Notas em Matemática Aplicada e-ISSN 2236-5915

Volume 18, 2012

Editado por

Eliana X.L. de Andrade
Universidade Estadual Paulista - UNESP
São José do Rio Preto, SP, Brasil

Rubens Sampaio
Pontifícia Universidade Católica do Rio de Janeiro -
Rio de Janeiro, RJ, Brasil

Geraldo N. Silva
Universidade Estadual Paulista - UNESP
São José do Rio Preto, SP, Brasil

Sociedade Brasileira de Matemática Aplicada e Computacional

2012

A Sociedade Brasileira de Matemática Aplicada e Computacional - SBMAC publica, desde as primeiras edições do evento, monografias dos cursos que são ministrados nos CNMAC.

Para a comemoração dos 25 anos da SBMAC, que ocorreu durante o XXVI CNMAC em 2003, foi criada a série **Notas em Matemática Aplicada** para publicar as monografias dos minicursos ministrados nos CNMAC, o que permaneceu até o XXXIII CNMAC em 2010.

A partir de 2011, a série passa a publicar, também, livros nas áreas de interesse da SBMAC. Os autores que submeterem textos à série Notas em Matemática Aplicada devem estar cientes de que poderão ser convidados a ministrarem minicursos nos eventos patrocinados pela SBMAC, em especial nos CNMAC, sobre assunto a que se refere o texto.

O livro deve ser preparado em **Latex (compatível com o Miktex versão 2.7), as figuras em eps** e deve ter entre **80 e 150 páginas**. O texto deve ser redigido de forma clara, acompanhado de uma excelente revisão bibliográfica e de **exercícios de verificação de aprendizagem** ao final de cada capítulo.

Veja todos os títulos publicados nesta série na página
http://www.sbmac.org.br/notas.php

Sociedade Brasileira de Matemática Aplicada e Computacional

2012

Introdução à Construção de Modelos de Otimização Linear e Inteira

Socorro Rangel - UNESP
socorro@ibilce.unesp.br

Sociedade Brasileira de Matemática Aplicada e Computacional

São Carlos - SP, Brasil
2012

Coordenação Editorial: Véra Lucia da Rocha Lopes

Coordenação Editorial da Série: Geraldo Nunes Silva

Editora: SBMAC

Capa: Matheus Botossi Trindade

Patrocínio: SBMAC

Catalogação elaborada pela Biblioteca do IBILCE/UNESP
Bibliotecária: Maria Luiza Fernandes Jardim Froner

Rangel, Socorro.
Introdução à Construção de Modelos de Otimização Linear e Inteira - São Carlos, SP : SBMAC, 2012, 82 p., 20.5 cm - (Notas em Matemática Aplicada; v. 18)

e-ISBN 978-85-86883-84-2

1. Otimização Linear Inteira. 2. Aplicações. 3. Sistemas Algébricos. de Modelagem.
I. Rangel, Socorro. V. Título. VI. Série.

CDD - 51

Esta é uma republicação em formato de e-book do livro original do mesmo título publicado em 2005 nesta mesma série pela SBMAC.

Ao Elso pelo carinho e compreensão.

Prefácio

A otimização está presente no nosso dia a dia. Sempre que queremos ou precisamos tomar uma decisão, procuramos escolher entre as várias alternativas aquela que, naquele momento, nos dê maior satisfação.

Na matemática, a área que estuda problemas de otimização é classicamente chamada de Programação Matemática . Esta denominação identifica uma ampla classe de problemas. O nome *programação* foi empregado porque os militares se referem ao planejamento de atividades como "programa". Boa parte dos acontecimentos que culminaram com a criação desta importante área da matemática, se deram durante a Segunda Guerra Mundial. De fato, George B. Dantzig usou o termo Programação Linear (mais especificamente *Programming in a linear Structure*, mais tarde resumido para *Linear Programming* e generalizado como *Mathematical Programming*) para analisar um problema de planejamento para a força aérea americana. Com a disseminação do uso do computador, *programação* passou a ser entendido como a codificação de um algoritmo em uma determinada linguagem (e.g. FORTRAN, C, C++) e às vezes a Programação Matemática é confundida com programação de computadores. Linguagens de programação e computadores são muito usados no estudo de problemas de otimização, mas a Programação Matemática é muito mais do que a codificação de um algoritmo.

O termo Otimização é empregado às vezes em referência a uma classe específica de problemas de Programação Matemática: problemas de Programação não-linear. No entanto, *Otimização* pode ser usado para designar *Programação Matemática* de forma a tornar o significado do termo mais compreensível. Neste sentido, optamos por usar Otimização Linear, Otimização Inteira, Otimização não-linear para nomear os problemas classicamente conhecidos como programação linear, programação inteira, e programação não linear, respectivamente.

Neste texto desenvolvemos a metodologia básica de construção de modelos de otimização linear e linear inteira, apresentando ferramentas computacionais que podem auxiliar no desenvolvimento de modelos eficientes e realistas. A modelagem de problemas é um importante tópico que tem sido pouco explorado nos livros textos e/ou disciplinas de otimização linear e linear inteira. A ênfase, em geral, é dada ao desenvolvimento de métodos de solução.

Considerada por alguns como uma "arte", talvez com o intuito de evitar o tema, a modelagem matemática pode e deve ser sistematizada. Naturalmente, como em outras áreas do conhecimento, quanto maior a experiência, maior a facilidade de construir modelos que representem matematicamente situações do nosso dia-a-dia.

A partir de modelos clássicos é possível aprender o processo de modelagem e desenvolver habilidades para a criação de bons modelos matemáticos. Ferramentas computacionais (e.g. linguagens algébricas de modelagem e sistemas de resolução) permitem experimentar diversos métodos de solução, avaliar as soluções obtidas e validar um modelo. Explorando as limitações dos recursos computacionais disponíveis somos motivados a pesquisar e desenvolver novos métodos de solução e novas ferramentas computacionais.

São José do Rio Preto, agosto de 2005.

Agradecimentos

- aos alunos e alunas que cursaram as disciplinas de graduação Métodos de Otimização I e II, Programação Linear e Programação Matemática entre 2000 e 2005 que de uma forma ou de outra incentivaram a redação deste texto;

- ao apoio financeiro do CNPq (Proc. nº 473001/2004-7);

- ao apoio institucional da UNESP;

- à Ana Paula Ximenes Flores e aos pareceristas AD HOC pela leitura cuidadosa do texto.

Conteúdo

Capítulo 1

Modelos de Otimização

> *"Existem duas maneiras de aumentar a eficiência de uma loja, empresa, ou indústria. Uma delas requer a melhoria tecnológica, isto é, atualização dos equipamentos, mudança no processo tecnológico, descoberta de novos e melhores tipos de matéria prima. A outra maneira, até hoje muito menos utilizada, envolve melhorias na organização do planejamento e da produção. Isto é, melhorias no processo de distribuição do trabalho entre as máquinas da empresa, distribuição de matéria prima, combustível, entre outros fatores."*[1]

Diversas situações podem ser estudadas de forma mais abrangente se representadas através de modelos que capturem seus principais elementos. Um modelo matemático de otimização envolve a representação de um problema ou situação através de um conjunto de relações matemáticas tais como: equações, inequações, dependências lógicas, e funções. São várias as razões para a construção de modelos de otimização, entre elas podemos destacar: aumentar o grau de entendimento da situação estudada; analisar a situação e propor soluções que não sejam aparentes; experimentar diversos cenários que de outra forma não seria possível, ou recomendável [56]. Taube [52] destaca outras razões sob o ponto de vista da gestão empresarial. Modelos de otimização linear e otimização linear inteira podem ser úteis na resolução de um grande número de problemas em diversas áreas. Neste texto iremos discutir algumas destas aplicações.

Para construir um modelo de otimização linear que represente um determinado problema é necessário identificar inicialmente quais são os *elementos conhecidos* geralmente associados ao que sabemos sobre o problema, e quais são os *elementos desconhecidos* associados ao que queremos determinar ao solucionar o problema. Esta fase inicial em geral é realizada de forma conjunta através de reuniões com o pessoal envolvido na resolução do problema, e envolve o conhecimento da situação

[1] Traduzido de (Kantarovich (1939) in Dantzig [12]-pg 22)

estudada. Nem sempre a identificação destes elementos é imediata. A facilidade de obtenção destas informações depende diretamente do grau de organização do setor em estudo. Para encontrá-las, algumas simplificações iniciais da situação estudada são necessárias. Estes elementos serão importantes para a construção do modelo matemático, é através deles que serão definidos os objetos matemáticos: constantes, incógnitas e funções que representem o problema.

Os *elementos desconhecidos*, em geral associados à decisão a ser tomada através da solução do problema, são modelados em termos de incógnitas, que chamaremos de *variáveis de decisão* . Em geral, um grande número de valores podem ser associados a estas variáveis e conseqüentemente é necessário definir um critério, ou objetivo, para seleção da melhor alternativa. Este critério será traduzido através de uma função linear das variáveis de decisão que será chamada de *função-objetivo* . Neste ponto surge a questão: "Quais são os impedimentos que restringem a tomada de decisões?". Estes impedimentos dão origem às equações e inequações lineares que formam o conjunto de restrições do modelo. Os *elementos conhecidos*, chamados de dados, fornecerão os coeficientes das variáveis e os termos constantes nas restrições e na função-objetivo. Os principais elementos de um modelo de otimização estão resumidos na Figura 1.1 a seguir.

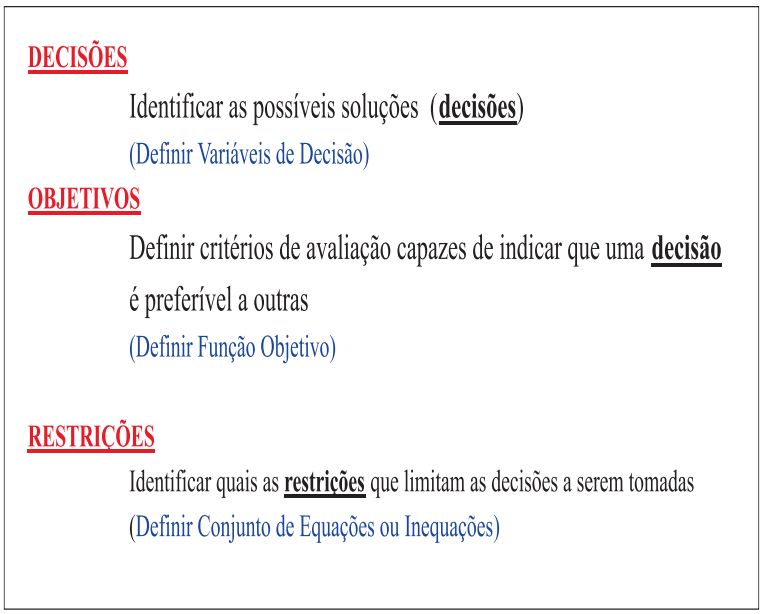

Figura 1.1: Principais Elementos de um modelo de otimização

É importante ressaltar que o processo de construção do modelo é iterativo e pode ser dividido em três fases: *modelagem, validação e implementação*. Na fase de *modelagem*, um conjunto de variáveis de decisão são definidas, uma função-objetivo

e um conjunto de restrições são inicialmente propostos. É possível que durante a definição da função-objetivo e das restrições, novas variáveis sejam necessárias, outras restrições sejam identificadas, e/ou novos termos devam ser considerados na função-objetivo. Ou seja a fase de *modelagem* deve ser repetida até que se obtenha um modelo bem representativo da situação estudada.

Uma vez obtido um modelo inicial, começa a fase de *validação* do modelo. O modelo inicial é resolvido e a solução obtida deve ser analisada para verificar se ela é aceitável para a situação em estudo. Os dados utilizados nesta fase podem ser colhidos junto aos responsáveis pelo problema, ou gerados aleatoriamente. Neste último caso, é necessário um certo cuidado para que os números gerados reflitam grandezas próximas dos dados reais. Esta é uma fase importantíssima, pois é nela que acharemos os principais problemas do modelo proposto. É muito improvável que o primeiro modelo construído reflita de forma satisfatória o problema. Novas rodadas de reuniões e depuração do modelo inicial com a inclusão e/ou remoção de variáveis, constantes e restrições são necessárias para a obtenção de um modelo mais próximo da realidade estudada. As fases de *modelagem* e *validação* são repetidas até que as partes envolvidas estejam satisfeitas com o modelo resultante.

A terceira e última fase, *implementação*, se inicia quando o modelo matemático construído é usado como ferramenta em um sistema de apoio à decisões (SADE). Para tanto, é necessário que uma interface seja construída entre o modelo matemático, o sistema de resolução e o usuário final. Esta interface pode ser construída de diversas maneiras, neste texto estaremos dando ênfase ao uso dos sistemas algébricos de modelagem (SAM) para facilitar a documentação, utilização e manutenção dos modelos de otimização que iremos construir. Através desta interface, os dados referentes a uma situação particular são recuperados e transferidos para o modelo matemático. Um exemplar do problema é assim obtido e então traduzido para um formato especial que será lido pelo sistema de resolução. Uma vez obtida a solução deste exemplar, a interface deverá recuperá-la e traduzi-la para um formato apropriado. É nesta fase que a solução matemática obtida deverá ser analisada, avaliada de acordo com critérios políticos, econômicos, científicos e possivelmente utilizada na prática. Mudanças na realidade podem requerer que o modelo seja reavaliado, entramos então na fase de *manutenção*. O processo de construção de um modelo de otimização pode ser resumido e representado através da Figura 1.2 a seguir.

Os modelos matemáticos usados em otimização seguem em geral um padrão composto por uma função-objetivo, um critério de otimização, minimizar (min) ou maximizar (max), o termo *sujeito a (s.a)* que indica que os valores aceitos para otimizar a função-objetivo devem satisfazer um conjunto de restrições, a descrição matemática das restrições na forma de equações ou inequações, e a definição do tipo das variáveis. O formato geral é dado pela Expressão (1.1) a seguir.

$$\begin{array}{ll} \text{min (ou max)} & \text{função-objetivo} \\ \text{sujeito a} & \\ & \text{restrições principais: equações ou inequações} \\ & \text{tipo das variáveis} \end{array} \qquad (1.\text{-}1)$$

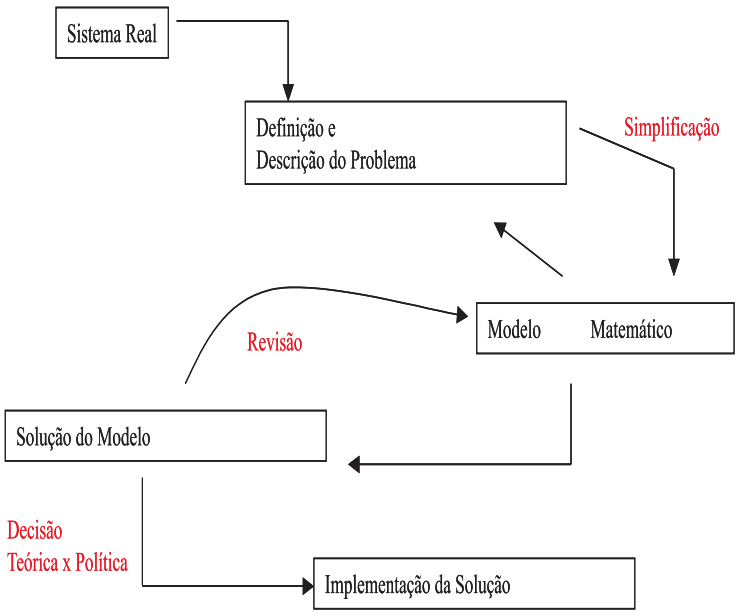

Figura 1.2: Processo de construção de um modelo de otimização

A forma matemática que a função-objetivo a ser minimizada (ou maximizada) irá tomar, bem como as restrições e o tipo de variável, irão definir os diversos modelos de otimização. Se a função-objetivo e as restrições forem lineares, e as variáveis puderem assumir valores reais, isto é $x \in \Re$, temos um modelo de otimização linear contínuo no formato padrão (Modelo 1.-1).

$$\begin{aligned} \min \quad & z = cx \\ \text{sujeito a} \quad & \\ & Ax = b \\ & x \geq 0, x \in \Re^n \end{aligned} \tag{1.-3}$$

onde A é uma matriz m x n, x e c são vetores n-dimensionais e b é um vetor m-dimensional. Se no Modelo (1.-1) restringirmos as variáveis de forma que só possam assumir valores inteiros, $x \in \mathrm{Z}$, teremos um modelo de otimização linear inteira (Modelo 1.-3).

$$\begin{aligned} \min \quad & z = cx \\ \text{sujeito a} \quad & \\ & Ax = b \\ & x \geq 0, x \in \mathrm{Z}^n \end{aligned} \tag{1.-5}$$

Em determinadas circunstâncias interessa que apenas um subconjunto de variáveis esteja restrito a assumir valores inteiros. Neste caso, temos um modelo de otimização

inteira mista (modelo 1.-5).

$$\begin{aligned} &\min && z = cx \\ &\text{sujeito a} \\ & && Ax = b \qquad (1.\text{-}7) \\ & && x \geq 0, x_j \in \mathrm{Z}, j = 1, \ldots, p; x_j \in \Re, j = p+1, \ldots, n. \end{aligned}$$

Note que no Modelo (1.-5) x_j indica a j-ésima componente do vetor x. Para facilitar a referência a estes modelos usaremos a abreviação (OL), (OI), (OIM) respectivamente. Modelos tais que a função-objetivo é não linear e/ou o conjunto de restrições é formado por equações ou inequações não lineares são chamados de modelos de otimização não-lineares (ONL). Situações que envolvam modelos não-lineares e que não possam ser representadas por modelos lineares fogem do escopo deste texto e não serão discutidas. O leitor interessado nesta classe de modelos poderá consultar por exemplo as seguintes obras: [56], [26], [31], [17].

Antes de prosseguirmos explorando os modelos descritos acima, gostaríamos de chamar atenção para algumas confusões existentes a respeito da validade e uso de modelos matemáticos. Uma das críticas apontadas diz respeito a quantificação de parte dos dados usados no modelo, por exemplo como atribuir um custo a um valor social, outra está associada à precisão dos dados utilizados. Williams [56] responde a estas críticas considerando que uma série de decisões associadas a conceitos "não-quantificáveis"precisam ser tomadas, e são baseadas numa caracterização implícita que não pode ser evitada. Incorporar esta decisão explicitamente num modelo matemático parece ser uma forma científica e honesta de lidar com a questão. A precisão dos dados deve ser considerada em relação a cada modelo especificamente. Apesar de uma parcela dos coeficientes de um modelo não serem precisos, ainda assim é possível que o modelo matemático produza uma boa solução para o problema. De qualquer forma é importante ter claro que o modelo matemático deve ser usado como uma de diversas ferramentas disponíveis para a tomada de decisão. A qualidades das respostas que um modelo produz depende da precisão e da estrutura dos dados do modelo. No caso dos modelos de otimização a definição da função-objetivo também afeta fortemente a resposta do modelo.

Williams [56] observa ainda que a resposta fornecida por um modelo matemático deve ser analisada cuidadosamente. Se ela representa uma decisão que não pode ser tomada, as razões para a não aceitação desta solução devem ser analisadas e possivelmente incorporadas num novo modelo através da modificação do conjunto de restrições e/ou da função-objetivo. Se a resposta for aceitável, pode ser sábio tomá-la apenas como uma opinião. Modificar a função-objetivo (e conseqüentemente o modelo) pode resultar em uma outra opção. Questionando as respostas fornecidas pelo modelo e modificando-o de forma adequada pode tornar mais visível as diversas possibilidades existentes e aumentar o grau de entendimento do problema.

Nos próximos capítulos desenvolvemos a metodologia básica de construção de modelos de otimização linear e linear inteira, apresentamos ferramentas computacionais que podem auxiliar no desenvolvimento de modelos eficientes e discutimos alguns problemas clássicos que servem de ponto de partida para a solução de diversos problemas no nosso dia a dia.

Capítulo 2

Um primeiro modelo de otimização linear: o problema da dieta

O problema da dieta consiste em, dado um conjunto de alimentos, escolher quais e quanto usar de cada um para compor uma dieta alimentar que atenda quantidades pré-determinadas de nutrientes, de acordo com algum critério. Esse é um enunciado muito geral, no entanto serve de base para apresentarmos um problema que possui uma importância especial. Num excelente histórico sobre o problema da dieta, Namem e Bornstein [40] destacam que este foi um dos primeiros problemas a ser usado para testar o método simplex (e.g. [13], [21], [6]) desenvolvido por George B. Dantzig em 1947 para resolver problemas de otimização linear. Além disso, o modelo original apresentado por Stigler em 1945, apesar de nem sempre resultar em uma resposta satisfatória, é simples e ilustra muito bem o processo de construção de um modelo discutido no Capítulo 1 e as principais suposições (ou axiomas) necessárias para se obter um modelo de otimização linear.

2.1 Axiomas: proporcionalidade, aditividade, divisibilidade

Vamos ilustrar a construção de um modelo de otimização linear considerando a seguinte adaptação do problema da dieta proposto por Stigler (uma versão do enunciado original que envolvia 77 alimentos pode ser encontrada em [40]).

Exemplo 2.1 *Paula deseja balancear os alimentos que consume de forma a obter uma dieta alimentar que forneça diariamente toda a energia, proteína e cálcio que necessita. Seu médico recomendou que ela se alimente de forma a obter diariamente no mínimo 2000 kcal de energia, 65g de proteína e 800 mg de cálcio. O Valor*

nutritivo e o preço (por porção) de cada alimento a ser considerado na dieta é dado na Tabela 2.1. Quanto de cada alimento Paula deve consumir para obter uma dieta que atenda a recomendação médica e que tenha o menor custo possivel?

Tabela 2.1: Composição nutricional e custo dos alimentos

Tipo de alimento	tamanho da porção	energia (kcal)	Proteína (g)	cálcio (mg)	preço p/ porção (centavos)
arroz	100g	170	3	12	14
ovos	2un	160	13	54	13
leite	237ml	160	8	285	9
feijão	260g	337	22	86	19

Para iniciar a construção do modelo matemático que represente o problema da Paula, precisamos distinguir os *elementos conhecidos* e os *desconhecidos* conforme a discussão que fizemos no Capítulo 1 (ver Figura 1.1). Analisando o enunciado do Exemplo 2.1 temos:

Elementos conhecidos

- alimentos a serem considerados na elaboração da dieta;

- composição nutricional dos alimentos;

- quantidade mínima de nutrientes que a dieta deve satisfazer diariamente;

- custo dos alimentos (por porção);

Elementos desconhecidos

- número de porções de cada alimento a ser usado na dieta.

De acordo com os dados da Tabela 2.1 podemos verificar que existem diversas combinações dos alimentos que fornecem os níveis recomendados de nutrientes. Também é possível calcular o custo associado a cada uma delas. Precisamos então definir um critério para a seleção do melhor combinação possível. Antes é necessário esclarecer o que consideramos como a melhor combinação possível. Dentro do presente contexto, a questão do custo associado à combinação dos alimentos parece ser um item importante para a Paula. Assim podemos dizer:

Objetivo

- obter uma dieta com o menor custo possível.

O menor custo possível na presente situação é custo zero, mas é claro que esta solução não atende a recomendação médica. Uma solução do problema deve satisfazer algumas restrições e portanto para completar a definição dos elementos principais do modelo é necessário definir:

Restrições

- a dieta deve fornecer uma quantidade mínima dos nutrientes pré-especificados.

Uma vez entendido o problema, podemos iniciar a construção do modelo de otimização traduzindo para a matemática os elementos acima. Vamos usar um modelo de otimização linear (OL). O modelo definido no Capítulo 1 pela Expressão (1.-1) também pode ser escrito no seguinte formato:

$$\min \quad z = \sum_{j=1}^{n} c_j x_j \tag{2.1}$$

sujeito a

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\sim b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\sim b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\sim b_m \end{aligned} \tag{2.-1}$$

$$x_j \geq 0, x_j \in \Re \tag{2.-1}$$

onde o símbolo $\sim$ pode ser substituído por "$\leqq$", "$\geqq$", ou "$=$". Qualquer problema de otimização linear pode ser escrito no formato padrão apresentado no Capítulo 1 (e.g. [13], [21]).

É importante neste momento detalharmos um pouco mais a definição de um modelo de otimização linear. A função-objetivo (2.1) e as restrições (2.-1) do modelo acima são lineares pois não apresentam produtos de variáveis, xy, potências de variáveis, $x^n, n > 1$, ou ainda combinações de variáveis do tipo $(x + logy)$. De uma maneira geral, uma forma linear pode ser caracterizada por certas propriedades multiplicativas e aditivas [25]. No contexto do Exemplo 2.1 se Paula incluir na dieta uma porção de arroz ela obtém 170 kcal de energia, se incluir duas irá obter o dobro de energia. A quantidade de energia obtida na dieta referentes ao arroz é proporcional ao número de porções incluídas. De maneira similar, para calcular o custo da dieta, podemos supor que se uma porção de arroz custa 14 centavos, duas porções custarão 28 centavos. Assim, um modelo linear pressupõe que o custo total da compra de um determinado item é *proporcional* ao número de unidades compradas.

A propriedade *aditiva* pode ser ilustrada considerando que se Paula incluir na dieta uma porção de arroz e uma de feijão a energia total obtida será igual à soma das quantidades individuais fornecidas por cada alimento, isto é 507 kcal, e o custo total igual à soma dos custos individuais, 33 centavos. Usando as propriedades de *aditividade* e *proprocionalidade*, podemos concluir que dobrando o consumo de arroz e triplicando o consumo de feijão, teremos uma dieta de 1351 kcal a um custo de 85 centavos.

No modelo (OL) que vamos construir para representar o problema da Paula, a letra j será usada para indexar os elementos do conjunto de alimentos que poderão ser usados para compor a dieta. No Exemplo 2.1 temos quatro alimentos, o índice

j poderá então assumir valores inteiros entre 1 e 4 para representar cada um deles. De forma similar definiremos um segundo indice, i, para indexar os nutrientes que devem ser obtidos pela dieta. Assim temos:

Índices

$j = 1, \ldots, 4$ representa respectivamente os alimentos: arroz, ovos, leite e feijão;

$i = 1, \ldots, 3$ representa respectivamente os nutrientes: energia, proteina e cálcio.

Verificamos acima que um *elemento desconhecido* importante no problema enunciado no Exemplo 2.1 é: quantas porções de cada alimento devem fazer parte da dieta. Assim podemos definir:

Variáveis de decisão

x_j : número de porções do alimento j a ser incluído na dieta.

Note aqui que os *elementos conhecidos* do problema estão fornecendo os dados do modelo. Até aqui o número e o tipo de alimentos e nutrientes.

Enunciamos acima que o critério de escolha dos alimentos é obter a dieta de menor custo possível. O preço de cada alimento é conhecido e assim usando as propriedades de *proporcionalidade* e *aditividade* podemos escrever matemáticamente o critério de escolha dos alimentos. Considerando que se comprarmos x_1 porções de arroz iremos gastar $14x_1$ centavos e que se comprarmos x_2 porções de ovos iremos gastar $13x_2$ centavos, temos que o custo total da compra de x_1 porções de arroz e x_2 porções de ovos é $14x_1 + 13x_2$. Usando o mesmo raciocínio e considerando os demais ingredientes, temos que o custo total da dieta é dado por:

$$z = 14x_1 + 13x_2 + 9x_3 + 19x_4.$$

Como queremos obter a dieta de menor custo possível, temos:

Função-objetivo

$\min z = 14x_1 + 13x_2 + 9x_3 + 19x_4$.

As propriedades de *aditividade* e *proporcionalidade* também serão úteis para expressar matematicamente as restrições do modelo. A dieta precisa satisfazer níveis mínimo de cada um dos nutrientes. Em relação à energia temos que:

quantidade total de energia da dieta $\geq$ 2000 kcal.

A quantidade total de energia pode ser calculada de forma similar ao cálculo do custo total. Considerando que se for consumido x_1 porções de arroz teremos $170x_1$ kcal e x_2 porções de ovos teremos $160x_2$ kcal, temos um total de $170x_1 + 160x_2$ kcal. Se chamarmos de a_{ij} a quantidade de nutriente i no alimento j, a quantidade total do nutriente i obtida pelo consumo de x_j porções do alimento j é igual a $a_{ij}x_j$. Considerando o consumo dos quatro alimentos temos que a quantidade total do nutriente i é igual a $a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + a_{i4}x_4$. Usando os dados da Tabela 2.1 podemos explicitar o conjunto de restrições:

Restrições

energia: $170x_1 + 160x_2 + 160x_3 + 337x_4 \geq 2000$

proteína: $3x_1 + 13x_2 + 8x_3 + 22x_4 \geq 65$

cálcio: $12x_1 + 54x_2 + 285x_3 + 86x_4 \geq 800$.

Para completar o nosso primeiro modelo, temos ainda que considerar os valores que as variáveis de decisão podem assumir. Naturalmente, nesta situação estamos interessados apenas em valores não-negativos que satisfaçam os níveis mínimos de nutrientes. Podemos também considerar que a variável x_j pode receber qualquer valor real. Esta é a terceira propriedade que caracteriza um modelo de otimização linear: *divisibilidade*. Se impormos como restrição adicional, por exemplo, que as variáveis assumam apenas valores inteiros não teremos mais um problema de otimização linear, mas sim um problema de otimização inteiro. Algumas situações reais são melhor representadas se estas restrições forem impostas, o que discutiremos com mais detalhes no Capítulo 5. No presente caso:

Tipo das Variáveis

$x_j \geq 0, x_j \in \Re$

O Modelo de otimização que representa o Problema da Dieta do Exemplo 2.1 é então dado pelo problema de otimização linear (2.-1) abaixo.

$$\begin{aligned} \min \quad & z = 14x_1 + 13x_2 + 9x_3 + 19x_4 \\ \text{sujeito a} \quad & \\ & 170x_1 + 160x_2 + 160x_3 + 337x_4 \geq 2000 \\ & 3x_1 + 13x_2 + 8x_3 + 22x_4 \geq 65 \\ & 12x_1 + 54x_2 + 285x_3 + 86x_4 \geq 800 \\ & x_j \geq 0, j = 1 \ldots, 4 \end{aligned} \tag{2.-4}$$

Note que a definição do tipo das variáveis fornece um conjunto de restrições adicionais ao modelo. Este conjunto é descrito de forma separada porque estas informações são em geral tratadas de forma implícita. A condição $x_j \in \Re$ é considerada implicitamente nos modelos de otimização linear e por isso foi omitida do problema (2.-1).

Uma vez obtido o modelo de otimização linear que represente, pelo menos a princípio, a situação estudada precisamos verificar se as respostas obtidas são satisfatórias. Isto é, iniciamos a fase de *validação* do modelo. Esta é uma fase que pode demandar bastante esforço da equipe envolvida. Ferramentas computacionais tais como linguagens de programação, linguagens de modelagem, bem como sistemas computacionais desenvolvidos para a resolução de problemas de otimização podem facilitar o trabalho desta fase. No Capítulo 3 apresentaremos algumas ferramentas computacionais que estão disponíveis para esta tarefa. Neste momento, vamos supor que dispomos de um aplicativo, chamado c_otim, que irá receber um problema

de otimização linear, resolvê-lo e retornar a solução ou informar que a mesma não existe. A solução de um problema de otimização, simplesmente *solução ótima*, é uma atribuição de valores para as variáveis de decisão de forma que o conjunto de restrições é satisfeito e o valor da função-objetivo é otimizado, no presente caso, minimizado. Qualquer outra atribuição de valores às variáveis de decisão tal que o conjunto de restrições seja satisfeito é chamada de *solução viável*.

Resolvendo o problema de otimização linear (2.-1) pelo sistema c_otim obtemos a seguinte solução:

$$z = 112,5, x_1 = 0, x_2 = 0, x_3 = 12,5, x_4 = 0. \tag{2.-5}$$

que "traduzida"quer dizer que a dieta deverá ser composta por 12,5 porções de leite correspondendo a aproximadamente 2,96 litros de leite (x_3 representa o número de porções do alimento leite e uma porção de leite é igual a 237ml) a um custo de 112,5 centavos.

O que dizer desta solução? Apesar de ser a de menor custo, a dieta sugerida é composta por apenas um alimento em uma quantidade que não é adequada. Quando tomou conhecimento desta sugestão, a nossa personagem Paula imediatamente disse: "Eu não consigo tomar esta quantidade de leite!". Uma pergunta natural para fazer a Paula é: "Quantas porções de leite voce tomaria?". Este diálogo fictício ilustra bem a necessidade de se avaliar as respostas que um modelo produz. Se lembrarmos do processo de construção de um modelo discutido no Capítulo 1, a fase de *validação* pode resultar em uma revisão do modelo proposto. A resposta da Paula à pergunta acima implica em impor um limite para o número de porções de leite a serem incluídas na dieta. Isto é, novos *elementos conhecidos* passam a fazer parte do nosso modelo: número máximo de porções do leite. Podemos prever que limitar apenas o consumo do leite pode gerar novas soluções que sugiram um consumo igualmente inaceitável dos demais alimentos. Assim, para obter um modelo que reflita melhor a situação precisamos conhecer os limites máximos aceitáveis para o consumo de cada um dos alimentos. Obtendo estes dados da Paula, podemos representar matematicamente esta nova restrição impondo um limite superior para o valor das variáveis no Modelo (2.-1). Considerando que o limite máximo para o consumo de arroz, ovos, leite e feijão são 1,2,2,3 respectivamente, temos:

Reformulação do modelo: novas restrições

$x_1 \leq 1$, $x_2 \leq 2$, $x_3 \leq 3$, $x_4 \leq 3$.

Este novo conjunto de restrições é incluído no modelo na seção Tipo de Variável (ver Modelo (1.1)). O modelo continua sendo linear, pois apesar de restritas, as varíáveis continuam podendo receber qualquer valor real dentro de um determinado

intervalo. O novo modelo é então:

$$\begin{aligned} \min \quad & z = 14x_1 + 13x_2 + 9x_3 + 19x_4 \\ \text{sujeito a} \quad & \\ & 170x_1 + 160x_2 + 160x_3 + 337x_4 \geq 2000 \\ & 3x_1 + 13x_2 + 8x_3 + 22x_4 \geq 65 \\ & 12x_1 + 54x_2 + 285x_3 + 86x_4 \geq 800 \\ & x_1 \leq 1, x_2 \leq 2, x_3 \leq 3, x_4 \leq 3, x_j \geq 0, \quad j = 1 \ldots, 4. \end{aligned} \tag{2.-7}$$

Diversas outras considerações poderiam ser feitas a respeito da validade da nova solução, não apenas dentro do contexto sugerido pelo Exemplo 2.1 mas também em relação ao problema da dieta de forma mais geral. Sugerimos ao leitor o texto de Namem e Bornstein [40] para aprofundar a discussão especificamente sobre a modelagem do problema da dieta. Nos demais capítulos voltaremos a usar este problema para ilustrar outros aspectos associados à construção e solução de modelos de otimização.

Capítulo 3

Ferramentas Computacionais

O problema da dieta foi uma das primeiras aplicações da otimização linear a ser resolvida pelo método simplex. Em 1947, uma versão do problema contendo 9 restrições e 77 variáveis, considerado um problema de grande porte para a época, foi resolvida usando calculadoras de mesa em 120 dias de trabalho [13]. Hoje, os sistemas computacionais são capazes de manipular problemas com milhares de variáveis em poucos segundos. De fato, o porte dos problemas a serem resolvidos por um determinado sistema depende principalmente da disponibilidade de memória da máquina onde o sistema está instalado. No entanto, manipular as informações associadas a um modelo de otimização de grande porte, nos padrões de hoje, não é uma tarefa trivial. E isso diz respeito ao armazenamento e manipulação dos dados do problema, bem como à geração e documentação das variáveis e das restrições do modelo associado, além de outros fatores como fornecer e obter informações dos sistemas computacionais de resolução.

Diversas ferramentas computacionais de caráter geral e específico podem auxiliar no processo de construção, manutenção e solução de modelos de otimização. Podemos dividir a discussão deste tópico em dois blocos: ferramentas de modelagem e ferramentas de resolução. No primeiro bloco estão as linguagens de programação (e.g. C, C++, Fortran), as planilhas de cálculo (e.g. EXCEL, Lotus 123) e as linguagens algébricas de modelagem (LAM). No bloco das ferramentas de resolução entram novamente as linguagens de programação, usadas para a implementação de algoritmos e os os sistemas de resolução comerciais e de pesquisa. Em [28] a planilha de cálculo EXCEL é usada para modelar e resolver problemas. Nas próximas seções faremos uma breve discussão sobre linguagens de modelagens (Seção 3.1) e sistemas de resolução (Seção 3.2).

3.1 Linguagens algébricas de modelagem

Em geral, estamos interessados em usar um determinado modelo matemático para simular diversas possibilidades ou cenários antes de tomar uma decisão. No caso

do problema da dieta discutido no Capítulo 2, construímos um modelo de otimização linear para representar especificamente a situação descrita no Exemplo 2.1. No entanto, é possível descrever o Modelo (2.-1) em um formato mais geral. Se considerarmos que temos:

- n alimentos;
- m nutrientes;
- $c_j, j = 1, \ldots, n$: o custo do alimento j;
- $b_i, i = 1, \ldots, m$: o nível mínimo do nutriente i;
- $a_{ij}, i = 1, \ldots, m, j = 1, \ldots, n$: quantidade do nutriente i presente no alimento j;

o modelo (2.-1) pode ser generalizado como:

$$\begin{array}{ll} \min & z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ \text{sujeito a} & \\ & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \geq b_1 \\ & a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \geq b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \geq b_m \\ & x_j \geq 0, x_j \in \Re. \end{array} \tag{3.-3}$$

Quando definimos $n = 4$ e $m = 3$, e usamos os dados da Tabela 2.1 particularizamos o Modelo (3.1) acima e obtemos o Problema 2.-1. Neste caso dizemos que (2.-1) é um exemplar do problema da dieta. O Modelo (3.1) pode ser usado para simular o problema da dieta em diversos cenários, para isto basta atribuirmos diferentes valores para os *elementos conhecidos* do problema. Por exemplo, podemos verificar qual é a sugestão de dieta se ao invés de arroz, ovos, leite e feijão, quiséssemos compor uma dieta usando macarrão, brócolis, cenoura, ervilha e ovos.

Neste ponto é conveniente distinguir entre um modelo matemático e um exemplar do mesmo. Um modelo é uma representação algébrica abstrata do problema e um exemplar é a descrição explícita ou uma atribuição de valores para os dados abstratos de um modelo. Assim, no caso do problema da dieta podemos dizer que (3.1) é um modelo para o problema da dieta e que (2.-1) é um exemplar. Assim quando dizemos que queremos resolver um modelo de otimização estamos nos referindo a resolução de um exemplar do mesmo.

O objetivo inicial para o desenvolvimento das linguagens algébricas de modelagem foi o de facilitar o processo de comunicação de dados entre o usuário e os sistemas de resolução. A evolução das linguagens ampliaram estes objetivos e podemos dizer que o seu uso facilita a construção e a documentação de um modelo de otimização e serve de interface com sistemas de resolução, não apenas para fornecer dados e receber a solução do exemplar resolvido, mas também para interagir do ponto de vista algoritmico com o sistema.

Existem disponíveis hoje uma grande variedade de linguagens de modelagem com diferentes habilidades (e.g. [19], [49], [36]). Nosso objetivo aqui não é fazer uma descrição comparativa entre elas, ou mesmo fornecer um manual de uso. Queremos sim, fazer uma breve introdução e mostrar como estas linguagens podem auxiliar o processo de construção e validação de modelos de otimização. Para tanto, vamos utilizar três linguagens: MPL [34], AMPL [18] e XPRESS-MOSEL [11] que juntas ilustram bem as possibilidades de uso das linguagens de modelagem em geral.

Estas três linguagens foram escolhidas por várias razões. Atualmente, todas elas estão disponíveis gratuitamente para *download*(em versões resumidas), o que permite que voce leitor, possa utilizá-las para construir seus próprios modelos. Em geral, um sistema de resolução (também em versão resumida) acompanha a versão disponibilizada para *download*. A linguagem MPL é bastante didática e indicada para um primeiro curso de modelagem. As linguagens XPRESS-MOSEL e AMPL permitem uma boa manipulação de conjuntos e índices, e fornecem um conjunto de comandos para interação algoritmica entre o modelo e os sistemas de resolução. Os comandos disponíveis para definição do modelo também podem ser usados para escrever relatórios da solução de um exemplar. Duas linguagens de modelagem não-comerciais (ZIMPL e GMPL) disponíveis via *www* possuem sintaxe similar à linguagem AMPL [30]. Licença gratuita para a versão completa do XPRESS-MOSEL pode ser obtida através do *Academic Partner Program* mantido por seus proprietários.

O processo de descrição de um modelo de otimização através de uma linguagem de modelagem segue, na maioria dos casos, um padrão similar ao que usamos para construir o modelo da dieta discutido no Capítulo 2. A estrutura geral pode ser resumida pela definição dos seguintes itens:

- **conjuntos e índices**

  locais: {Rio, SP, Goiânia}

  códigos: {A11, B45}

  $i : 1 \ldots 4$;

- **dados, parâmetros, tabelas**

  podem ser fornecidos diretamente no modelo, lidos em arquivos livres de formato (tipo .txt), ou retirados de planilhas de cálculo ou banco de dados;

- **variáveis de decisão**

  podem ser do tipo inteiras, binárias ou contínuas, e definidas ou não para todos os elementos de um conjunto;

- **função-objetivo**

  linear ou não linear;

- **restrições**

podem ser agrupadas por tipo para expansão a posteriori, e definidas para subconjuntos de índices.

Nas próximas seções, vamos usar o problema da dieta enunciado no Capítulo 2 (Exemplo 2.1) para ilustrar a estrutura e algumas das facilidades das três linguagens mencionadas acima.

3.1.1 A linguagem MPL

Faremos aqui uma breve apresentação da linguagem MPL[1] (*Mathematical Programming System*)[34]. Para facilitar o entendimento a exposição esta dividida em três partes discutidas a seguir: Operação do Sistema, Sintaxe do MPL e Recursos do Sistema.

Operação do Sistema

Por ser tratar de um aplicativo para o sistema operacional *Windows*, sua operação básica segue o padrão de outros aplicativos para este sistema operacional, como por exemplo WORD, EXCEL. A barra de ferramentas principal do aplicativo, a barra de menus (ver Figura 3.1), é composta de comandos usuais de aplicativos para Windows: FILE (e.g. abrir, fechar, salvar), EDIT (e.g. cortar, copiar, colar); e comandos específicos do sistema: RUN e GRAPH que serão descritos a medida que forem sendo usados. Na barra principal também estão dispostos uma série de ícones que facilitam o acesso aos diversos comandos do sistema.

Figura 3.1: Menu principal do MPL

Sintaxe do MPL

Os principais comandos usados para descrever um modelo de otimização através do MPL são: TITLE, INDEX, DATA, VARIABLES, MODEL, SUBJECT TO, END. Cada um destes comandos são usado para definir uma seção do modelo, exceto os comandos TITLE e END que são usados para iniciar e encerrar a descrição do modelo respectivamente. Estes comando são passados para o sistema através de um arquivo tipo texto que será lido e interpretado pelo MPL. O sistema possui

[1] uma versão resumida do sistema (*Free Student/Trial version*) pode ser obtida gratuitamente no endereço http://www.maximal-usa.com/download/ juntamente com as instruções para instalação (última visita em 13/08/2005).

um editor de texto próprio, mas qualquer editor capaz de gerar arquivo livre de formatação (e.g. EDIT, PICO, NOTEPAD) pode ser usado. Usando o editor do MPL vamos criar um arquivo para definir o nosso modelo (no menu principal escolha FILE seguido de NEW e uma janela se abrirá para receber o texto). Usaremos letras maiúsculas para indicar palavras que são reservadas para a linguagem, isto é, quando usadas são interpretadas como comandos pelo MPL.

O comando TITLE é usado para dar um título e indicar o início da descrição do modelo. Para o Exemplo 2.1 vamos usar o nome dieta. Veja a seguir o primeiro comando do nosso arquivo.

```
TITLE dieta
```

Como fizemos na modelagem descrita no Capítulo 1, vamos agora definir os índices que serão usados para gerar as variáveis, restrições e estruturas de dados (vetores, matrizes) para armazenar os valores necessários para gerar um exemplar do modelo. Isto será feito através do comando INDEX. Quando definimos os índices do Modelo (2.-1) usamos as letras i e j e números inteiros para definir os valores que estes índices poderiam receber. Precisamos exercitar um pouco nossa memória para lembrar que a letra i representa os nutrientes, a letra j representa os alimentos, e que quando fazemos $j = 3$ estamos nos referindo ao alimento leite. Seria muito mais natural usar diretamente as palavras *alimento* e *nutriente* como índices, da mesma forma que usar os nomes dos alimentos ao invés de números para definir as atribuições aos índices. Esta forma mais natural de fazer referência aos elementos de um modelo é uma das vantagens de se usar uma linguagem de modelagem. Podemos então acrescentar o comando INDEX ao nosso arquivo, substituindo os índices$i$ e j pelas palavras *nutriente* e *alimento*, respectivamente.

```
TITLE dieta

INDEX
      alimento = (arroz, ovos, leite, feijao)
      nutriente = (energia, proteina, caloria)
```

É importante observar a necessidade de se incluir comentários na descrição do modelo, pois eles ajudarão a documentá-lo e torná-lo auto-explicativo. Os comentários, isto é, partes do texto que serão ignorados pelo aplicativo, podem ser adicionados no MPL de duas formas: texto entre chaves e texto precedido de um ponto de exclamação. As chaves são, em geral, usadas no caso de comentários longos, de mais de uma linha, e o ponto de exclamação é usado no caso de comentários de uma única linha. Vamos então incluir comentários na descrição do modelo da dieta.

```
{ Arquivo: dieta.mpl
    Problema da Dieta: Determinar uma combinação de alimentos que forneça
    uma quantidade mínima de nutrientes }

TITLE dieta
```

INDEX
alimento = (arroz, ovos, leite, feijao)
nutriente = (energia, proteina, caloria)

Podemos iniciar agora a definição dos *elementos conhecidos* do modelo, ou seja a seção DATA. É neste ponto que serão definidas as estruturas que irão armazenar os dados do exemplar que iremos gerar posteriormente. Na generalização do Modelo 2.-1 definimos c_j, b_i, a_{ij} para representar o custo do alimento j, o nível do nutriente i e a quantidade do nutriente i presente no alimento j, respectivamente. Estes elementos podem ser pensados como componentes dos vetores c,b e da matriz A. Este mesmo recurso será usado agora definindo os nomes *custo*, *nivel* e *quant* no lugar de c, b e A. É necessário deixar claro a dimensão destas estruturas, e isto é feito associado a cada uma delas um ou mais dos índices definidos na seção INDEX. O comando VARIABLES abre a seção de definição das variáveis de decisão do modelo. Vamos usar o nome *comprar* no lugar de x para representar o número de porções de cada alimento a ser incluído na dieta.

Já incluímos muitas informações na descrição do modelo, e podemos ter cometido alguns erros de sintaxe, isto é comandos que o sistema MPL não é capaz de interpretar. Assim, antes de prosseguir, é conveniente verificar se a descrição do modelo feita até o momento está de acordo com os requisitos da linguagem. Para tanto acionamos na barra do menu principal o comando RUN seguido do comando CHECK SYNTAX (ver Figura 3.2).

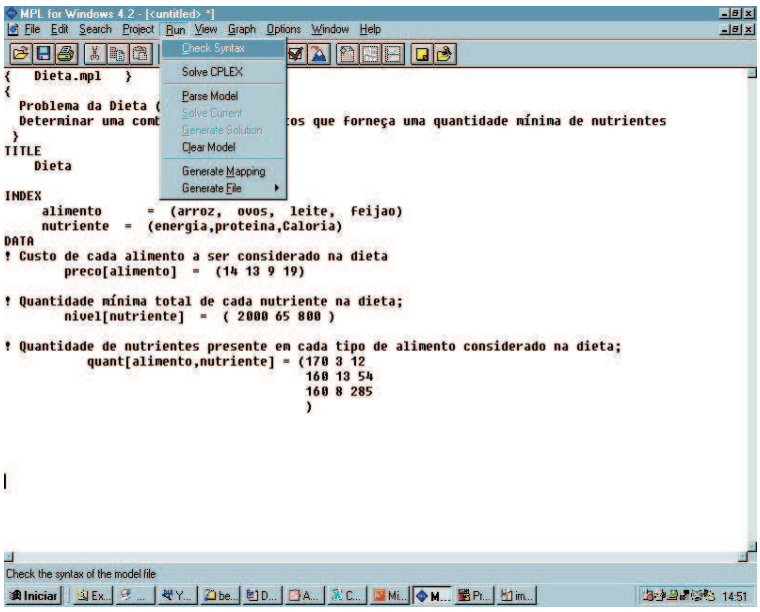

Figura 3.2: Verificando erros de sintaxe

Como o arquivo ainda não foi salvo, antes de proceder a análise da sintaxe, o sistema automaticamente aciona os comando FILE, seguido de SAVE AS do menu principal, sugerindo que a extensão do arquivo a ser salvo seja (.mpl). Após salvar o arquivo com o nome indicado pelo usuário, no nosso caso dieta.mpl, a sintaxe do modelo é verificada. Ao transcrever o trecho de modelo acima para o sistema cometemos um pequeno erro. Uma nova janela é aberta com uma descrição aproximada do erro cometido e uma indicação da posição do erro (ver Figura 3.3). Neste caso,

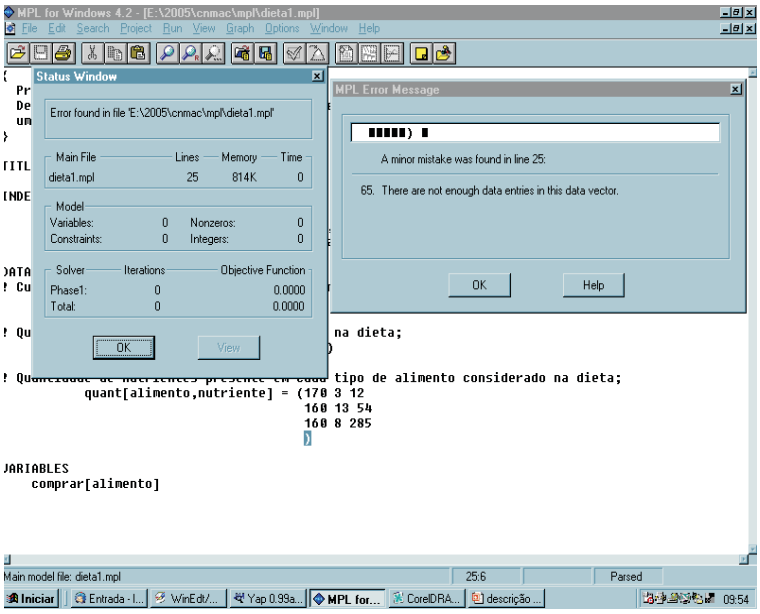

Figura 3.3: Erro de sintaxe

a mensagem do sistema indica que não fornecemos todos os dados necessários para compor a matriz *quant*. A matrix *quant* foi indexada por *nutriente* e *alimento*, portanto devem ser forncecidos 3 x 4 valores. Fornecemos apenas 9. Apesar de não se tratar de uma linguagem de programação de uso geral, a linguagem de modelagem pode ser tratada como tal pois possui uma estrutura específica a ser seguida. Boas práticas de programação e codificação podem e devem ser usadas. É conveniente observar que o MPL verifica a sintaxe do modelo linha por linha. Não processa uma nova linha se for encontrado algum erro na linha anterior. Esta forma de verificação de sintaxe muda de uma linguagem de modelagem para outra.

Incluindo o comando MODEL iniciamos a descrição do modelo de otimização propriamente dito: definição da função-objetivo e das restrições. Uma representação mais compacta e simples para o Modelo (3.1) pode ser obtida se usarmos o símbolo $\sum$ para representar as somas. Isto é, podemos reescrever o modelo para o problema

da dieta como (3.-4) abaixo.

$$\begin{aligned} \min \quad & z = \sum_{j=1}^{n} c_j x_j \\ \text{sujeito a} \quad & \\ & \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, \ldots, m \\ & x_j \geq 0, x_j \in \Re \end{aligned} \tag{3.-6}$$

É exatamente este tipo de construção compacta que iremos reproduzir na descrição do modelo usando o MPL. O sistema irá, uma vez acionado o comando apropriado, expandir os índices para gerar o Exemplar (2.-1). O símbolo de soma $\sum$ é representado na linguagem MPL através do comando SUM, e os índices usados para a expansão bem como a expressão a ser expandida são definidos entre parênteses. A função-objetivo é identificada através dos comandos MAX ou MIN, que já indicam também o critério de otimização. Assim de acordo com a sintaxe da linguagem MPL precisamos acrescentar as linhas a seguir ao arquivo.

```
MODEL
    MIN Custo_total = SUM(alimento: preco*comprar)
```

As restrições são precedidas do comando SUBJECT TO. Para garantir que o sistema entenda a necessidade de expansão das restrições é necessário definí-las usando um dos índices da seção INDEX. Por exemplo, se escrevermos simplesmente as linhas:

```
SUBJECT TO
    SUM(alimento: quant*comprar) > nivel[nutriente]
```

obteremos uma mensagem de erro pois o sistema não entende que será necessário criar uma restrição deste tipo para cada um dos elementos do índice *nutriente*. Para corrigir basta nomear as restrições e indexar por *nutriente*. Assim a descrição das restrições de acordo com a sintaxe do MPL é feita através do comandos a seguir.

```
SUBJECT TO
    N_[nutriente] : SUM(alimento: quant*comprar) > nivel[nutriente]
```

Não há necessidade de indicar a não-negatividade das variáveis. Esta restrição é assumida implicitamente pelo MPL. Caso as variáveis sejam livres ou não-positivas, aí sim é necessário explicitar o tipo. Para finalizar a descrição do modelo incluímos o comando END. O modelo completo é dado na Figura 3.4 a seguir.

```
{ Arquivo: dieta.mpl
Problema da Dieta: Determinar uma combinação de alimentos que forneça
uma quantidade mínima de nutrientes }
TITLE dieta

INDEX
  alimento = (arroz, ovos, leite, feijao)
  nutriente = (energia, proteina, caloria)

DATA
 ! Custo de cada alimento a ser considerado na dieta
  preco[alimento] = (14 13 9 19)
 ! Quantidade mínima total de cada nutriente na dieta;
  nivel[nutriente] = ( 2000 65 800 )
 ! Quantidade de nutrientes presente em cada tipo de alimento
  quant[alimento,nutriente] = (170 3 12
                        160 13 54
                        160 8 285
                        337 22 86)

VARIABLES
! Número de porções de cada alimento que ira compor a dieta
  comprar[alimento]

MODEL
! Obter a dieta de menor custo possível
  MIN Custo_total = SUM(alimento: preco*comprar)
SUBJECT TO
! A dieta deve satisfazer níveis minimos de cada nutriente
  N_[nutriente] :SUM(alimento: quant*comprar) > nivel[nutriente]
END
```

Figura 3.4: Problema da Dieta na Sintaxe MPL

Recursos da Linguagem

Um vez definido um modelo, diversas ações podem ser executadas: análise da estrutura do modelo, resolução de um exemplar entre outras. Apesar do MPL facilitar a interação entre um modelo e um sistema de resolução, o MPL é uma linguagem para descrever, documentar e analisar um modelo de otimização, não é um sistema para resolver problemas de otimização. Na Seção 3.2 iremos descrever brevemente alguns sistemas de resolução que podem ser acionados via MPL.

Formatos *mps* e *lp*

Se a sintaxe do modelo estiver correta, é possível gerar o exemplar do modelo associado aos dados definidos na seçao DATA em diversos formatos: *mps*, *lp*, *cplex*, *lindo*, entre outros. Este recurso é importante, pois, além de permitir que o exemplar possa ser usado como dado de entrada em diversos sistemas de otimização, permite também que seja lido e entendido sem um conhecimento prévio da sintaxe de uma determinada linguagem. A maioria das linguagens de modelagem possuem este recurso. Para gerar o exemplar acione o comando RUN no menu principal, seguido de GENERATE FILE, e escolha o formato desejado. As Figuras 3.5 e 3.6 mostram o Exemplar (2.-1) nos formatos *mps* e *lp*.

```
NAME                 DIETA              (MIN)
ROWS
 N                   CUSTO_TOTAL
 G                   N_ENERGIA
 G                   N_PROTEINA
 G                   N_CALORIA
COLUMNS
 COMPRAR_ARROZ       CUSTO_TOTAL        14
 COMPRAR_ARROZ       N_ENERGIA          170
 COMPRAR_ARROZ       N_PROTEINA         3
 COMPRAR_ARROZ       N_CALORIA          12
 COMPRAR_OVOS        CUSTO_TOTAL        13
 COMPRAR_OVOS        N_ENERGIA          160
 COMPRAR_OVOS        N_PROTEINA         13
 COMPRAR_OVOS        N_CALORIA          54
 COMPRAR_LEITE       CUSTO_TOTAL        9
 COMPRAR_LEITE       N_ENERGIA          160
 COMPRAR_LEITE       N_PROTEINA         8
 COMPRAR_LEITE       N_CALORIA          285
 COMPRAR_FEIJAO      CUSTO_TOTAL        19
 COMPRAR_FEIJAO      N_ENERGIA          337
 COMPRAR_FEIJAO      N_PROTEINA         22
 COMPRAR_FEIJAO      N_CALORIA          86
RHS
 RHS1                N_ENERGIA          2000
 RHS1                N_PROTEINA         65
 RHS1                N_CALORIA          800
ENDATA
```

Figura 3.5: Problema da dieta no formato *mps*

```
MINIMIZE Custo_total =
14 comprar_arroz + 13 comprar_ovos + 9 comprar_leite + 19 comprar_feijao

SUBJECT TO

N_energia:
170 comprar_arroz + 160 (comprar_ovos + comprar_leite)
+ 337 comprar_feijao >= 2000

N_proteina:
3 comprar_arroz + 13 comprar_ovos + 8 comprar_leite
+ 22 comprar_feijao >= 65

N_Caloria:
12 comprar_arroz + 54 comprar_ovos + 285 comprar_leite +
86 comprar_feijao >= 800
```

Figura 3.6: Problema da dieta no formato *lp*

O formato *mps* é o formato mais usado para representar um exemplar de um problema. Foi criado para unificar a comunicação de dados entre os diversos tipos de sistemas de resolução e facilitar a comunicação de dados na comunidade científica. Possui um formato simples, mas tem algumas desvantagens: não é possível identificar a estrutura do modelo e possui informações redundantes. Dada a proliferação, nos últimos anos, de linguagens de modelagem e sistemas de resolução, o formato *mps* perdeu um pouco do seu status de 'formato padrão' devido a pequenas diferenças nos arquivos *mps* gerados e manipulados pelos diversos sistemas [19]. Ainda assim, é um formato bastante usado nas bibliotecas de problemas disponibilizadas na *www*. O formato *lp* é o mais próximo da estrutura algébrica que usamos normalmente para definir um problema de otimização e tem sido usado por vários sistemas de modelagem.

Acionando o sistema de resolução

Um outro recurso importante dos sistemas de modelagem é a interface com sistemas de resolução. A maioria dos sistemas de modelagem estão acoplados a sistemas de resolução e/ou permitem que este acoplamento seja feito. O sistema MPL pode ser acoplado a diversos sistemas de otimização, por exemplo: Canopt, CPLEX, FortMP [48]. Para fazer a conexão, acesse o comando OPTION no menu principal seguido de SOLVER MENU e escolha o sistema desejado ou disponível. Apenas um sistema pode ser definido de cada vez. Se algum sistema de otimização estiver acoplado ao MPL, o exemplar pode então ser resolvido através do comando RUN seguido do SOLVE. Note que quando um sistema de resolução está acoplado ao sistema de modelagem não há necessidade de gerar o exemplar no formato *mps* ou *lp*. O exemplar é passado internamente para o sistema de resolução no formato adequado. Recentemente Fourer [19] fez uma proposta de padronização da forma de comunicação de dados entre os sistemas de modelagem e os sistemas de resolução que pode facilitar a decisão de adquirir um sistema ou mesmo o processo de desenvolvimento de ferramentas computacionais para otimização (sistema de modelagem e/ou resolução).

Relatório da solução

Uma vez acionado o sistema de otimização, o sistema MPL abre uma janela para mostrar o andamento do processo de solução. Isto é detalhes sobre o exemplar (número de linhas, colunas, elementos diferentes de zero, número de iterações, entre outros). Quando o processo de solução se encerra é possível ver a solução, caso haja, através do comando VIEW. A apresentação da solução é pré-definida e o usuário tem pouco controle sobre o formato do relatório. A linguagem MPL possui diversos outros comando e recursos que iremos apresentar à medida que sejam necessários. Um tutorial sobre o uso da linguagem, mantido pelos proprietários, está disponível em [34].

3.1.2 As linguagem XPRESS-MOSEL e AMPL

Nesta seção faremos uma breve introdução às linguagens de modelagem XPRESS-MOSEL e AMPL. A sintaxe destas duas linguagens, apesar de bem distintas, possuem uma estrutura próxima das linguagens de programação estruturadas (C, PASCAL, FORTRAN90). Ambas permitem a inserção de algoritmos que manipulam as estruturas definidas para descrever o modelo, o que pode ser muito útil quando modelamos algumas classes de problemas de otimização linear inteira (e.g. corte de estoque, roteamento, dimensionamento de lotes). Possuem dois modos de uso, via interface gráfica ou através de linhas de comando, e estão disponíveis para várias plataformas. A linguagem XPRESS-MOSEL permite ainda que o modelo seja manipulado dentro de um código escrito na linguagem C ou FORTRAN. Nesta seção damos ênfase à sintaxe da linguagem para a descrição de um modelo.

Estrutura da linguagem XPRESS-MOSEL

A linguagem XPRESS-MOSEL[2] pode ser vista como uma linguagem de modelagem e como uma linguagem de programação pois possui comandos que facilitam a declaração e manipulação de um modelo matemático (definição de variáveis e restrições) bem como comandos para controle (seleção e loops) que permitem a codificação de algoritmos e manipulação de sistemas de resolução [11]. É uma linguagem 'procedural' e não declarativa, isto é os comandos são compilados ao invés de interpretados. Possui uma interface gráfica, XPRESS-IVE, com diversos recursos interessantes que facilitam a utilização do sistema XPRESS. Neste texto iremos nos ater na descrição da sintaxe da linguagem. Maiores informações sobre o XPRESS-IVE pode ser obtido em [11].

A estrutura da linguagem XPRESS-MOSEL é resumida a seguir.

MODEL nome do modelo

Instruções para compilação

Definição de parâmetros

Definição do modelo e/ou algoritmos

END-MODEL

O comando MODEL é usado para nomear o modelo e determinar o início da descrição do programa, o fim do mesmo é indicado através do comando END-MODEL. As instruções para o compilador são fornecidas através do comando USES e OPTIONS, e são optativos. A definição das estruturas para o armazenamento de dados e definição das variáveis são iniciadas e finalizadas com os comandos DECLARATION e END-DECLARATION, respectivamente. Estas definições são opcionais e podem ser feitas em qualquer parte do código. A definição do modelo e/ou algoritmo é

[2] Para fazer o *Download* da versão resumida (*Free student version*)ou licença academica gratuita use o endereço http://www.dashoptimization.com (última vista em 13/06/2005).

feita através de comandos similares aos usados em linguagens de programação, com algumas palavras chaves reservadas para manipulações algébricas. Comentários, isto é, trechos de texto a serem ignorados pelo compilador, são precedidos do ponto de exclamação, !, ou escritos entre os símbolos (! !). Qualquer editor de texto que gere arquivos livres de formatação (tipo .txt) pode ser usado para criar o arquivo.

O modelo da dieta (3.1) na sintaxe do XPRESS-MOSEL pode ser lido nas Figuras 3.7 e 3.8 a seguir. Todas as palavras reservadas para a linguagem, (e.g. ARRAY, FORALL, SUM) estão escritas em letras maiúsculas.

```
{ Arquivo: dieta.mos
Problema da Dieta: Determinar uma combinação de alimentos que forneça
uma quantidade mínima de nutrientes }
MODEL dieta
USES "mmxprs"! define o sistema mmxprs para resolver o exemplar
DECLARATIONS
  ! definição dos índices
  alimento = {"arroz", "ovos", "leite", "feijao"};
  nutriente = {"energia", "proteina", "Caloria"};

  ! definição das estruturas para receber dados do exemplar
 ! Custo de cada alimento a ser considerado na dieta
  preco: ARRAY (alimento) OF REAL;
 ! Quantidade mínima total de cada nutriente na dieta;
  nivel: ARRAY (nutriente) OF REAL;
  (! Quantidade de nutrientes presente em cada tipo de alimento
 considerado na dieta !)
  quant: ARRAY (alimento,nutriente) OF REAL;

 ! definição das variáveis de decisão
 ! Número de porções de cada alimento que ira compor a dieta
  comprar: ARRAY (alimento) OF MPVAR;
END-DECLARATION
```

Figura 3.7: Problema da Dieta na Sintaxe XPRESS-MOSEL

```
! continuação do arquivo dieta.mos
  ! dados do exemplar
 ! Custo de cada alimento a ser considerado na dieta
        preco := [14, 13, 9, 19];
 ! Quantidade mínima total de cada nutriente na dieta;
        nivel := [ 2000, 65, 800];
  ( ! Quantidade de nutrientes presente em cada tipo de alimento considerado
na dieta ! )
        quant := (170, 3, 12,
                  160, 13, 54,
                  160, 8, 285,
                  337, 22, 86)

 ! definição da função-objetivo
        Custo_total := SUM(j in alimento) preco(j)*comprar(j);

 ! definição das restrições
! A dieta deve satisfazer níveis minimos de cada nutriente
    FORALL(i in nutriente)
      SUM(j in alimento) quant(j,i)*comprar(j) >= nivel(i);

 ! resolve o exemplar e critério de otimização
        MINIMIZE (Custo_total)

 ! relatório da solução
  WRITELN("Custo_total: ", GETOBJVAL)
  WRITELN("Numero de porções a ser incluída na dieta: ")
  FORALL(j in alimento) WRITELN(j, ": ", GETSOL(comprar(j)))

END-MODEL
```

Figura 3.8: Problema da Dieta na Sintaxe XPRESS-MOSEL (cont. da Figura 3.7)

Note que apesar de manter a mesma estrutura que o MPL, a descrição do modelo na sintaxe do XPRESS-MOSEL é muito diferente. A definição das estruturas para armazenamento de dados e variáveis, precedidas do comando DECLARATIONS, segue o padrão das linguagens de programação. Os vetores e matrizes são definidos pelo comando ARRAY seguido da definição do tipo que pode ser: real (valor real entre -1.7e+308 e 1.7e+308, integer (valor inteiro entre -214783648 e 2147483647) e boolean (verdadeiro ou falso: resultado de uma expressão boleana ou lógica) e MPVAR para definir as variáveis do modelo. O comando END-DECLARATIONS finaliza esta seção do código.

A descrição da função-objetivo e das restrições é mais livre do que no MPL e pode ser feita em qualquer local do código. A função-objetivo só é identificada como tal através do comando MINIMIZE (ou MAXIMIZE) que possui duas funções: invocar o sistema de resolução, definido através do comando USES, e definir o critério de otimização. O comando FORALL foi usado para expandir o conjunto de restrições. Observe que a sintaxe para definição das restrições é bem diferente da sintaxe do MPL. Além do comando FORALL, é necessário definir para que índices a restrição será expandida. No nosso exemplo para todo valor i do índice *nutriente*. Note também o uso do símbolo SUM. Na linguagem XPRESS-MOSEL o usuário possui diversas formas de gerar o relatório da solução. Através dos comandos GETOBJVAL e GETSOL é possível recuperar a solução e escrevê-la no formato desejado através do comando WRITELN. A linguagem XPRESS-MOSEL possui diversos outros comandos e recursos que iremos apresentar à medida que sejam necessários. Uma descrição completa desta linguagem pode ser lida em [11].

Estrutura da linguagem AMPL

A linguagem AMPL[3] possui uma sintaxe bem próxima da descrição algébrica dos modelos de otimização que estamos habituados a usar. Como o XPRESS-MOSEL, o AMPL também é uma linguagem procedural, isto é os comandos são compilados ao invés de interpretados. Possui uma interface gráfica para ambiente *Windows*, porém com menos recursos do que as interfaces do MPL e do XPRESS-MOSEL.

O modelo é criado em um arquivo tipo texto escrito de acordo com a sintaxe do AMPL (nomemodelo.mod). Os dados do exemplar a ser resolvido são fornecidos em um outro arquivo tipo texto (dadosmodelo.dat). O aplicativo AMPL é então executado criando um ambiente próprio para receber comandos associados com a compilação e resolução do modelo. A forma de manipulação desta linguagem é bem próxima da forma de manipulação das linguagens de programação em geral, e pode variar de um sistema operacional para outro. Ao invocar o aplicativo no *prompt* do sistema operacional, o ambiente AMPL é criado para receber os comandos necessários. A seguir os comandos MODEL e DATA podem ser usados para compilar o modelo e acessar os dados de um exemplar. Veja abaixo.

ampl: model *nomemodelo.mod*;

[3] Uma versão resumida pode ser obtida gratuitamente no endereço http://www.ampl.com/DOWNLOADS/index.html (última visita em 13/08/2005)

ampl: data *dadosmodelo.dat*;

ampl: option solver *nome do sistema de resolução*;

ampl: show;

ampl: solve;

ampl: display *nomevariavel*;

O comando *option solver* define o sistema de resolução que será usado para resolver o exemplar do modelo especificado em *dadosmodelo.dat*. O comando *show* exibe as principais estruturas do modelo. Os comandos *solve* e *display* são usados para resolver o exemplar e mostrar a solução do mesmo, respectivamente.

O arquivo *nomemodelo.mod* contendo a descrição do modelo na sintaxe do AMPL deve seguir a estrutura geral da linguagem composta dos comandos SET, PARAM, VAR, MINIMIZE (ou MAXIMIZE) e SUBJECT TO. Diferente da sintaxe do MPL, cada índice, vetor, variável ou restrição deve ser precedido do comando que o define. No MPL estes comandos abrem uma seção no código onde todos as definições são feitas. A estrutura geral para a descrição de um modelo é dada a seguir.

- SET

define um índice;

- PARAM

define uma estrutura (vetor ou matriz) que irá armazenar os elementos conhecidos do exemplar, fornecidos no arquivo nomemodelo.dat;

- VAR

define variáveis de decisão;

- MINIMIZE (ou MAXIMIZA)

define a função-objetivo e o critério de otimização;

- SUBJECT TO

define um conjunto de restrições.

O conteúdo do arquivo dieta.mod contendo o problema da dieta (3.1) na sintaxe do AMPL é mostrado na Figura 3.9 a seguir. Os comentários são precedidos do símbolo #, e todos os comandos devem finalizar com ponto e vírgula (;). Os dados necessários para gerar um exemplar (dieta.dat) pode ser visto na Figura 3.10.

A linguagem AMPL possui diversos outros comandos e recursos que serão apresentados a medida que forem necessários. Mais detalhes sobre a sintaxe e recursos desta linguagem podem ser obtidos em [18]. Na próxima seção iremos discutir alguns sistemas de resolução que podem ser acoplados às linguagens de modelagem.

```
# Arquivo: dieta.mod
# Problema da Dieta: Determinar uma combinação de alimentos que forneça
# uma quantidade mínima de nutrientes
# Definição dos índices
        SET alimento;
        SET nutriente;
# Definição das estruturas para receber dados do exemplar
        PARAM preco { alimento }
        PARAM nivel { nutriente };
        PARAM quant {nutriente, alimento };
# Definição das variáveis de decisão
        VAR comprar { j in alimento } >=0;
# definição da função-objetivo
        minimize custo_total: SUM { j in alimento } preco[j] * comprar[j];
# A dieta deve satisfazer níveis minimos de cada nutriente
        SUBJECT TO N_ { i in nutriente }:
        SUM { j in alimento } quant[i,j] * comprar[j] >= nivel[i];
# fim arquivo dieta.mod
```

Figura 3.9: Problema da Dieta na Sintaxe AMPL

```
# Arquivo: dieta.dat
# Problema da Dieta:
# Determinar uma combinação de alimentos que forneça
# uma quantidade mínima de nutrientes

# Índices
        SET alimento := arroz ovos leite feijao;
        SET nutriente := energia proteina caloria;

# Dados: Custo dos alimento
        PARAM preco :=
        arroz 14
        ovos 13
        leite 9
        feijao 19;

# Dados: Nível mínimo de nutrientes na dieta
        PARAM nivel :=
        energia 2000
        proteina 65
        caloria 800;

# Quantidade de nutirente presente em cada alimento
        PARAM quant:
                        arroz ovos leite feijao :=
        energia                 170 80 130 100
        proteina                  3 6 6.1 6
        caloria                 12 25 232 28;
# fim arquivo dieta.dat
```

Figura 3.10: Dados para o Problema da Dieta na Sintaxe AMPL

3.2 Sistemas de resolução

Para validar um modelo de otimização é necessário analisar se a solução obtida fornece uma resposta adequada. Existem diversos sistemas computacionais comerciais e de pesquisa desenvolvidos para este fim. Uma revisão sobre sistemas não-comerciais pode ser encontrada em [30], e sistemas comerciais em [5] e [48]. Neste texto iremos utilizar dois sistemas: CPLEX [27] e o XPRESS-MP [11] . Estes dois sistemas estão disponíveis (em versões resumidas) via *www*, e podem ser acoplados às linguagens de modelagem descritas na Seção 3.1.

Além da versão *standalone*, os dois sistemas possuem biblioteca de subrotinas de apoio (entradas de dados, manipulação de dados, otimização, etc.) que permitem ao usuário desenvolver técnicas de resolução sem a necessidade de codificar uma grande quantidade de subrotinas, tornando o processo de testar novos algoritmos mais ágil e robusto. Estes dois sistemas tem sido usados na literatura como padrão de referência para a análise comparativa de novas técnicas de solução de problemas de otimização linear e linear inteira.

Os dois sistemas contém implementações dos principais algoritmos de solução para problemas de otimização linear: método simplex revisado (e.g. [21] e [6]) e métodos de ponto interior (e.g. [57] e [22]); e problemas de otimização linear inteira: planos de corte, *branch and bound* e *branch and cut*(e.g. [58], [45], [21]). Apesar de apresentarem desempenhos distintos, possuem recursos que permitem ajustar o desempenho dos algoritmos de acordo com o tipo do problema.

Neste texto, usaremos estes sistemas mantendo os parâmetros padrões sugeridos pelos proprietários. No entanto, chamamos a atenção de que nem sempre esses são os mais adequados. Uma vez concluído o processo de validação de um modelo de otimização, é necessário testar as diversas alternativas que os sistemas disponibilizam para determinar o conjunto de parâmetros que melhor se comportam na resolução do problema. Esta sugestão é particularmente útil quando o modelo de otimização inclui variáveis inteiras ou binárias. Além disso, quando o modelo construído possui uma estrutura especial (e.g. problema do transporte, da designação, de dimensionamento de lotes, da mochila, do caixeiro viajante, sequenciamento de tarefas [21], [58]) é recomendável desenvolver e implementar algoritmos que explorem esta estrutura. Nestes casos a solução fornecida pelos modelos de otimização servem de parâmetro para avaliar a qualidade das soluções obtidas por outros métodos.

Não nos preocupamos aqui em explicar a forma de operação dos aplicativos CPLEX e XPRESS-MP pois eles serão acionados e controlados através das linguagens de modelagem. Mais informações sobre estes sistemas e os métodos de solução que empregam podem ser obtidas nas referências citadas ao longo desta seção.

Capítulo 4

Aplicações de Otimização Linear

É vasto o campo de aplicações da otimização linear. Esta classe de modelos têm sido usada para representar problemas em áreas tão distintas quanto finanças [51] e medicina ([39],[32]). Williams [56] apresenta uma descrição resumida de um amplo espectro de aplicações com estudos de casos que incluem, entre outros, problemas na indústria petrolífera, problemas de distribuição e de manufatura. Caixeta-Filho [9] apresenta algumas aplicações no setor agroindustrial. Uma bibliografia comentada de artigos sobre formulações de modelos de otimização linear que cobre o período entre 1939 e 1979 pode ser encontrada em [38]. É um banco de dados em construção (mantido *online*) e artigos mais recentes estão sendo incluídos gradativamente. Murphy [39] observa que através destas aplicações é possível analisar a evolução das estruturas usadas para representar determinadas situações práticas. Sistematizando o processo de construção de modelos usado por estes autores é possível estabelecer alguns princípios que permitem o desenvolvimento de modelos eficientes. Neste capítulo ilustramos alguns destes princípios através da discussão de modelos para problemas clássicos que podem ser usados isoladamente, mas que também aparecem como sub-modelos na representação de situações mais complexas.

4.1 Planejamento da produção

Considere um consórcio de empresas do ramo madeireiro. Este consórcio opera, entre outros negócios, reservas florestais para extração de madeira ecológica, serrarias, fábricas de móveis e de papel, e gráficas. A operação destas empresas envolve decisões de natureza muito variada: contratação de pessoal, distribuição da madeira extraída nas reservas para as serrarias, novos investimentos, planejamento da produção, entre outras. Englobar todas estas decisões em único modelo pode se tornar uma tarefa árdua e ineficiente. Ao invés disto, vamos simplificar o problema, modelando separadamente algumas decisões a serem tomadas pelos administradores

do consórcio.

Vamos iniciar nosso estudo modelando um problema muito simples adaptado de um execício proposto em [55].

Exemplo 4.1 *O Presidente, Antônio Castor, da Companhia Ramos de Carvalho quer utilizar do melhor modo possível os recursos de madeira em uma de suas reservas florestais. Dentro desta região, há uma serraria e uma fábrica de compensados, isto é, as toras podem ser convertidas em madeira beneficiada ou compensado. Produzir uma mistura de produto beneficiado requer 1 metro cúbico de pinho e 4 metros cúbicos de canela. Produzir 100 metros quadrados de madeira compensada requer 2 metros cúbicos de pinho e 4 metros cúbicos de canela. Esta região tem disponível no momento 32 metros cúbicos de pinho e 72 metros cúbicos de canela. Compromissos de venda exigem que sejam produzidos pelo menos 5 metros cúbicos de madeira beneficiada e 1200 metros quadrados de madeira compensada. As contribuições ao lucro são 45 u.m. (unidades monetárias) por 1 metro cúbico de produtos beneficiados e 60 u.m. por 100 metros quadrados de compensado.*

Seguindo a estrutura usada para obter o modelo para o problema da dieta no Capítulo 2 vamos identificar no enunciado do Exemplo 4.1 os elementos conhecidos, os elementos desconhecidos, o objetivo e as restrições do problema.

Elementos conhecidos

- lucro associado aos produtos;

- quantidade necessária de cada tipo de insumo para a produção de cada item

- quantidade de insumo disponível.

Elementos desconhecidos

- quanto produzir de cada item.

Objetivo a ser alcançado

- obter o maior lucro possível.

Restrições

- quantidade de insumos limitada

- compromissos de venda.

O modelo de otimização linear pode ser construído definindo os índices j para representar o tipo de produto (madeira beneficiada, compensado) e i para representar o tipo de insumo (canela, pinho). Estes índices serão usados para definir os demais elementos do modelo.

Índices

$j = 1, \ldots, 2$ representa respectivamente os produtos: madeira beneficiada, compensado;

$i = 1, \ldots, 2$ representa respectivamente os insumos: canela, pinho.

Variáveis de decisão

Precisamos decidir quanto produzir de cada item.

x_1: quantidade de madeira beneficiada produzida (em m^3)

x_2: quantidade de compensado produzido em ($100m^2$)

Note que é necessário definir uma unidade de medida para cada uma das variáveis já que os produtos são medidos em unidades diferentes.

O critério para a tomada de decisão é maximizar o lucro obtido com a venda dos produtos. O retorno associado a cada produto é conhecido. Assim, se vendermos 1 m^3 de madeira beneficiada teremos um lucro de 45. Supondo válido o axioma de *proporcionalidade* temos que se a venda for duplicada, o lucro dobra. De uma maneira geral, se vendermos x_1 m^3, o lucro associado será de $45x_1$. Similarmente, o lucro associado a venda de compensado pode ser calculado como: $60x_2$. Usando o axioma de *aditividade* temos que o lucro total associado à venda dos dois produtos é $45x_1 + 60x_2$. Como queremos obter o maior lucro possível, temos:

função-objetivo

Maximizar o lucro total:

$\max z = 45x_1 + 60x_2$

A quantidade de recursos é limitada. Sabemos que a quantidade total de pinho utilizada para a produção dos dois items não pode exceder os $32\,m^3$ disponíveis. Lendo o enunciado do Exemplo 4.1 temos que são necessários $1m^3$ de pinho para a produção de $1m^3$ de madeira beneficiada e 2 m^3 para produção de $100m^2$ de compensado. Usando os axiomas de *proporcionalidade* e *aditividade* e estendendo o raciocínio para o uso de canela temos o seguinte conjunto de restrições:

Restrições relativas à disponibilidade de insumos

- Pinho:

$$x_1 + 2x_2 \leq 32 \tag{4.0}$$

- Canela:

$$4x_1 + 4x_2 \leq 72. \tag{4.0}$$

Compromissos de venda determinam uma quantidade mínima a ser produzida de cada um dos items. Temos então um segundo conjunto de restrições.

Restrições relativas à produção mínima

- Madeira beneficiada:

$x_1 \geq 5$

- Compensado:

$x_2 \geq 12$

Observe que a variável x_2 foi definida em $100m^2$, por isso usamos um limite inferior de 12 para a produção de compensado. O modelo de otimização linear que representa o problema pode ser resumido na Expressão 4.1 abaixo.

$$\begin{aligned} \min \quad & z = 45x_1 + 60x_2 \\ \text{sujeito a} \quad & \\ & x_1 + 2x_2 \leq 32 \\ & 4x_1 + 4x_2 \leq 72 \\ & x_1 \geq 5, \quad x_2 \geq 12. \end{aligned} \tag{4.-2}$$

O Modelo (4.1) envolve apenas duas variáveis e pode ser resolvido facilmente pelo método gráfico ou manualmente pelo método simplex (e.g. [21], [6]). Note que sua estrutura é muito similar ao modelo do problema da dieta (2.-1). De fato se considerarmos que temos n produtos, m insumos, o Modelo 4.1 pode ser generalizado e escrito como:

$$\begin{aligned} \max \quad & z = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ \text{sujeito a} \quad & \\ & a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n \leq b_1 \\ & a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n \leq b_2 \\ & \vdots \\ & a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \leq b_m \\ & x_j \geq u_j \end{aligned} \tag{4.-6}$$

onde: a_{ij} representa a quantidade de insumo i usada na produção do item j, c_j e u_j são o lucro e a demanda mínima do item j, e b_i é a quantidade de insumo i disponível.

4.1.1 Planejamento Multi-período

A resposta do Modelo 4.1 fornece uma sugestão para o planejamento considerando um único período. Seria interessante analisar como seria a produção se o planejamento for feito considerando um horizonte maior, por exemplo planejar a produção para os meses de janeiro, fevereiro, março e abril. Neste caso vamos supor que é conhecida uma previsão para a venda dos produtos e para a disponibilidade de

insumos em cada um desses meses (ver Tabelas 4.1 e 4.2). É possível fazer estoque de um mês para o outro e, então, a demanda do mês de março pode ser atendida, por exemplo, com produtos fabricados anteriormente (em janeiro ou fevereiro). O custo mensal para manter cada um dos produtos em estoque é de 4 u.m. para cada m^3 de madeira beneficiada e 3.5 u.m. para cada $100\,m^2$ de compensado.

Tabela 4.1: Previsão de Vendas Mensal

Produto	Mês			
	Janeiro	Fevereiro	Março	Abril
M. beneficiada	43	0	100	53
Compensado	45	100	0	72

Tabela 4.2: Disponibilidade de insumos mensal

Insumo	Mês			
	Janeiro	Fevereiro	Março	Abril
Pinho	32	32	20	32
Canela	72	80	40	72

Para adaptar o Modelo (4.1) de forma a refletir esta nova situação vamos precisar de novas classes de restrições que também poderão ser úteis em outros contextos. Observe inicialmente que temos um novo conjunto de decisões a tomar. Não queremos apenas decidir *quanto* produzir de cada item. Precisamos definir também *quando* produzir. Assim de duas passamos a ter oito variáveis (dois produtos x quatro períodos). Poderíamos simplesmente ampliar o índice $j = 1, 2$ para $j = 1, \ldots, 8$ e por exemplo definir x_3 como sendo o número de m^3 de madeira beneficiada produzida em fevereiro. No entanto é mais natural se mantivermos o índice j como definido anteriormente e criamos um novo índice para representar o período. Assim temos:

Novo índice

$t = 1, \ldots, 4$ representa os respectivamente os meses: Janeiro, Fevereiro, Março e Abril.

Redefinição da variável produção

- quanto e quando produzir de cada item em cada mês

x_{jt}: quantidade do item j produzida no mês t.

O critério de otimização também precisa ser reavaliado. Na definição anterior consideramos que tudo o que estava sendo produzido seria vendido. Esta suposição não é adequada pois parte da produção em um período pode ser armazenada e vendida no período seguinte. Precisamos de dois novos conjuntos de variáveis.

Novas variáveis de decisão

- Quanto e quando estocar de cada item em cada mês:

y_{jt}: quantidade do item j em estoque no final do mês t;

- Quanto e quando vender de cada item em cada mês:

s_{jt}: quantidade do item j vendido no mês t.

o lucro associado à venda dos produtos será dado pela diferença entre o valor total obtido com a venda dos produtos e o custo total de armazenamento.

Redefinição da função-objetivo

- Retorno total associado à venda dos produtos:

$$Totalvendas = \sum_{j=1}^{2}\sum_{t=1}^{4} p_j s_{jt} \quad .$$

- Custo total associado ao armazenamento dos produtos

$$Totalestoque = \sum_{j=1}^{2}\sum_{t=1}^{4} c_j y_{jt} \quad .$$

- Objetivo: maximizar Lucro total

$$max \quad Lucro = Totalvendas - Totalestoque$$

onde p_j e c_j são o preço de venda e o custo de estoque do produto j respectivamente. Note que estamos supondo que estes valores não mudam ao longo do horizonte de planejamento. O conjunto de restrições também precisa ser redefinido. A disponibilidade de insumos varia de mês a mês. Fazendo b_{it} a disponibilidade do insumo i no mês t as Restrições (4.1) e (4.1) podem ser redefinidas como:

Redefinição das restrições relativas à disponibilidade de insumo

- Pinho:

$$x_1 + 2x_2 \leq b_{1t}, \quad t = 1, \ldots, 4 \tag{4.-11}$$

- Canela:

$$4x_1 + 4x_2 \leq b_{2t}, \quad t = 1, \ldots, 4 \tag{4.-11}$$

Note que as restrições em (4.1.1) representam o conjunto das quatro restrições, uma para cada valor de t, relativas à disponibilidade de pinho em cada mês. Juntas as Restrições (4.1.1) e (4.1.1) representam oito restrições. Esta forma compacta de representação é muito útil, pois facilita a leitura e compreensão do modelo matemático. Antes de prosseguirmos, vamos avaliar o modelo que construímos até agora. Resumindo as informações acima temos:

$$\begin{aligned} \min \qquad & Lucro = \sum_{j=1}^{2}\sum_{t=1}^{4} p_j s_{jt} - \sum_{j=1}^{2}\sum_{t=1}^{4} c_j y_{jt} \\ \text{sujeito a} \\ & x_{1t} + 2x_{2t} \leq b_{1t}, \qquad t = 1, \ldots, 4 \\ & 4x_{1t} + 4x_{2t} \leq b_{2t}, \qquad t = 1, \ldots, 4 \\ & x_{jt} \geq 0, s_{jt} \geq 0, y_{jt} \geq 0, j = 1, 2; t = 1, \ldots, 4. \end{aligned} \tag{4.-13}$$

De acordo com a função-objetivo do Modelo (4.-10) queremos obter o maior valor possível para a diferença entre o retorno das vendas e o custo do armazenamento. O conjunto de restrições está nos dizendo apenas que as variáveis de venda e estoque devem ser não-negativas. Ora, o lucro máximo será obtido atribuindo valor zero (limite inferior das variáveis) para as variáveis de estoque, y_{jt}, e valor infinito para as variáveis de venda s_{jt}. Mas obter lucro infinito não é realista. De fato, se tentarmos resolver o modelo acima obteremos a resposta que o lucro será tanto maior quanto for o valor atribuído às variáveis de venda, isto é o Problema (4.-10) é ilimitado. Em geral, apesar de matematicamente correto, encontrar a resposta de que um problema de otimização é ilimitado (ou mesmo inviável), em geral, indica um modelo mal formulado [56]. No nosso caso, é necessário analisar que:

1. não há limite para as vendas;
2. as variáveis de produção, venda e estoque não estão relacionadas, faltam restrições de acoplamento entre as variáveis.

Estas duas observações sugerem que precisamos rever o conjunto de restrições. Na Tabela 4.1 foi fornecida uma previsão para as vendas de cada produto em cada mês. Podemos pensar nestes valores como limite superior para as variáveis s_{jt}. Este novo conjunto de restrições torna o problema limitado, e a solução ótima será obtida atribuindo o limite superior para as variáveis de venda. Por exemplo, no mês de janeiro serão vendidos $43m^3$ de madeira beneficiada e $45m^2$ de compensado. Mas note que não temos insumo suficiente (pinho) para produzir esta quantidade de produtos. Ou seja, as vendas estão limitadas pela capacidade de produção, não podemos vender mais do que podemos produzir[1]. Observe na Tabela 4.1 que a previsão de vendas, d_{jt}, indica que não há demanda por madeira beneficiada no mês de fevereiro, mas ela é alta no mês de março. Talvez seja interessante produzir madeira beneficiada em fevereiro, armazenar e vender em março. O mesmo raciocínio pode ser estendido

[1] Em algumas situações pode-se supor que pedidos em atraso (*backorders*), venda de produtos produzidos em períodos posteriores ao considerado, são viáveis [16].

para os demais períodos. Assim, a quantidade produzida em um período t somado à quantidade em estoque no final do período anterior $(t-1)$ deve ser menor ou igual à quantidade de produto vendido. O que sobrar no final do período t será armazenado. Lembrando que y_{1t} é a quantidade de produtos em estoque no final do período t podemos representar a restrição de balanceamento relativa à madeira beneficiada como:

Novas restrições: *balanceamento*

- Madeira Beneficiada

$$y_{1t-1} + x_{1t} = s_{1t} + y_{1t}, \quad t = 1, \ldots, 4 \tag{4.-14}$$

Similarmente, temos:

- Compensado

$$y_{2t-1} + x_{2t} = s_{2t} + y_{2t}, \quad t = 1, \ldots, 4 \tag{4.-14}$$

Este conjunto de restrições é bastante comum em planejamento multi-período e aparece com freqüência na modelagem de diversos problemas. São chamadas de restrições de *balanceamento entre produção e estoque*, ou simplesmente restrições de *balanceamento*.

Naturalmente, surge a dúvida sobre o significado da variável y_{1t-1} quando o mês de janeiro, $t = 1$, é considerado. Apesar de definida implicitamente, a variável $y_{1,0}$ representa a quantidade de madeira beneficiada em estoque no período imediatamente anterior ao início do período considerado no planejamento. Em geral é necessário atribuir um valor para esta variável. Isto é, ela é considerada um *elemento conhecido* do problema. O modelo para o planejamento multi-período é então dado por:

$$\begin{aligned}
\min \quad & Lucro = \sum_{j=1}^{2}\sum_{t=1}^{4} p_j s_{jt} - \sum_{j=1}^{2}\sum_{t=1}^{4} c_j y_{jt} \\
\text{sujeito a} \quad & \\
& x_{1t} + 2x_{2t} \leq b_{1t}, \qquad t = 1, \ldots, 4 \\
& 4x_{1t} + 4x_{2t} \leq b_{2t}, \qquad t = 1, \ldots, 4 \\
& y_{jt-1} + x_{jt} = s_{jt} + y_{jt}, \\
& \qquad\qquad j = 1, 2; t = 1, \ldots, 4 \\
& \qquad\qquad x_{jt} \geq 0, s_{jt} \leq d_{jt}, y_{jt} \geq 0.
\end{aligned} \tag{4.-16}$$

O Modelo (4.-13) possui três tipos de variáveis (*produção*, *venda* e *estoque*) e dois tipos de restrições (*capacidade* e *balanceamento*) totalizando 24 variáveis e 16 restrições (desconsiderando o limite superior para as variáveis de venda). Este número

de variáveis e restrições sugere o uso de uma linguagem algébrica de modelagem para facilitar o trabalho de avaliação deste modelo. Vamos apresentar a seguir o processo de descrição do modelo na sintaxe do MPL (ver Seção 3.1.1) e mostrar um pouco mais dos recursos que esta linguagem oferece.

4.1.2 Modelo na sintaxe do MPL

Exceto pelas restrições de balanceamento, o Modelo (4.-13) é similar ao modelo do problema da dieta. Naturalmente os nomes atribuídos aos índices, vetores e matrizes devem ser diferentes para refletir que temos uma nova aplicação. Vamos utilizar novos recursos da linguagem. O primeiro é a abreviação dos nomes usados para os índices, variáveis e restrições. Isto é feito da seguinte forma:

INDEX

produto := (Madeira_Beneficiada, Compensado) $->$ (MB MC)

assim, usamos no modelo o nome estendido, mas ao gerar o exemplar em um formato específico, o comando $->$ fará com que o sistema use os nomes abreviados, MB no lugar de Madeira_Beneficiada e MC no lugar de Compensado. Isto pode ser útil porque alguns sistemas de resolução limitam o número de caracteres usados no nome de uma variável.

Outro recurso útil é a possibilidade de ler os dados do problema em um arquivo de dados, ao invés de digita-los diretamente no arquivo do modelo. O uso destes arquivos facilita a resolução do problema para diversos cenários. A leitura dos dados em arquivo externo será feita através do comando DATAFILE. Veja na Figura 4.1 a descrição do Modelo (4.-13) na sintaxe do MPL. Observe a inclusão da seção BOUNDS para definir o limite superior para a variável s_{jt}. O uso do comando MACROS permite manipular os dados no cálculo de expressões matemáticas. Note que a definição de *Totalvendas* e *Totalestoque* na seção MACROS permite uma definição mais clara da função-objetivo. Esta definição permite que junto com a solução tenhamos também o valor total arrecadado com as vendas e o valor gasto com os produtos mantidos em estoque.

```
{ Arquivo: Plan_mad_mult.mpl Planejamento da produção multi-periodo }
TITLE Mad_mult;
INDEX
  produto := (Madeira_Beneficiada, Compensado) -> (MB MC)
  mes := (janeiro, fevereiro, março, abril) -> (jan fev mar abr)
  insumo := (pinho, canela) -> (pin ca)
DATA
!Preço de venda de cada produto
  Preço[produto]:= DATAFILE (preco_mad.dat);
!Previsão de venda de cada produto em cada mes
  Demanda[produto,mes] := DATAFILE (max_mad.dat);
!quantidade de insumo necessaria para produzir cada tipo de madeira
  recurso[produto, insumo] := DATAFILE(recurs_mad.dat)
!disponibilidade de insumo
  Disp[insumo,mes] := DATAFILE (dispon_mad_mult.dat);
!custo de armazenamento
  Custoest[produto] := DATAFILE (cest_mad.dat);
VARIABLES
  x[produto,mes] ; ! quantidade produzida
  s[produto,mes] ; !quantidade vendida
  y[produto,mes] ; !quantidade estocada
MACROS
  Totalvendas := SUM(produto,mes: Preço * s);
  Totalestoque := SUM(produto,mes: Custoest * y);
MODEL
  !Maximizar o lucro total
  MAX Lucro = Totalvendas - Totalestoque;
SUBJECT TO
  !restrições de capacidade
    Cap[insumo,mes] : Sum(produto: recurso*x) <= Disp;
  !restrições de capacidade
    Bal[produto, mes] : y[mes-1] + x = s + y;
BOUNDS
    s <= Demanda; !Limite máximo para as vendas
END
```

Figura 4.1: Planejamento da produção muti-período - Sintaxe MPL

Para resolver o exemplar definido na seção DATA, acione o aplicativo CPLEX (Seção 3.2) através dos comandos RUN, Solve CPLEX do menu principal do MPL. Na Figura 4.2 temos o relatório parcial da solução com estatísticas do modelo e do processo de solução. São fornecidas informações sobre o tempo usado para processar o modelo e enviá-lo ao CPLEX (parsing time), tempo de resolução e o numero de iterações para obter a solução ótima. O valor da função-objetivo e das variáveis são mostrados na Figura 4.3.

```
MPL Modeling System - Copyright (c) 1988-2001, Maximal Software, Inc.
--------------------------------------------------------------------
MODEL STATISTICS
Problem name: Mad_mult
Filename: Plan_mad_mult.mpl Date: May 26, 2005 Time: 21:58
Parsing time: 0.05 sec
Solver: CPLEX Objective value: 3653.00000000
Iterations: 8 Solution time: 0.05 sec
Constraints: 16 Variables: 24 Nonzeros: 46
Density: 12
```

Figura 4.2: Relatório da solução

SOLUTION RESULT
Optimal solution found
MAX Lucro = 3653.0000
MACROS

Macro Name	Values
Totalvendas	3720.0000
Totalestoque	67.0000

DECISION VARIABLES
VARIABLE x[produto,mes] :

produto	mes	Activity
Madeira_Beneficiada	janeiro	4.0000
Madeira_Beneficiada	fevereiro	8.0000
Madeira_Beneficiada	abril	4.0000
Compensado	janeiro	14.0000
Compensado	fevereiro	12.0000
Compensado	março	10.0000
Compensado	abril	14.0000

VARIABLE s[produto,mes] :

produto	mes	Activity
Madeira_Beneficiada	janeiro	4.0000
Madeira_Beneficiada	março	8.0000
Madeira_Beneficiada	abril	4.0000
Compensado	janeiro	14.0000
Compensado	fevereiro	12.0000
Compensado	abril	24.0000

VARIABLE y[produto,mes] :

produto	mes	Activity
Madeira_Beneficiada	fevereiro	8.0000
Compensado	março	10.0000

Figura 4.3: Relatório da solução (cont. da Figura 4.2)

As informações contidas nas Figuras 4.2 e 4.3 constam de um único arquivo (Plan_mad_mult.sol) gerado automaticamente pelo aplicativo MPL. Observe que de fato, para um melhor aproveitamento dos recursos visando o lucro, a sugestão é produzir e estocar madeira beneficiada em fevereiro, quando a previsão de vendas é zero, para venda no período seguinte, quando a demanda é alta e a disponibilidade de insumos insuficiente.

Na manufatura de alguns tipos de produtos, muitas vezes as máquinas precisam de algum tipo de preparo antes que alguns items sejam produzidos. A este preparo pode estar associados um custo e um tempo de preparação. Em alguns setores, o tempo e o custo de preparo são dependentes da sequência em que os items são produzidos. Para controlar a produção nestes casos é necessário o uso de variáveis binárias , tema do próximo capítulo.

Capítulo 5

Aplicações de Otimização Linear Inteira

Enquanto problemas de otimização linear contínuo com algumas centenas de variáveis e restrições podem ser resolvidos facilmente, a resolução de algumas classes de problemas de otimização linear inteira não possuem este comportamento. Existe a possibilidade de construir um modelo de otimização linear inteira (ou inteira mista) e descobrir rapidamente o grau de dificuldade da sua resolução. De fato, uma 'boa' formulação do problema pode ser de crucial importância para a eficiência do processo de solução [41]. Contrastando com os problemas de otimização linear contínuo, não se sabe se existem ou não algoritmos polinomiais para resolver muitas classes de problemas de otimização linear inteira. Quando se usa esta classe de modelos é importante se ter em mente o grau de dificuldade associado à sua solução. Um estudo detalhado sobre a classificação de problemas de otimização inteira quanto ao grau de dificuldade de sua solução (complexidade computacional) pode ser visto em [20] e [10]. Saber que um problema é classificado como NP-difícil (*NP-Hard*[20]) é importante porque permite ao usuário entender que a resolução de alguns exemplares será difícil independentemente do algoritmo usado. No entanto, isto não quer dizer que exemplares de grande porte não possam ser resolvidos em um tempo computacional aceitável. Mesmo que a solução ótima não seja encontrada, é possível obter boas soluções viáveis e mostrar quão próximo da solução ótima podem estar.

Algoritmos eficientes para a solução de problemas envolvendo variáveis inteiras são baseados em técnicas de enumeração implícita e combinam vários métodos de solução. A maioria dos algoritmos empregam a técnica de "*relaxação* e *cálculo de limitantes*"([41], [45]) . Considere o problema de otimização inteira (1.-3) definido no Capítulo 1. A idéia é substituir o problema inteiro por um problema mais fácil (*relaxação*) que pode ser usado para obter um limite inferior (limite dual [58]) para o valor ótimo da função objetivo. Se a solução associada for viável, o problema original esta resolvido. Caso contrário a *relaxação* é refinada iterativamente para obter limites melhores. Soluções viáveis fornecem limites superiores (limite primal

[58]) para o valor ótimo da função objetivo. Bons limites superiores podem ser obtidos, por exemplo, através de algoritmos heurísticos [10].

Um problema inteiro pode ser *relaxado* de várias formas [58]. A relaxação linear, obtida substituindo a restrição $x_j \in \mathtt{Z}$ por $x_j \in \Re$, é conveniente para a maioria das aplicações e é a base dos algoritmos implementados nos sistemas de resolução de uso geral (ver Seção 3.2). Neste texto, iremos considerar que um modelo de otimização inteira é uma 'boa' formulação para uma aplicação se o limite inferior fornecido pela relaxação linear associada for bom. Isto é, se RLP_1 e RLP_2 representam o limite inferior de duas relaxações diferentes e tivermos que:

$$RLP_1 \leq RLP_2 \leq Z$$

diremos que a formulação associada a RLP_2 é a melhor das duas. Um modelo de otimização inteira pode ser reformulado de forma a fornecer bons limitantes inferiores através da inclusão de inequações válidas (planos de corte). Uma excelente discussão sobre a reformulação automática de modelos de otimização inteira e outros métodos de solução pode ser encontrada em [58] e [47]. Este e outros métodos de solução para problemas de otimização inteira também são apresentados em [21], [45], [41].

5.1 O problema da mochila

Vamos continuar discutindo as diversas decisões associadas à administração do consórcio de empresas do ramo madeireiro apresentado na Seção 4.1 e ver como estas decisões podem ser estudadas através de modelos de otimização para problemas clássicos.

Exemplo 5.1 *Seleção de Projetos - A companhia Ramos de Carvalho esta planejando seus investimento para o próximo ano. Existem 3 projetos prioritários, e o capital disponível é de 10 milhões de u.m.. O investimento necessário e retorno associado a cada projeto esta descrito na Tabela 5.1 abaixo. Não é possível fazer investimento parcial em um dado projeto. Em que projetos deve ser investido o capital disponível?*

Projeto	P1	P2	P3
Investimento	3	5	4
retorno	40	10	10

Tabela 5.1: Investimento e Retorno (em milhões de u.m.)

O enunciado deste problema é muito simples e podemos definir rapidamente qual é a decisão a ser tomada. Considerando que o investimento não pode ser parcial, a decisão a ser tomada é se investimos ou não em um determinado projeto. Decisões do tipo sim ou não são facilmente modeladas se introduzirmos variáveis binárias no modelo. Isto é, variáveis que podem assumir apenas dois valores: 0 ou 1. Definindo:

Índice

$j = 1, 2, 3$ para representar respectivamente os projetos P1, P2 e P3;

podemos representar a decisão a ser tomada como:

Variável de decisão

$$x_j = \begin{cases} 1, \text{ se o projeto } j \text{ for selecionado} \\ 0, \text{ caso contrário;} \end{cases}$$

neste caso, deixamos de lado o axioma de *divisibilidade* usado nos capítulos anteriores para construir modelos de otimização linear. O modelo que iremos construir será um modelo de otimização linear inteira, mais especificamente um modelo de otimização $0/1$, ou otimização binária. Os demais axiomas *proporcionalidade* e *aditividade* serão usados para obtermos modelos lineares.

O critério para a tomada de decisão é maximizar o retorno total obtido com o investimento. Usando o axioma da *aditividade* e os dados da Tabela 5.1 podemos construir a seguinte função objetivo:

Função Objetivo

max z= $40x_1 + 10x_2 + 10x_3$.

Naturalmente, se investirmos em todos os projetos teremos um lucro maior que se investirmos apenas em alguns. O capital disponível para os investimentos não é suficiente para investir em todos os projetos. Temos uma única restrição de capacidade, o valor total investido deve ser menor ou igual ao capital disponível. Obtemos assim o seguinte modelo de otimização binário para o problema de seleção de projetos:

$$\begin{aligned} \max \quad & z = 40x_1 + 10x_2 + 10x_3 \\ \text{sujeito a} \quad & \\ & 3x_1 + 5x_2 + 4x_3 \leq 10 \qquad (5.\text{-}2) \\ & x_j = 0 \text{ ou } 1 \end{aligned}$$

Este exemplar pode ser facilmente resolvido se avaliarmos o valor de z para cada uma dos oito possíveis valores para o vetor x. No entanto, se aumentarmos a dimensão do exemplar e considerarmos que existem n projetos , teremos 2^n possíveis soluções. Este número cresce exponencialmente quando o valor de$n$ cresce, e portanto métodos mais eficientes são necessários para resolvê-lo.

O modelo usado para representar a situação descrita no Exemplo 5.1 é também utilizado para representar um problema de otimização combinatória conhecido como O Problema da Mochila (*Knapsack Problem*). O enunciado clássico supõe que existem n items, cada um com peso, a_j, e valor, p_j, conhecidos. Deseja-se selecionar um subconjunto destes items para colocar em uma mochila que suporta um peso máximo, C, de forma que a soma dos valores dos items selecionados seja a maior

possível. O modelo matemático que descreve este problema e que generaliza o Modelo (5.0) é:

$$\begin{aligned} \max \quad & z = \sum_{j=1}^{n} p_j x_j \\ \text{sujeito a} \quad & \\ & \sum_{j=1}^{n} a_j x_j \leq C \\ & x_j = 0 \text{ ou } 1. \end{aligned} \tag{5.-4}$$

Em (5.-2) consideramos que não existem dois items com o mesmo par de valor e peso. Se existem vários items do mesmo tipo, temos a decisão de quantos de cada tipo devem ser incluídos na mochila. A variável de decisão passa a ser $x_j \in \mathbb{Z}$ e obtemos o Problema da mochila Inteiro.

Este problema possui um relacionamento muito grande com diversos outros problemas de otimização inteira . Sua importância se deve fortemente ao fato que além de modelar diversas situações, aparece como subproblema na formulação e solução de diversos outros problemas, por exemplo problemas de corte e empacotamento [3], e na geração de planos de corte (inequações de cobertura [58] e [45]). Sua estrutura especial permite que diversas classes de algoritmos sejam explorados na busca de um processo eficiente para resolvê-lo. Maiores detalhes sobre a modelagem e solução do problema da mochila e problemas relacionados podem ser encontrados em [33] e [21] entre outros.

5.2 O Problema do Caixeiro Viajante

O problema do Caixeiro Viajante é talvez um dos problemas de otimização combinatória mais estudados na literatura. Possui um enunciado muito simples, mas, um processo de solução complexo. Vejamos como este problema pode estar relacionado às decisões do consórcio de empresas que estamos estudando.

Exemplo 5.2 *O Presidente, Antônio Castor, da Companhia Ramos de Carvalho quer fazer uma visita às reservas florestais situadas nos estados do Amazonas e Pará, aos depósitos situados nos estados de São Paulo, Bahia, Minas Gerais e Rio de Janeiro. É possível determinar um roteiro de viagem tal que cada reserva e cada depósito sejam visitados exatamente uma vez, saindo e retornando à sede da empresa no Rio de Janeiro, e que minimize a distância total percorrida?*

A situação do Exemplo 5.2 pode ser representada através de um grafo valorado [50] $G(V, A)$ onde o conjunto de vértices, V, representa as capitais dos estados onde estão situadas as reservas e os depósitos, e o conjunto de arestas, A, representa a possibilidade de se viajar de uma cidade a outra (ver Figura5.1). O peso de uma aresta entre duas cidades é igual ao custo da viagem entre elas. Em termos de

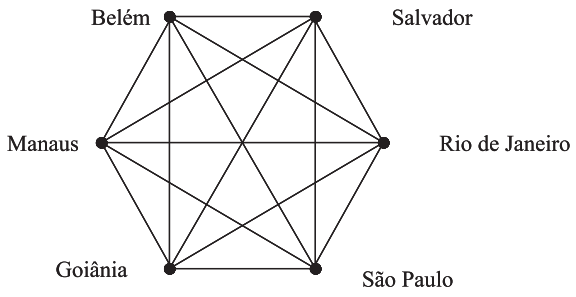

Figura 5.1: Cidades a serem visitadas

Teoria dos Grafos, a solução deste problema consiste em determinar o circuito[1] que inclua todos os vértices do grafo, um circuito hamiltoniano, de menor custo. O custo de um circuito é igual à soma dos pesos das aresta incluídas. Para modelar o problema vamos definir:

Índice

$i, j = 1, \ldots, 6$ para representar as cidades Rio de Janeiro , São Paulo, Goiânia, Manaus, Belém e Salvador respectivamente.

Note que os índices i e j representam o mesmo conjunto de cidades. Vamos usar uma variável binária para definir se a cidade i precede ou não a cidade j no roteiro. Isto é:

Variável de decisão

$$x_{ij} = \begin{cases} 1, \text{ se a cidade } i \text{ é visitada antes da cidade } j \\ 0, \text{ caso contrário;} \end{cases}$$

A variável x_{ii} não tem nenhum significado nesta aplicação, portanto só há variáveis quando $i \neq j$. O critério para a tomada de decisões está claramente definido, obter o circuito hamiltoniano de menor custo. Seja c_{ij} o custo da viagem entre a cidade i e a cidade j. A função objetivo pode então ser definida como:

Função Objetivo

$$\min \quad z = \sum_{i=1}^{6} \sum_{j=i}^{6} c_{ij} x_{ij}.$$

[1] Um circuito é uma seqüencia de vértices e arestas onde não há repetição de vértices, exceto pelo primeiro.

A restrição principal que temos é que cada cidade deve ser incluída exatamente uma vez no circuito. Vamos considerar, por exemplo, a cidade de São Paulo. Podemos chegar até ela vindo de qualquer uma das outras cinco cidades. Se fizermos$x_{12} = 1$ incluímos São Paulo imediatamente após o Rio de Janeiro no roteiro. Como cada cidade só pode aparecer uma vez no roteiro as demais variáveis que representam a chegada a São Paulo, x_{i2}, devem assumir valor zero. Como as variáveis de decisão são binárias, esta situação ($x_{12} = 1$ ou $x_{32} = 1$ ou $\ldots$ $x_{62} = 1$) pode ser representada através da seguinte equação:

$$x_{12} + x_{32} + x_{42} + x_{52} + x_{62} = 1. \tag{5.-5}$$

Pela Restrição (5.2) exatamente uma das variáveis $x_{i2} (i = 1 \ldots 6, i \neq 2)$ terá valor igual a 1, garantindo assim a chegada a São Paulo apenas uma vez. Naturalmente, saindo de São Paulo queremos ir até uma das outras cidades. Da mesma forma que chegamos apenas uma vez em São Paulo, iremos partir de lá uma única vez. Temos então que a equação:

$$x_{21} + x_{23} + x_{24} + x_{25} + x_{26} = 1$$

garante que sairemos de São Paulo uma única vez. O mesmo raciocínio vale para as demais cidades, assim temos o seguinte conjunto de restrições:

Restrições

Saída da cidade i:

$$x_{i1} + \ldots + x_{i6} = 1, \qquad i = 1 \ldots 6 \tag{5.-5}$$

e Chegada à cidade j:

$$x_{1j} + \ldots + x_{6j} = 1, \qquad j = 1 \ldots 6. \tag{5.-5}$$

Resumindo o modelo construído até agora temos:

$$\begin{aligned} \min z &= \sum_{i=1}^{6} \sum_{j=1}^{6} c_{ij} x_{ij} \\ \text{sujeito a} & \\ x_{i1} + \ldots + x_{i6} &= 1, \qquad i = 1 \ldots 6 \\ & \\ x_{1j} + \ldots + x_{6j} &= 1, \qquad j = 1 \ldots 6. \\ x_{ij} &= 0 \text{ ou } 1; \quad i, j = 1 \ldots 6, i \neq j. \end{aligned} \tag{5.-7}$$

As Restrições (5.2) e (5.2) são as restrições associadas ao problema da Designação (ver por exemplo [21]). A matriz associada possui estrutura e propriedades que permitem:

- que a restrição $x_{ij} = 0$ ou 1 possam ser relaxadas para $0 \leq x_{ij}$ e a solução ótima ser inteira.
- desenvolvimento de algoritmos especiais (e.g. algoritmo húngaro [21], [6]).

Vamos verificar então se este modelo também é apropriado para o Exemplo 5.2. Na Figura 5.2 temos a Modelo (5.-4) escrito na sintaxe do MPL. Observe o uso dos comandos EXCELRANGE e EXPORT TO EXCELRANGE usados para ler os dados e enviar a solução para o arquivo da planilha de cálculo EXCEL "caixeiro.xls". Note também a criação da variável Custo_Total para permitir que o valor ótimo da solução pudesse ser enviado junto com a solução para o arquivo do EXCEL.

```
{Problema do caixeiro viajante}
TITLE Roteiro;
INDEX
!cidades a serem incluídas no roteiro
  no =(Rio, SP, Go, Ma, Be, Sal);
   origem = no;
   destino = no;
DATA
!Leitura do custo da viagem entre as cidades na planilha caixeiro.xls
  custo[origem, destino] := EXCELRANGE ("caixeiro", "custo");
VARIABLES
! Solução do problema enviada para a planilha caixeiro.xls
   x[origem, destino]
      EXPORT TO EXCELRANGE("caixeiro","roteiro");
   Custo_Total
      EXPORT TO EXCELRANGE("caixeiro","Custo_Total");
MODEL
   MIN Custo_Total ;
SUBJECT TO
      Custo_Total = SUM(origem, destino: custo * x);
   saida [no] : SUM(origem=no, destino: x) =1;
   chegada[no] : SUM(origem, destino=no: x) =1;
END
```

Figura 5.2: Formulação Inicial : Problema da Designação - Sintaxe MPL

Acionando o CPLEX (Seção 3.2), através do MPL, para resolver um exemplar do problema obtemos a solução mostrada na Figura 5.3. Note que todas as cidades

MPL Modeling System - Copyright (c) 1988-2001, Maximal Software, Inc.
SOLUTION RESULT
Optimal solution found
MIN Z = 170.0000
DECISION VARIABLES
VARIABLE x[origem,destino] :

origem	destino	Activity
Rio	SP	1.0000
SP	Go	1.0000
Go	Rio	1.0000
Ma	Be	1.0000
Be	Sal	1.0000
Sal	Ma	1.0000

PLAIN VARIABLES

	Variable Name	Activity
	Custo_Total	170.0000

Figura 5.3: Solução do problema da designação - Modelo (5.-4) Formato MPL

foram incluídas no roteiro, mas não passamos por todas as cidades uma única vez pois o roteiro é sair do Rio de Janeiro ir para São Paulo e depois Goiânia. A próxima cidade é novamente o Rio de Janeiro (como queríamos, mas as demais cidades não foram visitadas ainda). O fato do modelo da designação permitir soluções que conduzem a sub-rotas (veja a Figura 5.4) mostra que ele não é apropriado para representar o problema que queremos resolver. Precisamos acrescentar restrições ao problema que tornem as soluções associadas a sub-rotas inviáveis. Existem na literatura diversas propostas de restrições para a eliminação de subrotas. Vamos apresentar a seguir duas delas.

5.2.1 Formulação de Miller, Tucker e Zemlin

A formulação proposta em 1960 por Miller, Tucker e Zemlin (vide e.g. [29] e [21]) resulta em um modelo de otimização inteira mista. Considere um novo conjunto de variáveis contínuas, isto é:

Novas variáveis

$u_j, j = 2 \dots 6$

A inclusão do seguinte conjunto de restrições ao Modelo 5.-4 elimina a possibilidade de sub-rotas:

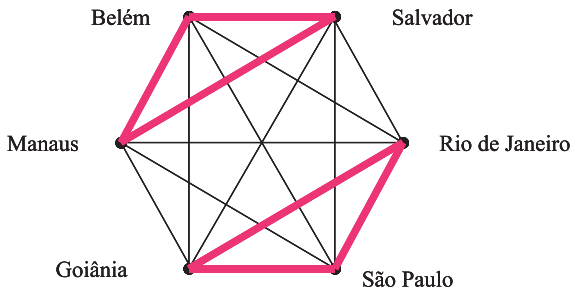

Figura 5.4: Solução do modelo da designação 5.-4: Subrotas

Novas Restrições

$$u_i - u_j + 6x_{ij} \leq 5, \qquad i, j = 2 \ldots 6; i \neq j.$$

A demonstração de que este conjunto de restrições é suficiente para eliminar as sub-rotas pode ser visto em [29] e [24]. O novo modelo para representar o Exemplo 5.2 é dado por:

$$\begin{aligned} \min \quad & z = \sum_{i=1}^{6} \sum_{j=i}^{6} c_{ij} x_{ij} \\ \text{sujeito a} \quad & \\ & x_{i1} + \ldots + x_{i6} = 1, \qquad i = 1 \ldots 6 \\ & x_{1j} + \ldots + x_{6j} = 1, \qquad j = 1 \ldots 6 \\ & u_i - u_j + 6x_{ij} \leq 5, \quad i, j = 2 \ldots 6; i \neq j \\ & x_{ij} = 0 \text{ ou } 1; \quad i, j = 1 \ldots 6, i \neq j \\ & u_i \geq 0; \quad i = 2 \ldots 6. \end{aligned} \tag{5.-11}$$

A modelagem de 5.-8 na sintaxe do MPL vai exigir novos recursos. Veja na Figura 5.5 o novo modelo. Observe que agora é necessário definir a variável$x$ como binária, pois a matriz de restrições não é mais totalmente unimodular[2].

[2]Uma matriz A é totalmente unimodular se o determinante de qualquer submatriz quadrada de A é 0, 1, ou -1 [41].

```
{Problema do caixeiro viajante} TITLE Roteiro;
INDEX
  no = EXCELRANGE ("caixeiro", "cidades");
  origem = no;
  destino = no;
DATA
  custo[origem, destino] := EXCELRANGE ("caixeiro", "custo");
  nc := EXCELRANGE ("caixeiro", "nc"); !número de cidades
VARIABLES
  x[origem , destino]
    EXPORT TO EXCELRANGE("caixeiro","roteiro");
  Custo_Total
    EXPORT TO EXCELRANGE("caixeiro","Custo_Total");
    u[no > Rio];
MODEL
  MIN Custo_Total ;

SUBJECT TO
    Custo_Total = SUM(origem, destino: custo * x);
  saida [no] : SUM(origem=no, destino: x) =1;
  chegada[no] : SUM(origem, destino=no: x) =1;
!restrições para eliminação de sub-rotas
 sub[origem,destino > Rio] where (origem <> destino):
        u[no:=origem] - u[no:=destino] + nc*x[origem,destino]<= nc-1;
BINARY
        x
BOUNDS
        u >=2
END
```

Figura 5.5: Caixeiro Viajante : Formulação de Miller, Tucker e Zemlin - Sintaxe MPL

Todos os dados necessários, incluindo o número de cidades, são lidos na planilha do EXCEL "caixeiro.xls". A variável u_j será gerada apenas para os índices definidos no Modelo 5.-8. Da mesma forma, o comando WHERE gera o conjunto de restrições de sub-rotas apenas quando $i \neq j$. Acionando mais uma vez o CPLEX para resolver o exemplar obtemos a solução mostrada na Figura 5.6. Na seção MODEL STATISTICS, além do número de restrições e variáveis do exemplar, o relatório do MPL fornece também informações sobre o processo de solução do problema. Para resolver este exemplar foi necessário examinar 12 nós na árvore de enumeração implícita associada ao algoritmo *branch and bound* (ver Seção 3.2). É interessante observar que as variáveis u_j apesar de definidas como contínuas receberam valores

inteiros. O limite inferior igual a dois definido para elas na seção BOUNDS permite interpretar seu valor como sendo a ordem em que a cidade j é visitada, isto é: São Paulo, Goiânia, Manaus, Belém e Salvador. É mera coincidência esta ordem ser a mesma em que os índices foram definidos.

MPL Modeling System - Copyright (c) 1988-2001, Maximal Software, Inc.
MODEL STATISTICS
Solver: CPLEX

Iterations: 98	Integer nodes: 12	
Constraints: 38	Variables: 42	Integers:36
Nonzeros: 179	Density: 11%	

SOLUTION RESULT
Optimal integer solution found
MIN Z = 200.0000
DECISION VARIABLES
VARIABLE x[origem,destino] :

origem	destino	Activity
Rio	SP	1.0000
SP	Go	1.0000
Go	Ma	1.0000
Ma	Be	1.0000
Be	Sal	1.0000
Sal	Rio	1.0000

VARIABLE u[no=SP..Sal] :

no	Activity
SP	2.0000
Go	3.0000
Ma	4.0000
Be	5.0000
Sal	6.0000

Figura 5.6: Solução do modelo de de Miller, Tucker e Zemlin - Formato MPL

O modelo mostrado na Figura 5.5 representa tanto o Exemplar (5.-8) como um problema mais geral (5.-13) que pode ser enunciado como: encontre o roteiro de menor custo que passe por n cidades exatamente uma vez e retorne à cidade de

origem.

$$
\begin{aligned}
\min \quad & z = \sum_{i=1}^{n} \sum_{j=i}^{6} c_{ij} x_{ij} \\
\text{sujeito a} & \\
& \sum_{j=1}^{n} x_{ij} = 1, \qquad\qquad i = 1 \ldots n \\
& \sum_{i=1}^{6} x_{ij} = 1, \qquad\qquad j = 1 \ldots n \qquad (5.\text{-}16) \\
& u_i - u_j + n x_{ij} \leq n - 1, \quad i, j = 2 \ldots n; i \neq j \\
& x_{ij} = 0 \text{ ou } 1; \quad i, j = 1 \ldots n, i \neq j \\
& u_i \geq 0; \quad i = 2 \ldots n.
\end{aligned}
$$

Classicamente conhecido como "O Problema do Caixeiro Viajante", este problema possui diversas aplicações no nosso dia-a-dia. É muito comum que empresas de diversos setores, por exemplo de bebidas, se deparem com a necessidade de determinar o melhor roteiro para a entrega da mercadoria em pontos de revenda (bares, armazéns, supermercados). Naturalmente, é mais econômico um roteiro que inclua cada ponto de revenda uma única vez. A coleta de leite em sítios e fazendas é um outro exemplo onde é desejável encontrar um roteiro que saia da fábrica de laticínios, passe em cada ponto de coleta apenas uma vez e retorne ao ponto de partida. Mas, não é apenas no setor de distribuição que temos exemplos de aplicação deste importante problema de Otimização Combinatória. Aplicações mais sutis podem ser encontradas na indústria de manufatura onde a seqüência em que um determinado conjunto de tarefas é realizada pode contribuir para diminuir os custos da produção [29]. Indo um pouco mais além, podemos encontrar exemplos na biologia computacional, onde definir um circuito hamiltoniano esta associado ao problema de desvendar os mistérios do DNA [23].

O Modelo (5.-13) possui $(n-1)^2$ restrições para eliminação de sub-rotas. Esta é uma formulação bastante compacta e pode ser usada para modelar situações onde a cidade de origem pode ser visitada diversas vezes e nenhuma sub-rota pode conter mais do que p cidades [21]. Uma formulação contendo um número maior de restrições, porém com uma relaxação linear melhor será apresentada na próxima seção.

5.2.2 Formulação de Dantzig, Fulkerson e Johnson

Dantzig, Fulkerson e Johnson (1954)(vide e.g. [29] e [21]) propuseram duas maneiras de eliminar as sub-rotas permitidas no Modelo da designação (5.-4). Uma delas consiste em limitar o número de variáveis associadas a um subconjunto de cidades S que pode receber valor diferente de zero. Considerando o roteiro mostrado na Figura 5.4 e o subconjunto $S = \{Ma, Be, Sal\}$, se impormos a restrição:

$$x_{Ma,Be} + x_{Ma,Sal} + x_{Be,Ma} + x_{Be,Sal} + x_{Sal,Ma} + x_{Sal,Be} \leq 2.$$

a solução mostrada na Figura 5.3 deixará de ser viável. De fato, acrescentar esta única restrição ao exemplar 5.-4 é suficiente para obter a solução ótima do problema. Veja este novo modelo escrito na sintaxe do XPRESS-MOSEL (Seção 3.1.2) na Figura 5.7 a seguir.

```
(! Problema do Caixeiro Viajante: Formulação de Dantzig,
Fulkerson e Johnson (1954)!)
MODEL "Roteiro"
USES "mmxprs"
DECLARATIONS
  no : set of string;
  custo: array(no,no) of integer
END-DECLARATIONS
! Leitura de dados em arquivo
INITIALIZATIONS FROM 'custo.dat'
  custo;
END-INITIALIZATIONS
DECLARATIONS
  x: ARRAY(no,no) of MPVAR ! 1 se a cidade i antecede a viagem j
END-DECLARATIONS
! Objetivo: encontrar roteiro de menor custo
Custo_Total:= SUM(i,j in no | i<>j) custo(i,j)*x(i,j)
! Visitar cada cidade exatamente uma vez
FORALL (i in no) chegada(i):= SUM (j IN no | i<>j) x(i,j) = 1
FORALL (j in no) saida(j):= SUM (i IN no | i<>j) x(i,j) = 1
FORALL (i,j IN no | i<>j) x(i,j) IS_BINARY
!Gerar inequações para eliminar subrotas
DECLARATIONS
  subrota= {"Ma", "Be", "Sal"}
END-DECLARATIONS
  sb(1):= SUM (i,j in subrota | i<>j) x(i,j) <= 3-1
! Resolva o problema
  minimize(Custo_Total)
!Relatório da Solução
writeln ("Custo_Total= ", getobjval)
FORALL(i in no) do
  FORALL (j in no) do
   if getsol(x(i,j)) > 0
    THEN writeln ('viaje de: ', i, ' para: ', j, ' com custo: ',custo(i,j))
   END-IF
  END-DO
END-DO
!gera o mps
exportprob(EP_MPS,"caix.mps",Custo_Total)
END-MODEL
```

Figura 5.7: Caixeiro Viajante - Formulação de Dantzig, Fulkerson e Johnson - Sintaxe XPRESS-MOSEL

No caso geral o modelo é dado por:

$$\begin{aligned}
\min \quad & z = \sum_{i=1}^{n} \sum_{j=i}^{6} c_{ij} x_{ij} \\
\text{sujeito a} & \\
& \sum_{j=1}^{n} x_{ij} = 1, && i = 1 \ldots n \\
& \sum_{i=1}^{6} x_{ij} = 1, && j = 1 \ldots n \\
& \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, && \vee S \subset \{1 \ldots n\} \\
& x_{ij} = 0 \text{ ou } 1; \quad i, j = 1 \ldots n, i \neq j.
\end{aligned} \tag{5.-21}$$

A formulação (5.-18) possui $(2^n - 2)$ restrições de eliminação de sub-rotas e $2n$ restrições de designação. Este número de restrições cresce exponencialmente quando o valor de n cresce. Por isso, as inequações de sub-rotas são geradas a medida que vão sendo necessárias. Inicialmente resolve-se a relaxação do Problema (5.-18) obtida eliminando-se a restrição de sub-rotas e as restrições de integralidade. A solução do problema da designação resultante é então analisada para ver se forma um circuito hamiltoniano. Caso forme, esta é a solução ótima do problema. Senão uma restrição de sub-rota não satisfeita pela solução atual é incluída e o novo problema é resolvido. Esta técnica aplicada iterativamente e combinada com método *branch and bound*, método conhecido hoje como *branch and cut* (ver Seção 3.2), foi proposta em 1954 por Dantzig, Fulkerson e Johnson para resolver com sucesso um problema com 49 cidades. Mais recentemente, Padberg e Rinaldi (1991) [42] usaram esta técnica combinada com métodos de pré-processamento e outras classes de inequações para resolver exemplares com até 2392 cidades. Em um artigo publicado em 2003, Applegate et al. [1] relatam a solução de exemplares com um milhão de cidades ou mais.

Boa parte do sucesso na resolução de exemplares de grande porte do problema do caixeiro viajante se deve à integração de diversas metodologias. De fato, um dos objetivos dos autores em [1] foi explorar a eficiência e os limites destas técnicas. As linguagens de modelagem XPRESS-MOSEL e AMPL possuem recursos que permitem implementar o método *cut and branch*: inclusão iterativa das restrições de sub-rotas (planos de corte) para obter bons limites duais, seguido do método *branch and bound* (ver Seção 3.2 e referências citadas).

5.3 Dimensionamento de lotes com tempos de preparo

Vamos explorar um pouco mais o modelo de planejamento multiperíodo apresentado na Seção 4.1 discutindo uma aplicação na indústria de bebidas. O modelo

apresentado a seguir é baseado no modelo clássico de dimensionamento de lotes e foi usado por Rangel e Ferreira [46] para estudar o setor de produção de uma fábrica de refrigerantes de médio porte situada na região de São José do Rio Preto-SP. A empresa é organizada em três setores: produção, armazenamento e transporte. Considerando a complexidade de otimizar simultaneamente as operações dos três setores, foi abordado inicialmente o setor de produção, levando em consideração a existência de um espaço limitado para armazenamento dos produtos.

5.3.1 A linha de produção de uma Fábrica de Refrigerantes

A empresa produz refrigerantes em 12 sabores que são embalados em diferentes tipos de vasilhames. São utilizadas garrafas de plástico descartáveis e garrafas de vidro que precisam ser esterilizadas antes de receberem o líquido. Existem no total 10 tipos diferentes de tamanhos entre as embalagens de plástico e as de vidro. Os insumos necessários para a produção são: xarope de diversos sabores, vasilhames descartáveis e vasilhames recicláveis, rótulos variados, tampas e água gaseificada.

A produção dos refrigerantes é realizada em três etapas: preparo do líquido, ajuste das máquinas e finalização do produto. Os xaropes são preparados em quantidades pré determinadas (1310 litros), chamadas de*tachadas*, com capacidade para produzir 7860 litros de refrigerante. O xarope utilizado na produção dos refrigerantes é armazenado em tanques que são ligados através de mangueiras especiais às linhas de produção. A fábrica possui três tanques pequenos e quatro tanques grandes. Os tanques devem trabalhar com uma quantidade mínima de líquido suficiente para cobrir a hélice que mistura os ingredientes e assim garantir a homogeneidade do xarope. Na linha de produção, um proporcionador mistura o xarope composto com água tratada. Esta mistura recebe gás carbônico e se torna a bebida que vai para a máquina que enche os vasilhames.

As máquinas são ajustadas inicialmente para produzir refrigerantes de um determinado sabor em um determinado tamanho. Se for necessário fabricar outro tipo de refrigerante é preciso parar a linha de produção e fazer ajustes nas máquinas para produzir um item de outro sabor e/ou tamanho.

Toda a produção da fábrica é realizada por três linhas de produção. Uma 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 com o líquido, fechar, rotular, codificar e empacotar os refrigerantes. Ao final do processo, os pacotes de refrigerantes são colocados nos paletes e estocados. Existe apenas uma entrada e uma saída de vasilhames. Um esquema do setor de produção da fábrica está ilustrado na Figura 5.8.

O gargalo de uma linha de produção é a máquina que enche os vasilhames (enchedora), pois é a sua capacidade que determina a produção. A máquina enchedora possui várias válvulas, o que possibilita o enchimento de mais de uma garrafa quase que simultaneamente. A enchedora das linhas de produção 1 e 2 possuem 42 válvulas e da linha 3 possui 64 válvulas. A parte pós-enchedora (fechar, rotular, codificar, empacotar) deve ter capacidade maior que a da enchedora, assim como a parte pré-enchedora (lavagem, preparo do líquido), pois se a enchedora estiver trabalhando

bem o restante da linha de produção deve acompanhá-la.

Os pedidos de refrigerantes são recebidos diariamente. O gerente da produção anota os pedidos do dia anterior no início do 1º turno de cada dia, verifica a parcela do pedido que pode ser atendida usando produtos em estoque e determina o que será produzido. O espaço para o armazenamento dos refrigerantes é restrito.

O problema é determinar quanto produzir em cada dia de forma a satisfazer a demanda e minimizar os custos de produção e armazenamento.

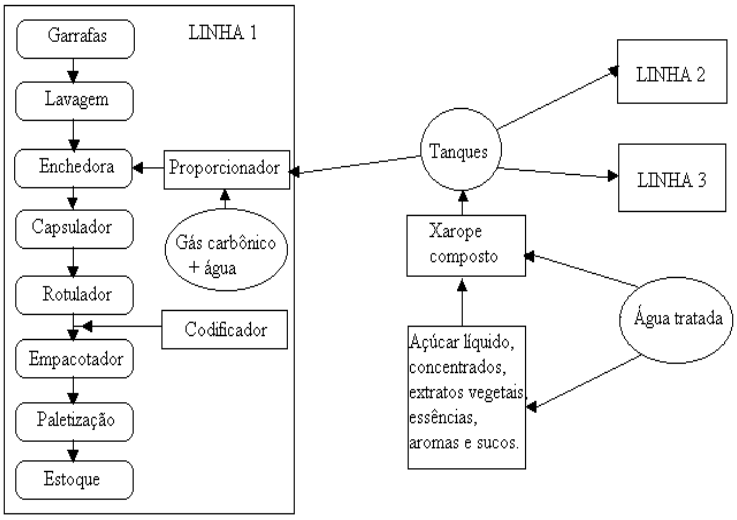

Figura 5.8: Esquema do setor de Produção da Fábrica de Refrigerantes [46]

5.3.2 Um modelo de otimização inteira mista

Para construir o modelo, vamos simplificar mais uma vez o problema e considerar o planejamento para uma única linha de produção. 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.

Pela descrição do problema na Seção 5.3.1 podemos identificar:

Elementos conhecidos

- tipo de sabores e tamanhos de vasilhames;
- horizonte de planejamento;
- número necessário de litros de xarope para a produção de uma caixa (pacote) de refrigerantes;

- tempo de produção de uma caixa de refrigerantes;
- tempo de preparação da máquina: troca de sabor;
- tempo de preparação da máquina: troca de tamanho;
- tempo disponível para produção em cada período;
- custo de produção de uma caixa de refrigerante;
- custo de armazenamento de uma caixa de refrigerante;
- custo de preparação da máquina: troca de sabor;
- custo de preparação da máquina: troca de tamanho;
- previsão de demanda de cada tipo (sabor, tamanho) de refrigerante em cada período;
- capacidade de produção em número de *tachadas*;
- quantidade mínima (máxima) de xarope no tanque;
- capacidade do estoque;
- espaço ocupado por uma caixa de refrigerante no estoque;

Elementos desconhecidos

- número de caixas de refrigerante de cada tipo (sabor e tamanho) a serem produzidas em cada período;
- número de *tachadas* de xarope de cada sabor usados em cada período;
- número de caixas de refrigerantes de cada sabor e tamanho em estoque ao final de cada período;

Objetivo a ser alcançado

- minimizar o custo total de produção, preparo e armazenamento.

Restrições

- capacidade de produção: tempo e insumos disponíveis
- capacidade do estoque
- homogeneidade do xarope

Considerando que a fábrica produz s sabores de refrigerantes em q tamanhos e que o planejamento será feito para n períodos, podemos definir o seguinte conjunto de índices:

Índices

- $i = 1, ..., s$ representa o sabor do refrigerante;
- $j = 1, ..., q$ representa o tamanho da embalagem;
- $t = 1, ..., n$ representa os períodos considerados no horizonte de planejamento.

Vamos considerar inicialmente o seguinte conjunto de variáveis:

Variáveis de decisão

- x_{ijt} = número de caixas de refrigerante do sabor i, de tamanho j, produzidas no período t;
- y_{it} = número de *tachadas* de xarope do sabor i usado no período t;
- e_{ijt} = número de caixas de refrigerantes do sabor i, de tamanho j, em estoque ao final do período t.

Para descrever a função-objetivo vamos desconsiderar no momento os custos de preparo:

função-objetivo

- Custo de produção

$$\sum_{i=1}^{s}\sum_{j=1}^{q}\sum_{t=1}^{n} c_{ijt}^{p} x_{ijt}$$

- Custo de estoque

$$\sum_{i=1}^{s}\sum_{j=1}^{q}\sum_{t=1}^{n} c_{ijt}^{a} x_{ijt}$$

- Minimizar custo de produção e armazenamento

$$\min Custo\_Total = \sum_{i=1}^{s}\sum_{j=1}^{q}\sum_{t=1}^{n} (c_{ijt}^{p} x_{ijt} + c_{ijt}^{a} x_{ijt}) \qquad (5.\text{-}23)$$

onde c_{ijt}^{p} e c_{ijt}^{a} são respectivamente o custo de produção e o custo de estoque.

Podemos iniciar a descrição do conjunto de restrições. A demanda no período$t$ pode ser atendida por refrigerantes produzidos neste período ou por produtos em estoque. O que sobrar poderá ser usado no período seguinte. Esta é exatamente a situação que modelamos na Seção 4.1.1 através das restrições de balanceamento. Assim, considerando que d_{ijt} é a previsão de demanda de refrigerantes de cada tipo (sabor i, tamanho j) no período t, temos um primeiro conjunto de restrições:

Restrições de balanceamento

$$e_{ijt-1} + x_{ijt} = d_{ijt} + e_{ijt}, \qquad i = 1, ..., s; j = 1, ..., q;\ t = 1, ..., n. \quad (5.\text{-}23)$$

Precisamos considerar também a disponibilidade de xarope e o o espaço disponível para armazenamento. Três novos conjuntos de restrições são descritos a seguir.

Restrições de espaço para armazenamento de produtos

$$\sum_{i=1}^{s}\sum_{j=1}^{q} a_j e_{ijt} \leq L, \quad t = 1 \ldots n \quad (5.\text{-}23)$$

onde a_j é o espaço ocupado por uma caixa de refrigerantes do tamanho j e L é a capacidade do armazém.

A disponibilidade de xarope será tratada através de dois conjuntos de restrições. Inicialmente precisamos considerar a capacidade de produção (c_t) em termos do número de *tachadas* de xarope que a linha de produção consegue engarrafar em cada período. Esta restrição irá ser usada para definir o número de *tachadas* (y_{it}) de cada sabor de xarope que deverão ser preparados em cada período. Naturalmente, não poderá ser usado mais do que $1310y_{it}$ litros de xarope.

Restrições de disponibilidade de xarope

- Capacidade de produção

$$\sum_{i=1}^{s} y_{it} \leq c_t, \quad t = 1 \ldots n. \quad (5.\text{-}23)$$

- Quantidade de xarope usada na produção

$$\sum_{j=1}^{q} b_{ij} x_{ijt} = 1310 y_{it}, \quad i = 1 \ldots s; t = 1 \ldots n \quad (5.\text{-}23)$$

onde b_{ij} é a quantidade de xarope do sabor i para a produção do tamanho j.

Precisamos considerar ainda o tempo disponível para a produção. Seja k_j o tempo necessário para produzir uma caixa de refrigerante do tamanho j. O tempo total para a produção dos refrigerantes no período t deve ser menor ou igual ao tempo disponível para a produção. Temos então a seguinte restrição de capacidade.

Restrições de capacidade de tempo de produção

$$\sum_{\mathrm{i=1}}^{\mathrm{s}} \sum_{\mathrm{j=1}}^{\mathrm{q}} k_j x_{ijt} \leq h_t,\ t = 1, ..., n. \quad (5.\text{-}23)$$

Note no entanto, que na Restrição (5.3.2) não foram considerados os tempos para preparação da linha de produção para receber um novo tamanho e/ou sabor. Se este tempo não for considerado, a sugestão de produção obtida com o modelo formado pelas restrições apresentadas até aqui pode não ser realizável [54]. Precisamos então considerar dois novos fatores no modelo:

- só pode haver produção de refrigerante de um determinado tipo se a linha de produção estiver preparada;
- o tempo de troca restringe a capacidade de produção.

Para considerar estes elementos no modelo vamos precisar introduzir novas variáveis que irão controlar se a linha de produção esta preparada ou não para produzir refrigerantes de uma determinada combinação de sabor/tamanho. Ao modelar o problema da mochila na Seção 5.0 vimos que decisões do tipo sim ou não podem ser modeladas usando variáveis binárias. Precisamos do seguinte conjunto de variáveis:

Variáveis de troca

- Sabor:

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

- Tamanho:

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

Precisamos garantir que só pode haver produção de um refrigerante do sabor $i = 2$, $x_{2jt} > 0$ se a linha de produção estiver ajustada para tal, $z_{2t} = 1$. Caso contrário, se $z_{2t} = 0$, temos que ter $x_{2jt} = 0$. Esta condição pode ser satisfeita se incluirmos o seguinte conjunto de restrições, para $i = 1 \ldots s$; $j = 1 \ldots q$ e $t = 1 \ldots n$:

Restrições de preparo

Sabor:

$$x_{ijt} \leq M z_{it} \tag{5.-22}$$

Tamanho:

$$x_{ijt} \leq M w_{jt} \tag{5.-21}$$

A constante M representa um valor conhecido para o limite superior da produção de um dado item. Desta forma, se $z_{it} = 0$ as restrições (5.-22) impõem $x_{ijt} = 0$, garantindo que não haverá produção de refrigerante do sabor i. Quando $z_{it} = 1$ as restrições (5.-22) se tornam redundantes, $x_{ijt} \leq M$, e poderá haver produção de refrigerante do sabor i. As restrições (5.-21) controlam de forma similar a produção

de refrigerante do tamanho j. Este tipo de restrição também pode ser útil em outros contextos, e recebe o nome de restrições VUB (do inglês: *Variable Upper Bound*, limite superior variável[41]).

As variáveis z_{it} e w_{jt} são conhecidas na literatura como variáveis de preparação das máquinas (*set-up variables* [54]). Neste modelo se considera que o tempo de ajuste da máquina para um determinado sabor (tamanho) é independente do sabor (tamanho) usado anteriormente, de acordo com a prática da empresa estudada. Elas serão usadas para controlar de forma mais realista a capacidade de produção da fábrica. Considerando o_i e f_j como sendo o tempo de troca de sabor e o tempo de troca do tamanho, respectivamente, podemos reformular as restrições de capacidade (5.3.2) como:

Restrições de capacidade com tempo de produção e preparo

$$\sum_{\mathrm{i}=1}^{\mathrm{s}} \sum_{\mathrm{j}=1}^{\mathrm{q}} k_{ij}x_{ijt} + \sum_{\mathrm{i}=1}^{\mathrm{s}} o_i z_{it} + \sum_{\mathrm{j}=1}^{\mathrm{q}} f_j w_{jt} \leq h_t,\ t = 1, ..., n. \qquad (5.\text{-}21)$$

Para garantir a homogeneidade deve ser mantida uma quantidade mínima de xarope nos tanques. Isto é, se houver produção do sabor i no período t é necessário que a quantidade de xarope y_{it} seja maior ou igual que uma quantidade pré-especificada (q_{min}). Naturalmente, a capacidade do tanque, q_{max}, também deve ser respeitada. Assim se $z_{it} = 1$ devemos ter $q_{min} \leq y_{it} \leq q_{max}$; caso contrário não há necessidade de preparar o xarope i no período t. Assim temos mais um conjunto de restrições:

Restrições de homogeneidade do xarope

$$q_{min}z_{ij} \leq y_{it} \leq q_{max}z_{ij}, \qquad i = 1, ..., s;\ t = 1, ..., n.$$

Finalmente, podemos redefinir a Função-objetivo (5.3.2) para considerar também os custos de preparo. Assim temos mais dois fatores para incluir na expressão do Custo_Total:

Reformulação da função-objetivo

- Custo de troca de sabor

$$\sum_{i=1}^{s}\sum_{t=1}^{n} c_i^{sab} z_{it}$$

- Custo de troca de tamanho

$$\sum_{j=1}^{q}\sum_{t=1}^{n} c_j^{tam} w_{jt}$$

- Minimizar custo total de produção, armazenamento e preparo

$$\min Custo\_Total = \sum_{i=1}^{s}\sum_{j=1}^{q}\sum_{t=1}^{n}(c_{ijt}^{p}x_{ijt} + c_{ijt}^{a}x_{ijt} + c_{i}^{sab}z_{it} + c_{j}^{tam}z_{jt}) \quad (5.\text{-}21)$$

onde c_i^{sab} e c_j^{tam} são respectivamente os custos de troca de sabor e de tamanho.

Podemos resumir as expressões acima no seguinte modelo de otimização inteira mista cujo objetivo é definir um programa para a produção dos refrigerantes que minimize os custos de produção, armazenamento, e preparo das máquinas (FR):

$$\min \sum_{\mathrm{i=1}}^{\mathrm{s}}\sum_{\mathrm{j=1}}^{\mathrm{q}}\sum_{\mathrm{t=1}}^{\mathrm{n}}\left(\mathrm{c}_{ij}^{p}\mathrm{x}_{ijt}+\mathrm{c}_{j}^{a}\mathrm{e}_{ijt}+\mathrm{c}_{i}^{sab}z_{it}+\mathrm{c}_{j}^{tam}w_{jt}\right)$$

Sujeito a:

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

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

$$\sum_{\mathrm{i=1}}^{\mathrm{s}}\sum_{\mathrm{j=1}}^{\mathrm{q}} k_{ij}x_{ijt} + \sum_{\mathrm{i=1}}^{\mathrm{s}} o_i z_{it} + \sum_{\mathrm{j=1}}^{\mathrm{q}} f_j w_{jt} \leq h_t, \ t = 1,...,n.$$

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

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

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

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

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

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

Considerando apenas os conjuntos de restrições (5.3.2), (5.-22), (5.-21) e (5.3.2) obtemos um modelo que é conhecido na literatura como modelo de planejamento da produção multi-item capacitado com tempos e custos de preparo (PMC) (*Multi-item Production Planning with set-up* [54] e [2]).

Observe que o modelo (FR) permite a produção de refrigerantes de sabores e tamanhos diferentes em um mesmo período. Uma importante questão que surge é como fazer o sequenciamento dos itens diferentes que são produzidos em um mesmo período. No presente trabalho, estamos supondo que o planejamento será feito em dois estágios. Um estágio determina o dimensionamento dos lotes (estágio A), e outro estágio (estágio B), feito antes ou depois do estágio A, determina a

sequência em que os itens serão produzidos. O modelo (FR) é usado no estágio A. Na próxima seção apresentamos modelos para o problema de sequenciamento de itens, estágio B. Modelos de planejamento da produção que incluem o sequenciamento e o dimensionamento dos lotes simultaneamente podem ser vistos em [14], [16] e [53].

5.4 Problema do Escalonamento de Tarefas

Problemas de seqüenciamento desempenham um papel importante tanto em indústrias de manufatura como em empresas de serviço. Empresas devem entregar seus produtos ou serviços dentro de um prazo pré-determinado, ou mesmo escalonar as tarefas para usar de forma eficiente os recursos disponíveis [43]. Do ponto de vista matemático, construir modelos de otimização que representem de forma adequada o problema continua sendo um desafio. Nesta seção apresentaremos dois modelos diferentes: modelo com restrições disjuntas [47] e o modelo indexado por tempo [4]. Sendo que este último pode servir de base para a consideração de restrições de tempo de preparo das máquinas (ver Seção 5.3.1).

Vamos começar por um pequeno problema estudado por Bezerra e Rangel [7].

Exemplo 5.3 *Uma fábrica produz panelas de metal de cinco modelos diferentes em uma única máquina. Cada panela é confeccionada da seguinte maneira: primeiro são cortados elementos circulares e em seguida cada um deles é modelado na forma da panela desejada. As etapas do processo não podem ser interrompidas, ou seja, depois de iniciada a fabricação de um lote de determinado modelo de panelas, este deve ser totalmente concluído. A fábrica realiza um expediente de até 22h, devendo produzir diariamente um lote de cada modelo. Cada lote possui horários de entrega pré-definidos pelos compradores. Caso a fábrica entregue algum lote depois do prazo deverá pagar uma multa, por cada hora de atraso, ao cliente prejudicado. Os dados referentes ao tempo necessário para a produção de cada lote, horário de entrega e multas estão resumidos na Tabela 5.2 abaixo. O gerente de produção precisa definir a seqüência em que ele irá produzir os lotes de forma a minimizar a demora na entrega dos lotes e as respectivas multas.*

Tabela 5.2: Fábrica de Panelas: Tempo de produção, horário de entrega e multa

	Modelo				
	P1	P2	P3	P4	P5
Tempo de Produção (horas)	6	5	4	3	2
Horário de Entrega	10	11	15	5	5
Multa (u.m./hora)	1	2	3	6	1

Uma decisão importante na fabricação dos lotes de panelas é a escolha da seqüência em que estes devem ser produzidos. Problemas deste tipo são chamados Problemas de Sequenciamento (ou escalonamento) de tarefas [43].

5.4.1 Classificação do Problema - $(\alpha\ |\beta\ |\gamma)$

Existe uma nomenclatura própria para descrever Problemas de Sequenciamento, formada a partir do preenchimento de três campos $(\alpha\ |\beta\ |\gamma)$. O campo α representa as características de quem executa as atividades, denominado por convenção como máquina(s). O campo β descreve as atividades e como estas estão relacionadas. O último campo, γ, se refere ao tipo de objetivo utilizado para determinar a seqüência de produção.

Vamos avaliar as características do problema das panelas do Exemplo 5.3. Existe apenas uma máquina na fábrica, neste caso utiliza-se a denominação problema de máquina única ($\alpha = 1$). A máquina deve produzir cinco lotes diferentes, assim, ao utilizar a palavra atividade estaremos nos referindo a produção de um lote de um determinado modelo de panela. Além disso, sabemos que não deve haver preempção, ou seja, o processo de produção não pode ser interrompido. Também não há relação de precedência entre as atividades, nenhum dos modelos deve obrigatoriamente ser produzido antes de outro ($\beta = \emptyset$). Levando em consideração todos os dados do problema, o objetivo do gerente de produção é minimizar as multas relacionadas à demora na entrega de cada um dos lotes ($\gamma = wTmax$, onde: $Tmax$ = max {0, atraso na entrega do lote i}). Um valor estritamente negativo para $Tmax$ quer dizer que a atividade foi concluída antes da data de entrega. Segundo a nomenclatura apresentada em [37] e [43], este problema é então classificado como $1|\ |wTmax$.

5.4.2 Modelo com restrições disjuntas

Para construir o modelo de otimização vamos identificar no enunciado do Exemplo 5.3 os principais elementos do problema.

Elementos conhecidos - Dados de um exemplar

- n : número de atividades
- p_i : tempo de processamento da atividade $i, i = 1 \ldots n$
- d_i : data de entrega da atividade $i, i = 1 \ldots n$
- w_i : multa (peso) associado à atividade $i, i = 1 \ldots n$

Variáveis de decisão: tempo de início

- x_i : início do processamento da atividade $i, i = 1 \ldots n$

O gerente de produção deseja minimizar as multas relacionadas à demora na entrega de cada um dos lotes. A demora ponderada da atividade i pode ser calculada como:

Demora Ponderada

$$\max\ \{0, w_i(x_i + p_i - d_i)\}$$

e o critério de otimização é minimizar a demora. Isto é:

função-objetivo linear por partes

$$\min \max\{0, w_i(x_i + p_i - d_i)\} \tag{5.-21}$$

A função-objetivo definida acima é linear por partes, mas pode ser linearizada se criarmos uma nova variável, t_i, tal que:

Restrições para linearização da Função-objetivo

$$t_i \geq x_i + p_i - d_i, \quad t_i \geq 0, \quad i = 1 \ldots n;$$

e substituirmos a expressão 5.4.2 por:

Função-objetivo Linear

$$\min z = \sum_{i=1}^{n} w_i t_i \tag{5.-21}$$

Outros critérios de otimização podem levar a sequenciamentos equivalentes. Maiores detalhes sobre a equivalência entre critérios de otimização para problemas de sequenciamento pode ser encontrados em [43] e [7].

Precisamos garantir que a máquina produzirá apenas um lote de cada vez. Isto é, se a máquina estiver executando a atividade 1, ela só poderá iniciar uma outra atividade j quando a tiver terminado, isto é $x_1 + p_1 \leq x_j$. De uma maneira mais geral, vamos considerar as atividade i e j. Para garantir que a máquina processe apenas uma delas de cada vez devemos ter:

Restrições disjuntas

$$x_i + p_i \leq x_j \tag{5.-21}$$

ou

$$x_j + p_j \leq x_i. \tag{5.-21}$$

Isto é, apenas uma das duas restrições acima deve estar ativa, a outra deve ser redundante. Mais uma vez variáveis binárias serão úteis para fazer este controle. Vamos definir:

Variável de decisão: controle de execução

$$y_{ij} = \begin{cases} 1, \text{ se a atividade } i \text{ começa antes da atividade } j \\ 0, \text{ caso contrário;} \end{cases}$$

Se $y_{ij} = 1$ temos que a atividade i começa antes da j e portanto a restrição (5.4.2) associada deve estar ativa para garantir que a atividade j comece depois que a atividade i estiver completa, a outra restrição deve ser redundante. Similarmente se $y_{ij} = 0$ a restrição (5.4.2) deve ser satisfeita. Assim, podemos escrever:

Restrições: controle de atividades na máquina

$$x_i + p_i \leq x_j + M(1 - y_{ij}) \tag{5.-21}$$

$$x_j + p_j \leq x_i + M(y_{ij}). \tag{5.-21}$$

onde M é uma constante real de valor suficientemente grande. Note que se$y_{ij} = 1$ a restrição (5.4.2) fica ativa e a restrição (5.4.2) apenas fornece um limite superior para x_j.

Resumindo as informações obtidas até agora temos um modelo com restrições disjuntas para representar o problema de sequenciamento:

$$\begin{aligned}
\min z &= \sum_{i=1}^{n} w_i t_i \\
\text{sujeito a} & \\
t_i &\geq x_i + p_i - d_i, \\
& \qquad i = 1 \ldots n \\
x_i + p_i &\leq x_j + M(1 - y_{ij}), \\
& \qquad i = 1 \ldots n, j = 1 \ldots n \\
x_j + p_j &\leq x_i + M(y_{ij}) \\
& \qquad i = 1 \ldots n, j = 1 \ldots n \\
x_i &\geq 0, t_i \geq 0, i = 1 \ldots n \\
y_{ij} &= 0 \text{ ou } 1, i = 1 \ldots n, i = 1 \ldots n.
\end{aligned} \tag{5.-24}$$

Se a atividade j é a última a ser executada, temos que $x_i + p_i \leq x_j + p_j$ para todo $i = 1 \ldots n, i \neq j$. Portanto se $TTotal$ representa o tempo total necessário para a execução de todas as atividades:

$$TTotal = \sum_{i=1}^{n} p_i; \tag{5.-29}$$

qualquer valor maior ou igual a $TTotal$ é conveniente para a constante M. De fato a qualidade da relaxação linear associada ao Modelo (5.-20) depende do valor atribuído à M.

5.4.3 Modelo Indexado pelo tempo

Outra forma de modelar problemas de sequenciamento é utilizando o tempo em que cada atividade inicia como índice das variáveis. Esta estrutura permite construir modelos que incluam restrições adicionais ao problema, por exemplo: restrições de Tempo de Preparo das máquinas (ver Seção 5.3.2) que surgem quando é necessário preparar a máquina antes de começar a executar cada atividade.

Para construir o modelo vamos precisar calcular o tempo total necessário para o processamento de todas as tarefas. Este valor será usado para definir o tamanho

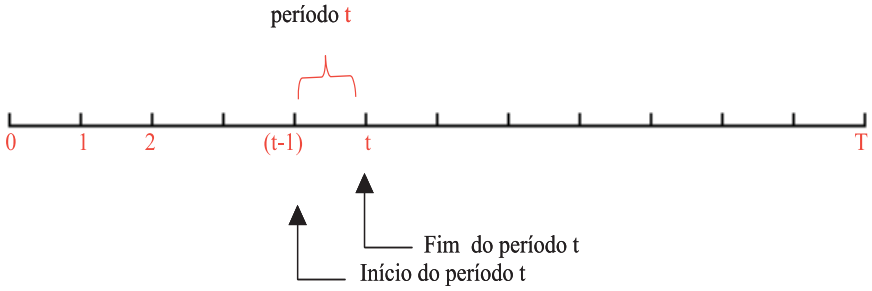

Figura 5.9: Horizonte de planejamento

do horizonte de planejamento T, isto é T períodos de tempo (ver Figura 5.9). Um período pode ser definido em horas, turno de trabalho, dias ou outra unidade de tempo de acordo com a aplicação estudada. No caso do Exemplo 5.3 o período de tempo será medido em horas. Se tomarmos (T) como sendo exatamente o valor calculado na Expressão (5.4.2) não será permitido tempo ocioso na máquina, um valor menor, não permite a execução de todas as tarefas.

Temos n atividades que podem começar em qualquer um dos períodos de 0 a T. A decisão a ser tomada então é se a atividade i começa ou não no período de tempo t. Assim temos:

Variável de decisão indexada pelo tempo

$$x_{it} = \begin{cases} 1, \text{ se a atividade i começa no período t} \\ 0, \text{ caso contrário;} \end{cases}$$

Precisamos garantir que cada atividade seja executada exatamente uma vez. Usando restrições similares às restrição de designação usadas para modelar o problema do caixeiro viajante (ver Seção 5.2) temos:

Restrições: Todas as atividades devem ser executadas

$$\sum_{t=1}^{T} x_{it} = 1 \quad i = 1 \ldots n. \tag{5.-29}$$

Mas, considere por exemplo, que a atividade 3, que tem tempo de processamento igual a 4 (ver Tabela 5.2), comece a ser executada no período $t = 18$ ($x_{3,18} = 1$). Não será possível concluí-la pois ela só terminará de ser executada no período $t = 18 + 4 - 1 = 21$, fora do horizonte de planejamento ($T = 20$). Para evitar este fato, a atividade 3 deverá começar em um tempo anterior ou igual a $T - p_3 + 1$. Assim o limite superior da soma em (5.4.3) deve ser $T - p_i + 1$, para garantir que cada uma das atividades seja concluída dentro do horizonte de planejamento.

Reformulação: Todas as atividades devem ser executadas e concluídas

$$\sum_{t=1}^{T-p_i+1} x_{it} = 1 \quad i = 1 \ldots n. \tag{5.-29}$$

Precisamos também garantir que no máximo uma atividade esteja sendo executada em cada período de tempo t:

Restrições de capacidade: No máximo uma tarefa em cada período

$$\sum_{i=1}^{n} x_{it} \leq 1, \quad t = 1 \ldots T. \tag{5.-29}$$

A solução $x_{23} = 1$ e $x_{34} = 1$, isto é a atividade 2 começa no período 3 e a atividade 3 começa no período 4, satisfazem o conjunto de restrição definidos acima (5.4.3 e 5.4.3). No entanto, de acordo com os tempos de execução dados na Tabela 5.2, no período $t = 7$ a máquina ainda estará ocupada com a atividade 2, esta atividade só estará completa no final deste período ($t = 3 + p_2 - 1 = 7$). Portanto apenas uma destas duas variáveis pode assumir valor 1. De fato, é necessário garantir que nenhuma atividade comece enquanto a máquina estiver ocupada. Por exemplo, a restrição mostrada na Figura 5.10 garante que se a atividade 2 começar no período 3, nenhuma outra atividade pode começar antes do final do período 7. Estendendo

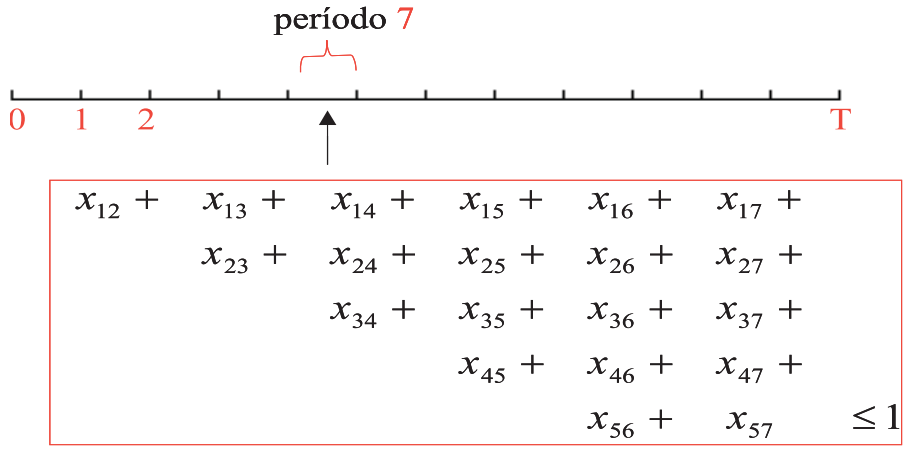

Figura 5.10: Restrição de capacidade associada ao período $t = 7$

este raciocínio para os demais períodos, as restrições (5.4.3) devem ser reformuladas como:

Restrições de capacidade: Reformulação

$$\sum_{i=1}^{n} \sum_{s=t-p_i+1}^{t} x_{it} \leq 1, \quad t = 1 \ldots T. \tag{5.-29}$$

O critério de otimização que vamos usar é minimizar os custos relativos à demora na entrega, isto é, minimizar as multas pelo atraso na entrega das atividades. Uma atividade i está atrasada se $t + p_i - 1 > d_i$, onde t é o período quando ela começa a ser executada. Assim a matriz de custos associada aos dados apresentados na Seção 5.4.2 pode ser calculada como:

Cálculo do custo de atraso

$$c_{it}^{m} = \begin{cases} w_i((t + p_i - 1) - d_i), \text{ se } t + p_i - 1 > d_i \\ 0 \text{ caso contrário;} \end{cases}$$

A função objetivo pode ser escrita como:

Função-objetivo

$$\min z = \sum_{i=1}^{n} \sum_{t=1}^{T-p_i+1} c_{it}^{m} x_{it}. \tag{5.-29}$$

Reunindo as informações acima chegamos ao modelo Indexado por Tempo para o Problema de sequenciamento:

$$\begin{aligned} \min\ z = & \sum_{i=1}^{n} \sum_{t=1}^{T-p_i+1} c_{it}^{m} x_{it} \\ \text{sujeito a} & \\ & \sum_{t=1}^{T-p_i+1} x_{it} = 1 \quad i = 1 \ldots n \\ & \sum_{i=1}^{n} \sum_{s=t-p_i+1}^{t} x_{it} \leq 1, \quad t = 1 \ldots T \\ & x_{it} = 0 \text{ ou } 1, i = 1 \ldots n, t = 1 \ldots T. \end{aligned} \tag{5.-31}$$

Para um exemplar com n atividades, esta formulação fornece um modelo com aproximadamente nT variáveis e $(n + T)$ restrições. No modelo com restrições disjuntas (5.-20) temos $(n^2 + 2n)$ variáveis e $(2n^2 + n)$ restrições. O exemplar definido pelos dados mostrados na tabela 5.2 tem 100 variáveis e 25 restrições para o modelo indexado por tempo e 35 variáveis e 55 restrições para o modelo de restrições disjuntas. Uma grande vantagem da formulação indexada por tempo (5.-28) é que a Relaxação Linear associada fornece bons limitantes para o problema inteiro, de fato os resultados de um estudo computacional comparando estes dois modelos apresentado em [7] demonstrou sua superioridade sobre o Modelo (5.-20). O alto

número de variáveis presentes no Modelo (5.-28), dependente do número de períodos considerado no horizonte de planejamento, sugere o método de geração de colunas para resolvê-lo (e.g. [4], [44]).

Devido à complexidade computacional dos Problemas de sequenciamento, os principais métodos de solução descrito na literatura são baseados em algoritmos heurísticos que fornecem soluções aproximadas para o problema (e.g. [35], [37] e [43]. A vantagem da utilização de modelos de otimização inteira é poder resolvê-los usando sistemas gerais de otimização. Estes sistemas tem alcançado um alto grau de sofisticação, o que tem permitido a solução de exemplares de grande porte em tempo viável (e.g. [4]). Além disto, estes modelos podem fornecer bons limitantes para o valor ótimo, que servem de parâmetros para avaliação das soluções heurísticas.

Bibliografia

[1] Applegate, D., Bixby, R., Chvatal, V., e Cook, W., Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems,*Mathematical Programming*, Ser. B, 97, 91-153, 2003.

[2] 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 - SOBRAPO*, 20, No. 2, 287-306, 2000.

[3] Arenales, M.N., Morabito, R. e Yanasse, H., Problemas de Corte e Empacotamento, *Anais Do XXXVI Simpósio Brasileiro De Pesquisa Operacional*, SOBRAPO, 2004.

[4] van den Akker, J. M, Hurkens, C.A.J. e Salvelsbergh, M.W.P., Time-Indexed Formulations for Machine Scheduling Problems: Column Generation,*Informs Journal on Computing*, 12(2), pg. 111-124, 2000.

[5] Atamtäurk, A. e Savelsbergh, M.W.P., Integer Programming Software Systems, Research Report BCOL.03.01, IEOR, University of California at Berkeley, 2004. A ser publicado em *Annals of Operations Research*.

[6] Bazaraa, M.J. e Jarvis, J.J., *Linear Programming and Network Flows*, J. Wiley & Sons, N.Y., 2^a edição, 1990.

[7] Bezerra, S. e Rangel S., Problemas De Sequenciamento: Modelos e Métodos de Resolução, *Anais do CIC - Congresso de Iniciaçõa Científica da UNESP*, Ilha Solteira, 2004.

[8] Boaventura, P. O., *Grafos : teoria, modelos, algoritmos*, Edgard Blucher, 2001.

[9] Caixeta-Filho, J.V., *Pesquisa Operacional: Técnicas de Otimização Aplicadas a sistemas Agroindustriais*, Editora Atlas, 2001.

[10] Campelo, R.E e Maculan,N., *Algoritmos e Heurísticas* , Editora da Universidade Federal Fluminense, 1994.

[11] Dash Optimization, *Applications Of Optimization With XpressMP*, Tradução para o inglês de Programmation Linéaire de C. Guéret, C. Prins E M. Sevaux, Dash Optimization Ltda, 2000.

[12] Dantzig, G. B., *Linear Programming and Extensions*, Princeton Universty Press, 1963.

[13] Dantzig, G. B. e Thappa,M.N., *Linear Programming 1: Introduction*, Springer, 1997.

[14] Drexl, A. e Kimms, A., Lot sizing and scheduling - survey and extensions, *European Journal of Operational Research*, 99, 1997, 221-235.

[15] Ferris, M.C, e Zhang, Y. Foreword: Special issue on mathematical programming in biology and medicine, *Mathematical Programming*, 101(2), 297-299, 2004.

[16] Ferreira, D.; Rangel, S.; Morabito, R. Um Modelo Integrado De Dimensionamento E Sequenciamento De Lotes Para A Produção De Bebidas, *Anais Do XXXVI Simpósio Brasileiro De Pesquisa Operacional*, SOBRAPO, 2335-2335, 2004.

[17] Friedlander, A., *Elementos de programação não-linear*,Editora da UNICAMP, 1994.

[18] Fourer, R., D.M. Gay, B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, Duxbury Press / Brooks/Cole Publishing Company, 2002.

[19] Fourer, R., Lopes,L., Martin, K. LPFML: A W3C XML Schema for Linear Programming, Relatório Técnico, Department of Industrial Engineering and Management Sciences, McCormick School of Engineering and Applied Science, Northwestern University, USA, 2004.

[20] Garey, M. R. e Johnson D.S., *Computer Intractability - A Guide to the Theory of NPCompleteness*, W.H. Freeman Company, 1979.

[21] Goldbarg, M.C. E Luna, H.P.L., *Otimização Combinatória e Programação Linear*, Editora Campus, 2000.

[22] Gonzaga, C. C., *Algoritmos de Pontos Interiores para Proigramação Linear*, IMPA, 1989.

[23] Gonnet G.H., Korostensky C., Benner S., Evaluation measures of multiple sequence alignments *Journal of Comupational Biology*, 7, (1-2), 261-276, 2000.

[24] Gurreiro, J., Magalhães, A. e Ramalho, M., *Programação Linear*, Vol II, Mac Graw Hill.

[25] Hadley, G., *Programação Linear*, Ed. Guanabara Dois, 1982.

[26] Hilllier, F.S. E Lieberman, G.J., *Introdução à Pesquisa Operacional*, Ed. Campus, 1988.

[27] Ilog, *ILOG CPLEX 7.1: Getting Started*, ILOG, 2001.

[28] Lachtermacher, G., *Pesquisa Operacional na Tomada de Decisões*, Ed. Campus, 2002.

[29] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. e Shmoys, D.B. (ed.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization Problems*, Wiley, 1990.

[30] Linderoth,J. T. e Ralphs, T. K., Noncommercial Software for Mixed-Integer Linear Programming December, Relatório Técnico, Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015, 2005.

[31] Luenberger, D.G., *Linear and nonlinear programming*,Addison-Wesley, 1984.

[32] Mangasarian OL, Street WN, Wolberg WH, Breast-Cancer Diagnosis And Prognosis Via Linear-Programming *Operations Research* 43 (4): 570-577 Jul-Aug 1995.

[33] Martello, S. e Toth, P., *Knapsack problems*, John Wiley, 1990.

[34] Maximal Software, *MPL Modeling System*, v. 4.2, 2002. (http://www.Maximal-Usa.Com)

[35] Morton , E. T. e Pentico, D. W., *Heuristic Scheduling Systems*, Wiley-Interscience Publication, 1993.

[36] Moody, S., *Methods and Tools for Modelling Linear and Integer Programming Problems*, Ph.D. Thesis, Department of Mathematics and Statistics, Brunel University, Uxbridge, UK, March, 141p, 1994.

[37] Muller, F. M. *Introdução aos Problemas de Sequenciamento*, Minicurso, Congresso Nacional de Matemática Aplicada e Computacional - Setembro/2002.

[38] Murphy, F.H., Annotated Bibliography on Linear Programming Models, *Interactive Transactions of ORMS*, v.1, n.4. http://catt.bus.okstate.edu/itorms/volumes/vol1/papers/murphy/ (última consulta: 25/05/2005)

[39] Murphy, F.H., Understanding Linear Programming Modeling Through an Examination of Early Papers on Model Formulations, *Operations Research*, 45, 3, 341-356, 1997.

[40] Namem, A.A.A. e Bornstein, C., Uma Ferramenta para Avaliação de Resultados de Diversos Modelos de Otimização de Dietas, *Pesquisa Operacional*, v.24, n.3, p.445-465, 2004

[41] Nemhauser, G.L. e Wolsey, L., *Integer and Combinatorial Otimization*, Wiley, 1988.

[42] Padberg, M. e Rinaldi, G., A Branch and Cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems, *SIAM Review*, 33(1), 66-100, 1991.

[43] Pinedo, M., *Scheduling - Theory, Algorithms and Systems*, Prentice Hall, 1995.

[44] Rangel, S., Estratégias de decomposição aplicadas ao problema de sequenciamento de tarefas para máquina única, *Resumos do XXVII CNMAC*, Porto Alegre, SBMAC, p. 450-450, 2004.

[45] Rangel, S., *Solving Integer Programming Problems using Preprocessing and Cutting Planes: Theory and Implementation of Branch and Cut*, Tese de doutorado, Brunel University, Uxbridge, Inglaterra, 1995.

[46] Rangel, S., Ferreira, D., Um modelo de dimensionamento de lotes para uma fábrica de refrigerantes, *Tema - Têndencias Em Matemática Aplicada e Computacional*, SBMAC, v. 4, n. 2, p.237-246, 2003.

[47] Rardin, R.L., *Optimization in Operations Research*, Prentice Hall, 1998.

[48] Saltzman, M.J., Broad Review of Software Packages available, *OR/MS Today*, ORSA/TIMS, pg. 42-51, 1994.

[49] Sharda, R. e Rampal. G., Algebraic Modeling Languages on PCs, *OR/MS Today*, 22(3), 58-63, 1995.

[50] Szwarcfiter, J.L., *Grafos e algoritmos computacionais*, Ed. Campos, 1988.

[51] Sodhi, M.S., LP modeling for asset-liability management: A survey of choices and simplifications, *Operations Research*, 53 (2), 181-196, 2005.

[52] Taube, M., Matemática Para Produtividade, *Com Ciência - Revista Eletrônica*, SBPC/LBJOR, 2002.

[53] Toledo,C.F.M., França, P. e Morabito, R., Proposta de um modelo conjunto de programação da produção e dimensionamento de lotes aplicado a uma indústria de bebidas, em *Anais do XXII ENEGEP*, Curitiba, PR, outubro, 2002.

[54] Trigeiro,W.W., Thomas, L.J. e Mcclain,J.O. , Capacitated lot sizing with setup times, *Management Science*, 35, No. 3 , 353-366, 1989.

[55] Wagner, H., *Pesquisa Operacional*, Prentice Hall do Brasil, 1986.

[56] Williams, H.P., *Model Bulding In Mathematical Programming*, John Wiley & Sons, 1990.

[57] Wright, S.J., *Primal-Dual Interior-Point Mthods*, SIAM, 1997.

[58] Wolsey, L., *Integer Programming*, Ed. John Wiley & Sons, 1998.

Índice

Notas em Matemática Aplicada

1. Restauração de Imagens com Aplicações em Biologia e Engenharia
   Geraldo Cidade, Antônio Silva Neto e Nilson Costa Roberty
2. Fundamentos, Potencialidades e Aplicações de Algoritmos Evolutivos
   Leandro dos Santos Coelho
3. Modelos Matemáticos e Métodos Numéricos em Águas Subterrâneas
   Edson Wendlander
4. Métodos Numéricos para Equações Diferenciais Parciais
   Maria Cristina de Castro Cunha e Maria Amélia Novais Schleicher
5. Modelagem em Biomatematica
   Joyce da Silva Bevilacqua, Marat Rafikov e Cláudia de Lello Courtouke Guedes
6. Métodos de Otimização Randômica: algoritmos genéticos e "simulated annealing"
   Sezimária F. Pereira Saramago
7. "Matemática Aplicada à Fisiologia e Epidemiologia"
   H.M. Yang, R. Sampaio e A. Sri Ranga

8. Uma Introdução à Computação Quântica

   Renato Portugal, Carlile Campos Lavor, Luiz Mariano Carvalho e Nelson Maculan

9. Aplicações de Análise Fatorial de Correspondências para Análise de Dados

   Dr. Homero Chaib Filho, Embrapa

10. Modelos Matemáticos baseados em autômatos celulares para Geoprocessamento

    Marilton Sanchotene de Aguiar, Fábia Amorim da Costa, Graçaliz Pereira Dimuro e Antônio Carlos da Rocha Costa

11. Computabilidade: os limites da Computação

    Regivan H. N. Santiago e Benjamín R. C. Bedregal

12. Modelagem Multiescala em Materiais e Estruturas

    Fernando Rochinha e Alexandre Madureira

13. Modelagem em Biomatemática

    1 - "Modelagem matemática do comportamento elétrico de neurônios e algumas aplicações"

    2 - "Redes complexas e aplicações nas Ciências"

    3 - "Possíveis níveis de complexidade na modelagem de sistemas biológicos"

    Coraci Malta, 1 - Reynaldo D. Pinto, 2 - José Carlos M. Mombach e 3 - Henrique L. Lenzi, Waldemiro de Souza Romanha e Marcelo Pelajo-Machado

14. A lógica na construção dos argumentos

    Angela Cruz e José Eduardo de Almeida Moura