estágio anterior ou posterior. Uma boa revisão sobre o problema de sequenciamento pode ser vista em (Muller, 2001).



Atualmente existem muitos softwares de otimização tais como CPLEX (ILOG, 2000), LINDO (Lindo system, 1999), que resolvem problemas gerais de otimização inteira mista. A cada dia, novas técnicas de otimização são incorporadas aos softwares tornando-os mais eficientes. Se olharmos por exemplo, o uso dos Planos de Corte no CPLEX, observamos que na versão 4.0 haviam os Cortes de Cobertura Mínima (Rangel, 1995) e os Corte de Cliques (Nemhauser e Wolsey, 1988), na versão 6.6 além dos anteriores foram incorporados os Cortes de Gomory, Cobertura de Fluxo, Cobertura GUB (Nemhauser e Wolsey, 1988) e na versão 7.0 já estão implementadas as Inequações de Caminho em Rede (Van Roy e Wolsey, 1987) entre outras.

Capítulo 4

Experimentos Computacionais

Atualmente existem muitos softwares de otimização tais como CPLEX (ILOG, 2000), LINDO (Lindo system, 1999), que resolvem problemas gerais de otimização inteira mista. A cada dia, novas técnicas de otimização são incorporadas aos softwares tornando-os mais eficientes. Se olharmos por exemplo, o uso dos Planos de Corte no CPLEX, observamos que na versão 4.0 haviam os Cortes de Cobertura Mínima (Rangel, 1995) e os Corte de Cliques (Nemhauser e Wolsey, 1988), na versão 6.6 além dos anteriores foram incorporados os Cortes de Gomory, Cobertura de Fluxo, Cobertura GUB (Nemhauser e Wolsey, 1988) e na versão 7.0 já estão implementadas as Inequações de Caminho em Rede (Van Roy e Wolsey, 1987) entre outras.

Alguns softwares possuem também bibliotecas de subrotinas que facilitam a implementação de algoritmos específicos para resolução de classes especiais de problemas de otimização. Neste trabalho utilizamos as bibliotecas de subrotinas do CPLEX 6.6 para implementar o algoritmo SolFR que utiliza iniquações supermodulares na solução do problema FR. Este algoritmo visa melhorar o processo de solução de Problemas de Planejamento da Produção de Grande Porte.

Na seção 4.1 são descritos os parâmetros usados na geração das instâncias testadas, na seção 4.2 são analisados os resultados obtidos com o software CPLEX. Na seção 4.3 discutimos os resultados do uso do algoritmo SolFR e damos uma



breve descrição dos passos iniciais para usar a biblioteca de subrotinas do CPLEX.

Em todos os testes computacionais o tempo de solução foi limitado em meia hora e o número de nós em 500000. O computador utilizado foi um PC pentium III, 128 Mb, 800 Mhz.

A tabela 4.1 a seguir mostra os dados das grades.

Tabela 4.1. Dados das grades

Dist.	Setores	Instâncias	Per	Total Vals	Var. Binárias	Rest.	Temp.
1	6	1	7	132	112	20	13
2	12	4	15	2100	608	3192	45
3	12	4	15	8764	1440	7324	60
4	3	2	24	500	240	260	4
5	2	5	30	1000	500	500	10
6	12	1	7	504	192	312	14

O seguinte conjunto de instâncias (Tabela 4.1) foi utilizado para a resolução dos problemas de programação linear inteira. O conjunto de instâncias foi dividido em dois grupos de problemas de programação linear inteira. O primeiro grupo contém problemas de programação linear inteira com um número de variáveis binárias entre 100 e 1000. O segundo grupo contém problemas de programação linear inteira com um número de variáveis binárias entre 1000 e 10000. Os problemas de programação linear inteira foram resolvidos usando o solver CPLEX 12.1. Os resultados dos testes são apresentados na Tabela 4.2.

Os parâmetros utilizados para a resolução dos problemas de programação linear inteira são os seguintes:

4.1 Geração de Dados

Para verificar a dificuldade de solução do modelo testamos instâncias geradas aleatoriamente em 6 dimensões diferentes. Para cada dimensão descrita na tabela 4.1 geramos 10 instâncias. Testando assim 60 instâncias diferentes.

A tabela 4.1 a seguir mostra as dimensões geradas.

Tabela 4.1: Dimensões geradas							
Dim.	Sabores	Tamanhos	Per.	Total Var.	Var. Binárias	Rest.	Itens
1	6	2	7	322	112	483	12
2	12	4	15	2100	480	3105	48
3	12	8	36	8784	1440	13068	96
4	3	2	20	500	240	780	6
5	2	5	30	1080	420	1680	10
6	14	1	7	504	136	728	14

O segundo conjunto de exemplares (Dim 2) do modelo FR tem dimensões de um problema real, considerando a produção de vasilhames de plástico. O primeiro e o terceiro tipo foram gerados como sendo exemplares pequenos e grandes com relação ao número de itens. Ao utilizar as inequações supermodulares não conseguimos resolver os exemplares tipo 2 e 3 por falta de memória. Por isso, geramos uma nova classe de problemas, tipo 4, 5 e 6 que são modelos que consideramos serem pequenos, médios e grandes.

Os parâmetros utilizados para a geração dos dados estão descritos na tabela 4.2 a seguir.



Tabela 4.2: Parâmetros usados para a geração de instâncias do modelo FR.

Tipo de dados	Nome	Intervalo
Quantidade de Xarope	b_{ij}	[5, 10]
Tempo setup: sabor	o_i	[1, 20]
Tempo setup: tamanho	f_j	[1, 20]
Tempo: produção	k_{ij}	[1, 5]
Custo: produção	c_{ij}^p	[1, 3]
Custo: armazenagem	c_j^a	[0.5, 1.5]
Custo setup: sabor	c_i^{ssab}	[10, 20]
Custo setup: tamanho	c_j^{stam}	[10, 20]
Nível máximo de produção (sabor)	$M1_{ijt}$	[700, 750]
Nível máximo de produção (tamanho)	$M2_{ijt}$	[700, 750]
Volume	vol_{tam}	[0.4, 1]
Tempo total: disponível	h_t	$\frac{(\sum_i \sum_j \sum_t d_{ijt} k_{ij} + \sum_i o_i + \sum_j f_j)}{((0.97)T)}$
Capacidade total: produção	c_t	$\frac{(\sum_i \sum_j b_{ij} * d_{ijt})}{T}$
Capacidade total: estoque	1	500000
Demanda	d_{ijt}	[0, 700]

A obtenção de instâncias factíveis para este modelo foi um trabalho complicado. Os parâmetros usados foram estabelecidos através de testes computacionais. Para definir a demanda estabelecemos que se a instância a ser gerada envolvesse mais que 4 períodos, a demanda dos 4 primeiros períodos estaria entre 0 e 100 e a dos demais períodos entre 500 e 700. Este processo de construção de instâncias factíveis foi baseado em parâmetros usados em (Araújo e Arenales, 2000).

Para utilização das Inequações Supermodulares como Planos de Corte é necessário que a demanda seja sempre menor ou igual a M1 e M2. Se a demanda for maior que M1 ou M2, uma reformulação dos dados deve ser incluída na implementação



do método de solução (Constantino, 1996).

4.2 Solução do Modelo FR

Nesta seção apresentamos resultados da resolução do modelo FR obtidos com o CPLEX. Este software de otimização resolve problemas de otimização inteira mista e possui várias opções de métodos para resolvê-los, como o método Branch and Bound, método de Planos de Corte e Heurísticas. Além de podermos optar pelo método, ou métodos pois é possível utilizá-los de maneira híbrida, o CPLEX possui várias opções que podem melhorar o desempenho do algoritmo usado. Este é o caso das opções de busca na árvore do método Branch and Bound e das opções de tipos de Planos de Corte a serem aplicados.

Resolvemos as instâncias com 9 combinações diferentes de estratégias do software CPLEX 6.6.

Dividimos os testes em estratégias tipo 1, que testam os efeitos da heurística e de planos de corte no modelo; estratégias tipo 2, que envolvem estratégias diferentes de busca na árvore do Branch and Bound; e tipo 3 que combina os parâmetros da melhor estratégia do tipo 1 e do tipo 2. O padrão do CPLEX, que chamaremos de estratégia 1.1, utiliza os planos de corte, a heurística e a seleção de variáveis automaticamente; para selecionar o nó da árvore ele opta pelo nó de melhor limite, veja seção 2.3. As estratégias usadas estão descritas na tabela 4.3 a seguir.



Tabela 4.3: Estratégias para Solução do Modelo FR.						
Estratégia	PC	H	SVMI	SVS	SNML	SNME
Est.1.1	auto.	auto.	auto.	auto.	X	-
Est.1.2	-	auto.	auto.	auto.	X	-
Est.1.3	-	-	auto.	auto.	X	-
Est.1.4	auto.	-	auto.	auto.	X	-
Est.2.1	auto.	auto.	X	-	X	-
Est.2.2	auto.	auto.	X	-	-	X
Est.2.3	auto.	auto.	-	X	X	-
Est.2.4	auto.	auto.	-	X	-	X
Est.3.1	auto.	-	-	X	X	-

X: Utiliza esta opção;

-: Nunca utiliza esta opção;

auto.: A decisão de usar ou não é feita automaticamente pelo CPLEX.

PC: Planos de Corte;

H: Heurística;

SVMI (Seleção de variável): a mais infactível;

SVS (Seleção de variável): strong branch;

SNML (Seleção de nó): melhor limite;

SNME (Seleção de nó): melhor estimativa;

Separamos os resultados obtidos em: instâncias com solução ótima para todas as estratégias; e instâncias para as quais o pacote CPLEX falha em encontrar a solução ótima em todas as estratégias.

Analizamos os resultados obtidos em termos de melhor solução inteira encontrada, tempo de solução, número de nós examinados e nó da melhor solução inteira obtida. Os resultados (valor médio e desvio padrão) para as estratégias tipo 1 estão descritos nas tabelas 4.4 a 4.11 e para a estratégia tipo 2 nas tabelas 4.12 a 4.19, para cada grupo de instâncias. No cálculo da média o número de



instâncias utilizados é indicado entre chaves {}, abaixo da média é dado o desvio padrão entre parênteses ().

Estratégias tipo 1:

• Função Objetivo (Valor Ótimo)

Observamos uma certa variação no valor da função objetivo nas soluções de algumas instâncias que o CPLEX apontou como ótimas. Porém, foi uma variação pequena como podemos ver na tabela 4.4.

Tabela 4.4: Valor médio da FO (Desvio Padrão) (Sol. Ótima)				
Modelo	Estra.1.1	Estra.1.2	Estra.1.3	Estra.1.4
Dim. 1 {10}	91.875 (7.284)	91.875 (7.284)	91.875 (7.284)	91.875 (7.284)
Dim. 2 {10}	933.182 (80.562)	933.084 (80.542)	933.077 (80.540)	933.077 (80.540)
Dim. 4 {10}	195.153 (21.565)	195.155 (21.564)	195.155 (21.564)	195.155 (21.564)
Dim. 5 {10}	477.399 (46273,211)	477.401 (46281,756)	477.398 (46275,708)	477.397 (46271,085)
Dim. 6 {5}	115.596 (5.513)	115.596 (5.513)	115.596 (5.513)	115.596 (5.513)



• Função Objetivo (Melhor Solução Inteira)

A diferença no valor da melhor solução inteira obtida em cada estratégia é pequena (tabela 4.5). A estratégia 1.1 (default), que permite o uso de heurística e planos de corte, encontra em média as melhores soluções.

Não houve muita variação no valor da solução encontrada usando as diferentes estratégias para a dimensão 6.

Tabela 4.5: Valor médio da FO (Desvio Padrão)(Melhor Sol. Int.)				
Modelo	Estra.1.1	Estra.1.2	Estra.1.3	Estra.1.4
Dim 3	5.960.129	5.978.515	5.970.484	5.969.712
{10}	(672.515)	(678.307)	(678.648)	(678.890)
Dim 6	111.062	111.062	111.062	111.062
{5}	(474.258)	(474.259)	(474.258)	(474.258)



• Tempo de Solução (Solução Ótima)

Tendo em vista que todas as soluções das instâncias da tabela 4.6 abaixo são ótimas, o tempo de solução é um fator importante.

Dos cinco tipos de intâncias com solução ótima em todas as estratégias, três tiveram suas soluções encontradas mais rapidamente com a estratégia 1.4 e duas com a estratégia 1.3. Lembremos que estas estratégias não permitem o uso de heurística.

As instâncias que a estratégia 1.3 obteve os menores tempos são as instâncias tipo 2 e 6, que são duas das três instâncias com maior número de itens.

Tabela 4.6: Tempos Médios de Solução em Segundos (Sol. Ótima)				
Modelo	Estra.1.1	Estra.1.2	Estra.1.3	Estra.1.4
Dim 1 {10}	0,852 (0,352792)	0,886 (0,379918)	0,812 (0,371806)	0,696 (0,255786)
Dim 2 {10}	91,9 (55,183431)	94,4 (46,45715)	64,8 (37,36844)	76,5 (55,10243)
Dim 4 {10}	3,8 (4,0221608)	4,1 (4,254409)	3,1 (3,142893)	3 (3,527668)
Dim 5 {10}	11,8 (5,4934304)	48,6 (55,29155)	55,7 (95,94912)	10,6 (6,552353)
Dim 6 {5}	777,6 (471,26882)	700,8 (346,9462)	571,8 (262,5218)	624,4 (296,4537)



• Tempo de Solução (Sem Solução Ótima)

Para as instâncias de dimensão 3 as estratégias 1.1 e 1.4 se destacam por terem obtido em média os dois melhores tempos (tabela 4.7). A estratégia 1.1 encontra soluções ótimas para 4 instâncias e a estratégia 1.4 encontra solução ótima para 3 instâncias. Lembrando da qualidade das soluções estas também foram as estratégias com melhores soluções inteiras (tabela 4.4 e 4.5). Nas instâncias de dimensão 6 apenas a estratégia 1.3 consegue um tempo melhor pois esta estratégia encontra solução ótima para uma instância.

Tabela 4.7: Tempos Médios de Solução em Segundos (Melhor Sol. Int.)				
Modelo	Estra.1.1	Estra.1.2	Estra.1.3	Estra.1.4
Dim 3	1564,2	1800	1800	1578,1
{10}	(316,74659)	(0)	(0)	(386,1386)
Dim 6	1800	1800	1662	1800
{5}	(0)	(0)	(308,5774)	(0)



• **Número de Nós Examinados (Solução Ótima)**

Para instâncias com solução ótima em todas as estratégias a estratégia 1.4 avalia menos nós em três dos cinco tipos de instâncias. A estratégia 1.1, avalia menos nós para os demais tipos (tabela 4.8).

Tabela 4.8: Número de Nós examinados (Sol. Ótima)				
Modelo	Estra.1.1	Estra.1.2	Estra.1.3	Estra.1.4
Dim 1 {10}	34,2 (5,672546)	38,6 (5,125102)	41,8 (6,545567)	85,5 (152,8749)
Dim 2 {10}	463,1 (603,44059)	921,3 (1220,618)	896,3 (1180,001)	456,7 (538,1691)
Dim 4 {10}	84,6 (68,838297)	117,7 (86,44851)	131,8 (98,13687)	96,2 (79,20971)
Dim 5 {10}	193,4 (226,60842)	3547,3 (4664,188)	5172,4 (10166,84)	292,2 (482,6208)
Dim 6 {5}	140609,8 (82614,519)	151480,4 (82661,91)	142281,4 (68786,84)	138274,2 (66944,29)

• **Número de Nós Examinados (Melhor Solução Inteira)**

Para as instâncias sem solução ótima a estratégia 1.1 também avalia menos nós (tabela 4.9).

Tabela 4.9: Número de Nós examinados (Melhor Sol. Int.)				
Modelo	Estra.1.1	Estra.1.2	Estra.1.3	Estra.1.4
Dim 3 {10}	1142,9 (596,25189)	1620,3 (1048,058)	3077,3 (1381,415)	2154,7 (1574,515)
Dim 6 {5}	334866,2 (16582,753)	377136,2 (50453,47)	424193 (55347,16)	416777 (30339,88)



• **Nó da melhor solução inteira (Ótima)**

Na tabela 4.10 observamos que o valor médio do nó da melhor solução inteira para todas as estratégias possui um desvio padrão alto, indicando uma grande variação entre as instâncias. A estratégia 1.1 encontra a solução ótima primeiro que as outras estratégias.

Tabela 4.10: Nó da melhor solução inteiras (Sol. Ótima)				
Modelo	Estra.1.1	Estra.1.2	Estra.1.3	Estra.1.4
Dim 1	0	0	35,5	33,1
{10}	(0)	(0)	(2,505549)	(5,95259)
Dim 2	279,7	685,8	503,3	205
{10}	(514,22737)	(825,4909)	(683,7301)	(171,0179)
Dim 4	18,4	111,6	70,1	68,2
{10}	(27,512421)	(122,3185)	(12,56494)	(46,69237)
Dim 5	133,2	2452	4090	256,9
{10}	(138,54787)	(2890,321)	(8153,541)	(468,0352)
Dim 6	82256,4	138743,4	128586	88444,2
{5}	(92596,924)	(93255,87)	(70298,76)	(92978,29)

• **Nó da melhor solução inteira**

O mesmo não ocorre para as instâncias de dimensão 3 e 6 (tabela 4.11). Para estas instâncias a melhor solução foi encontrada primeiro com a estratégia 1.3. Porém, se observarmos o valor da função objetivo (tabela 4.7) a qualidade da solução para as instâncias de dimensão 3 é pior com esta estratégia.

Tabela 4.11: Nó da melhor solução inteiras (Melhor Sol. Int.)				
Modelo	Estra.1.1	Estra.1.2	Estra.1.3	Estra.1.4
Dim 3	933,3	578,2	415,7	910,8
{10}	(648,58822)	(444,2699)	(36,42359)	(924,2901)
Dim 6	203116,2	297694,6	175708	304679,8
{5}	(86272,902)	(89915,27)	(63592,49)	(166963,6)



Estratégias tipo 2:

Assim como no primeiro tipo de estratégias, iremos dividir a análise nos casos com ou sem solução ótima em todas as instâncias. Neste segundo teste o CPLEX não obteve solução ótima para todas as instâncias de dimensão 6 e para uma das instâncias de dimensão 5.

• **Função Objetivo (Solução Ótima)**

Observamos uma pequena variação nas soluções ótimas com as estratégias tipo 2 também (tabela 4.12).

Tabela 4.12: Valor médio da FO (Sol. Ótima)				
Modelo	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim 1 {10}	91875,51 (892,3505)	91875,51 (1199,616)	91875,51 (211,1686)	91875,51 (217,9309)
Dim 2 {10}	933175,5 (1201,544)	933175,5 (1624,698)	933180,5 (278,1288)	933175,5 (290,2394)
Dim 4 {10}	195153,6 (6668,739)	195153,6 (10342,28)	195153,6 (16,72689)	195153,6 (21,66538)
Dim 5 {9}	478385,7 (1558,262)	478379 (2807,622)	478381,6 (65,63874)	478385,2 (89,94999)



• Função Objetivo (Melhor Solução Inteira)

A estratégia 2.2 encontrou as melhores soluções para as instâncias de dimensão 3. Para as instâncias de dimensão 6 a estratégia 2.4 foi melhor (tabela 4.13). Das 10 instâncias de dimensão 5, dentro dos limites de tempo e nós, o CPLEX não pode provar que a solução obtida é ótima.

Tabela 4.13: Valor médio da FO (Melhor Sol. Int.)				
Modelo	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim. 5 {1}	468.482	468.482	468.482	468.482
Dim 3 {10}	5.970.312 (678.956)	5.969.213 (678.607)	5.971.809 (678.889)	5.971.372 (679.272)
Dim 6 {10}	113.339 (5.243)	113.330 (5.243)	113.332 (5.243)	113.328 (5.242)



• Tempo (Solução Ótima)

Para as instâncias de dimensão 1 e 2 a estratégia 2.1 o tempo de CPU foi menor (tabela 4.14). Com esta estratégia o CPLEX obteve a solução ótima em quase um terço do tempo que as estratégias 2.3 e 2.4. Porém, nos problemas de dimensão 4 e 5 as estratégias 2.3 e 2.4 foram muito mais rápidas.

Tabela 4.14: Tempos de Solução em Segundos (Sol. Ótima)				
Modelo	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim 1	59	61	190,6	183
{10}	(65,5)	(64)	(235)	(239)
Dim 2	57,8	59,7	184,4	175,8
{10}	(72)	(70,7)	(258,7)	(262,9)
Dim 4	22,8	34	2	2
{10}	(69)	(105)	(2,9)	(3)
Dim 5	20,3	29	17,5	17,9
{9}	(22,8)	(41,7)	(16)	(16,2)

• Tempo (Sem Solução Ótima)

No caso dos problemas sem solução ótima, as estratégias 2.3 e 2.4 foram as únicas com destaque de menor tempo (tabela 4.15). O CPLEX encontra solução ótima para duas instâncias em cada estratégia.

Tabela 4.15: Tempos de Solução em Segundos (Melhor Sol. Int.)				
Modelo	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim 3	1800	1800	1782,098	1800
{10}	(0)	(0)	(56,61109)	(0)
Dim 6	1800	1800	1800	1797,239
{10}	(0)	(0)	(0)	(8,731049)
Dim5	1800	1800	116,1	102,14
{1}				



• **Número de Nós examinados (Solução Ótima)**

O número de nós gerado na árvore Branch and Bound é menor com as estratégias 2.3 e 2.4, cada uma tem menos nós examinados em dois tipos de dimensão (tabela 4.16).

Tabela 4.16: Número de nós examinados (Sol. Ótima)				
Modelo	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim 1 {10}	500,2 (892,4)	639,5 (1199,7)	119,3 (211,2)	118,5 (217,9)
Dim 2 {10}	838,7 (1201,6)	1077,2 (1624,7)	207,9 (278,1)	205,3 (290,2)
Dim 4 {10}	2182,5 (6668,7)	3347,4 (10342,3)	14,7 (16,7)	16,5 (21,7)
Dim 5 {9}	1281,9 (1558,3)	1945 (2807,6)	72,2 (65,6)	82,3 (89,9)

• **Número de Nós Examinados (Melhor Solução Inteira)**

A estratégia 2.3 consegue avaliar menos nós que as outras estratégias e sua diferença com relação as estratégias 2.1 e 2.2 é bem grande (tabela 4.17).

Tabela 4.17: Número de nós examinados (Melhor Sol. Int.)				
Modelo	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim 3 {10}	2.614,6 (1800,204)	4.025,7 (2071,755)	353,9 (72,586)	364,5 (84,786)
Dim 6 {10}	235.983,5 (9240,444)	306.102,4 (11271,45)	31.457,3 (3739,506)	43.945,2 (4381,755)
Dim 5 {1}	135.615	146.178	677	751



• **Nó da melhor solução inteira (Ótima)**

De maneira geral a estratégia 2.3 encontra a melhor solução inteira (tabela 4.18) primeiro, exceto para os problemas de dimensão 2.

Tabela 4.18: Nó da melhor sol inteira encontrada (Sol. Ótima)				
Modelo	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim 1	7	1	1	1
{10}	(22,1)	(3,2)	(3,3)	(3,2)
Dim 2	103	43	254,7	66,1
{10}	(210,9)	(62,9)	(587)	(86,4)
Dim 4	12	7	6,9	6,9
{10}	(24,9)	(10,6)	(11,3)	(11,3)
Dim 5	874,7	197	55	60,7
{9}	(925,2)	(139,9)	(47,8)	(54,2)

• **Nó da melhor solução inteira**

Para os casos sem solução ótima as estratégias 2.3 e 2.4 são melhores (tabela 4.19). As estratégias 2.1 e 2.2 encontram suas melhores soluções inteiras em nós bem mais fundos da árvore.

Tabela 4.19: Nó da melhor sol inteira encontrada (Melhor Sol. Int.)				
Modelo	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim 3	1051,6	1493,7	173	190
{10}	(924,695169)	(1259,324)	(103,0696)	(116,9045)
Dim 6	140.357	102.572,4	18.792,1	14.884,9
{10}	(79662,1919)	(88101,15)	(11373,86)	(14177,74)
Dim 5	134846	18300	555	160
{1}				

• **Flow Cuts**

O número de planos de corte gerados em cada dimensão não variou quando houve mudança de estratégia (ver tabela 4.20 abaixo). O corte gerado é sempre



do tipo flow cut, apesar do CPLEX incluir outros tipos de planos de corte.

Tabela 4.20: Número de Flow Cuts gerados						
Modelo	Estra.1.1	Estra.1.4	Estra.2.1	Estra.2.2	Estra.2.3	Estra.2.4
Dim 1	80,4	80,4	80,4	80,4	80,4	80,4
{10}	(6,2)	(6,2)	(6,2)	(6,2)	(6,2)	(6,2)
Dim 2	542,5	542,5	542,5	542,5	542,5	542,5
{10}	(20,2)	(20,2)	(20,2)	(20,2)	(20,2)	(20,2)
Dim	1317,2	1317,2	1317,2	1317,2	1317,2	1317,2
{10}	(114,8)	(114,8)	(114,8)	(114,8)	(114,8)	(114,8)
Dim	100,9	100,9	100,9	100,9	100,9	100,9
{10}	(31,7)	(31,7)	(31,7)	(31,7)	(31,7)	(31,7)
Dim 5	372,8	372,8	372,8	372,8	372,8	372,8
{10}	(32,6)	(32,6)	(32,6)	(32,6)	(32,6)	(32,6)
Dim	648	48	48	48	48	48
{10}	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)	(4,1)

Pelos resultados obtidos nas tabelas 4.6, 4.7, 4.14 e 4.15 observamos que a variação de tempos de solução entre as estratégias tipo 1 é menor que entre as estratégia tipo 2 para a maioria dos tipos de instâncias consideradas. O tempo de CPU necessário para resolver as instâncias de dimensão 2 usando as estratégias tipo 1 varia entre 64,8 e 94,4 enquanto que usando as estratégias tipo 2 o tempo varia entre 57,8 e 184,4. Nas instâncias de dimensão 4 o tempo de solução usando a estratégia 2.2 é 17 vezes maior que usando as estratégias 2.3 e 2.4. Podemos notar também que apesar das estratégias 2.1 e 2.2 terem sido melhores para as instâncias de dimensão 2 e, as estratégias 2.3 e 2.4 terem sido melhores para as instâncias de dimensão 4, de uma maneira geral as estratégias tipo 2 aumentaram o tempo de CPU necessário para resolver os problemas. Isto sugere que a estratégia de busca na árvore de decisão do Branch and Bound pode influenciar muito a solução do



modelo.

Constatamos que usando a estratégias 1.1 (padrão), o CPLEX avalia menos nós na árvore Branch and Bound para a maioria das instâncias e com a estratégia 1.4 o CPLEX usa o menor tempo de CPU para a maioria das instâncias. Estas duas estratégias fazem uso de planos de corte.

O uso das estratégias 2.3 e 2.4 gerou uma árvore de busca menor com qualidade de solução similar. No entanto a estratégia 2.3 é que se destaca, quando comparada também com as estratégias 1.1 e 1.4, em relação ao número de nós examinados em problemas que não possuem a solução ótima (tabela 4.17).

Um novo teste foi realizado unindo as estratégias 1.4 e 2.3 (Estratégia 3.1). Os resultados obtidos em termos de número de nós examinados, nó da melhor solução inteira encontrada, nó da primeira solução inteira encontrada, tempo de CPU e o valor da função objetivo estão nas tabelas 4.21 e 4.22 a seguir.

Tabela 4.21: Estratégia 3.1 (Sol. Ótima)				
Modelo	Nos Exam	No Sol. Int.	Tempo	F.O.
Dim 1 {10}	35,3 (4,1)	35,2 (4,2)	1,6 (0,6)	91875,5 (7283,8)
Dim 2 {10}	252 (252,2)	140,6 (35,1)	179,6 (227)	933183,7 (80567,7)
Dim 4 {10}	57,7 (33)	52 (15,9)	6,4 (4,8)	195153,8 (21565,4)
Dim 5 {10}	144,4 (192)	120,5 (153,9)	31,9 (33,5)	477391 (46275,7)
Dim 3 {1}	350	271	1445	7.027.572



Tabela 4.22: Estratégia 3.1 (Melhor Sol. Int.)				
Modelo	Nos Exam	No Sol. Int.	Tempo	F.O.
Dim 3	414,4	243,1	1800	5852579,5
{9}	(81)	(32,3)	(0)	(602665,6)
Dim 6	31309,2	14611,1	1800	113332
{10}	(3892,1)	(13595,23)	(0)	(5242,3)

Em termos de número de nós examinados e o nó da melhor solução inteira, a estratégia 3.1 foi melhor que 1.1 e 1.4 na maioria das instâncias e melhor que 2.3 nas instâncias de dimensão 6. Esta nova estratégia apresenta também o melhor valor médio da função objetivo para as instâncias de dimensão 3 e 6, quando comparadas com as estratégias 1.4 e 2.3 utilizadas isoladamente. Para os casos de dimensão 3, a estratégia 3.1 obteve o mesmo resultado que a estratégia 2.3. O uso da estratégia 3.1 impediu que o CPLEX encontrasse a solução ótima para as instâncias de dimensões 6.

Fazendo uma análise dos resultados obtidos acima considerando as dimensões das instâncias, observamos uma variação no processo de solução. Por exemplo as instâncias de dimensão 4 e 6 possuem o número total de variáveis e restrições muito similares, porém o grau de dificuldade de solução dos dois modelos é muito diferente. Apenas 5 das 10 instâncias de dimensão 6 foram resolvidas, e mesmo assim com um tempo médio de CPU 190 vezes maior. Observe ainda que as instâncias de dimensão 5, que possuem por volta do dobro do número total de variáveis e restrições que as instâncias de dimensão 6, são resolvidas 102 vezes mais rápido que as instâncias de dimensão 6. Este fato sugere que a dificuldade deste modelo não depende apenas do número de variáveis e restrições mas, depende também da interrelação entre as variáveis e restrições. Uma observação que podemos fazer é que um horizonte de planejamento mais longo permite uma flexibilidade de planejamento e portanto torna-se mais fácil de ser feito.

Na próxima seção apresentamos os resultados obtidos com o uso das inequa-



ções supermodulares que foram deduzidas explorando a estrutura do problema, como planos de corte. Nos testes foram usadas as instâncias de dimensão 1, 4, 5 e 6. As instâncias de dimensão 2 e 3 não foram utilizadas devido a problemas de memória.

4.3. O uso da Biblioteca de Soluções do CPLEX

Para resolver o problema de otimização linear inteira, foi utilizado o solver CPLEX da IBM.

Para permitir a compilação do programa, foi necessário utilizar o compilador CPLEX e o gerador de código de máquina. O compilador CPLEX é um compilador de código de máquina que gera código de máquina para o processador de 32 bits.

O programa principal para implementar o modelo de programação linear inteira é o programa principal. O programa principal é o programa principal que resolve o problema de otimização linear inteira. O programa principal é o programa principal que resolve o problema de otimização linear inteira.

Exemplo 4.1 (versão 1): Para Subproblema 1

Definir o tipo de

problema de otimização

de otimização

Definir o tipo de

problema de

problema de

problema de

4.3 O Uso de Inequações Supermodulares na Solução do Modelo FR

O objetivo desta seção é apresentar os resultados da solução do modelo FR utilizando planos de corte gerados a partir da subestrutura do modelo e dar uma breve explicação da implementação realizada.

4.3.1 O uso da Biblioteca de Subrotinas do CPLEX

Daremos a seguir os passos iniciais necessários para utilização da biblioteca de subrotinas do CPLEX 6.6.

Para permitir a compilação do programa fonte que utiliza as subrotinas do CPLEX é necessário que os arquivos `cplex.dll`, `cplex.lib` e `cplex.h` estejam na mesma pasta que o arquivo fonte. Além disso, a biblioteca `cplex.lib`, deve ser colocada junto com as bibliotecas `.lib` do compilador em uso.

Os primeiros passos para implementar um programa usando a linguagem C (Schildt, 1997) são: Incluir as bibliotecas básicas da linguagem `< ctype.h >`, `< stdio.h >`, `< stdlib.h >`; abrir e fechar o programa principal que tem o nome `main`. Um `return(0)` indica ao compilador o fim do programa principal.

Exemplo 4.1 (versão 1): *Prog_SubrotinaCPX*

```
#include < ctype.h >
#include < stdio.h >
#include < stdlib.h >
```

```
int main( void )
{
    return(0);
}
```



O primeiro passo para usar as subrotinas do CPLEX é incluir o arquivo `< cplex.h >` e abrir e fechar o ambiente CPLEX.

Exemplo 4.1 (versão 2): *Prog_SubrotinaCPX*

```
#include < cplex.h >
#include < ctype.h >
#include < stdio.h >
#include < stdlib.h >

int main( void )
{
    \*Declaramos as variáveis de parâmetros*\
    CPXENVptr env;
    int status;

    \*Rotina que abre o CPLEX*\
    env = CPXopenCPLEX (&status);

    \*Rotina que fecha o CPLEX*\
    status = CPXcloseCPLEX (&env);

    return(0);
}
```

A seguir incluímos a subrotina que cria o problema, `CPXcreateprob`. Esta subrotina é extremamente necessária e sem ela não há sucesso com os programas. Ela cria um problema imaginário que será substituído por um problema real através da subrotina de leitura `CPXreadcopyprob`. O manual do CPLEX contém informações detalhadas sobre os parâmetros necessários.



Exemplo 4.1 (versão 3): *Prog_SubrotinaCPX*

```
#include < cplex.h >
#include < ctype.h >
#include < stdio.h >
#include < stdlib.h >

int main( void )
{
    \*Declaramos as variáveis de parâmetros*\

    CPXENVptr env;

    int status;
    char *filename;

    \*Alocação de memória para o nome do problema.*\

    filename=malloc(22*sizeof(char));

    \*Rotina que abre o CPLEX*\

    env = CPXopenCPLEX (&status);

    \*Leitura do nome do problema.*\
    printf("Nome do arquivo para leitura: ");
    scanf("%s",filename);

    lp = CPXcreateprob (env, &status, filename);

    if (lp == NULL )
    {
        printf("Falha na criação do problema");
        printf("ABANDONANDO O PROGRAMA");
        status = CPXcloseCPLEX (&env);
        exit(1);
    }

    status = CPXreadcopyprob (env, lp, filename, NULL);
```



```
\*Rotina que fecha o CPLEX*\nstatus = CPXcloseCPLEX (&env);\nreturn(0);\n}
```

Observe que colocamos uma mensagem de erro ao utilizar cada subrotina. Para identificar erro as rotinas retornam um valor zero ou diferente de zero dependendo se houve sucesso ou não. Na subrotina que cria o problema por exemplo, é retornado um valor nulo se houver erro. É importante sempre fechar o ambiente CPLEX antes de finalizar o programa.

Após compilar, testamos o programa com exemplos diferentes. Observamos que apesar de estarmos alocando um bom espaço para o nome do problema o programa não lia nomes com mais de 7 caracteres, desconsiderando o ponto e a extensão do problema como caracter. Sendo assim, é melhor criar problemas com nomes menores.

Estes são os principais cuidados observados durante a construção de nosso algoritmo.

4.3.2 O algoritmo SolFR

Utilizamos as subrotinas do CPLEX para facilitar o trabalho de implementação do algoritmo SolFR.

Observando o modelo FR descrito na seção 3.2, vemos que as restrições (3.2), (3.6 ou 3.7), (3.8 ou 3.11) e (3.16) são as mesmas do modelo PMCSS (seção 1.1). Isto é, o modelo FR possui como subestrutura o modelo PMCSS. Se fixarmos um i e um j temos também como subestrutura o modelo PUCSS, seção 1.1. As Inequações Supermodulares, descritas na subseção 2.2.3, são válidas para PUCSS e portanto podem também ser utilizadas como planos de corte para FR.

A geração de um plano de corte é feita escolhendo inicialmente um sabor e um tamanho de refrigerante, índices i e j respectivamente, entre as variáveis



binárias mais infactíveis. Estas variáveis são as variáveis de setup e startup que dependem do sabor ou do tamanho. O período t é determinado pela variável de estoque "mais infactível".

Se o uso das inequações supermodulares como planos de corte não resolve o problema após um número máximo de iterações, o processo de solução continua com o algoritmo Branch and Cut do CPLEX. Aplicamos portanto as inequações supermodulares apenas no nó raiz. O método Branch and Bound é livre para gerar outros tipos de planos de corte nos nós onde identificar necessidade e a possibilidade.

A seguir está apresentado o algoritmo SolFR implementado, em cada passo indicamos a subrotina do CPLEX usada.

O número máximo de iterações para gerar inequações supermodulares é definido pelo parâmetro $Max1$.

Algoritmo SolFR

Inicialização: *Abrir o CPLEX. It=1.*

Passo 1: *Cria o problema. (Subrotina CPXcreatprob)*

Passo 2: *Lê um problema inteiro misto (MIP). (Subrotina CPXreadcopyprob)*

Passo 3: *Relaxa o problema. (Subrotina CPXchgprobtype)*

Passo 4: *Resolve a relaxação linear. (Subrotina CPXprimopt)*

Se a solução é inteira, para. O problema está resolvido.

Se não, guarda a solução da RL para ser usada nos Passos 5 e 6.

Passo 5:

Se $It \leq Max1$.

*Escolhe uma combinação sabor/tamanho para identificar
as inequações válidas.*

Identifica as inequações válidas e passa para o passo 6.

Senão vai para o passo 9.

Passo 6: *Verifica se as inequações válidas são violadas pela solução da RL.*

Se violadas vá para o passo 7.



Senão, $it=it+1$ e volta para o passo 5.

Passo 7: *Retorna o problema para MIP. (Subrotina CPXchgprobtype)*

Passo 8: *Adiciona os planos de corte ao problema; $it=it+1$ e volta para o passo 3. (Subrotina CPXaddrows)*

Passo 9: *Aplica o método Branch and Bound. (Subrotina CPXmipopt)*

O passo 7 é necessário pois após utilizar as subrotinas de relaxação não é possível utilizar nenhuma outra subrotina de modificação do problema antes de retornar o problema para a forma original.

Os passos 5 e 6 resolvem o problema de separação associado à identificação de planos de corte (Algoritmo IISE - subseção 2.2.3).

Aplicamos o algoritmo SolFR para resolver as instâncias de dimensão 2, 4, 5 e 6 cujos dados estão descritos na tabela 4.1. Testamos duas estratégias diferentes:

- estratégia 4.1 - inequações supermodulares e a estratégia 1.1 na subrotina CPXmipopt (passo 9).
- estratégia 4.2 - inequações supermodulares e a estratégia 3.1 na subrotina CPXmipopt (passo 9).

O número de iterações para a geração de inequações supermodulares no nó raiz foi fixado em 30 ($Max1 = 30$).

Apresentamos nas tabelas 4.23 a 4.26 o número de flow cuts (Flow C.) e o número de inequações supermodulares gerados (Ineq.Super.), o total de cortes adicionados (TC), o nó da melhor solução inteira (N.Int.), o número de nós examinados (N.Exam.) e o valor da função objetivo obtidos (F.O.). Neste experimento não conseguimos computar o tempo de CPU. Os resultados foram divididos em instâncias com e sem solução ótima.



4.3. O Uso de Inequações Supermodulares na Solução do Modelo FR

Tabela 4.23: Estratégia 4.1 (Sol. Ótima)

Modelo	Flow C.	Ineq. Super.	T. C.	N. Int.	N. Exam.	F.O.
Dim 1	0	54,3	54,3	1	20	91875,5
{10}	(0)	(10,52)	(10,52)	(3,2)	(15,7)	(7283,8)
Dim 4	0,1	128,2	128,3	26,2	129,7	195153,7
{10}	(0,32)	(22,25)	(22,2)	(21,5)	(270,4)	(21565,3)
Dim 5	2,375	251,625	260,25	235,625	247,125	482983,4
{8}	(1,9)	(32,8)	(4295,742)	(227,2)	(218,7)	(50239)

Tabela 4.24: Estratégia 4.1 (Melhor Sol. Int.)

Modelo	Flow C.	Ineq. Super.	T. C.	N. Int.	N. Exam.	F.O.
Dim 5	3,5	244,5	248	62,5	925,5	455062
{2}	(3,4)	(21,9)	(18,4)	(17,7)	(1149,05)	(19003,9)
Dim 6	0	81,2	81,2	117820,5	191609,6	113336,2
{10}	(0)	(8,08)	(8,08)	(51317)	(21665,2)	(5241)

Tabela 4.25: Estratégia 4.2 (Sol. Ótima)

Modelo	Flow C.	Ineq. Super.	T. C.	N. Int.	N. Exam.	F.O.
Dim 1	0	54,3	54,3	23,2	23,7	91875,5
{10}	(0)	(10,5)	(10,5)	(9,1)	(9,8)	(7283,8)
Dim 4	0,1	128,2	128,3	45,5	47,7	195153,8
{10}	(0,32)	(22,3)	(22,2)	(39,8)	(46,1)	(21565,35)
Dim 5	2,6	250,2	252,8	153,7	192,4	477390,7
{10}	(2,1)	(30)	(29,7)	(273,1)	(270)	(46276,2)
Dim 6	0	86	86	8447,5	17107,5	117781,8
{2}	(0)	(4,24)	(4,24)	(2758,424)	(84,146)	(8735,147)



Tabela 4.26: Estratégia 4.1 (Melhor Sol. Int.)						
Modelo	Flow C.	Ineq. Super.	T. C.	N. Int.	N. Exam.	F.O.
Dim 6	0	80	80	9017,5	16718,25	99501,47
{8}	(0)	(8,552)	(8,552)	(4695,868)	(969,854)	(35877,53)

Observe que o CPLEX gerou cortes apenas nas instâncias de dimensão 4 e 5. A inclusão das Inequações Supermodulares nas instâncias 1 e 6 que possuem menos restrições inibiu a geração de cortes do CPLEX.

O uso da estratégia 4.1 diminui o número de nós examinados nas instâncias de dimensão 1 e nas instâncias de dimensão 6, se comparadas com o número de nós examinados pela estratégia 1.1 nas instâncias de dimensão 6 sem solução ótima (tabela 4.5). Porém, com esta estratégia o número de instâncias com solução ótima nas dimensões 5 e 6 é reduzido. Utilizando a estratégia 1.1, dez instâncias com dimensão 5 e cinco instâncias com dimensão 6 tem solução ótima (ver tabelas 4.4 e 4.5).

A aplicação da estratégia 4.2 nas instâncias 5 e 6 melhorou o número de instâncias com solução ótima com relação as estratégias 4.1 e 2.3. Todas instâncias de dimensão 5 e duas instâncias de dimensão 6 foram resolvidas até a prova da otimalidade.

O tamanho da árvore de decisão que a estratégia 4.2 gera é menor que a estratégia 1.4 para as dimensões 1, 4, 5 e 6; menor que a árvore de decisão gerada pela estratégia 2.3 para instâncias de dimensão 1 e 6; e menor que a árvore gerada pela estratégia 3.1 em todos os tipos de instâncias 1, 4, 5 e 6. A melhor solução inteira foi obtida mais rapidamente, isto é em níveis menores da árvore de busca, quando comparamos a estratégia 4.2 com a estratégia 1.4, em quatro tipos de instâncias testadas; em relação a estratégia 3.1 a estratégia 4.2 encontrou a melhor solução inteira em três tipos de instâncias.

Obsevamos então que o uso de planos de corte específicos para o modelo FR e uma estratégia de escolha de nós e variáveis podem diminuir o tamanho da árvore



de decisão do Branch and Bound e resultar em soluções de boa qualidade.

Capítulo 5

Considerações Finais

Neste capítulo, foram discutidos os aspectos de implementação do algoritmo de um programa para a solução do modelo FR. O programa foi desenvolvido em linguagem C++ e foi executado no ambiente de desenvolvimento de código de programação e gerenciamento de projetos, o Visual Studio 2010.

Trabalhou-se com o algoritmo de Branch and Bound, que é um algoritmo de busca em árvore de decisão. O algoritmo de Branch and Bound é um algoritmo de busca em árvore de decisão que utiliza a técnica de poda para reduzir o número de nós a serem explorados. O algoritmo de Branch and Bound é um algoritmo de busca em árvore de decisão que utiliza a técnica de poda para reduzir o número de nós a serem explorados.

O modelo FR foi implementado no programa de computador. O programa de computador foi desenvolvido em linguagem C++ e foi executado no ambiente de desenvolvimento de código de programação e gerenciamento de projetos, o Visual Studio 2010. O programa de computador foi desenvolvido em linguagem C++ e foi executado no ambiente de desenvolvimento de código de programação e gerenciamento de projetos, o Visual Studio 2010. O programa de computador foi desenvolvido em linguagem C++ e foi executado no ambiente de desenvolvimento de código de programação e gerenciamento de projetos, o Visual Studio 2010.

O desenvolvimento de um programa de computador para a solução do modelo FR é um trabalho que requer um conhecimento profundo em programação e em matemática. O programa de computador foi desenvolvido em linguagem C++ e foi executado no ambiente de desenvolvimento de código de programação e gerenciamento de projetos, o Visual Studio 2010.

Capítulo 5

Considerações Finais

Neste trabalho formulamos um modelo de planejamento da produção de um problema real da empresa de refrigerantes Arco Íris, situada na região de São José do Rio Preto. A empresa colaborou nas discussões sobre a formulação do modelo fornecendo detalhes da linha de produção e permitindo visitas as suas instalações.

Testamos a solução do modelo obtido usando inequações válidas derivadas a partir da estrutura do modelo; e através do algoritmo branch and cut incluído no software CPLEX. Os resultados obtidos indicam que explorar a estrutura do problema pode contribuir no processo de solução.

O modelo FR formulado permite ainda vários estudos. Neste trabalho abordamos apenas o dimensionamento dos lotes com custo de setup fixo; os custos de start up não foram considerados. Além disso, consideramos tempos/custos de setup independentes da sequência da produção. Por isso, é interessante rever a contagem do tempo e dos custos de setup e start up. Um outro importante tópico para trabalhos futuros seria incluir o sequenciamento da produção no modelo FR, a exemplo do modelos descritos na seção 1.2.

O desenvolvimento de uma heurística baseada na estrutura do modelo e que gere boas soluções inteiras é um aspecto importante a ser considerado. Esta solução poderia ser útil não apenas como um retorno rápido para o Engenheiro de Produção mas também para acelerar o algoritmo Branch and Bound.



Todos os estudos, as pesquisas realizadas e as dificuldades encontradas foram importantes para meu amadurecimento matemático e de pesquisa de uma maneira geral.

Referências Bibliográficas

- [1] ALFREDO F. RAYDON, E. - SILVA G. Planejamento de Produção de uma Fábrica de Bolachas. *Revista de Estatística da UNICAMP*, Campinas, 10(1), setembro, p. 189, 1974.
- [2] ALMEIDA M. T. e OLIVEIRA M. A. Correlação linear por meio de máquinas/programa de solução de problemas, *Computers Ops Res*, 10(1), 1983.
- [3] AMARAL L.E. Introdução a Pesquisa Operacional. Método e Modelos para a Análise de Decisão. Editora Mac Graw Hill, 1994.
- [4] ARABO E. A. e AMARAL M. E. Problemas de Distribuição de Itens. *Memórias em Honra do Professor Modestino de Faria*, Universidade Federal de Pernambuco, Recife, 1984.
- [5] BRAPPO, V.L. M., R. Z. de Aguiar, 2001, p. 101-104.
- [6] DAHL G. G., RITENBERG L.E. e COSTA J. S. D. Determining Job Size and Resource Requirements: A Review, *Journal of Operational Research Society*, 13(1), 1967.
- [7] DANTZIG, G.B. e WOLFE, P. *Linear Programming and Extensions*, John Wiley & Sons, 1963.



Referências Bibliográficas

- [1] AKIYAMA F., RANGEL S. e SILVA G. Planejamento da Produção de Uma Fábrica de Refrigerantes, **Anais do XXI CNMAC**, Caxambú, MG, setembro, p.189, 1998.
- [2] ALMEIDA M. T. e CENTENO M. A composite heuristic for the single machine early/tardy job scheduling problem, **Computers Ops Res**, 625-635, 1998.
- [3] ANDRADE L.E. **Introdução à Pesquisa Operacional Métodos e Modelos para a Análise de Decisão**, Editora, Rio de Janeiro, 2a ed, 2000.
- [4] ARAÚJO S. A. e ARENALES M. N. Problema de Dimensionamento de Lotes Monoestágio com Restrição de Capacidade: Modelagem, Método de Resolução e Resultados Computacionais, **Pesquisa Operacional SO-BRAPO**, Vol. 20, No 2, dezembro, 2000, p.287-306.
- [5] BAHL H.C., RITZMAN L.P. e GUPTA J.N.D. Determining Lot Sizes and Resource Requirements: A Review, **Journal of Operational Research Society**, 35(3), 329-345, 1987.
- [6] BARANY I., VAN ROY T. e WOLSEY L. Uncapacitated Lot-Sizing: The convex hull of solutions, **Mathematical Programming Study**, North Holland, 22, 32-43, 1984.



- [7] BELVAUX G. e WOLSEY L.A. bc-prod: A specialized Branch and Cut system for lot-sizing Problem, **Management Science**, 46(5), 724-738, 2000.
- [8] BERTSEKAS D., **Linear Network Optimization - Algorithms and Codes**, Massachusetts Institute of Technology, 1992.
- [9] BERTSIMAS D. e TSITSIKLIS J.N., **Introduction to Linear Optimization**, Athena Scientific, 1997.
- [10] BITRAN G.R. e YANASSE H.H., Computational Complexity of the Lot Size Problem, **Management Science**, 28(10), 1174-1186, 1982.
- [11] CAMPELLO R.E. e MACULAN N. **Algoritmos e Heurísticas - Desenvolvimento e Avaliação de Performance**, Editora Universidade Federal Fluminense, Niterói, RJ, 1994.
- [12] CLARK A. R. Diminishing Neighbourhood search in Canning Line Production Planning, **Engineering and Mathematical Science**, 2001.
- [13] CONSTANTINO M. A cutting plane approach to capacitated lot-sizing with start-up costs, **Mathematical Programming**, 353-376, 1996.
- [14] CROWDER H., JOHNSON E.L., e PADBERG M.W. Solving Large-Scale Zero One Linear Programming Problems, **Operations Research**, 31(5), 803-834, 1983.
- [15] DILLENBERGER C., et al On Practical resource allocation for production planning and scheduling with period overlapping setups, **European Journal of Operational Research**, 275-286, 1994.
- [16] DREXL A. e HAASE K. Proportional Lotsizing and Scheduling, **Int. J. Production Economics**, 40, 73-87, 1995.
- [17] DREXL A. e KIMMS A. Lot Sizing and Scheduling - Survey and Extensions, **European Journal of Operational Research**, 99 (1997), 221-235.



- [18] FERREIRA C. E. e WAKABAYASHI Y. *Combinatória Poliédrica e Planos de Corte Faciais*, **X Escola de Computação**, ed. IMECC-UNICAMP, Campinas, SP, 1996.
- [19] FISHER M.L., The Lagrangian Relaxation Method for Solving Integer Programming Problems, **Management Science**, 27 (1), 1981.
- [20] FLEISCHMANN B. The Discrete Lot-sizing and Scheduling Problem, **European Journal of Operational Research**, 44 (1990), 337-348.
- [21] FLEISCHMANN B. e MEYR H. The General Lotsizing and Scheduling Problem, **OR Spektrum**, 19 (1997), 11-21.
- [22] GAREY M. R. e JOHNSON D.S. **Computers and Intractability - A Guide to the Theory of NP-Completeness**, W.H. Freeman and Company, New York, 1977.
- [23] GOLDBARG M.C. e LUNA H.P.L. **Otimização Combinatória e Programação Linear - Modelos e Algoritmos**, Campus, 2000.
- [24] GOPALAKRISHNAN M. et all. A Tabu-Search Heuristic for the Capacitated Lot-Sizing Problem with Set-up Carryover, **Management Science**, 47 (6), 851-863, 2001.
- [25] HAASE K. Capacitated Lot-Sizing with Sequence Dependent Setup Costs, **OR Spektrum**, 18 (1996), 51-59.
- [26] HAASE K. e KIMMS A. Lot Sizing and Scheduling With Sequence-dependent Setup Costs and Times and Efficient Rescheduling Opportunities, **Int. J. Production Economics**, 66 (2000), 159-169.
- [27] ILOG **Using the CPLEX6.6 Callable Library**, Copyright ILOG, 2000.
- [28] KARMARKAR U.S. e SCHRAGE L. The Deterministic Dynamic Product Cycling Problem, **Operations Research**, 33 (2), march-april, 1985.



- [29] KUIK R., SALOMON M. e VAN WASSENHOVE L.N. Batching Decision: Structure and Models, **European Journal of Operational Research**, 75, 234-263, 1994.
- [30] LASDON L. S. **Optimization Theory for Large Systems**, Macmillan, New York, 449-455, 1970.
- [31] LINDO SYSTEMS **Lingo - The Modeling Language and Optimizer**, versão 6.0, 1999.
- [32] MAXIMAL SYSTEM, **MPL Modeling System**, Maximal Software, versão 4.0, 1998.
- [33] MAZZINI R. e ARMENTANO V.A. A heuristic for single machine scheduling with early and tardy costs, **European Journal of Operational Research**, 129-146, 2001.
- [34] MEYR H. Simultaneous Lotsizing and Scheduling by Combining Local Search with Dual Reoptimization, **European Journal of Operational Research**, 120 (2000), 311-326.
- [35] METROPOLIS N. et all. Equation of state calculation by fast computing machines, **J. of Chem. Phys.**, 21, 1087-1091, 1953.
- [36] MÜLLER F.M. Introdução aos Problemas de Sequenciamento, Minicurso apresentado junto ao DCCE/IBILCE/UNESP, in Seminários do Grupo de Otimização, São José do Rio Preto, outubro, 2001.
- [37] NEMHAUSER G.L. e WOLSEY L.A. **Integer and Combinatorial Optimization**, John Wiley & Sons, 1988.
- [38] OZDAMAR L. e ALI BOZYEL M. The capacitated lot sizing problem with overtime decisions and setup times, **IIE Transactions**, 32 (11), 1043-1057, 2000.



- [39] PADBERG M. e RINALDI G., A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems, **Society for Industrial and Applied Mathematics**, 33 (1), 60-100, 1991.
- [40] PARKER R.G. e RARDIN R. L. **Discrete Optimization**, Academic Press, New York, 1988.
- [41] PATTLOCH M., SCHMIDT G. e KOVALYOV Y. Heuristic algorithms for lotsize scheduling with applications in the tobacco industry, **Computer & Industrial Engineering**, 39, 235-253, 2001.
- [42] POCHET Y e WOLSEY L. A. . Solving Multi-item Lot-Sizing Problems using strong cutting planes, **Management Science**, vol 37, No 1, Janeiro, 1991.
- [43] POTTS C.N. e VAN WASSENHOVE L.N. Integrating Scheduling with Batching and Lot-Sizing: a Review of Algorithms and Complexity, **Journal of Operational Research Society**, 43 (5), 395-406, 1992.
- [44] PUCCINI A.L. e PIZZOLATO N.D. **Programação Linear**, Livros Técnicos e Científicos Editora S.A., RJ, SP, 1987.
- [45] RANGEL S. **Solving Integer Programming Problems Using Preprocessing and Cutting Planes: Theory and Implementation of Branch and Bound**, Tese Ph.D., Departamento de Matemática e Estatística, Universidade de Brunel, Uxbridge, UK, 1995.
- [46] REEVES C. (editor) **Modern Heuristic Techniques for Combinatorial Problems**, John Wiley & Sons, 1993.
- [47] SCHRAGE L. **Optimization Modeling with Lingo**, Lindo Systems, 2001.



- [48] SIKORA R., CHHJED D. e SHAW M. J. Integrating the lot-sizing and sequencing decisions for scheduling a capacitated flow line, **Computers & Industrial Engineering**, 30 (4), 659-679, 1996.
- [49] SORIC K. A cutting plane algorithm for a single machine scheduling problem, **European Journal of Operational Research**, 383-393, 2000.
- [50] STAGGEMEIER, A.T. e CLARK, A.R. A survey of lot-sizing and scheduling models, **Anais do XXXIII SBPO**, Campos do Jordão, SP, novembro, 938-947, 2001.
- [51] TOLEDO C.F.M. Problema Dimensionamento de Lotes e Programação da Produção, Seminário apresentado junto ao DCCE/IBILCE/UNESP, in Seminários do Grupo de Otimização, São José do Rio Preto, novembro, 2001.
- [52] TRIGEIRO W.W., THOMAS L.J. e McCLAIN J.O. Capacitated Lot Sizing With Setup Times, **Management Science**, 35(3), 353-366, 1989.
- [53] VANDERBECK F. e WOLSEY L. A. Valid inequalities for the Lasdon-Terjung production model, **J. Opl. Res. Soc.**, Vol 43, 435-441, 1992.
- [54] VANDERBEI R.J., **Linear Programming - Formulations and Extensions**, Kluwer Academic Publishers, 2ª ed., 2001.
- [55] VAN ROY T. J. e WOLSEY L. A. Solving mixed integer programming problems using automatic reformulation, **Operations Research**, vol. 35, No 1, Janeiro-Fevereiro, 1987.
- [56] VON ELLENRIEDER A., **Pesquisa Operacional**, Ed. Almeida Neves, Rio de Janeiro, 1971.
- [57] WAGNER H.M. **Pesquisa Operacional**, 2ª edição, editora Prentice/Hall do Brasil, 1986.



- [58] WILLIAMS H.P., **Model Building in Mathematical Programming**,
John Wiley & Sons, 3ª ed., 1991.



