Instituto de Matemática e Estatística - IME/USP
Universidade de São Paulo

Estudo e modelagem de soluções de otimização discreta para grades horárias

Trabalho de Conclusão do Curso de
Bacharelado em Ciência da Computação

Aluno: Nilo Cesar Teixeira
<nilo.teixeira@gmail.com>

Orientador: Prof. Alfredo Goldman

17 de fevereiro de 2012

Sumário

I Parte objetiva **3**

1 Introdução **4**
1.1 Motivação 4
1.2 Organização do texto 4
1.3 Conceitos 5
1.3.1 Álgebra Linear 5
1.3.2 Programação linear 6
1.3.3 Teoria dos Grafos 6

2 Escolha do *solver* **7**
2.1 Formato da saída glpk 9

3 Modelando um problema linear inteiro **10**
3.1 Formulações de PLI com variáveis binárias 10
3.1.1 Exemplo: Emparelhamento máximo 11

4 Modelando uma grade horária **13**
4.1 Definindo a função objetivo 13
4.2 Modelando restrições 13
4.2.1 Restrição objetiva 1 - Viabilidade de sala 14
4.2.2 Restrição objetiva 2 - Viabilidade de professor 15
4.2.3 Restrição objetiva 3 - Cotas semanais de aulas de cada matéria 16
4.2.4 Restrição objetiva 4 - Cotas diárias de aulas de cada matéria 17
4.2.5 Restrição objetiva 5 - "Convexidade" de aulas 18
4.2.6 Restrição objetiva 6 - Aulas sucessivas de uma mesma matéria ("*dobradinhas*") 19
4.3 Restrições subjetivas 20

II Parte subjetiva **21**

5 Sobre o curso **22**
5.1 Desafios do trabalho de conclusão 22
5.2 Frustrações 22
5.2.1 No divã 22
5.2.2 A retomada 23
5.3 Disciplinas relevantes e aplicação de conceitos 23
5.4 Aperfeiçoamento dos conceitos estudados e continuidade do trabalho 23

6 Conclusão **24**
6.1 Considerações finais 24
6.2 Agradecimentos 24

A Código completo da modelagem Mathprog **25**

Referências Bibliográficas **29**

Parte I

Parte objetiva

Capítulo 1

Introdução

1.1 Motivação

Muitos problemas práticos na indústria, empresas de prestação de serviços e poder público envolvem a busca pela melhor maneira de se alocar recursos operacionais ao longo de horários pré-estabelecidos.

Equipes de professores, médicos, enfermeiras, policiais, entre outros, devem coordenar a periodicidade de seus horários de trabalho, buscando alguma forma de **otimização**, que pode ser expressa como a **minimização** de custos, a **maximização** de horários com algum recurso operacional utilizado, ou alguma destas duas operações (*min/max*) sobre alguma **função objetivo** que defina o critério utilizado.

Em problemas de **otimização discreta**, cada um dos elementos x para os quais a função objetivo é definida pertencem a um **conjunto discreto** F. Ou seja, cada coordenada destes elementos geralmente é um número natural ou inteiro, já que representam quantias físicas ou enumeráveis.

Uma maneira sistemática de modelar uma ampla classe de tais problemas é expressá-los como problemas de **programação linear inteira**, e resolvê-los através de alguma biblioteca ou software (denominado "*solver*"), após descrever o problema na linguagem ou interface fornecida pelo "*solver*".

Neste trabalho, tal metodologia foi aplicada ao problema de alocação de recursos humanos de acordo com uma grade horária, problema recorrente em instituições de ensino, onde os docentes representam o recurso, e a grade é composta pelos horários de aulas.

1.2 Organização do texto

No **Capítulo 1**, apresentamos a motivação para a modelagem descrita neste trabalho, além dos conceitos matemáticos e teóricos que consideramos necessários ao entendimento do texto.

Os capítulos seguintes refletem a ordem cronológica das etapas deste trabalho: o **Capítulo 2** enumera os critérios e razões que basearam a escolha do "*solver*" para o problema proposto. O **Capítulo 3** ilustra alguns padrões de modelagem de problemas de programação linear inteira. No **Capítulo 4**, detalhamos o modelo para o problema descrito neste trabalho, apresentando também a implementação do mesmo em Mathprog, linguagem do "*solver*" utilizado neste projeto.

Por fim, nos **Capítulos 5 e 6** é apresentada a parte subjetiva, onde analiso minha experiência pessoal ao longo do curso, e concluímos o trabalho e alguns agradecimentos.

1.3 Conceitos

1.3.1 Álgebra Linear

Combinação linear

Um vetor $y \in \Re^n$ é uma **combinação linear** de vetores $x_i \in \Re^n$, $i \in \{1, ..., k\}$, $k \in \mathbb{N}$, se[1] existem escalares $\alpha_i \in \Re$ tais que $y = \sum_{i=1}^{k} \alpha_i x_i$.

Uma combinação linear é chamada de:

- **afim**, se $\sum_{i=1}^{k} \alpha_i = 1$;
- **cônica**[2], se $\alpha_i \geq 0$;
- **convexa**, se for afim e cônica.

Dependência e independência linear

Um **conjunto** de vetores $V = \{v_i \in \Re^n,\ i = 1, \ldots, k,\ k \in \mathbb{N}\}$ é **linearmente dependente** se, e somente se:

- Existem $\alpha_i \in \Re$, com algum deles diferente de zero, tais que $\sum_{i=1}^{k} \alpha_i v_i = 0$;
- Um dos vetores do conjunto (e.g. v_1) pode ser escrito como combinação linear dos outros vetores.

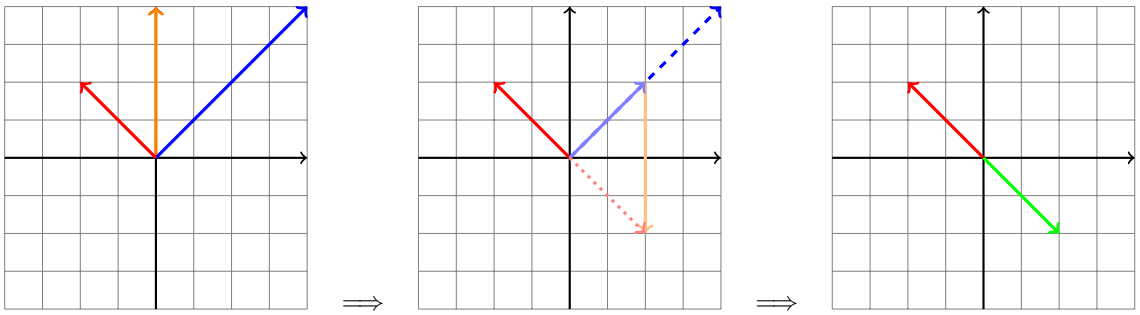

Figura 1.1: Três vetores linearmente dependentes no $\Re^2$. Um vetor oposto a v_1 (vermelho) é sintetizado a partir de v_2 (laranja) e v_3 (azul). Somando os vetores verde e vermelho, temos o vetor nulo.

Intuitivamente, a figura acima ilustra que um conjunto linearmente dependente permite **sintetizar** o vetor nulo a partir dos elementos desse conjunto, **mantendo** o tamanho de pelo menos um deles diferente de zero. O "truque" consiste em construir, por uma combinação linear, um vetor **de mesmo tamanho e direção**, mas de sentido **contrário**, a esse vetor selecionado cujo tamanho foi mantido positivo, e **somá-los** vetorialmente, assim sintetizando o vetor nulo.

Um conjunto de vetores é **linearmente independente** se ele não for linearmente dependente.

[1] Ou seja, se y pode ser expresso como a soma vetorial dos x_i, com cada uma de suas magnitudes (e sentidos) alterados pelo fator correspondente α_i.

[2] Não confundir com côncava

Subespaços e bases

Um subespaço $S \in \Re^n$ é o conjunto formado por todas as possíveis combinações lineares de um número finito de vetores $v_1 \dots v_k$, $k \leq n$. Note que todo subespaço contém o vetor nulo [1]. Dizemos que S é **gerado** por estes vetores.

Dizemos alternativamente que S é o **fecho linear** (ou "*span*", ou **subespaço linear**) de tais vetores. Analogamente, define-se [2] os termos **subespaço afim**, **cone**, e **fecho convexo**, respectivamente, para as combinações lineares especiais descritas no item anterior.

Poliedro

Um subconjunto $P \subseteq \Re^n$ é denominado **poliedro** se

$$P = \{x \in \Re^n \mid Ax \leq b,\ A \in \Re^{m \times n},\ b \in \Re^m\}$$

e é denotado por $P(A, b)$.

Um poliedro da forma $\{x \in \Re^n \mid a'x \leq \alpha,\ \alpha \in \Re,\ a \in \Re^n\}$ é denominado **semi-espaço**. Logo, segue que $P(A, b)$ é uma intersecção de semi-espaços.

1.3.2 Programação linear

Define-se então um problema de programação linear PL como:

$$PL = \max \{c'x \mid x \in P(A, b),\ c \in \Re^n,\ x \geq 0\}$$

E um problema de **programação linear inteira** PLI como:

$$PLI = \{PL \mid x \in \mathbb{Z}^n\}$$

Denominamos $c'x$ a **função objetivo**, e x, **variável de decisão** do PL. O **fecho inteiro** de $P(A, b)$ equivale ao seu fecho convexo para $x \in PLI$.

1.3.3 Teoria dos Grafos

Um **digrafo** $DG(V, A)$ consiste de um conjunto finito não-vazio V de elementos chamados **vértices** e um conjunto A de **pares ordenados** de elementos distintos de V, chamados **arcos**. Um **grafo** $G(V, A') \supseteq DG(V, A)$ pode ser modelado a partir de um digrafo construindo, para cada arco $a \in A$, um arco a^* no sentido contrário, e adicionando este **par** de arcos (representando uma **aresta**) ao conjunto A'.

Um **emparelhamento** em um grafo é um conjunto de arestas $X \subseteq A$ onde nenhum par de arestas possui vértices em comum. Um emparelhamento é **maximal** se, ao acrescentar qualquer outra aresta de A em X, este conjunto deixa de ser um emparelhamento, e **máximo** se, para qualquer outro emparelhamento $X' \neq X$, $|X'| \leq |X|$.

Capítulo 2

Escolha do *solver*

Inicialmente, a modelagem deste trabalho foi formulada em **Lingo** [4], pois foi o primeiro *solver* com que tomei contato que dispunha de uma linguagem de modelagem e que resolvia problemas de programação inteira.

Uma alternativa *open-source*, com desempenho equivalente e com uma linguagem de modelagem superior, foi encontrada após algum tempo: o **GNU Linear Programming Kit** (glpk) [3], que substituiu o Lingo neste trabalho.

Ambos os "solvers" possuem ajuda disponível na Internet para resolução de problemas. Notavelmente, o *newsgroup* do glpk mostrou-se muito útil, com usuários dispostos a tirar dúvidas iniciais ingênuas, (o que também ocorreu com o suporte do Lingo, apesar de pago e não tão rápido em responder). Os "solvers" não são perfeitos, contudo. Ambos possuem uma deficiência evidente, na opinião deste autor, que é a ausência de construções para modularizar/encapsular modelos, o que não foi fator impeditivo para que as demais qualidades do glpk pudessem sobressair.

Como ilustração, mostramos abaixo o output do modelo completo (presente no apêndice desta monografia), e em seguida um texto auxiliar para a interpretação da saída do programa:

```
Generating obj...
Generating Restricao_1...
Generating Restricao_2...
Generating Restricao_3i...
Generating Restricao_3s...
Generating Restricao_4i...
Generating Restricao_4s...
Generating Restricao_6...
Model has been successfully generated
GLPK Integer Optimizer, v4.43
14726 rows, 2625 columns, 57750 non-zeros
2625 integer variables, all of which are binary
Preprocessing...
14350 rows, 2625 columns, 52500 non-zeros
2625 integer variables, all of which are binary
Solving LP relaxation...
GLPK Simplex Optimizer, v4.43
14350 rows, 2625 columns, 52500 non-zeros
      0: obj =   0.000000000e+00  infeas =  1.750e+02 (0)
    500: obj =   3.925000000e+02  infeas =  6.500e+00 (0)
*   781: obj =   4.000000000e+02  infeas =  5.542e-15 (0)
*   785: obj =   4.000000000e+02  infeas =  1.415e-15 (0)
OPTIMAL SOLUTION FOUND
```

```
Integer optimization begins...
+   785: mip =     not found yet <=              +inf        (1; 0)
+  1892: mip =     not found yet <=   4.000000000e+02        (5; 0)
(...)
+ 11101: mip =   4.000000000e+02 <=     tree is empty   0.0% (0; 347)
INTEGER OPTIMAL SOLUTION FOUND
Time used:   107.5 secs
Memory used: 22.0 Mb (23104703 bytes)
SALA 1
```

	SEGUNDA	TERCA	QUARTA	QUINTA	SEXTA
HORARIO1	CAPELA	FISICA	MATEMATICA	PORTUGUES	PORTUGUES
HORARIO2	MATEMATICA	QUIMICA	QUIMICA	BIOLOGIA	MATEMATICA
HORARIO3	MATEMATICA	RELIGIAO	FISICA	BIOLOGIA	QUIMICA
HORARIO4	INGLES	INGLES	BIOLOGIA	GEOGRAFIA	HISTORIA
HORARIO5	PORTUGUES	EDUC_FISICA	HISTORIA	RELIGIAO	ARTES
HORARIO6	PORTUGUES	EDUC_FISICA	GEOGRAFIA	FISICA	FISICA
HORARIO7	ESPANHOL	PORTUGUES	PORTUGUES	FILOSOFIA	SOCIOLOGIA

SALA 2

	SEGUNDA	TERCA	QUARTA	QUINTA	SEXTA
HORARIO1	GEOGRAFIA	FILOSOFIA	PORTUGUES	BIOLOGIA	FISICA
HORARIO2	BIOLOGIA	FISICA	MATEMATICA	PORTUGUES	QUIMICA
HORARIO3	FISICA	MATEMATICA	QUIMICA	GEOGRAFIA	CAPELA
HORARIO4	FISICA	PORTUGUES	SOCIOLOGIA	MATEMATICA	PORTUGUES
HORARIO5	RELIGIAO	QUIMICA	BIOLOGIA	EDUC_FISICA	PORTUGUES
HORARIO6	MATEMATICA	INGLES	HISTORIA	EDUC_FISICA	ESPANHOL
HORARIO7	PORTUGUES	ARTES	HISTORIA	INGLES	RELIGIAO

SALA 3

	SEGUNDA	TERCA	QUARTA	QUINTA	SEXTA
HORARIO1	PORTUGUES	PORTUGUES	FISICA	FISICA	BIOLOGIA
HORARIO2	RELIGIAO	ESPANHOL	HISTORIA	MATEMATICA	PORTUGUES
HORARIO3	QUIMICA	INGLES	MATEMATICA	EDUC_FISICA	PORTUGUES
HORARIO4	QUIMICA	FISICA	GEOGRAFIA	BIOLOGIA	QUIMICA
HORARIO5	GEOGRAFIA	RELIGIAO	EDUC_FISICA	HISTORIA	INGLES
HORARIO6	ARTES	BIOLOGIA	PORTUGUES	FILOSOFIA	MATEMATICA
HORARIO7	FISICA	SOCIOLOGIA	CAPELA	PORTUGUES	MATEMATICA

SALA 4

	SEGUNDA	TERCA	QUARTA	QUINTA	SEXTA
HORARIO1	QUIMICA	MATEMATICA	BIOLOGIA	SOCIOLOGIA	INGLES
HORARIO2	FISICA	BIOLOGIA	PORTUGUES	CAPELA	ESPANHOL

```
HORARIO3 |PORTUGUES    |BIOLOGIA     |FILOSOFIA    |MATEMATICA   |FISICA       |
HORARIO4 |PORTUGUES    |RELIGIAO     |EDUC_FISICA  |HISTORIA     |EDUC_FISICA  |
HORARIO5 |ARTES        |PORTUGUES    |INGLES       |PORTUGUES    |RELIGIAO     |
HORARIO6 |GEOGRAFIA    |QUIMICA      |QUIMICA      |PORTUGUES    |HISTORIA     |
HORARIO7 |MATEMATICA   |FISICA       |MATEMATICA   |FISICA       |GEOGRAFIA    |
-------------------------------------------------------------------------------

SALA 5
===============================================================================
         |SEGUNDA      |TERCA        |QUARTA       |QUINTA       |SEXTA        |
-------------------------------------------------------------------------------
HORARIO1 |INGLES       |QUIMICA      |SOCIOLOGIA   |MATEMATICA   |HISTORIA     |
HORARIO2 |QUIMICA      |HISTORIA     |FISICA       |FISICA       |FISICA       |
HORARIO3 |BIOLOGIA     |PORTUGUES    |PORTUGUES    |PORTUGUES    |BIOLOGIA     |
HORARIO4 |FILOSOFIA    |GEOGRAFIA    |PORTUGUES    |INGLES       |ESPANHOL     |
HORARIO5 |CAPELA       |FISICA       |MATEMATICA   |BIOLOGIA     |GEOGRAFIA    |
HORARIO6 |EDUC_FISICA  |MATEMATICA   |ARTES        |QUIMICA      |PORTUGUES    |
HORARIO7 |RELIGIAO     |MATEMATICA   |RELIGIAO     |EDUC_FISICA  |PORTUGUES    |
-------------------------------------------------------------------------------
```

2.1 Formato da saída glpk

A resolução de um problema de Programação Linear Inteira Mista (ou MIP, do inglês "*Mixed Integer Programming*") pode tomar bastante tempo, então o "*solver*" exibe certas informações periodicamente sobre as melhores soluções parciais encontradas. Tais informações são enviadas à saída-padrão (terminal), e têm o seguinte formato:

```
+nnn: mip = xxx <rho> yyy gap (ppp; qqq)
```

Onde: *nnn* é o valor atual do contador de iterações do simplex; *xxx* é o valor da função objetivo para a melhor solução viável básica inteira conhecida (ou "*not found yet*", se nenhuma solução viável básica inteira tiver sido encontrada até o momento); *rho* é a cadeia de caracteres >= (no caso de um problema de minimização) ou <= (no caso de um problema de maximização); *yyy* é um limitante global para o valor ótimo inteiro (ou seja, o valor ótimo inteiro exato está sempre no intervalo fechado entre *xxx* e *yyy*); *gap* é a distância relativa, em porcentagem, do limitante global ótimo para a melhor solução viável básica encontrada, medida como

$$\textbf{gap} = |xxx - yyy|/(|xxx| + DBL_EPSILON) * 100\%$$

(se *gap* é maior do que $999,9\%$, não é exibido); *ppp* é o número de subproblemas ativos, *qqq* é o número de subproblemas que já foram completamente sondados (no inglês, "*fathomed*") e portanto removidos da árvore de busca "*branch-and-bound*".

Capítulo 3

Modelando um problema linear inteiro

3.1 Formulações de PLI com variáveis binárias

Uma variável de decisão $x_k \in \{0, 1\}$, pode compôr restrições no modelo de diversas formas. Quando ela representa a ocorrência ou não de um evento, temos comumente alguns padrões de modelagem [1]:

- No máximo um entre os n eventos ocorre:

$$\sum_{k=1}^{n} x_k \leq 1$$

- Ou ambos eventos ocorrem (e.g. para $n = 2$), ou nenhum ocorre:

$$x_1 = x_2$$

- Se um evento x_1 ocorre, o evento x_2 necessariamente ocorre:

$$x_1 \leq x_2$$

A título de exemplo, apresentamos um modelo que encontra, se existente, uma ocorrência de um emparelhamento máximo em um grafo, e o output de sua execução.

3.1.1 Exemplo: Emparelhamento máximo

Modelo para Emparelhamento máximo em um grafo - versão Mathprog

```
#Vértices
set V;

#Arestas
param A {i in V, j in V}, binary;

#Variável de decisão
var X {v1 in V, v2 in V}, binary;

#Função objetivo
maximize obj: sum {v1 in V, v2 in V} X[v1,v2] * A[v1,v2];

## Restrições ##
#Cada aresta deve ter duas pontas distintas
check {v in V} : A[v, v] = 0;

#Modelando grafo a partir de um digrafo
check {v1 in V, v2 in V} : A[v1,v2] = A[v2,v1];

#Arcos duplos (representando arestas), ou nenhuma aresta
s.t. r1 {v1 in V, v2 in V} : X[v1,v2] * A[v1,v2] = X[v2,v1] * A[v2,v1];

#Modelando emparelhamento
s.t. r2 {v1 in V} : sum {v2 in V} (X[v1,v2] * A[v1,v2] + X[v2,v1] * A[
    v2,v1]) <= 2;

#Resolve o problema linear
solve;

#Mostra resultado
display X;

data;

#Definição de parâmetros
set V := V1 V2 V3 V4;

param A : V1 V2 V3 V4 :=
       V1 0  1  0  0
       V2 1  0  1  0
       V3 0  1  0  1
       V4 0  0  1  0;

#Resposta para esta instância: {{1,2},{3,4}}
end;
```

```
INTEGER OPTIMAL SOLUTION FOUND
Time used:   0.0 secs
Memory used: 0.1 Mb (143961 bytes)
Display statement at line 30
X[V1,V1] = 0
X[V1,V2] = 1
X[V1,V3] = 0
X[V1,V4] = 0
X[V2,V1] = 1
X[V2,V2] = 0
X[V2,V3] = 0
X[V2,V4] = 0
X[V3,V1] = 0
X[V3,V2] = 0
X[V3,V3] = 0
X[V3,V4] = 1
X[V4,V1] = 0
X[V4,V2] = 0
X[V4,V3] = 1
X[V4,V4] = 0
```

Capítulo 4

Modelando uma grade horária

4.1 Definindo a função objetivo

Na modelagem proposta, temos a seguinte função objetivo:

$$f(H, Peso) = H_{hr,\,d}^{m,\,sl} \ * \ Peso_{hr,\,d}^{m,\,sl}$$

$$(H_{hr,\,d}^{m,\,sl} \in \{0, 1\}, Peso_{hr,\,d}^{m,\,sl} \in \mathbb{Z})$$

Os índices representam o seguinte:

- **m** $\in MATERIA$: Matéria lecionada;
- **sl** $\in SALA$: Sala de aula;
- **hr** $\in HORARIO$: Horário da aula;
- **d** $\in DIA$: Dia da semana.

Assim, formulamos o seguinte PLI, sujeito a restrições de viabilidade:

$$\max \sum_{hr,\,d}^{m,\,sl} H_{hr,\,d}^{m,\,sl} \ * \ Peso_{hr,\,d}^{m,\,sl}$$

Assim, sugerindo $Pesos$ atrelados às variáveis de decisão $H_{hr,\,d}^{m,\,sl}$, podemos indicar preferências na grade horária.

4.2 Modelando restrições

Definir restrições em um programa linear inteiro significa **definir o poliedro** das soluções viáveis, que atendem aos requisitos do problema. Sem elas, a solução ótima não necessariamente seria viável. Portanto, representam as limitações reais do problema, que devem ser obedecidas.

Neste modelo, classificamos as restrições como **objetivas** (quando dizem respeito a limitações físicas ou regras do problema), e **subjetivas**, quando a elaboração da grade horária deve seguir regras pessoais de algum docente.

A apresentação das restrições a seguir é feita da seguinte maneira: definimos cada uma delas em **linguagem natural**, depois escrevemos seu **pseudo-código**, então exemplificamos o comportamento do modelo para certos **valores de entrada** e, finalmente, apresentamos o **código-fonte** da restrição em **Mathprog**, com construções auxiliares (como parâmetros e conjuntos, etc.) quando necessário.

4.2.1 Restrição objetiva 1 - Viabilidade de sala

Esta restrição diz ao "*solver*" que, ao tentar maximizar o número de aulas lecionadas, não permita que mais de uma matéria seja lecionada em uma mesma sala, em um horário e dia da semana.

$$\textbf{para cada } sl \in SALA$$
$$\textbf{para cada } hr \in HORARIO$$
$$\textbf{para cada } d \in DIA$$
$$\sum_{m \in MATERIA} H_{hr,\,d}^{m,\,sl} \leq 1$$

Por exemplo, supondo um modelo apenas com aulas de exatas, fixados horário (1º), sala (1ª) e dia da semana (SEGUNDA), temos que a restrição definida acima garante que:

$$\mathbf{H}[\mathsf{MATEMATICA}, 1, 1, \mathsf{SEGUNDA}] + \mathbf{H}[\mathsf{FISICA}, 1, 1, \mathsf{SEGUNDA}] + \mathbf{H}[\mathsf{QUIMICA}, 1, 1, \mathsf{SEGUNDA}] <= 1$$

(no máximo uma matéria é lecionada nessa posição da grade horária)

Restrição de viabilidade de sala - versão Mathprog

```
param SALAS integer := 1;
param HORARIOS integer := 1;

set MATERIA;
set SALA := {i in 1..SALAS};
set HORARIO := {i in 1..HORARIOS};
set DIA;

var H {MATERIA, SALA, HORARIO, DIA} binary;

#Omitindo pesos por simplicidade
maximize obj:
  sum {M in MATERIA, SL in SALA, HR in HORARIO, D in DIA}
      H[M, SL, HR, D];

#Restrição 1) Uma sala não pode ter duas matérias lecionadas ao mesmo
   tempo
s.t. Restricao_1 "AULA"
  {SL in SALA, HR in HORARIO, D in DIA} :
      sum {M in MATERIA} H[M, SL, HR, D] <= 1;

data;

set DIA := SEGUNDA;
set MATERIA :=
MATEMATICA
FISICA
QUIMICA;

end;
```

4.2.2 Restrição objetiva 2 - Viabilidade de professor

Esta restrição do modelo impõe uma limitação física importante:

> "Cada professor leciona apenas **uma** matéria, e cada matéria é lecionada por apenas **um** professor"

Assim, temos uma correspondência biunívoca entre professores e matérias, que é útil para simplificar a modelagem para efeito deste trabalho, mas que deve ser revista em casos reais mais complexos.

Como consequência, não podemos ter uma matéria sendo lecionada **ao mesmo tempo** em mais de uma sala.

$$\begin{array}{c} \textbf{para cada } m \in MATERIA \\ \textbf{para cada } hr \in HORARIO \\ \textbf{para cada } d \in DIA \\ \sum_{s \in SALA} H_{hr,\,d}^{m,\,sl} \leq 1 \end{array}$$

Por exemplo, supondo uma instituição com mais de uma sala (salas 1, 2 e 3), fixados horário (1º), dia da semana (SEGUNDA) e matéria (FISICA), temos que a restrição definida acima garante que:

$$\mathbf{H}[\mathsf{FISICA}, 1, 1, \mathsf{SEGUNDA}] + \mathbf{H}[\mathsf{FISICA}, 2, 1, \mathsf{SEGUNDA}] + \mathbf{H}[\mathsf{FISICA}, 3, 1, \mathsf{SEGUNDA}] <= 1$$

(Uma matéria não pode ser lecionada em duas salas ao mesmo tempo)

Restrição de viabilidade de professor - versão Mathprog

```
param SALAS integer := 3;

## ...Demais definições iguais à restrição anterior... ##

#Restrição 2) Uma matéria não pode ser lecionada em duas salas ao mesmo
    tempo (com apenas um professor para cada matéria)
s.t. Restricao_2 "DISCIPLINA"
  {M in MATERIA, HR in HORARIO, D in DIA} :
      sum {SL in SALA} H[M, SL, HR, D] <= 1;

data;

## ...Demais parâmetros iguais à restrição anterior... ##

set MATERIA := FISICA;

end;
```

4.2.3 Restrição objetiva 3 - Cotas semanais de aulas de cada matéria

Nesta restrição, começamos a tornar o modelo condizente com a realidade de uma instituição de ensino, no que tange a estipulação de **cotas semanais** de aulas para cada matéria, em cada sala. Para manter o modelo simples, consideramos que cada uma das salas da instituição têm as **mesmas cotas** por matéria. Cada cota define uma quantidade **mínima** e **máxima** de aulas de cada matéria.

$$\begin{aligned}
&\textbf{para cada } m \in MATERIA \\
&\quad \textbf{para cada } sl \in SALA \\
&\qquad \sum_{hr \in HORARIO,\, d \in DIA} H_{hr,\,d}^{m,\,sl} \geq CTA_INF[m] \\
&\qquad \sum_{hr \in HORARIO,\, d \in DIA} H_{hr,\,d}^{m,\,sl} \leq CTA_SUP[m]
\end{aligned}$$

Aqui, assumindo fixadas (e.g.) cotas inferior (2 aulas) e superior (6 aulas) de uma matéria (PORTUGUÊS), para uma instituição com uma sala, três dias na semana (SEGUNDA, QUARTA e SEXTA) e apenas um horário (1º), temos que a restrição definida acima garante que:

$$\begin{gathered}
\text{CTA_INF[PORTUGUES]} = 2 \\
\text{CTA_SUP[PORTUGUES]} = 6
\end{gathered}$$

$$\text{CTA_INF[PORTUGUES]} \leq \begin{array}{l} \mathbf{H}\text{[PORTUGUES, 1, 1, SEGUNDA]} + \\ \mathbf{H}\text{[PORTUGUES, 1, 1, QUARTA]} + \\ \mathbf{H}\text{[PORTUGUES, 1, 1, SEXTA]} \end{array} \leq \text{CTA_SUP[PORTUGUES]}$$

(Cotas semanais de aulas de cada matéria)

Cotas semanais de aulas de cada matéria - versão Mathprog

```
## ...Após definir horários e salas cf. exemplo restrição 1... ##
#Cotas
param COTA_INFERIOR {MATERIA} integer;
param COTA_SUPERIOR {MATERIA} integer;

#Restrição 3) Cota semanal de aulas por matéria, por sala
s.t. Restricao_3i {M in MATERIA, SL in SALA} :
  sum {HR in HORARIO, D in DIA} H[M, SL, HR, D] >= COTA_INFERIOR[M];
s.t. Restricao_3s {M in MATERIA, SL in SALA} :
  sum {HR in HORARIO, D in DIA} H[M, SL, HR, D] <= COTA_SUPERIOR[M];

data;
#Demais matérias têm CT_INF=1
param COTA_INFERIOR default 1 := PORTUGUES 2;
#Demais matérias têm CT_SUP=4
param COTA_SUPERIOR default 4 := PORTUGUES 6;
set DIA := SEGUNDA QUARTA SEXTA;
end;
```

4.2.4 Restrição objetiva 4 - Cotas diárias de aulas de cada matéria

Além das cotas semanais estipuladas acima, temos requisitos que determinam que muitas matérias terão dias da semana sem aulas, e cada uma não deve ultrapassar um limite máximo de aulas por dia. Para permitir tal restrição, são construídos dois parâmetros, CTA_INFD[m, sl, d] e CTA_SUPD[m, sl, d], semelhantes aos da restrição anterior, só que desta vez definidos para **cada** matéria **m**, sala **sl**, e dia **d**. Para manter o exemplo simples, atribuiremos por padrão cotas com as mesmas quantidades para todas as matérias, e cotas inferior e superior diferentes para um caso específico ilustrativo (FISICA).

$$\begin{aligned}
&\textbf{para cada } m \in MATERIA \\
&\quad \textbf{para cada } sl \in SALA \\
&\quad\quad \textbf{para cada } d \in DIA \\
&\quad\quad\quad \sum_{hr \in HORARIO} H_{hr,\,d}^{m,\,sl} \geq CTA_INFD[m, sl, d] \\
&\quad\quad\quad \sum_{hr \in HORARIO} H_{hr,\,d}^{m,\,sl} \leq CTA_SUPD[m, sl, d]
\end{aligned}$$

Exemplificando, fixamos cotas diárias inferior (0 aulas) e superior (2 aulas) de todas as matérias de exatas novamente (e.g.), para uma instituição com uma sala (1ª), dois dias na semana (SEGUNDA e QUARTA) e três horários, e faremos uma disciplina ter uma cota especial (FISICA) em um dia da semana (QUARTA):

$$\begin{gathered}
\text{CTA_INFD[\{QUIMICA, FISICA, MATEMATICA\}, 1, \{SEGUNDA, QUARTA\}]} = 0 \\
\text{CTA_SUPD[\{QUIMICA, FISICA, MATEMATICA\}, 1, \{SEGUNDA, QUARTA\}]} = 2 \\
\text{CTA_INFD[FISICA, 1, QUARTA]} = 0 \\
\text{CTA_SUPD[FISICA, 1, QUARTA]} = 4
\end{gathered}$$

$$\text{CTA_INFD[FISICA, 1, QUARTA]} \leq \begin{array}{l} \mathbf{H}\text{[FISICA, 1, 1, QUARTA]} + \\ \mathbf{H}\text{[FISICA, 1, 2, QUARTA]} + \\ \mathbf{H}\text{[FISICA, 1, 3, QUARTA]} \end{array} \leq \text{CTA_SUPD[FISICA, 1, QUARTA]}$$

(Cotas diárias de aulas de cada matéria)

Cotas diárias de aulas de cada matéria - versão Mathprog

```
#Cotas
param COTA_INFERIOR_DIA {MATERIA, SALA, DIA} integer;
param COTA_SUPERIOR_DIA {MATERIA, SALA, DIA} integer;

#Restrição 4) Cota diária de aulas por matéria, por sala
s.t. Restricao_4i {M in MATERIA, SL in SALA, D in DIA} : sum {HR in
   HORARIO} H[M, SL, HR, D] >= COTA_INFERIOR_DIA[M, SL, D];
s.t. Restricao_4s {M in MATERIA, SL in SALA, D in DIA} : sum {HR in
   HORARIO} H[M, SL, HR, D] <= COTA_SUPERIOR_DIA[M, SL, D];

data;
param COTA_INFERIOR_DIA default 0;
param COTA_SUPERIOR_DIA default 2;
set DIA := SEGUNDA QUARTA;
end;
```

4.2.5 Restrição objetiva 5 - "Convexidade" de aulas

Esta restrição diz ao "*solver*" que, caso o **total** de aulas lecionadas na semana em uma sala seja **menor** do que o total de **horários** disponíveis na grade horária, então os horários vagos (ou "**janelas**", sem aulas), devem ocorrer no início ou no fim do dia. Intuitivamente, tal requisito é equivalente a dizer que as aulas, em uma sala e dia da semana, devem ser "**convexas**". Ou seja, não deve haver *lacunas* entre aulas[1].

A modelagem desta restrição difere das demais, no que aproveita o fato de que o "*solver*" sempre busca o valor ótimo da função objetivo. Assim, aplicando **pesos** a cada uma das variáveis de decisão **H**, de acordo com o horário do dia que representam, fazemos com que horários no meio do dia sejam "mais valorizados" do que os demais. Em pseudo-código:

```
1  para cada m ∈ MATERIA
2      para cada sl ∈ SALA
3          para cada d ∈ DIA
4              para cada hr ∈ HORARIO
5                  se hr ≤ ⌊HORARIOS / 2⌋
6                      PESO[m, sl, hr, d] = hr
7                  senão
8                      PESO[m, sl, hr, d] = HORARIOS − hr + 1
```

Por exemplo, fixados matéria (MATEMATICA), sala (1ª) e dia da semana (SEGUNDA), e supondo cinco horários no dia, temos os seguintes valores para o parâmetro auxiliar PESO:

PESO[MATEMATICA,1,1,SEGUNDA] = 1
PESO[MATEMATICA,1,2,SEGUNDA] = 2
PESO[MATEMATICA,1,3,SEGUNDA] = 3 *("Convexidade" de aulas)*
PESO[MATEMATICA,1,4,SEGUNDA] = 2
PESO[MATEMATICA,1,5,SEGUNDA] = 1

Restrição de viabilidade de sala - versão Mathprog

```
#Peso finalmente aplicado à função objetivo
maximize obj:
  sum {M in MATERIA, SL in SALA, HR in HORARIO, D in DIA}
    H[M, SL, HR, D] * PESO[M, SL, HR, D];

#Restrição 5) Janelas devem ocorrer no início ou no final do dia
param PESO "Restricao_5"
  {M in MATERIA, SL in SALA, HR in HORARIO, D in DIA},
  integer,
  >=0,
  default
    (if HR <= card(HORARIO) div 2 then HR else card(HORARIO) - HR + 1);

    end;
```

[1] ao menos na grade horária planejada :)

4.2.6 Restrição objetiva 6 - Aulas sucessivas de uma mesma matéria ("*dobradinhas*")

Como consequência das restrições anteriores que definem cotas, podemos ter **mais de uma aula** em um certo dia, de uma determinada matéria. Assim, esta restrição objetiva visa forçar que tais aulas ocorram **em sequência**, fazendo o modelo ficar inviável (i.e. sem solução) caso contrário. Para tanto, usamos o mesmo conceito de convexidade da restrição anterior, mas desta vez aplicado aos sucessivos valores da variável de decisão **H**, fixados matéria, sala e dia.

```
1 para cada m ∈ MATERIA
2     para cada sl ∈ SALA
3         para cada d ∈ DIA
4             para cada i ∈ HORARIO
5                 para j = i + 2 até HORARIOS
6                     para k = i + 1 até j − 1
7                         H[m, sl, k, d] ≥ H[m, sl, i, d] + H[m, sl, j, d] − 1
```

O pseudo-código acima funciona da seguinte maneira: para todos os casos de duas aulas A_i e A_j de uma matéria m, fixados sala e dia, com um ou mais horários **entre** elas, temos que cada uma das aulas intermediárias A_k deve ser da matéria m.

Por exemplo, fixados matéria (QUIMICA), sala (1ª), dia (SEGUNDA) e para três horários, se há aula no primeiro e último horários, então devemos ter:

$$\mathbf{H}[\mathsf{QUIMICA}, 1, 2, \mathsf{SEGUNDA}] >= \mathbf{H}[\mathsf{QUIMICA}, 1, 1, \mathsf{SEGUNDA}] + \mathbf{H}[\mathsf{QUIMICA}, 1, 3, \mathsf{SEGUNDA}] - 1$$

(Aulas sucessivas de uma mesma matéria)

Restrição de viabilidade de sala - versão Mathprog

```
s.t. Restricao_6 {M in MATERIA, SL in SALA, D in DIA, I in HORARIO, J
   in I+2..card(HORARIO), K in I+1..J-1} : H[M, SL, K, D] >= H[M, SL,
   I, D] + H[M, SL, J, D] - 1;
end;
```

4.3 Restrições subjetivas

O modelo até então consegue compôr uma grade horária sem intervenção de detalhes pelo usuário, como pode ser comprovado pelo output no Capítulo 2. Mesmo assim, certamente a grade precisará de ajustes para que se acomode a necessidades específicas de docentes, ou até mesmo viabilidade de salas físicas.

Para isto definimos **restrições subjetivas**, que são da forma:

$$H[MATERIA, SALA, HORARIO, DIA] = 1$$

*(Positiva, quando queremos forçar que uma aula necessariamente **aconteça**, ou o modelo se inviabilize c.c.)*

OU

$$H[MATERIA, SALA, HORARIO, DIA] = 0$$

*(Negativa, quando queremos forçar que uma aula **não** aconteça, ou o modelo se inviabilize c.c.)*

Restrições subjetivas - versão Mathprog

```
#Restrição subjetiva positiva (Matemática na primeira aula de segunda,
   sala 1)
s.t. foo: H["MATEMATICA","1","1","SEGUNDA"] = 1;

#Restrição subjetiva negativa (Educação física NUNCA ocorrerá na
   primeira aula de segunda, sala 1)
s.t. foo: H["EDUC_FISICA","1","1","SEGUNDA"] = 0;
```

Parte II

Parte subjetiva

Capítulo 5

Sobre o curso

5.1 Desafios do trabalho de conclusão

Acredito que o maior desafio neste trabalho foi escolher e permanecer em um tema: foram três anos de amadurecimento na escolha e tentativa de preparar algo que fosse diretamente relacionado a alguma disciplina do curso.

Ao mesmo tempo, queria algo que despertasse interesse pessoal e representasse uma proposta factível em um projeto de um ano (que, devido aos compromissos profissionais, sabidamente não seria exclusivamente dedicado a este trabalho, apesar das tentativas de planejamento).

5.2 Frustrações

5.2.1 No divã

Falar sobre frustrações requer disposição a lembrar do que não aconteceu conforme esperado, e achar um porquê.

Acredito que minha maior fonte de problemas no curso foi ter demorado para aceitar o fato de que não iria me formar com a minha turma original (de 2003). Isso fez com que eu ignorasse o oferecimento de disciplinas em que fui reprovado, no ano seguinte à reprovação, para eliminá-las o quanto antes. Ao invés disso, optei por continuar fazendo as disciplinas da grade ideal, junto com a turma. Invariavelmente isso resultou em mais reprovações.

Os principais fatores que acredito terem contribuído para as reprovações, em ordem decrescente de importância, foram:

1. **Falta de assiduidade nas aulas presenciais:** Presença em todos os dias pela manhã foi o objetivo mais difícil de cumprir, que só fui alcançar nos últimos anos do curso. Morando longe da USP, chegar no horário das aulas era "matar um leão por dia". Assistir aulas torna o semestre muito mais viável, e uma solução encontrada durante os semestres finais para obter 100% de frequência foi dormir temporariamente fora de casa;

2. **Trabalho simultâneo ao curso:** Comecei a estagiar no segundo semestre do curso, e foram poucos os meses em que não tinha compromissos externos ao curso. O tempo útil do dia, portanto, nem sempre era voltado às preocupações com o curso;

3. **Ausência de planejamento de conclusão:** Nesse período de indefinição, enquanto prevaleciam as condições acima, ainda não havia traçado um plano considerando os oferecimentos da grade curricular, para a minha formatura.

5.2.2 A retomada

A partir do segundo semestre de 2006 (em que minha turma original se formou), tomei consciência e fiz um plano de conclusão de curso, retomando seriamente as atividades. Felizmente a jornada restante foi conduzida com novas amizades e acredito que estas contribuíram muito para meu amadurecimento.

5.3 Disciplinas relevantes e aplicação de conceitos

Acredito que o fato que ocasionou a maior mudança de paradigma para mim, durante o curso, foi aprender a usar software livre. Antes de falar sobre disciplinas, o uso constante, a troca de conhecimento com meus colegas (por grupos de discussão e "Installfests") e as tarefas práticas do curso foram excelentes contribuições para este fim.

As disciplinas que considero importantes para este trabalho:

- **MAC0122 - Princípios de desenvolvimento de algoritmos:** (*2003/2 - Carlinhos*)
  Primeiro contato com problemas cuja complexidade não havia encontrado fora da faculdade;
- **MAC0211 - Laboratório de Programação I:** (*2004/1 - Gubi*)
  Primeiro trabalho prático em linguagem C (em que o "solver" é escrito);
- **FLC0474 - Língua Portuguesa:** (*2006/1 - Reginaldo*)
  Oportunidade de praticar e ser avaliado por dissertações;
- **MAC0315 - Programação Linear:** (*2006/1 - Leônidas; 2008/1 - Marcelo Queiroz*)
  O "solver" glpk inicia sua busca pela solução ótima usando o método Simplex relaxado, cujo algoritmo foi coberto nesta disciplina;
- **MAC0422 - Sistemas Operacionais:** (*2008/2 - Alan; 2009/2 - Marcel; 2010/2 - Marcel*)
  Programação avançada em C (extensões do Minix), além de praticar e dominar o uso de ferramentas open-source para desenvolvimento nesta linguagem (Vim, ctags);
- **MACWEB:** (*2009/1 - Gerosa*) Primeiro contato com Jquery;
- **Curso de Latex do PoliGNU:** (*2011/2 - Thiago "Tecepe" Paiva*)
  Este curso surgiu na hora necessária. Aprendi construções em Latex utilizadas na escrita do pôster, apresentação e monografia.

5.4 Aperfeiçoamento dos conceitos estudados e continuidade do trabalho

Pretendo continuar lendo toda a bibliografia ao longo deste ano, e aprimorar meu entendimento no tema estudado, possivelmente aplicando-o em uma versão mais robusta e amigável da modelagem.

Capítulo 6

Conclusão

6.1 Considerações finais

A modelagem proposta supõe alguns requisitos limitantes, como o fato de não poder haver mais de um professor da mesma matéria, mas considerando a evolução da pesquisa até chegar em uma modelagem que descrevesse e resolvesse o problema, mesmo com tais limitações, já foi para o autor um motivo de contentamento com o resultado do trabalho.

6.2 Agradecimentos

Acredito que o fato de ter cursado esta disciplina mais de uma vez foi um aprendizado penoso, mas que agregou valor à escolha do tema proposto neste trabalho. Acredito que cada docente com que tive decepções, alegrias, aprendizagens e até mesmo desavenças contribuiu para que noções, livros, tecnologias e linguagens de programação pudessem aprimorar meu amadurecimento como aluno acadêmico e como futuro bacharel.

Agradeço ao meu orientador, Alfredo, à Ellen Hidemi Fukuda e Thiago "Tecepe" Paiva pelas leituras, dicas e cursos ministrados, que me auxiliaram na elaboração desta monografia, e também à minha família e filha queridas, pelo apoio incondicional em todas as circunstâncias por que passei até agora.

Obrigado!

Apêndice A

Código completo da modelagem Mathprog

```
#Copyright 2011 by Nilo Cesar Teixeira
#
#This program is free software: you can redistribute it and/or modify
#it under the terms of the GNU General Public License as published by
#the Free Software Foundation, either version 3 of the License, or
#(at your option) any later version.
#
#This program is distributed in the hope that it will be useful,
#but WITHOUT ANY WARRANTY; without even the implied warranty of
#MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#GNU General Public License for more details.
#
#You should have received a copy of the GNU General Public License
#along with this program.  If not, see <http://www.gnu.org/licenses/>.

param SALAS integer := 5;
param HORARIOS integer := 7;

set MATERIA;
set SALA := {i in 1..SALAS};
set HORARIO := {i in 1..HORARIOS};
set DIA;

param LH := length("HORARIO") + length(HORARIOS) + 1; #padding
param MSZ := max {M in MATERIA} length(M) + 2; #padding
param HSZ integer := LH + 1 + card(DIA) * (MSZ + 1);
param SFMT symbolic := "%-" & MSZ & "s";
param SFMTH symbolic := "%-" & LH & "s";
param FMT symbolic := SFMT & "|";
param FMTH symbolic := SFMTH & "|";

var H {MATERIA, SALA, HORARIO, DIA} binary;

#Restrição 5) Janelas devem ocorrer no início ou no final do dia
param PESO "Restricao_5" {M in MATERIA, SL in SALA, HR in HORARIO, D in
    DIA},
    integer,
    >=0,
```

```
    default (if HR <= card(HORARIO) div 2 then HR else card(HORARIO) -
    HR + 1);

#Cotas
param COTA_INFERIOR {MATERIA} integer;
param COTA_SUPERIOR {MATERIA} integer;
param COTA_INFERIOR_DIA {MATERIA, SALA, DIA} integer;
param COTA_SUPERIOR_DIA {MATERIA, SALA, DIA} integer;

#Função objetivo, maximizar aulas
maximize obj: sum {M in MATERIA, SL in SALA, HR in HORARIO, D in DIA} H
    [M, SL, HR, D] * PESO[M, SL, HR, D];

#Restrição 1) Uma sala não pode ter duas matérias lecionadas ao mesmo
    tempo
s.t. Restricao_1 "AULA" {SL in SALA, HR in HORARIO, D in DIA} : sum {M
    in MATERIA} H[M, SL, HR, D] <= 1;

#Restrição 2) Uma matéria não pode ser lecionadas em duas salas ao
    mesmo tempo (com apenas um professor para cada matéria)
s.t. Restricao_2 "DISCIPLINA" {M in MATERIA, HR in HORARIO, D in DIA} :
     sum {SL in SALA} H[M, SL, HR, D] <= 1;

#Restrição 3) Cota semanal de aulas por matéria, por sala
s.t. Restricao_3i {M in MATERIA, SL in SALA} : sum {HR in HORARIO, D in
     DIA} H[M, SL, HR, D] >= COTA_INFERIOR[M];
s.t. Restricao_3s {M in MATERIA, SL in SALA} : sum {HR in HORARIO, D in
     DIA} H[M, SL, HR, D] <= COTA_SUPERIOR[M];

#Restrição 4) Cota diária de aulas por matéria, por sala
s.t. Restricao_4i {M in MATERIA, SL in SALA, D in DIA} : sum {HR in
    HORARIO} H[M, SL, HR, D] >= COTA_INFERIOR_DIA[M, SL, D];
s.t. Restricao_4s {M in MATERIA, SL in SALA, D in DIA} : sum {HR in
    HORARIO} H[M, SL, HR, D] <= COTA_SUPERIOR_DIA[M, SL, D];

#Restrição 6) Se mais de uma aula no dia, devem ser compactas, em
    sequencia;
s.t. Restricao_6 {M in MATERIA, SL in SALA, D in DIA, I in HORARIO, J
    in I+2..card(HORARIO), K in I+1..J-1} : H[M, SL, K, D] >= H[M, SL,
    I, D] + H[M, SL, J, D] - 1;

solve;

#### Imprimindo resultado ####

for {SL in SALA} {
  printf "%s\n", "SALA " & SL;
  printf {i in {1..HSZ}} "%s", "=";
  printf "\n";
  printf FMTH, "";
  for {D in DIA} {
    printf FMT, D;
  }
```

```
  printf "\n";
  printf {i in {1..HSZ}} "%s", "-";
  printf "\n";
  for {HR in HORARIO} {
    printf FMTH, "HORARIO" & HR;
    for {D in DIA} {
      printf {i in 1..MSZ} (if card({M in MATERIA: H[M, SL, HR, D] =
   1}) = 0 then "␣" else "");
      printf {M in MATERIA : H[M, SL, HR, D] = 1} SFMT, M;
      printf "|";
    }
    printf "\n";
  }
  printf {i in {1..HSZ}} "%s", "-";
  printf "\n\n";
}


##### Seção data ####

data;

param COTA_INFERIOR default 1 :=
  MATEMATICA 4
  FISICA 4
  QUIMICA 3
  BIOLOGIA 3
  INGLES 2
  HISTORIA 2
  PORTUGUES 6
  RELIGIAO 2
  EDUC_FISICA 2
  GEOGRAFIA 2;
param COTA_SUPERIOR default 6 :=
  MATEMATICA 4
  FISICA 4
  QUIMICA 3
  BIOLOGIA 3
  INGLES 2
  HISTORIA 2
  PORTUGUES 6
  RELIGIAO 2
  EDUC_FISICA 2
  GEOGRAFIA 2;

param COTA_INFERIOR_DIA default 0;
param COTA_SUPERIOR_DIA default 2;

set MATERIA :=
  MATEMATICA
  FISICA
  QUIMICA
  BIOLOGIA
```

```
  INGLES
  ESPANHOL
  HISTORIA
  PORTUGUES
  RELIGIAO
  EDUC_FISICA
  GEOGRAFIA
  SOCIOLOGIA
  CAPELA
  FILOSOFIA
  ARTES;

set DIA :=
  SEGUNDA
  TERCA
  QUARTA
  QUINTA
  SEXTA;

end;
```

Referências Bibliográficas

[1] Dimitris Bertsimas e John N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, Belmont, Massachusetts, USA, 3ª impressão, 1997.

[2] Carlos Eduardo Ferreira e Yoshiko Wakabayashi. *Combinatória Poliédrica e Planos de Corte Faciais*. UNICAMP, Instituto de Computação, Campinas, SP, Brasil, 1996.

[3] Andrew Makhorin. Glpk (gnu linear programming kit). http://www.gnu.org/s/glpk/, 2011. [Online; acessado em 01/Nov/2011].

[4] Lindo Systems. Lingo optimization modeling software. http://www.lindo.com/, 2011. [Online; acessado em 01/Set/2011].