INSTITUTO DE CIÊNCIAS EXATAS

UNIVERSIDADE FEDERAL RURAL DO RIO DE JANEIRO

CURSO DE GRADUAÇÃO EM MATEMÁTICA

Otimização da grade de horários do curso de Matemática da Universidade Federal Rural do Rio de Janeiro

Amanda Ferreira de Azevedo

Seropédica - RJ, 2019

Amanda Ferreira de Azevedo

Otimização da grade de horários do curso de Matemática da Universidade Federal Rural do Rio de Janeiro

Monografia apresentada à Banca Examinadora da Universidade Federal Rural do Rio de Janeiro, como requisito parcial para obtenção do título de Bacharel em Matemática Aplicada e Computacional sob orientação do Prof. Dr.Vinícius Leal do Forte

Agosto de 2019

"There is a purpose of life in the nerd world, which is treating reality as code, and optimizing it. Life becomes a problem-solving activity, and the problem is some sort of lack of optimization."

Ellen Ullman, Close to the Machine: Technophilia and Its Discontents

AGRADECIMENTOS

Agradeço aos meus pais, Silvana e Antônio pelo apoio, força e amor incondicional. A minha irmã, Ana Paula, que sempre acreditou em mim e me incentivou em todos os momentos, além de ser minha maior inspiração.

Aos meus amigos de graduação, especialmente Suellen de Melo, Erick Henrique, Mariane Silva e Matheus Lopes que compartilharam alegrias, angústias, conhecimentos, ideias, infinitas horas de estudo, copos de café e viradas de noite. Foi uma convivência maravilhosa e enriquecedora. Sem dúvidas foram fundamentais para minha formação, por isso merecem o meu eterno agradecimento.

Agradeço a Galere, por entenderem os momentos de ausência durante a graduação. Vocês nunca negaram uma palavra de apoio, força e cumplicidade ao longo dessa etapa em minha vida. Obrigada pela companhia durante esse processo.

Agradeço ao Prof. Vinícius Leal do Forte, pelo incentivo, orientação, seu grande desprendimento em ajudar e amizade sincera. Também sou grata a todos os professores que contribuíram com a minha trajetória acadêmica, principalmente aos professores Douglas Monsôres de Melo e Luciano Viana Félix, pois sempre acreditaram no meu potencial e souberam me amparar nos momentos mais difíceis.

E a todos que direta e indiretamente fizeram parte da minha formação, o meu muito obrigada.

RESUMO

No campo da otimização, a construção de grades horárias em universidades é um problema conhecido na literatura como *University Timetabling Problem* (UTP), que consiste em distribuir horários relacionados a turmas e disciplinas para cada professor no ensino superior. Sendo assim, este trabalho utilizará como referência o curso de Matemática da Universidade Federal Rural do Rio de Janeiro, campus Seropédica, propondo uma formulação de programação linear inteira para este local, respeitando suas especificidades e maximizando a preferência dos professores. Utilizando a implementação computacional, o trabalho apresenta resultados para o segundo semestre de 2018 e uma simulação de horários aos professores, evidenciando suas satisfações.

Palavras-chave: UTP; Otimização; Programação linear inteira; Grade de horários.

ABSTRACT

In the field of optimization, the construction of university schedules is a well-known problem in the literature as University Timetabling Problem (UTP), which consists of distributing class-related schedules and disciplines for each professor in higher education. Thus, this work will use as reference the Mathematics course of the Universidade Federal Rural do Rio de Janeiro, Seropédica campus, proposing a formulation of integer linear programming for this place, respecting its specificities and maximizing teacher's preference. Using computational implementation, the work presents results for the second semester of 2018 and a simulation of schedules for professors, showing their satisfactions.

Key-words: UTP. Optimization. Integer linear programming. Schedule.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

UFRRJ	-	Universidade Federal Rural do Rio de Janeiro
DEMAT	-	Departameto de Matemática
UTP	-	*University Timetable Problem*
STP	-	*School Timetable Problem*
POC	-	Problema de Otimização Combinatória
PPL	-	Problema de Programação Linear
PPI	-	Problema de Programação Inteira
PLI	-	Programação Linear Inteira

Conteúdo

INTRODUÇÃO

A distribuição de horários, dias e aulas para cada professor é uma tarefa árdua que se repete semestralmente. Apesar de todo avanço computacional, a maioria das universidades ainda utiliza métodos manuais para a construção desta grade, o que implica em grande esforço por parte da gestão, que enfrenta esse problema pelo menos duas vezes ao ano. A construção de uma grade de horários ineficiente pode acarretar em diversos problemas, que refletem diretamente nos discentes e docentes. Esse problema é conhecido, na literatura, como *University Timetable Problem*, definido por Wren (1995) como a alocação de objetos que existem de forma a satisfazer o máximo possível de objetivos desejáveis sujeita a restrições. Assim, para cada professor atribui-se uma ou mais disciplinas em uma semana, sujeito a condições que façam uma grade de horários ter sentido, o que aumenta a complexidade do problema. Durante o processo de construção da grade de horários, busca-se reduzir a insatisfação dos professores e, por consequência, a insatisfação dos alunos. Será utilizado como objeto de estudo, para o presente trabalho, o Departamento de Matemática da Universidade Federal Rural do Rio de Janeiro, campus Seropédica. Assim, com a formulação criada e fazendo uso da Programação Linear Inteira (PLI), o trabalho cria uma grade de horários semanal que relaciona professores com seus respectivos horários. Dessa forma, o trabalho busca maximizar a satisfação dos professores, utilizando ferramentas como formulários para obter os dados necessários. A motivação que impulsiona essa pesquisa se dá através do local de estudo. No Departamento de Matemática desta universidade, a geração de horários é manual, gerando grande esforço para a montagem de horários satisfatórios. De acordo com Murray (et al., 2016), poucos são os trabalhos que resolvem problemas de UTP desse tipo. Desta forma, a pesquisa se torna importante, pois digitalizando o processo, otimiza-se o tempo e a quantidade de funcionários envolvidos, diminuindo custos. Além de resolver um problema interno e muito pertinente, o presente trabalho servirá como proposta de implementação para os outros cursos da universidade e os resultados encontrados servirão para pesquisas futuras,

pois aperfeiçoa o conhecimento sobre UTP na literatura. Esse projeto teve como objetivo geral construir um modelo de geração de horários com o intuito de apoiar e propor novas metodologias aos administradores e gestores da universidade, para a otimização da construção de horários dos períodos. No Capítulo 1, faz-se uma breve introdução aos Problemas de Programação Linear, no intento de apresentar ao leitor conceitos básicos que serão necessários para o entendimento da pesquisa. No capítulo 2, são apresentados conteúdos de Programação Inteira, sob a ótica de modelagem, formulações e o método de *Branch-and-Bound*, que será utilizado para resolver o problema em questão. No Capítulo 3, encontra-se a caracterização de problemas de geração de horários (*scheduling*) na otimização combinatória, fazendo um breve resumo, utilizando referências bibliográficas. O Capítulo 4 ficou dedicado à apresentação da formulação do problema no âmbito da otimização inteira. Nas considerações finais são mostrados os resultados, uma perspectiva para futuros melhoramentos e possíveis trabalhos com relação a essa pesquisa.

1 PROGRAMAÇÃO LINEAR

Um Problema de Programação Linear (PPL) consiste em minimizar ou maximizar uma função linear, chamada função objetivo, sujeita a uma série de restrições, que serão inequações ou equações também lineares. O problema é dado, então, da seguinte forma:

$$\begin{aligned} &\text{min ou max} \quad z = \sum_{j=1}^{n} c_j x_j \\ &\text{sujeito a} \qquad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad i = 1, ..., m \\ &x \in \mathbb{R}^n \qquad x_j \geq 0 \qquad j = 1, ..., n \end{aligned} \tag{1.1}$$

onde $x \in \mathbb{R}^n$ é o vetor de variáveis do problema, c_j, a_{ij}, e b_i são constantes. Um programa linear desse tipo é dividido em duas componentes: a função objetivo e um conjunto de restrições.

Para melhor compreensão, segue o exemplo abaixo:

$$\begin{aligned} \text{min} \quad & 4x_1 - 2x_2 + x_3 \quad \text{(Função objetivo)} \\ \text{sujeito a} \quad & x_1 + x_2 + \quad +3x_4 \leq 7 \quad \text{(Restrição 1)} \\ & \quad x_2 - 4x_3 \quad \leq -3 \quad \text{(Restrição 2)} \\ & \quad x_2 \quad \geq 2 \quad \text{(Restrição 3)} \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Neste exemplo, deseja-se obter uma combinação de valores que respeitem estas restrições e, ao mesmo tempo, o menor valor possível na função objetivo. Observe que, nesse caso, $x \in \mathbb{R}^4$. Pode-se verificar que existem restrições de desigualdades e de igualdades. Desta forma, reescreve-se o problema de forma a ter apenas desigualdades

de mesma natureza, isto é, desigualdades equivalentes, bastando multiplicar por -1.

Desta forma, pode-se observar que é possível escrever esse conjunto de restrições do exemplo a partir de notação matricial, sendo:

$$A = \begin{bmatrix} 1 & 1 & 0 & 3 \\ 0 & 1 & -4 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix} e \quad b = \begin{bmatrix} 7 \\ -3 \\ -2 \end{bmatrix}.$$

Reescreve-se o PPL (1.1), generalizando da seguinte forma:

$$\begin{array}{lll} \text{min ou max} & z = cx & \\ \text{Sujeito a} & Ax \leq b & \\ x \in \mathbb{R}^n & x_j \geq 0 & j = 1, ..., n \end{array} \tag{1.2}$$

Onde $A \in \mathbb{R}^{mxn}$, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ e $x \in \mathbb{R}^n$ o vetor que possui as variáveis de decisão do problema. Assim, segue a seguinte definição de Maculan (2004):

Definição 1: Seja X = $\{x \in \mathbb{R}^n | Ax = b, x \geq 0\}$. O conjunto X é denominado *conjunto ou região viável* do (PPL) e, se $x \in X$, então x é *solução viável* do problema. Dado $x^* \in X$, x^* é denominado uma *solução ótima* do (PPL), de maximização, se $cx^* \geq cx$, $\forall x \in X$ e se o problema for de minimização, vale $cx^* \leq cx$, $\forall x \in X$.

1.1 Solução gráfica de um PPL

Aqui, comenta-se sobre os aspectos gráficos da solução de um PPL, uma vez que proporciona uma melhor visualização dos conceitos acima citados. Considere o exemplo a seguir:

$$\max \quad z = 2x_1 + 4x_2$$

Sujeito a

$$0.5x_1 + 0.75x_2 \leq 2$$

$$x_1, x_2 \geq 0 \quad x_1, x_2 \in \mathbb{R}^2$$

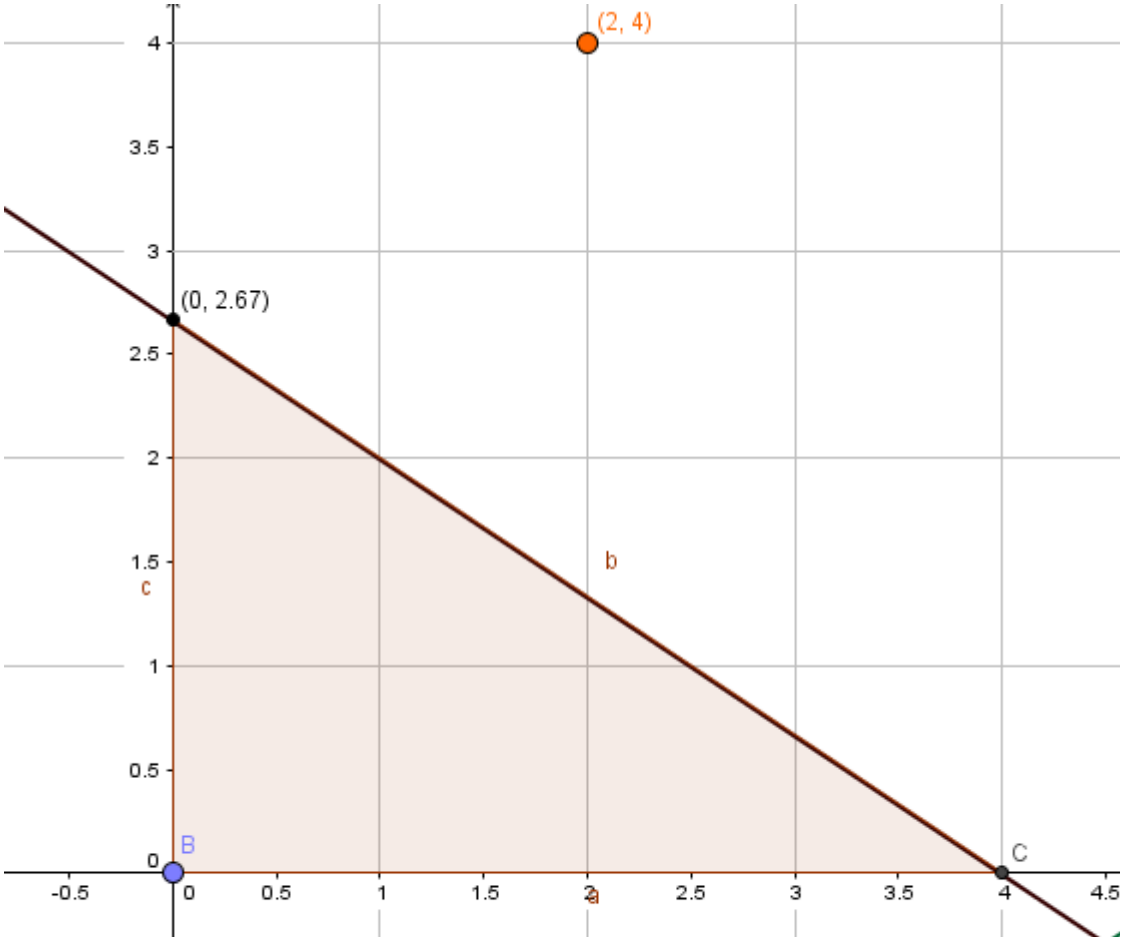

Figura 1.1: Geogebra: Solução gráfica para o PPL

Ao se tomar a interseção de todas as restrições do problema, tem-se a região viável, que está hachurada na Figura 1.1. Note que, o vetor (2,4) é o vetor gradiente da função objetivo, para todo x, neste exemplo. Já que a função objetivo é linear, todas as soluções ótimas se encontram nas extremidades do polígono; assim, traçam-se as curvas de nível da função objetivo, que serão, no geral, hiperplanos perpendiculares

ao gradiente, na direção de maior crescimento da função objetivo, até atingir a última extremidade do polígono, sendo assim, um único vértice ou uma aresta. Observa-se, na Figura 1.1 que é possível encontrar o valor ótimo no ponto (0, 2.67), com z = 10,68.

1.2 Algoritmo Simplex

De acordo com Luenberger (1973), a ideia do algoritmo simplex é partir de uma solução básica viável, a partir das restrições, caminhar pelos vértices, até que o mínimo (ou máximo) seja obtido, ou que se mostre ilimitado. Simplex é um algoritmo criado por George Bernard Dantzig, em 1947, amplamente utilizado em PPL com variáveis contínuas.

Seja o PPL padrão:

$$\begin{array}{lll} \text{max} & z = cx & \\ \text{Sujeito a} & Ax \leq b & \\ x \in \mathbb{R}^n & x_j \geq 0 & j = 1, ..., n \end{array} \tag{1.3}$$

Particionando A = (B N), $x = (x_b, x_N)$ e c = (c_b, c_N), de tal forma que B será uma matriz quadrada mxm denominada *base*, invertível, e x_b e c_b possuem m componentes relacionadas a B. Com isso, pode-se reescrever o PPL da seguinte forma:

$$max\ z = c_b x_b + c_N x_N \tag{1.4}$$

Assim, escreve-se explicitando x_b, a seguinte equação:

$$x_b = B^{-1}b - B^{-1}Nx_N$$

Dessa forma, segue a definição de Maculan (2004):

Definição 2: x é uma *solução básica* do PPL padrão se $x^t = (x_b^t 0)$. As variáveis associadas às componentes x_b são denominadas básicas e as demais não básicas. Quando x_b possuir ao menos uma componente nula, diremos que x é uma *solução básica degenerada*.

Definição 3: Se a solução não possui coordenadas negativas, esta é chamada de *solução básica viável*. E substituindo a expressão de x_b acima citada na equação (1.3), há outra forma de escrever este PPL:

$$max\ z = c_b(B^{-1}b - B^{-1}Nx_N) + c_N x_N = c_b B^{-1}b - (c_b B^{-1}N - c_N)x_N \qquad (1.5)$$

sujeito a:

$$x_b = B^{-1}b - B^{-1}Nx_N, \quad x_b \geq 0, x_N \geq 0$$

A ideia do algoritmo Simplex consiste em algumas etapas, de acordo com Ricardo Augusto Trindade (2015):

1) Achar uma solução viável;
2) Verificar se há vértices adjacentes viáveis;
3) Existindo um vértice adjacente que melhor otimiza a função objetivo, haverá troca de uma variável básica por uma não básica, modificando as colunas de B e N.
4) Testar se a solução atual é ótima, caso contrário, voltar ao passo 2.

Esse teste, na quarta etapa, consiste em testar se a base produz a solução ótima. Isso acontece se $x_B = B^{-1}b \geq 0$ e $c_b B^{-1}N - c_N \leq 0$, pela equação (1.5); esse é o teste de otimalidade. A demonstração pode ser encontrada em Maculan (2004). Caso não seja ótimo, primeiramente verifica-se qual variável entrará na base e depois qual sairá da base.

2 PROGRAMAÇÃO INTEIRA

Ao formular problemas de otimização, é comum encontrar problemas que envolvem contagem de elementos (quantidade de carros, lotes, equipamentos, pessoas etc.) ou problemas que envolvam decisões (sim ou não, por exemplo), onde as variáveis assumem valores binários. Isto é, problemas com domínios discretos. Assim, faz-se necessário o uso da Programação Inteira (PI). Utiliza-se aqui apenas o caso linear, onde maioria das aplicações de (PI) se encontram. Define-se Programação Linear Inteira (PLI) como:

$$\begin{array}{lll} \text{min ou max} & z = cx & \\ \text{Sujeito a} & Ax \leq b & \\ x \in \mathbb{Z}^n & x_j \geq 0 \qquad & j = 1, ..., n \end{array} \tag{2.1}$$

sendo $A \in \mathbb{R}^{mxn}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$ e $x \in \mathbb{R}^n$.

Perceba que o (PLI) é um caso particular de um (PPL), onde as variáveis de decisão são inteiras, isto é, restringe-se o domínio aos números inteiros. Considere o mesmo exemplo, do capítulo anterior:

$$\begin{array}{l} \text{max} \quad z = 2x_1 + 4x_2 \\ \text{Sujeito a} \\ 0.5x_1 + 0.75x_2 \leq 2 \\ x_1, x_2 \geq 0 \quad x_1, x_2 \in \mathbb{R}^2 \end{array}$$

Se utilizarmos a abordagem gráfica, tem-se:

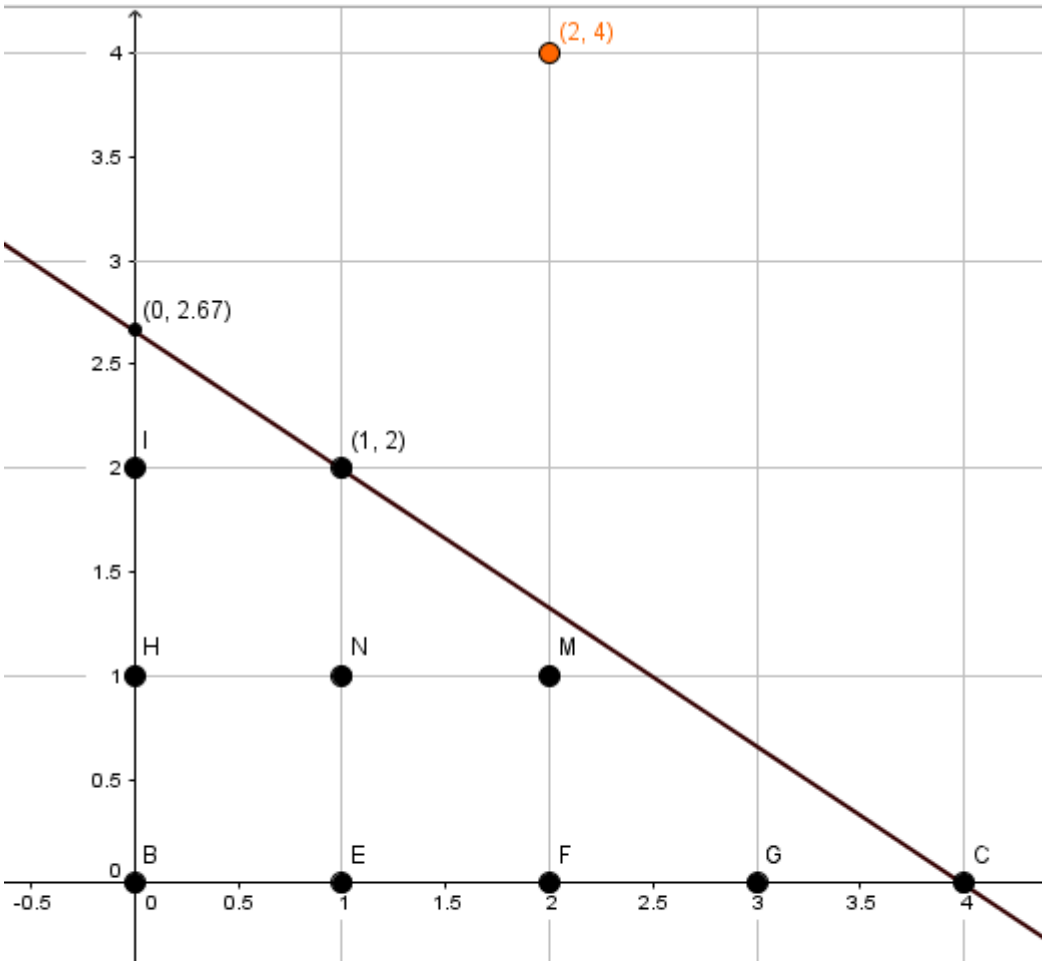

Figura 2.1: Geogebra: Solução gráfica para o PLI

Repare que o domínio, agora, é inteiro. O que faz com que a solução deste problema seja z = 10, no ponto (1,2).

Outras variantes de PLI:

Problema de Programação Inteira Mista (PPIM): Quando apenas algumas variáveis assumem valores inteiros.

$$\begin{array}{ll} \text{min ou max} & z = c^T x + h^T y \\ \text{Sujeito a} & Ax + Gy \leq b \\ x \in \mathbb{Z}^n, y \in \mathbb{R}^p & x \geq 0, y \geq 0 \end{array} \tag{2.2}$$

onde $G \in \mathbb{R}^{mxp}$, $h \in \mathbb{R}^p$, $y \in \mathbb{R}^p$

Problema de Programação Binária (PPB): Quando todas as variáveis do problema assumem valores binários, isto é, assumem valores no intervalo {0,1}.

$$\begin{aligned} &\text{min ou max} \quad z = c^T x \\ &\text{Sujeito a} \quad Ax \leq b \\ &x \in \{0,1\}^n \end{aligned} \tag{2.3}$$

2.1 Modelagem e Modelos

O processo de descrever matematicamente um fenômeno real é chamado modelagem matemática. De acordo com Marcos C. Goldbarg (2000), existe um processo para a construção de um modelo, ele consiste em:

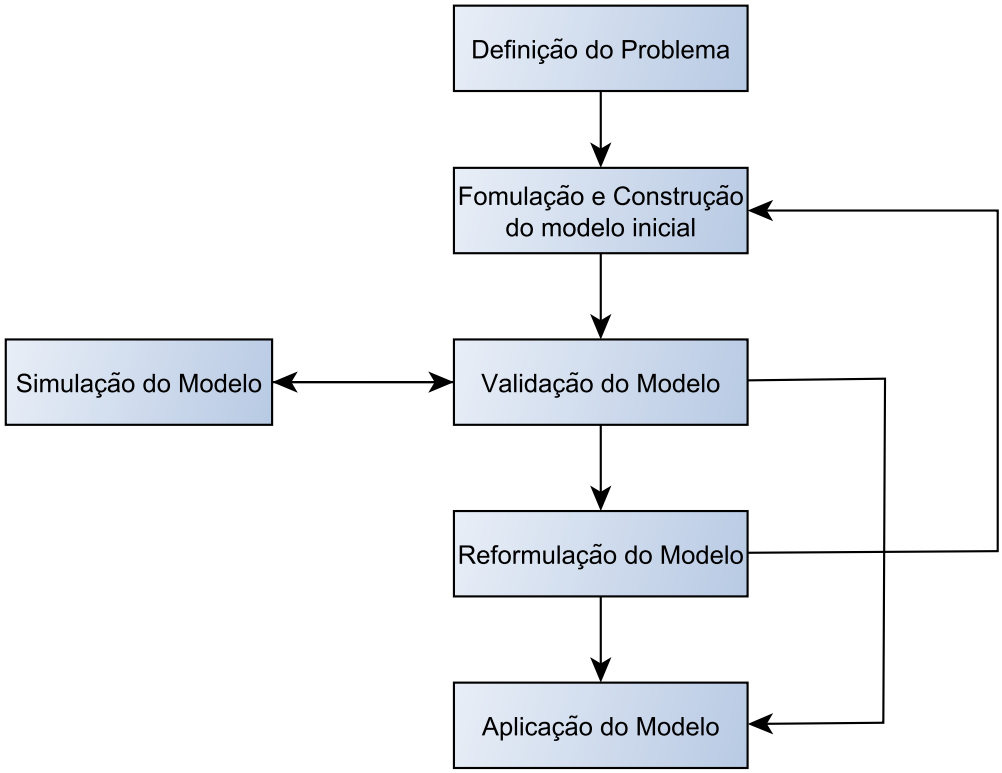

Figura 2.2: Esquema de Modelagem

Fonte: Marcos C. Goldbarg (2000)

Uma formulação, no âmbito da Programação Inteira, é um modelo matemático que segue os seguintes passos:

1) Definição do objetivo do problema;

2) Definição das variáveis de decisão envolvidas

3) Conhecimento das restrições a que está sujeito o problema.

Abaixo, serão apresentados dois modelos clássicos de PLI, que são problemas importantes e servem como base para outros modelos.

2.1.1 Problema de Atribuição

Mais conhecido na literatura como *Assignment Problem*. Consiste em relacionar pessoas a tarefas, lugares etc. Suponha que se deseja relacionar n indivíduos a n tarefas, onde definimos um custo c_{ij} de eficiência dessa relação (pode representar o lucro, por exemplo). Assim, segue a formulação:

$$x_{ij} \in \{0,1\} = \begin{cases} 1, \text{ se o indivíduo i for designado a tarefa j} \\ 0, \text{ caso contrário} \end{cases}$$

$$\begin{aligned} &\text{min ou max} \quad z = \sum_{i=1}\sum_{j=1} c_{ij}x_{ij} \\ &\text{Sujeito a} \qquad \sum_{j=1} x_{ij} = 1, \quad i = 1,...,n \quad \text{(Restrição 1)} \\ &\qquad\qquad\quad \sum_{i=1} x_{ij} = 1, \quad j = 1,...,n \quad \text{(Restrição 2)} \\ &x \in \mathbb{Z}^n \qquad x_{ij} \geq 0 \qquad j = 1,...,n \quad i = 1,...,n \end{aligned} \tag{2.4}$$

(Restrição 1) Cada indivíduo só pode estar designado a apenas uma tarefa.

(Restrição 2) Cada tarefa só pode estar designada a apenas um indivíduo.

2.1.2 Problema da Mochila

Este problema é caracterizado pela seguinte situação: um indivíduo dispõe de uma mochila e de vários objetos que podem estar nela. É conhecido o peso p_i e o preço v_i de cada objeto, além da capacidade máxima da mochila C. Busca-se escolher um conjunto de objetos dentre os n existentes de forma a maximizar o proveito com relação ao custo e peso.

Assim, define-se a seguinte formulação:

$$x_{ij} \in \{0,1\} = \begin{cases} 1, \text{ se o objeto i for escolhido para estar na mochila} \\ 0, \text{ caso contrário} \end{cases}$$

$$\begin{aligned} &\text{max} && z = \sum_{i=1}^{n} v_{ij} x_{ij} \\ &\text{Sujeito a} && \sum_{i=1} p_i x_{ij} \leq C, \quad i = 1, ..., n \end{aligned} \tag{2.5}$$

Abaixo, pode-se ver um esquema de Der-San Chen (2011), para visualizar melhor os conceitos anteriormente apresentados.

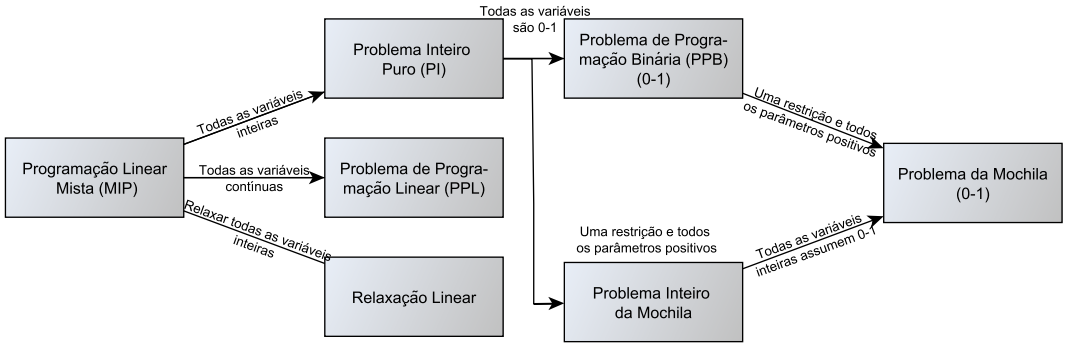

Figura 2.3: Fluxograma de problemas de programação

Fonte: Der-San Chen (2011)

Define-se um Problema de Otimização Combinatória (POC) como sendo os problemas que se caracterizam por possuírem um domínio discreto e, além disso, a solução surge na seleção de um conjunto finito de combinações entre os elementos. Exemplos de POC são os problemas anteriormente mostrados, como o Problema de Atribuição e o Problema da Mochila.

2.1.3 Formulação

Agora que se sabe como trazer um problema da vida real para o contexto da otimização, é natural estudar a qualidade das formulações e o que é de fato uma formulação válida. Como existem diversas formulações para um mesmo problema, pode-se pensar "qual será a melhor para um problema?" Abaixo, apresentam-se algumas definições que explicarão isso.

As definições (1), (2), (3), (4) e (5) podem ser encontradas em Laurence Wolsey (1998); bem como a definição (6) pode ser encontrada em Der-San Chen (2011).

Definição 1: Um subconjunto $P \subset \mathbb{R}^n$ é chamado um *poliedro* ou *politopo* se for definido por um número finito de restrições lineares, isto é:

$$P = \{x \in \mathbb{R}^n | Ax \leq b\}$$

Definição 2: O problema $z^R = \max\{f(x) : x \in T \subseteq \mathbb{R}^n\}$ é uma *relaxação* do (PI) $z = \max\{c(x) : x \in X \subseteq \mathbb{R}^n\}$ se:

- $X \subseteq T$, e
- $f(x) \geq c(x), \forall x \in X$.

Definição 3: O poliedro P $\subseteq \mathbb{R}^{n+p}$ é uma *formulação* de um conjunto $X \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ se $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$.

Definição 4: Dado um conjunto $X \subseteq \mathbb{R}^n$, a *envoltória convexa* de X, denotada por conv(X) é definida por: $conv(X) = \{x | x = \sum_{i=1}^{t} \lambda_i x^i \ , \sum_{i=1}^{t} \lambda_i = 1 \ , \lambda_i \geq 0$, para todo i = 1,...,t sob todos os subconjuntos finitos $\{x^1, x^2, ..., x^t\}$ de X$\}$.

Definição 5: Dado um conjunto de $X \subseteq \mathbb{R}^n$ e duas formulações P_1 e P_2 para X, P_1 é uma formulação melhor que P_2 se $P_1 \subset P_2$.

Definição 6: Seja $S = \{y : Gy < b \ , \ y$ inteiro$\}$. Uma formulação de S é ótima se todos os pontos extremos do poliedro forem inteiros.

2.2 Métodos de Solução

De acordo com Marcos C. Goldbarg (2000), as técnicas mais utilizadas para encontrar solução inteira são divididas em três classes:

Enumeração: *Branch-and-Bound* (BB), Enumeração Implícita;

Cortes: Cortes combinatórios, inteiros e de interseção, método de decomposição de *Benders*;

Híbridos: *Branch-and-Cut* e Teoria de Grupo.

Para a resolução do problema proposto, a próxima seção será dedicada ao detalhamento mais específico do método de *Branch-and-Bound*.

2.2.1 Branch-and-Bound

Foi apresentado um algoritmo denominado "Branch and Bound"(BB), por Leid, em 1960, para a resolução de problemas mistos. De acordo com Marcos C. Goldbarg (2000),

> "O método denominado de *Branch-and-Bound* (BB) baseia-se na ideia de desenvolver uma enumeração inteligente dos pontos candidatos à solução ótima inteira de um problema. O termo *branch* refere-se ao fato de que o método efetua partições no espaço das soluções. O termo *bound* ressalta que a prova da otimalidade da solução utiliza-se de limites calculados ao longo da enumeração."

Também, de acordo com Teixeira (2011), o algoritmo de BB é, atualmente, o procedimento mais utilizado para a resolução de problemas de programação linear inteira.

Neste capítulo, será apresentado o algoritmo dado, que será o método de resolução do problema em questão. A ideia do método é dividir o problema inicial em subproblemas mais fáceis de resolver, de forma a criar subconjuntos disjuntos. Laurence Wolsey (1998, p. 91) define isso como "dividir e conquistar". Deve-se definir o conceito de limitante superior e limitante inferior.

Seja um (POC) ou (PI): $\{\max\ z = cx : x \in \mathbb{Z}^n\}$. Um limitante superior z_{sup} é da forma que, para qualquer z, $z \leq z_{sup}$. De forma análoga, um limitante inferior z_{inf} se caracteriza como, para qualquer z, $z \geq z_{inf}$. Isso significa então que $z_{inf} \leq z \leq z_{sup}$. Diz-se que o problema é **ótimo** se, dado $\epsilon > 0$, $|z_{sup} - z_{inf}| < \epsilon$. Isto é, é ótimo quando a distância entre os limites é tão pequena quanto a escolha do ϵ, dando ideia de grande proximidade.

Dessa forma, enuncia-se as proposições (2.1) e (2.2) retiradas de Laurence Wolsey (1998):

Proposição 2.1: Seja S = $S_1 \cup S_2 \cup S_3 \cup ... \cup S_K$ uma decomposição de S em conjuntos menores e tome $z^k = max\{cx : x \in S_K\}$, para k=1,.., K. Então z = max_k z^k.

Demonstração: O problema inicial é dado por z = max{cx: $x \in S$}. Seja x^* a solução deste problema em S. Ora, se x^* está em S, o mesmo estará na união dos subconjuntos, o que significa que estará em algum subconjunto S_K. Cada subconjunto irá ter o seu máximo e o máximo de todos esses valores será exatamente x^*, pois não pode haver ninguém maior que ele na união, que é o máximo do próprio S. ■

Proposição 2.2: Seja S = $S_1 \cup S_2 \cup S_3 \cup ... \cup S_k$ uma decomposição de S em pequenos conjuntos e tome $z^k = max\{cx : x \in S_k\}$, para k=1,.., K. Seja z^k_{sup} um

limite superior em z^k e seja z^k_{inf} um limite inferior em z^k. Assim, $z_{sup} = max_k\ z^k_{sup}$ é um limite superior de z e $z_{inf} = max_k\ z^k_{inf}$ é um limite inferior de z.

Demonstração: Assim como a demonstração anterior, dado um limitante superior no conjunto S, este será um limitante superior para a união de seus subconjuntos, o que implica que será um limitante superior em cada um dos subconjuntos. Ora, tomando o máximo de todos os limitantes superiores dos subconjuntos, este só poderá ser o próprio limitante superior de S, pois não há ninguém maior que ele por definição. De forma análoga, tomando o mínimo de todos os limitantes inferior dos subconjuntos, este só poderá ser o próprio limitante inferior de S. ■

Sejam P_1 e P_2 duas formulações:

$P_1 = \max\ \{cx | Ax \leq b,\ x \geq 0,\ x \in \mathbb{Z}^n\}$

$P_2 = \max\ \{cx | Ax \leq b,\ x \geq 0,\ x \in \mathbb{R}^n\}$

Com a troca da restrição de $x \in \mathbb{Z}^n$ por $x \in \mathbb{R}^n$, pode-se afirmar que P_2 é chamado **relaxação linear** de P_1.

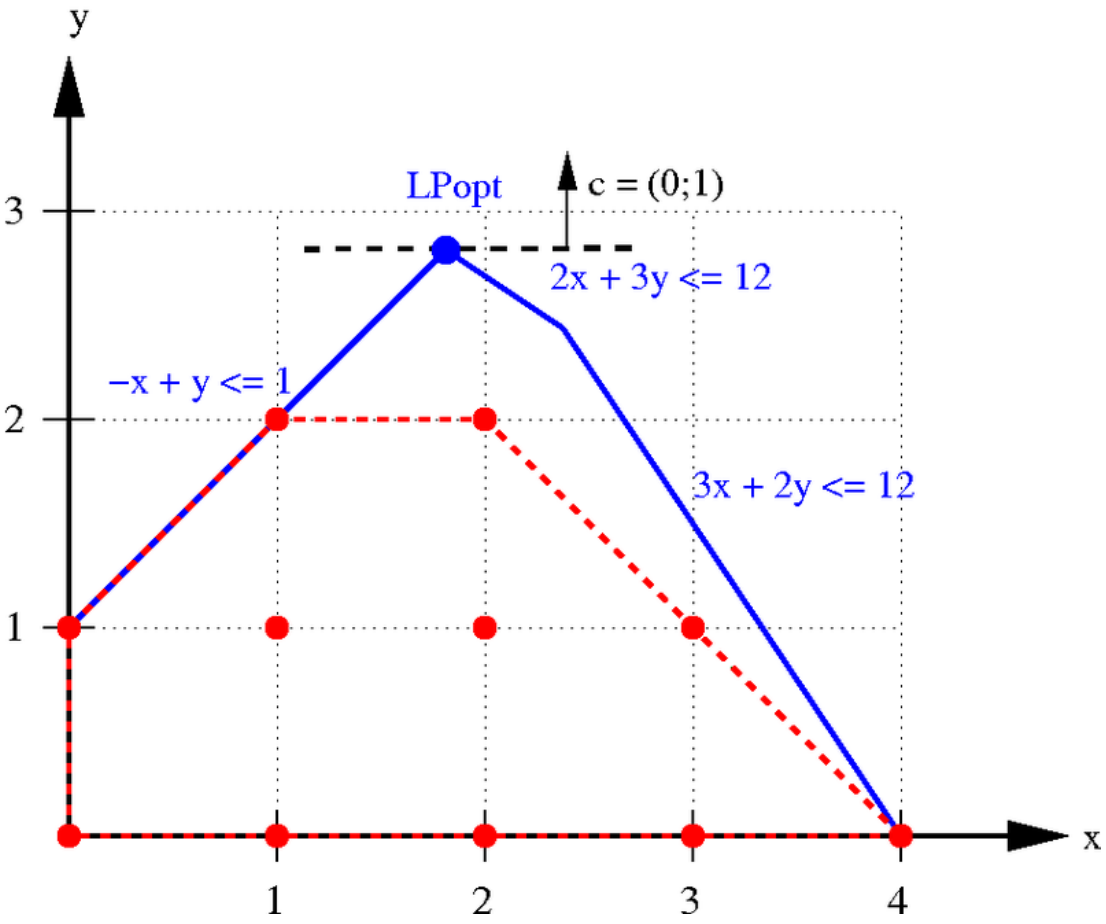

Figura 2.4: Wikipédia: ***"Linear programming relaxation"***

Observe, na Figura 2.4 que a parte azul se refere a relaxação linear e a vermelha, a região viável do problema inteiro. Assim, suponha que x_1 seja o ótimo de P_1 , x_2 o ótimo de P_2, e x qualquer solução viável de P_1 tem-se a seguinte relação:

$$cx \leq cx_1 \leq cx_2$$

Chama-se cx_2 de um **limitante dual** de P_1. E cx, que é qualquer solução viável, é dita um **limitante primal** de P_1.

2.2.2 Árvores

Na figura abaixo, tem-se um exemplo de árvore enraizada de enumeração:

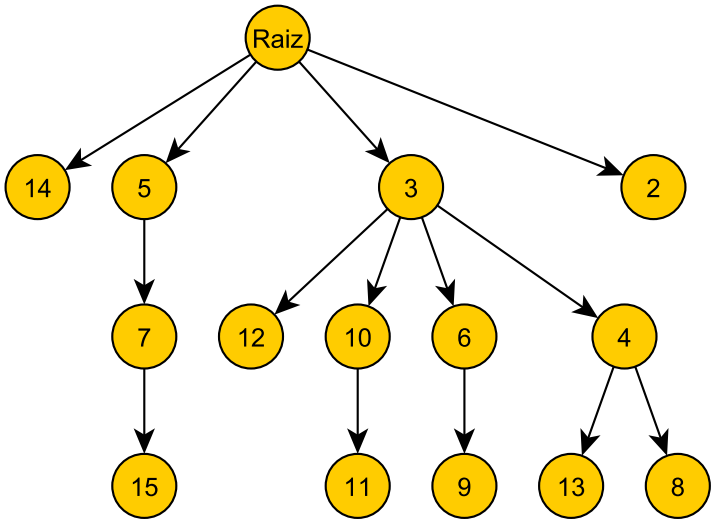

Figura 2.5: Árvore enraizada de Enumeração

Observe que os vértices estão enumerados, isto é, possuem uma ordem para serem visitadas a partir da raiz. Define-se cada vértice dessa árvore como um nó. Neste caso, trata-se o método de (BB) com a característica de enumeração implícita, isto é, gera-se todas as soluções possíveis mas sem explicitar elas.

Este trabalho apresenta um algoritmo para o (BB), na perspectiva da programação linear, utilizando busca em largura, isto é, examina-se todos os vértices a partir de uma estrutura de fila para visitação dos nós. Com isso, é natural discutir sobre os critérios que levam ao algoritmo parar. Laurence Wolsey (1998) define 3 tipos de critérios de parada para o Branch-and-Bound e lista dois exemplos, isto é, critérios para podar os ramos das árvores.

- Podado por otimalidade: $z^t = \{max\ cx : x \in S^t\}$ for resolvido.
- Podado por limitante: $z^t_{sup} \leq z_{inf}$.
- Podado por inviabilidade: $S^t = \varnothing$.

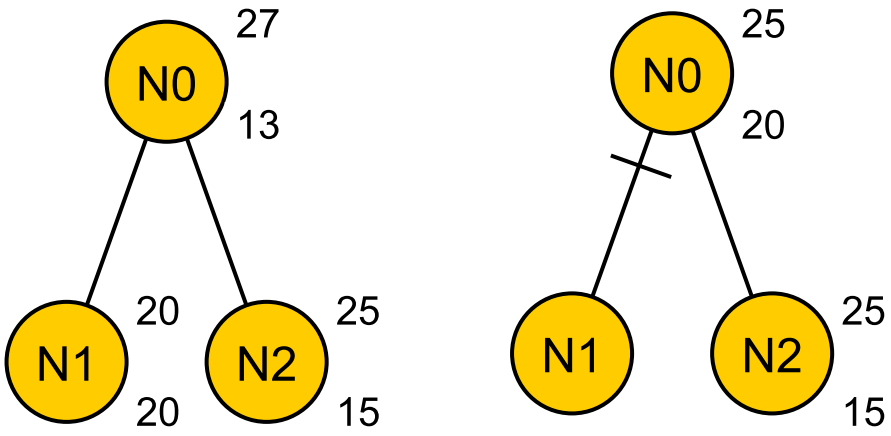

Figura 2.6: Podado por otimalidade

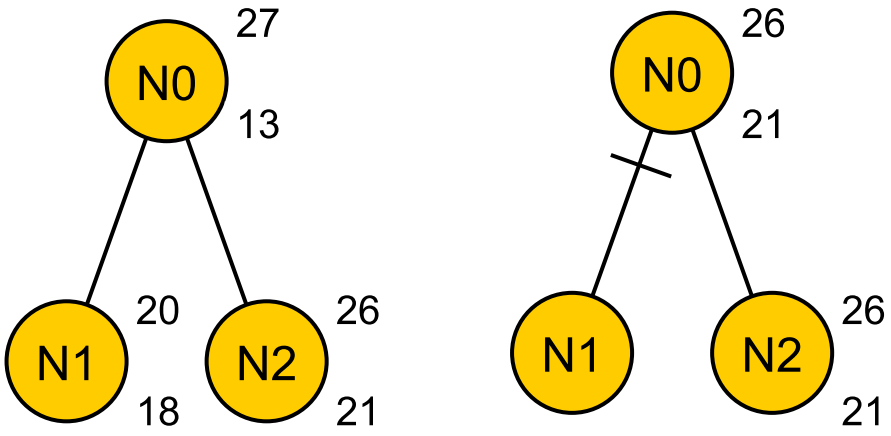

Figura 2.7: Podado por limite

Dado uma formulação P: $\{max\ z = cx : x \in S \subset \mathbb{Z}^n\}$, k sendo o número de iterações e S^k subconjuntos de S com formulação P^k.

Seja z_{inf} o limitante primal de z e z_{sup} limitante dual de z, N^k denotado pelo nó na iteração k e L definido pela lista de nós que ainda precisam ser visitados.

Algoritmo 1 BRANCH-AND-BOUND

início

inicialização

k = 0; $z_{inf} = -\infty$; $x_*^0 = 0$; $S^0 = S$;

Passo 1:

branch = true;

Escolha o nó N^k com a formulação P^k.

Resolva o problema relaxado P^k definido por: $\{max\ z : x \in S^k \subset \mathbb{R}^n\}$

se P^k *tiver solução* **então**

$z_{sup}^k = z_*^k$.

se x_*^k *for inteiro* **então**

$z_{inf}^k = z_*^k$

fim

fim

$z_{sup} = max\{z_{sup}^i; i = 0, ..., k\}$

$z_{inf} = max\{z_{inf}^i; i = 0, ..., k\}$

se $S^k = \varnothing$ **então**

branch = false;

O problema P^k não tem solução, pode o nó N^k por inviabilidade.

fim

se $z_{sup}^k \leq z_{inf}$ **então**

branch = false;

pode o nó N^k por limitante.

fim

se $z_{sup}^k = z_{inf}^k$ **então**

branch = false;

pode o nó N^k por otimalidade.

fim

Passo 2:

Faça k = k+1;

se *branch = true* **então**

Faça a separação do problema P^k, isto é, defina os problemas:

$$P^{2k-1} : \{max\ z : \{x \in S^{2k-1} | S^{2k-1} = S^{k-1} \cap \{x_r \in \mathbb{R} | x_r \leq \lfloor x_r^k \rfloor\}\}\}$$

$$P^{2k} : \{max\ z : \{x \in S^{2k} | S^{2k} = S^{k-1} \cap \{x_r \in \mathbb{R} | x_r \leq \lfloor x_r^k \rfloor + 1\}\}\}$$

(Onde x_r^k é uma componente não inteira de x^k)

Adicione a L os nós relativos aos problemas P^{k-1} e P^{2k}.

fim

se $L = \varnothing$ **então**

pare; $x_* = max\{x_*^i; i = 0, ..., k\}$ será a solução do problema P.

fim

Vá para o Passo 1.

fim

3 GERAÇÃO DE HORÁRIOS

Define-se *University Timetable Problem* (UTP) como a criação de grades de horários semanais para estudantes e professores de uma universidade, de forma a buscar uma grade viável, dentro de suas restrições. Na literatura, vários tipos de problemas de geração de horários são encontrados: esportes, ferrovias, projetos, ônibus, voos, enfermaria, universidades, escolas, dentre outras inúmeras variantes. Esse problema tem sido estudado desde 1950 por Tripathy (1984), gerando inúmeras formas de formulação, solução e, principalmente, vários tipos de geração de horários. De acordo com Tomáš Müller (2002), este problema consiste em alocar atividade a recursos, em uma janela de tempo. Problemas desse tipo, usualmente, relacionam alunos, professores e salas de aula. Com relação, especificamente, à geração de horários no ambiente acadêmico, tem-se STP (*School Timetable Problem*) e UTP. A principal diferença é que, nas universidades, os cursos podem ter estudantes em comum. Outra coisa que difere bastante é que no STP vê-se professores com cargas horárias pesadas, turmas pré-definidas e poucas opções de disciplinas. No UTP nota-se que as cargas são mais leves, com grande opção de disciplinas e turmas diversificadas. Laporte (1997) sugere características do STP e do UTP, conforme mostrado na tabela 3.1.

Tabela 3.1: STP x UTP

Características	STP	UTP
Agendamento	Por turma	Por aluno
Escolha	Poucas Opções Grades Curriculares rígidas	Muitas disciplinas eletivas Grades Curriculares flexíveis
Disponibilidade de Professores	Apertada (Carga Horária Pesada)	Flexível (Carga Horária Leve)
Locais de Aula	Poucos Locais Mesma Capacidade Localização Centrada	Muitos Locais Variedade de Capacidade Localização Descentralizada
Carga Horária do Estudante	Pesada (ocupada durante todo o dia)	Esparsa (várias lacunas de horários)
Restrições	Sem qualquer tipo de conflito	Um mínimo de conflitos é aceitável

3.1 Conceitos Importantes

Faz-se essa seção pois no decorrer do trabalho haverá a utilização de termos que serão melhor compreendidos se detalhados aqui. Assim, para tal, define-se a estrutura de um UTP. A **grade de horários**, em universidades, em geral, possui disciplinas, que são distribuídas em dias da semana, em algum intervalo de tempo, que são repetidas durante um semestre. Uma **aula** é a interseção entre um dia da semana e um intervalo de tempo, em que esta aula é ministrada por um professor, unicamente. Um **intervalo de tempo** é a janela de tempo na qual uma aula é ministrada. Possui um período de tempo com início e fim, podendo estar em três turnos distintos: manhã, tarde ou noite. Uma **disciplina** é um componente da matriz curricular, que possui um determinado número de aulas que são distribuídas durante a semana, ministrado por um professor. Uma grade de horários, nesse sentido, será uma matriz em que as linhas são os intervalos de tempo e as colunas, os dias da semana. Cada componente dessa matriz será

caracterizado por uma aula, em que a disciplina é ministrada pelo professor. Na tabela 3.2, mostra-se um exemplo de grade de horários fixando um professor:

Tabela 3.2: Grade de Horários semanal de um professor

x	Seg	Ter	Qua	Qui	Sex
8:00 - 10:00	AULA1	-	AULA1	-	AULA1
10:00 - 12:00	-	-	-	-	-
13:00 - 15:00	AULA2	-	AULA2	-	-
15:00 - 17:00	-	-	-	-	-
18:00 - 20:00	-	-	-	-	-
20:00 - 22:00	-	-	-	-	-

elaborado pelo autor

Repare que, como se fala de um problema de geração de horários de universidade, a matriz terá muitos espaçamentos entre as aulas. Desta forma, estes horários se repetirão durante todo período semestral. Esse problema é classificado como NP-hard (PONGCHAROEN et al., 2008), o que significa que requer uma grande quantidade de recursos computacionais. Em algumas instâncias, essa classe de problemas não consegue ser resolvido com algoritmos de tempos polinomiais.

Assim, Bucco (2014) monta uma tabela dos métodos mais utilizados na literatura para encontrar soluções para problemas com essa natureza, representada na Figura 3.1.

Método	Frequência
Programação linear inteira	14
Algoritmo genético	14
Simulated annealing	13
Busca tabu	12
Busca local	6
Constraint programming	6
Heurísticas desenvolvidas pelos autores	5
Great deluge	4
Outros	55

Figura 3.1: Métodos de solução mais utilizados na literatura

Fonte: Bucco (2014)

PLI é o único método exato, de resto, todos são métodos heurísticos, isto é, métodos que produzem soluções viáveis de qualidade. Observe que muitos autores utilizam de métodos heurísticos para resolução, pela natureza difícil do problema. Nas próximas seções, serão apresentados os conceitos de restrição forte e restrição fraca, no contexto da UTP.

3.2 Restrições Fortes

As restrições fortes são restrições que não podem ser violadas. Em outras palavras, se forem ignoradas, os horários não funcionam. Para viabilizar o problema, estas restrições precisam ser atendidas. A tabela 3.3, feita por Mirhassani e Habibi (2013), mostra algumas restrições fortes para o problema geral de UTP, que são geralmente utilizadas.

Tabela 3.3: Restrições Fortes

A capacidade de salas é limitada
Uma turma não pode ter a mesma disciplina por mais de dois períodos por dia
Uma turma não pode ter uma aula com mais de um professor
Um professor somente pode ministrar uma aula por vez
Cada aula tem exatamente a duração de um período
Os estudantes somente podem ter uma aula de cada vez
Cada professor deve ministrar um número específico de aulas por semana
Uma sala de aula somente pode ser usada para uma aula de cada vez
Aulas podem ser trancadas, se requerido
Aulas podem ser pré-agendadas para um período específico
A indisponibilidade dos professores é considerada
As salas alocadas devem ser grandes o suficiente para acomodar os estudantes
Apenas um tipo de aula, para uma disciplina específica, é permitido
Aulas duplas devem ser alocadas em dois períodos consecutivos
Pode haver um conjunto de requerimentos de precedência determinando que certos eventos devam ocorrer antes de outros

Traduzido por Bucco (2014)

Fonte: Mirhassani e Habibi (2013)

3.3 Restrições Fracas

As restrições fracas são aquelas que não são necessárias para que o problema possua solução viável aplicável ao problema, mas se você quiser uma boa grade de horários, elas, certamente, serão uma boa inclusão. Muitas restrições fracas de UTP são relacionadas aos *feedbacks* dos alunos, sendo assim, elas não precisam ser válidas para o funcionamento, mas se forem, deixarão o modelo mais satisfatório. De acordo com Woods e Trenaman (1999), a satisfação de restrições fracas é o que faz um ho-

rário válido melhor que outro. A tabela 3.4 mostra algumas restrições fracas para um problema geral de UTP, feita por Mirhassani Habibi (2013).

Tabela 3.4: Restrições Fracas

Algumas aulas requerem recursos de salas de aulas específicas
Os estudantes não podem ter períodos dispersos com aula única
Conflitos entre disciplinas opcionais escolhidas pelo estudantes devem ser evitados
Professores podem especificar períodos nos quais preferem não dar aula
Algumas aulas não devem acontecer tarde, no fim da noite
Grades horárias de professores devem evitar períodos desocupados
Salas devem ser, se possível, totalmente preenchidas
As grades horárias das salas os mais compactas possíveis
Disciplinas devem ser dispersas uniformemente ao longo da semana
Um intervalo de almoço deve ser agendado entre 12h e 14h
Os alunos tem uma lista de disciplinas preferidas
A carga de aulas de um professor deve ser observada
O número de aulas por dia pode ser limitado
O número de estudantes almoçando num determinado período deve ser controlado
Salas devem ser próximas ao seu departamento
Os estudantes devem ter aulas consecutivas no mesmo período
Turmas devem ter aulas nas manhãs ou nas tardes
Algumas turmas podem se dividir em grupos menores (para seminários, laboratórios)
Todas as aulas de uma disciplina devem ser agendadas na mesma sala e no mesmo período, mas dias diferentes
Algumas aulas são oferecidas com tutoriais, laboratórios ou ambos
Algumas disciplinas tem mais de um professor
Alguns grupos de disciplinas devem ser alocados em períodos específicos

Traduzido por Bucco (2014)

Fonte: Mirhassani e Habibi (2013)

Assim como as restrições, pode-se definir uma série de objetivos que os autores consideram na hora de formular o problema de timetable. Na tabela 3.5, criado por Mirhassani e Habibi (2013), pode-se ver isso.

Tabela 3.5: Funções objetivos usualmente utilizadas

Maximizar a preferência dos estudantes
Minimizar o número total de disciplinas não alocadas
A grade de horários obtida deve ser o mais compacta possível
Minimizar o número de estudantes sem assentos em sala de aula
Minimizar o número de alocações distintas de disciplinas-aula
Maximizar a alocação de disciplinas tidas como urgentes
Minimizar a ocorrência de encontros de disciplinas em dias contínuos
Minimizar o conflito de dias e períodos de disciplinas que possuem pré-requisito em comum
Maximizar a satisfação dos membros da instituição
Minimizar os períodos vagos para os professores
Minimizar os períodos vagos para estudantes, entre duas aulas
Minimizar os conflitos entre disciplinas pré-requisitos, que podem ser cursadas no mesmo semestre

Traduzido por Bucco (2014)

Fonte: Mirhassani e Habibi (2013)

4 DESCRIÇÃO E CONSTRUÇÃO DO MODELO

O objetivo deste problema é gerar uma grade de horários semanal para o departamento de Matemática, da Universidade Federal Rural do Rio de Janeiro. Para tal, fez-se necessário observar as características deste local. A partir de entrevista com gestores e participação na criação da grade de horários de 2018-2, decidiu-se por objetivar a maximizar a satisfação dos professores, desconsiderando, assim, a satisfação e necessidades dos alunos e salas de aulas, a princípio. A universidade em questão possui 6 intervalos de tempo, com duração de duas horas cada, distribuídos em 5 dias na semana; cada par ordenado dessa matriz é composto por uma aula, como mostra a figura 4.1.

Tabela 4.1: Grade de Horários: UFRRJ

x	Seg	Ter	Qua	Qui	Sex
8:00 - 10:00	-	-	-	-	-
10:00 - 12:00	-	-	-	-	-
13:00 - 15:00	-	-	-	-	-
15:00 - 17:00	-	-	-	-	-
18:00 - 20:00	-	-	-	-	-
20:00 - 22:00	-	-	-	-	-

elaborado pelo autora

Observe que cada professor estará, dessa forma, associado a uma ou mais disciplinas em um intervalo, distribuído pelos dias da semanas. Outra característica importante é que cada disciplina possui uma quantidade de créditos, equivalente a quantidades de horas semanais. Esses créditos são distribuídos duas vezes ou três vezes na semana. Também é importante destacar que as disciplinas nesta universidade são fixadas em horários e dias específicos. Para que fosse possível a criação da formulação no âmbito da PLI, foi necessário criar um ambiente de dados conhecido, onde seria possível acessar informações relevantes para o resultado. Dessa forma, foi feita a criação dos seguintes conjuntos:

Conjuntos:

$p \in P$: Conjunto de professores

$d \in D$: Conjunto de disciplinas ofertadas no semestre

$h \in H$: Conjunto de intervalos de horários

$s \in S$: Conjunto de dias úteis da semana

$Cred_d$ :: Conjunto de créditos de cada disciplina d

$M \subset D$ Subconjunto de disciplinas que estão no horário de 8:00-10:00

$N \subset D$ Subconjunto de disciplinas que estão no horário de 10:00-12:00

$O \subset D$ Subconjunto de disciplinas que estão no horário de 18:00-20:00

$Q \subset D$ Subconjunto de disciplinas que estão no horário de 20:00-22:00

$W \subset D$ Subconjunto de disciplinas que estão alocadas na segunda-feira e quarta-feira.

$T \subset D$ Subconjunto de disciplinas que estão alocadas na terça-feira e quinta-feira.

$Z \subset D$ Subconjunto de disciplinas que estão alocadas na segunda-feira, quarta-feira e sexta-feira.

$I \subset D$ Subconjunto de disciplinas que estão alocadas na quarta-feira e sexta-feira.

$K_h \subset D$ Subconjunto de disciplinas que possuem o horário $h \in H$.

$R_s \subset D$ Subconjunto de disciplinas que possuem o dia da semana $s \in S$.

Foi possível a criação desses conjuntos a partir dos dados compartilhados pelos gestores da universidade em questão. A próxima etapa foi definir que variáveis o problema poderia retornar para que fosse possível alcançar o objetivo de gerar os horários. Assim, criou-se uma variável de controle binária que informe se o professor vai ser alocado em uma disciplina, que por sua vez estará associada a um intervalo de tempo e a um dia da semana. Define-se:

Variáveis:

$$X_{pd} \in (0,1) = \begin{cases} 1, \text{ se o professor p for alocado na disciplina d} \\ 0, \text{ caso contrário} \end{cases}$$

Como visto anteriormente, além das restrições fortes, pode-se ter restrições fracas. Para isso, criou-se uma variável binária que escolhe um número finito de professores para ajudar no problema, tornando, assim, uma restrição forte em fraca.

Variáveis:

$$Y_p \in (0,1) = \begin{cases} 1, \text{ se o professor p for escolhido} \\ 0, \text{ caso contrário} \end{cases}$$

Outra característica da universidade nos faz criar dois parâmetros nesse problema, que seria o número de créditos mínimos que cada professor tem que cumprir em um semestre letivo e os créditos máximos.

Parâmetros:

$cred_{max}$: Número de créditos máximos: 12;

$cred_{min}$: Número de créditos mínimos: 8.

$qtdprof$: Número de professores a escolher: 1

Até aqui, definiu-se a estrutura do modelo, seus conjuntos, variáveis e parâmetros. Mas, como dito anteriormente, o objetivo desta formulação será a de maximizar a satisfação dos professores, sendo, então, necessária a criação de um parâmetro que mostre essa quantidade de satisfação de cada professor. Na próxima seção, constrói-se esse parâmetro.

4.1 Parâmetro Satisfação

Para definir a quantidade de satisfação de cada professor, este trabalho utiliza informações provindas do formulário já existente na criação de horários deste departamento, que está no Anexo 1, de onde é possível retirar informações sobre a preferência de disciplinas e de turno. Para poder acessar esses dados, criou-se os seguintes subconjuntos:

$T_p \subset H$ Preferência de turno de cada professor p;

$D_p \subset D$ Preferências de disciplinas de cada professor p;

$B_p \subset D$ Disciplinas nas quais o professor p está habilitado a dar aulas.

Com esses dados, foram criados parâmetros com pesos, para que fosse possível quantificar esta satisfação. Observou-se, através das entrevistas com os gestores, que atender à disciplina pedida pelo professor é mais relevante na satisfação do que atenderem ao pedido com relação ao turno. Por isso, a princípio, definiu-se os seguintes pesos:

$$b_p \in \{0,1\} = \begin{cases} 1, \text{ se } d \in B_p \\ 0, \text{ caso contrário.} \end{cases}$$

Se a disciplina for uma disciplina na qual o professor está habilitado a dar aula, este parâmetro retorna 1, caso contrário 0.

$$k_p \in \mathbb{Z} = \begin{cases} 70, \text{ se } d \in D_p \\ 0, \text{ caso contrário.} \end{cases}$$

Se a disciplina estiver no conjunto de preferências de disciplinas que o professor pediu, este parâmetro retorna 70, caso contrário, 0.

$$w_p \in \mathbb{Z} = \begin{cases} 30, \text{ se } h_d \in T_p \\ 0, \text{ se } h_d \in T_p \end{cases}$$

Se o horário associado à disciplina estiver no conjunto de preferências de turno que o professor pediu, este parâmetro retorna 30, caso contrário, 0. Com esses parâmetros foi possível criar a equação que retornasse o nível de satisfação de cada professor:

$$f_{pd} = b_p(k_p + w_p) \tag{4.1}$$

Observe que, propositalmente, no máximo f pode atingir 100 e no mínimo 0.

4.2 Restrições

Para obedecer a estrutura de um (PI), esta seção será dedicada a discutir as restrições deste problema.

4.2.1 Retrições Fortes:

RH1: O somatório de todos os professores tem que ser menor ou igual a um

$$\sum_{p \in P} Y_p \leq qtdprof \tag{4.2}$$

Como Y_p se trata de uma variável auxiliar para transformar uma restrição forte em fraca, esta restrição tem como principal objetivo controlar o número de professores no qual alguma restrição poderá relaxar, isto é, neste caso, o parâmetro $qtdprof$ tem como objetivo poder escolher professores os quais não obedecerão a uma certa restrição integralmente.

RH2: O número total de créditos tem que ser maior ou igual ao número de créditos mínimo.

$$\sum_{d \in D} Cred_d X_{pd} \geq Cred_{min} \qquad p \in P \tag{4.3}$$

Para cada professor p, esse somatório indica a quantidade de créditos relacionados às disciplinas e intervalos.

RH3: O número total de créditos tem que ser menor ou igual ao número de créditos máximo.

$$\sum_{d \in D} Cred_d X_{pd} \leq Cred_{max} \qquad p \in P \tag{4.4}$$

Análogo a restrição acima.

RH4: Uma disciplina deverá ser dada por um único professor.

$$\sum_{p \in P} X_{pd} = 1 \qquad d \in D \tag{4.5}$$

Fixada uma disciplina d, esta estará atrelada a um único professor. Dessa forma, esse somatório só poderá ser 1.

RH5: Um professor só poderá dar no máximo uma disciplina em um intervalo de tempo, ou nenhuma.

$$\sum_{d \in K_h \cap R_s} X_{pd} \leq 1 \qquad p \in P, s \in S, h \in H, d \in K_h \cap R_s \tag{4.6}$$

Fixado um professor p, um horário h e um dia da semana s, o professor só pode ser alocado a uma única disciplina atrelada a esse horário e dia da semana ou nenhuma.

RH6: O professor não pode dar aulas em turnos não-conseguintes.

$$\sum_{d_1 \in M} \sum_{d_2 \in O} X_{pd_1} + X_{pd_2} \leq 1 \qquad p \in P$$

$$\sum_{d_1 \in M} \sum_{d_2 \in Q} X_{pd_1} + X_{pd_2} \leq 1 \qquad p \in P$$

$$\sum_{d_1 \in N} \sum_{d_2 \in O} X_{pd_1} + X_{pd_2} \leq 1 \qquad p \in P$$

$$\sum_{d_1 \in N} \sum_{d_2 \in Q} X_{pd_1} + X_{pd_2} \leq 1 \qquad p \in P$$

Esta restrição impede que dado um professor p, este não lecione disciplinas no turno da manhã e no turno da noite, ao mesmo tempo. Assim, faz-se a combinação de todos os horários nos quais se encontra esse problema e essa soma não pode ser maior que 1.

RH7: O professor não pode dar aula segunda-feira e quarta-feira e, ao mesmo tempo, dar aula terça-feira e quinta-feira.

$$X_{pd_1} + X_{pd_2} \leq 1 \qquad p \in P, d_1 \in W, d_2 \in T \tag{4.7}$$

Para cada professor p, cada disciplina d_1 em W e cada disciplina d_2 em T, não poderá ter professor alocado nas duas disciplinas ao mesmo tempo, por conta de seus horários.

RH8: O professor não pode dar aula segunda/quarta/sexta e terça/quinta.

$$X_{pd_1} + X_{pd_2} \leq 1 \qquad p \in P, d_1 \in Z, d_2 \in T \tag{4.8}$$

Para cada professor p, cada disciplina d_1 em Z e cada disciplina d_2 em T, não poderá ter professor alocado nas duas disciplinas ao mesmo tempo, por conta de seus horários.

4.2.2 Restrições Fracas

RS1: Se a disciplina não for uma disciplina no qual ele está habilitado a lecionar, o mesmo não pode ser alocado nela

Se a disciplina não estiver no conjunto B_p, então:

$$X_{pd} \leq qtdprof \qquad p \in P, d \in D - B_p \tag{4.9}$$

Para cada professor e disciplina, se esta disciplina não estiver no conjunto que ele está apto a lecionar, esta variável pode atingir no máximo, o valor de $qtdprof$, onde $qtdprof$ é o parâmetro de controle para tornar essa restrição em fraca.

4.3 Função Objetivo

Dessa forma, chega-se a última etapa da formulação, que é a função objetivo, dada por:

$$MAX \sum_{p \in P} \sum_{d \in D} f_{pd} X_{pd} \tag{4.10}$$

Está coerente com o objetivo desta formulação, que é maximizar a satisfação dos professores, uma vez que o parâmetro f representa a quantidade de satisfação e a variável X retorna os professores que serão alocados. Sendo assim, o produto irá retornar a melhor combinação possível que atinja a maior satisfação possível.

5 RESULTADOS E CONSIDERAÇÕES FINAIS

5.1 Resultados

Este trabalho utilizou como referência os dados do segundo semestre de 2018 da Universidade Federal Rural do Rio de Janeiro, onde o departamento de Matemática é composto por 28 docentes, 33 disciplinas e 63 turmas. Esta foi a base de dados utilizada neste trabalho, para que se obtivessem os resultados. Essa base de dados está nos Anexos, onde se definem todos os conjuntos citados acima. No Apêndice 2 e 3, encontra-se o código que possui a formulação acima, no ambiente do *Gurobi*[1], através do uso da linguagem *Python*[2]. Assim, com os parâmetros de satisfação valendo respectivamente k=70 e w=30, obteve-se a solução em 6 segundos no computador Asus X200M com Processador Intel® Celeron® Dual-Core e DDR3 MHz SDRAM, 2 GB de memória. Através de mudanças nos parâmetros, observou-se uma possível melhora, a partir de conhecimentos e observações com relação a grades antigas, na solução através do aumento do parâmetro k e, consequentemente, da diminuição do parâmetro w. Isso significa que o peso dado aos horários não precisar ser, necessariamente, muito grande, uma vez que pouca quantidade já surte o efeito que esse problema busca. Assim, a partir dessas mudanças de parâmetros, fixou-se um resultado com os parâmetros k=73 e w=27, que está nos Apêndices.

Para efeito de validação do modelo, este resultado fora compartilhado com os docentes da universidade através de um formulário, que indaga o nível de satisfação desses professores, caso aquela fosse sua grade de horários, no período. Bem como pergunta-se se há alguma disciplina no horário que o faz se sentir satisfeito. Dentre o total de 28 professores do departamento, responderam 20 nesta pesquisa. Os resultados do formulário nos deram o seguinte gráfico:

[1] http://www.gurobi.com/
[2] https://www.python.org/

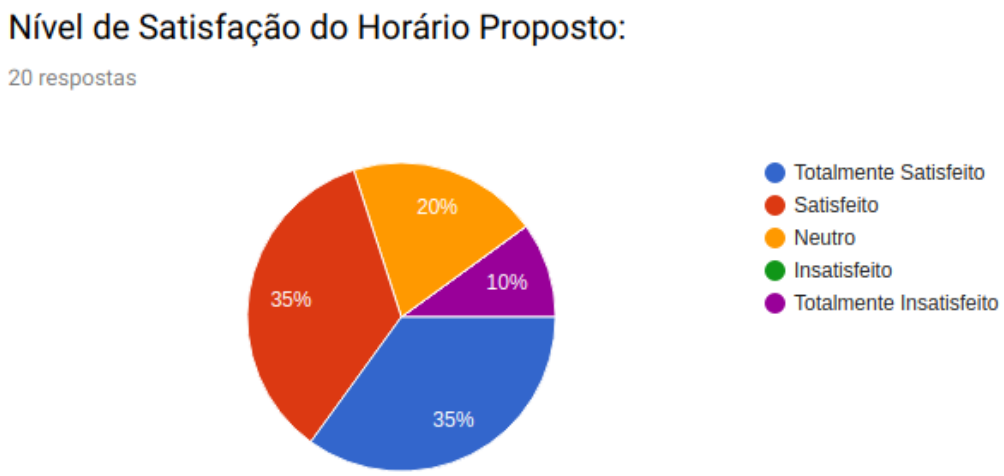

Figura 5.1: Gráfico de Satisfação do Horário Proposto

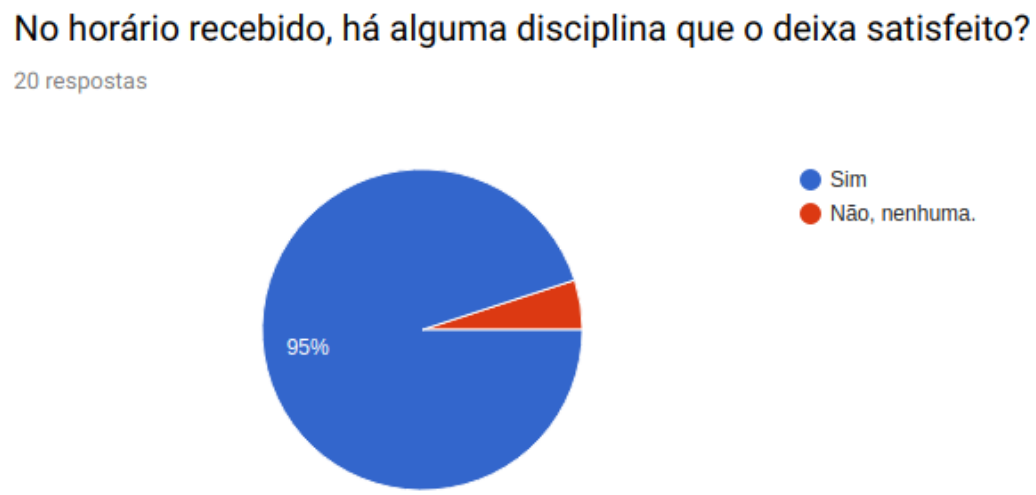

Figura 5.2: Gráfico de Satisfação por Disciplina

Observe que 70% dos professores ficaram satisfeitos ou muito satisfeitos. Nos outros casos, os professores relataram estar insatisfeitos com o espaçamento de horários entre as disciplinas e também por ter matérias repetidas. No segundo gráfico, nota-se que o horário gerado atende maioria das expectativas com relação a disciplinas. Como pôde ser observado, há ganhos neste modelo, no entanto ainda pode ser altamente aperfeiçoado.

5.2 Considerações Finais

Ao começo deste trabalho, esperava-se uma alternativa viável e que demorasse um tempo menor para geração de horários. Esse objetivo foi atingido, uma vez que o programa gera uma grade de horários em tempo curto e ainda consegue satisfazer a maioria dos professores. No processo da pesquisa, houve algumas limitações, uma delas se deu pelo fato de estar sendo baseado em um formulário já pré-existente, isto é, não foi criado um documento, especificamente, para os objetivos deste trabalho. Outra limitação foi o caso da subjetividade do termo "satisfação", acarretando uma dificuldade no pensamento de como quantificar isso da melhor forma possível. O trabalho tem inúmeras maneiras de ser aperfeiçoado, como por exemplo, adicionar informações sobre o nível de produção científica, orientações e projetos dos professores, sendo determinantes na carga horária. Também, pode-se pensar em adicionar ao problema as limitações dos professores com relação aos dias da semana ou horário, bem como pesos as disciplinas de escolha de cada professor. Outra sugestão para o futuro, é pensar em um modelo mais robusto, que envolva salas de aulas e a satisfação dos alunos.

REFERÊNCIAS

[1] BUCCO, Guilherme Brandelli. *Construção de um modelo de programação linear para o university timetabling problem.* M.Phil. thesis, Universidade Federal do Rio Grande do Sul, 2014.

[2] CARVALHO, Ricardo Augusto Trindade. *Aplicações de Programação Linear Inteira Mista à problemas de futebol*, 2015.

[3] DER-SAN CHEN, Robert G. Baston, DANG, Yu. *Applies Integer Programming, modeling and solution*, Wiley, 2011.

[4] GOLDBARG, Marcos C.; P. L. Henrique. *Otimização Combinatória e Programação Linear*, Rio de Janeiro:Campus, 2000.

[5] LAPORTE, Michael W. Carter Gilbert. *Recent Developments in Practical Course Timetabling*, 1997.

[6] LUENBERGER, David G.; YINYU, Ye. *Linear and Nonlinear Programming*, 1973.

[7] MACULAN, Nelson; FAMPA, Marcia H. Costa. *Otimização Linear*, 2004.

[8] MÜLLER, Tomáš; BARTÁK, Roman. *Interactive timetabling: Concepts, techniques, and practical results*, 2002.

[9] MURRAY, Keith; MÜLLER, Tomáš; RUDOVÁ, Hana. *Modeling and Solution of a Complex University Course Timetabling Problem*, 2016.

[10] PONGCHAROEN, P.; PROMTET, W.; YENRADEE, P.; HICKS, C. *Stocastic optimisation tool for university course scheduling*. International Journal of Productions Economics, 2008.

[11] S. A., Mirhassani; FARHANG, Habibi. *Solution approaches to the course timetabling problem*, 2013.

[12] TEIXEIRA, Vinicius Garcia. *Aplicação de Programação Linear na Alocação de Vagões Gôndola para o Transporte de Ferro GUSA na Mrs Logística S.A*, 2011.

[13] TRIPATHY, Arabinda. *School Timetabling—A Case in Large Binary Integer Linear Programming. Management Science*, 1984.

[14] WOLSEY, Laurence. *Integer programming*, 1998.

[15] WOODS, Damien; TRENAMAN, Adrian. *Simultaneous Satisfaction of Hard and Soft Timetable Constraints for a University Department Using Evolutionary Timetabling*, 1999.

[16] WREN, Anthony. *Scheduling, Timetabling and Rostering - A Special Relationship?* Vol. 1153. Lecture Notes in Computer Science, 1995.

Apêndice 1: Conjuntos

codigos/Conjuntos.py

```
D= {
'IC239T02' :['IC239', '8:00-10:00', 'TER,QUI', 4],
'IC239T03' :['IC239', '13:00-15:00', 'TER,QUI', 4],
'IC239T04':['IC239', '15:00-17:00', 'TER,QUI', 4],
'IC239T64':['IC239', '18:00-20:00', 'QUA,SEX', 4],
'IC241T01':['IC239', '10:00-12:00', 'SEG,QUA,SEX', 6],
'IC241T02':['IC241', '8:00-10:00', 'SEG,QUA,SEX', 6],
'IC241T03':['IC241','15:00-17:00', 'SEG,QUA,SEX', 6],
'IC241T04':('IC241', '13:00-15:00', 'SEG,QUA,SEX', 6),
'IC241T05':('IC241', '8:00-10:00', 'SEG,QUA,SEX', 6),
'IC241T06':('IC241', '20:00-22:00', 'SEG,QUA,SEX', 6),
'IC241T07':('IC241', '8:00-10:00', 'SEG,QUA,SEX', 6),
'IC242T01':('IC242', '10:00-12:00', 'SEG,QUA,SEX', 6),
'IC242T02':('IC242', '8:00-10:00', 'SEG,QUA,SEX', 6),
'IC242T04':('IC242', '13:00-15:00', 'SEG,QUA,SEX', 6),
'IC242T05':('IC242', '15:00-17:00', 'SEG,QUA,SEX', 6),
'IC242T64':('IC242', '20:00-22:00', 'SEG,QUA,SEX', 6),
'IC243T01':('IC243', '10:00-12:00', 'TER,QUI', 4),
'IC243T02':('IC243', '10:00-12:00', 'TER,QUI', 4),
'IC243T03':('IC243', '18:00-20:00', 'SEG,QUA', 4),
'IC244T01':('IC244', '8:00-10:00', 'TER,QUI', 4),
'IC244T02':('IC244', '8:00-10:00', 'SEG,QUA', 4),
'IC244T03':('IC244', '13:00-15:00', 'SEG,QUA', 4),
'IC251T01':('IC251', '15:00-17:00','SEG,QUA', 4),
'IC251T02':('IC251', '15:00-17:00', 'SEG,QUA', 4),
'IC251T03':('IC251', '10:00-12:00', 'SEG,QUA', 4),
```

```
'IC251T04':('IC251', '15:00-17:00', 'TER,QUI', 4),
'IC251T05':('IC251', '10:00-12:00', 'SEG,QUA', 4),
'IC251T06':('IC251', '15:00-17:00', 'TER,QUI', 4),
'IC251T07':('IC251', '13:00-15:00', 'SEG,QUA', 4),
'IC251T08':('IC251', '18:00-20:00', 'SEG,QUA', 4),
'IC251T09':('IC251', '18:00-20:00', 'TER,QUI', 4),
'IC251T10':('IC251', '15:00-17:00', 'TER,QUI', 4),
'IC252T01':('IC252', '13:00-15:00', 'TER,QUI',4),
'IC252T02':('IC252', '8:00-10:00', 'TER,QUI', 4),
'IC252T03':('IC252', '8:00-10:00', 'SEG,QUA', 4),
'IC252T06':('IC252', '18:00-20:00', 'TER,QUI', 4),
'IC260T01':('IC260', '8:00-10:00', 'SEG,QUA', 4),
'IC261T01':('IC261', '8:00-10:00', 'SEG,QUA,SEX', 6),
'IC267T01':('IC267','10:00-12:00', 'TER,QUI', 4),
'IC268T01':('IC268', '10:00-12:00', 'SEG,QUA', 4),
'IC269T01':('IC269', '15:00-17:00', 'SEG,QUA', 4),
'IC270T01':('IC270', '15:00-17:00', 'SEG,QUA', 4),
'IC272T01':('IC272', '8:00-10:00', 'TER,QUI', 4),
'IC275T01':('IC275', '13:00-15:00', 'SEG,QUA', 4),
'IC276T01':('IC276', '8:00-10:00', 'SEG,QUA', 4),
'IC277T01':('IC277', '15:00-17:00', 'SEG,QUA', 4),
'IC279T01':('IC279', '8:00-10:00', 'QUA,SEX', 4),
'IC294T01':('IC294', '13:00-15:00', 'TER,QUI', 4),
'IC297T01':('IC297', '15:00-17:00', 'SEG,QUA', 4),
'IC571T01':('IC571', '13:00-15:00', 'TER,QUI', 2),
'IC576T01':('IC576', '8:00-10:00', 'SEG,QUA', 4),
'IC577T01':('IC577', '8:00-10:00', 'TER,QUI', 4),
'IC579T01':('IC579', '10:00-12:00', 'TER,QUI', 4),
'IC801T01':('IC801', '13:00-15:00', 'TER,QUI', 4),
```

```
'IC802T01':('IC802', '13:00-15:00', 'SEG,QUA', 4),
'IC815T01':('IC815', '15:00-17:00', 'TER,QUI', 4),
'IC815T02':('IC815', '10:00-12:00', 'TER,QUI', 4),
'IC851T01':('IC851', '13:00-15:00', 'TER,QUI', 4),
'IC852T01':('IC852', '8:00-10:00', 'TER,QUI', 4),
'IC855T01':('IC855', '13:00-15:00', 'TER,QUI', 4),
'IC861T01':('IC861', '10:00-12:00', 'TER,QUI', 4),
'IC862T01':('IC862', '13:00-15:00', 'SEG,QUA', 4),
'IC863T01':('IC863', '10:00-12:00', 'SEG,QUA', 4),
}

#S conjunto de dias de uma semana, a=segunda, b=terça, c=
    quarta, d=quinta, e=sexta
S = ['SEG,QUA','TER,QUI','SEG,QUA,SEX', 'QUA,SEX']
#H conjunto de intervalos de tempo, 1i = 8:00-10:00, 2i =
     10:00-12:00, 3i = 13:00-15:00, 4i = 15:00-17:00, 5i =
     18:00 - 20:00, 6i = 20:00-22:00
H = ['8:00-10:00', '10:00-12:00', '13:00-15:00', '15:00-
    17:00', '18:00-20:00', '20:00-22:00']
#P conjunto de professores disponíveis no semestre de
    2018-2
P = [
'ALINE',
'ANDRÉMARTINS',
'ANDRÉSMAURÍCIO',
'ANGEL',
'CARLOSANDRÉS',
'CLÁUDIO',
'DANIEL',
```

```
80 'DOUGLAS',
81 'DUILIO',
82 'EDIVALDO',
83 'EULINA',
84 'GABRIEL',
85 'GISELA',
86 'LEANDRO',
87 'LUCIANO',
88 'MARCIA',
89 'MONTAUBAN',
90 'ORLANDO',
91 'PAULOPARGA',
92 'PEDRO',
93 'RENANTEIXEIRA',
94 'RENANVICENTE',
95 'RENATOAQUINO',
96 'ROSANE',
97 'SÉRGIOVENTURA',
98 'VALDOMIRO',
99 'VINICIUS',
100 'WILIAN']
101
102 #PrefD conjunto de preferencias de disciplinas de cada
      professor para o período de 2018-2
103 PrefD = {
104
105 'ALINE' : ('IC243', 'IC251', 'IC260', 'IC270', 'IC815'),
106 'ANDRÉMARTINS' : ('IC260', 'IC267', 'IC269', 'IC276', '
      IC862', 'IC863'),
```

```
'ANDRÉSMAURÍCIO' : ('IC243', 'IC260', 'IC270', 'IC855'),
'ANGEL' : ('IC251', 'IC252', 'IC278'),
'CARLOSANDRÉS' : ('IC243','IC244', 'IC297'),
'CLÁUDIO' : ('IC269', 'IC270', 'IC862', 'IC863'),
'DANIEL' : ('IC239', 'IC243', 'IC244', 'IC260', 'IC267',
    'IC270', 'IC851', 'IC852'),
'DOUGLAS' : ('IC269', 'IC268'),
'DUILIO' : ('IC239', 'IC243', 'IC267', 'IC268','IC851'),
'EDIVALDO' : ('IC239', 'IC242', 'IC243', 'IC244', 'IC279'
    , 'IC297', 'IC815'),
'EULINA' : ('IC270', 'IC802'),
'GABRIEL' : (),
'GISELA' : ('IC801', 'IC802','IC861'),
'LEANDRO' : ('IC251', 'IC252', 'IC276'),
'LUCIANO' : ('IC815'),
'MARCIA' : (),
'MONTAUBAN' : ('IC242'),
'ORLANDO' : ('IC244', 'IC268'),
'PAULOPARGA' : ('IC239', 'IC243', 'IC260', 'IC268', '
    IC270', 'IC851', 'IC852'),
'PEDRO' : (),
'RENANTEIXEIRA': ('IC243', 'IC244','IC272','IC287', '
    IC851'),
'RENANVICENTE' : ('IC244', 'IC278', 'IC851', 'IC852'),
'RENATOAQUINO' : ('IC577', 'IC802'),
'ROSANE' : ('IC239', 'IC272', 'IC297', 'IC851', 'IC852'),
'SÉRGIOVENTURA' : ('IC269', 'IC270', 'IC278', 'IC279', '
    IC297', 'IC815', 'IC851', 'IC852'),
'VALDOMIRO': (),
```

```
'VINICIUS' : ('IC277', 'IC279', 'IC294'),
'WILIAN' : ('IC239', 'IC241', 'IC251', 'IC252', 'IC276')
}

#PrefT preferência de turno de cada professor no semestre de 2018-2.
PrefT = {
'ALINE' : ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'ANDRÉMARTINS' : ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'ANDRÉSMAURÍCIO': ('18:00-20:00', '20:00-22:00'),
'ANGEL': ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'CARLOSANDRÉS': ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'CLÁUDIO': ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'DANIEL': ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'DOUGLAS': ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'DUILIO': ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'EDIVALDO': ('18:00-20:00', '20:00-22:00'),
'EULINA': ('8:00-10:00','10:00-12:00','13:00-15:00','15:00-17:00'),
'GABRIEL' : (),
```

```
149 'GISELA' : ('8:00-10:00','10:00-12:00','13:00-15:00','15:
        00-17:00'),
150 'LEANDRO' : ('8:00-10:00','10:00-12:00','13:00-15:00','15
        :00-17:00'),
151 'LUCIANO': (),
152 'MARCIA' : (),
153 'MONTAUBAN' : ('8:00-10:00','10:00-12:00','13:00-15:00','
        15:00-17:00'),
154 'ORLANDO': ('8:00-10:00','10:00-12:00','13:00-15:00','15:
        00-17:00'),
155 'PAULOPARGA': ('8:00-10:00','10:00-12:00','13:00-15:00','
        15:00-17:00'),
156 'PEDRO' : (),
157 'RENANTEIXEIRA' : ('18:00-20:00', '20:00-22:00'),
158 'RENANVICENTE' : ('8:00-10:00','10:00-12:00','13:00-15:00
        ','15:00-17:00'),
159 'RENATOAQUINO': ('8:00-10:00','10:00-12:00','13:00-15:00'
        ,'15:00-17:00'),
160 'ROSANE': ('8:00-10:00','10:00-12:00','13:00-15:00','15:
        00-17:00'),
161 'SÉRGIOVENTURA': ('8:00-10:00','10:00-12:00','13:00-15:00
        ','15:00-17:00'),
162 'VALDOMIRO': (),
163 'VINICIUS': ('8:00-10:00','10:00-12:00','13:00-15:00','15
        :00-17:00'),
164 'WILIAN': ('18:00-20:00', '20:00-22:00')
165 }
166
```

```
167 #Disciplinas divididas em áreas, 1-Métodos Numéricos,
      Otimização, Pura, Licenciatura e Básico.
168 AREAS = {
169 'METNUM': ('IC272', 'IC279', 'IC297'),
170 'OT': ('IC294', 'IC277', 'IC279'),
171 'PURA': ('IC267', 'IC268', 'IC269', 'IC270', 'IC862','
      IC863', 'IC855'),
172 'LIC': ('IC275', 'IC571', 'IC576', 'IC577', 'IC579', '
      IC801', 'IC802', 'IC861'),
173 'BASICO': ('IC239', 'IC241', 'IC242', 'IC243', 'IC244', '
      IC251', 'IC252', 'IC261', 'IC260', 'IC276', 'IC815', '
      IC851')
174 }
175
176 M1 = set(AREAS['BASICO']).union(AREAS['PURA'])
177 M2 = set(AREAS['BASICO']).union(AREAS['OT'])
178 M3 = set(AREAS['BASICO']).union(AREAS['LIC'])
179 M4 = set(AREAS['BASICO']).union(AREAS['METNUM'])
180
181
182 #PrefA conjunto que indica as disciplinas que cada
      professor está apto a lecionar.
183 PrefA = {
184 'ALINE' : (M1),
185 'ANDRÉMARTINS' : (M1),
186 'ANDRÉSMAURÍCIO': (M1),
187 'ANGEL': (M2),
188 'CARLOSANDRÉS': (M4),
189 'CLÁUDIO' : (M1),
```

```
190 'DANIEL' : (M1),
191 'DOUGLAS' : (M1),
192 'DUILIO' : (M4),
193 'EDIVALDO': (M4),
194 'EULINA' : (M1),
195 'GABRIEL': (AREAS['BASICO']),
196 'GISELA' : (M3),
197 'LEANDRO' : (M1),
198 'LUCIANO' : (M1),
199 'MARCIA' : (M1),
200 'MONTAUBAN': (M1),
201 'ORLANDO': (M1),
202 'PAULOPARGA': (M2),
203 'PEDRO' : (M3),
204 'RENANTEIXEIRA' : (M4),
205 'RENANVICENTE' : (M2),
206 'RENATOAQUINO': (M3),
207 'ROSANE' : (M4),
208 'SÉRGIOVENTURA' : (M2),
209 'VALDOMIRO' : (M2),
210 'VINICIUS' : (M2),
211 'WILIAN' : (M4)
212
213 }
```

Apêndice 2: Lista de Chaves

codigos/listsofkeys.py

```
"""Created on Fri Nov  9 18:06:03 2018

@author: mndzvd
"""

def getKeysByValues(dictOfElements, listOfValues):
    listOfKeys = list()
    listOfItems = dictOfElements.items()
    for item  in listOfItems:
        if item[1][2] == listOfValues:
            listOfKeys.append(item[0])
    return  listOfKeys

W = getKeysByValues(D, 'SEG,QUA')

T = getKeysByValues(D, 'TER,QUI')

Z = getKeysByValues(D, 'SEG,QUA,SEX')

I = getKeysByValues(D, 'QUA,SEX')


def getKeysByValues1(dictOfElements, listOfValues):
    listOfKeys = list()
    listOfItems = dictOfElements.items()
    for item  in listOfItems:
```

```
27            if item[1][1] in listOfValues:
28                listOfKeys.append(item[0])
29        return  listOfKeys
30
31 M = getKeysByValues1(D,'8:00-10:00')
32 N = getKeysByValues1(D,'18:00-20:00')
33 O = getKeysByValues1(D,'20:00-22:00')
34 Q = getKeysByValues1(D,'10:00-12:00')
35
36
37 def getKeysByValues2(dictOfElements, listOfValues):
38     listOfKeys = list()
39     listOfItems = dictOfElements.items()
40     for item  in listOfItems:
41         if item[1][2] in listOfValues:
42             listOfKeys.append(item[0])
43     return  listOfKeys
44
45
46 def getKeysByValues3(dictOfElements, listOfValues):
47     listOfKeys = list()
48     listOfItems = dictOfElements.items()
49     for item  in listOfItems:
50         if item[1][2] == listOfValues:
51             listOfKeys.append(item[0])
52     return  listOfKeys
53
54 def getKeysByValues4(dictOfElements, listOfValues):
55     listOfKeys1 = list()
```

```
56      listOfItems = dictOfElements.items()
57      for item  in listOfItems:
58          if item[1][1] == listOfValues:
59              listOfKeys1.append(item[0])
60      return  listOfKeys1
61
62
63  #Iterate over the list of values
64  for key  in W:
65      print(key)
```

Apêndice 3: Código principal

codigos/horariosfixados.py

```
#from SETS.py import *
from gurobipy import *
from numpy import *

m= Model("Scheduling")


#Função satisfação
f = {}
X = {}
Y = {}
for p in P:
    f[p] = {}
    for d in D.keys():
        f[p][d] = {}
        f[p][d][D[d][2]] = {}
        f[p][d][D[d][2]][D[d][1]] = {}

        SB = set(PrefA.get(p))
        if D[d][0] in SB:
            a = 1
        else:
            a = 0
        SD = set(PrefD.get(p))
        if D[d][0] in SD:
            k = 73
```

```
27            else:
28                k = 0
29            ST = set(PrefT.get(p))
30            if D[d][1] in ST:
31                w = 27
32            else:
33                w = 0
34            f[p][d][D[d][2]][D[d][1]] = a*(k + w)
35
36 #Variável X binária e função objetivo.
37 for p in P:
38     X[p] = {}
39     for d in D.keys():
40         X[p][d] = {}
41         X[p][d][D[d][2]] = {}
42         X[p][d][D[d][2]][D[d][1]] = {}
43         X[p][d][D[d][2]][D[d][1]] = m.addVar(vtype=GRB.
               BINARY, name="%s'%s'%s'%s'"% (p, d, D[d][2], D
               [d][1]))
44
45 for p in P:
46     Y[p] = {}
47     Y[p] = m.addVar(vtype=GRB.BINARY, name="%s'"% (p))
48
49
50 m.setObjective(quicksum((X[p][d][D[d][2]][D[d][1]])*(f[p]
    [d][D[d][2]][D[d][1]]) for p in P for d in D.keys()),
    GRB.MAXIMIZE)
51
```

```
52
53 """RESTRIÇOES"""
54
55
56 m.addConstr(quicksum(Y[p] for p in P) <= 1)
57
58
59 m.update()
60
61
62 """RESTRIÇOES FRACAS"""
63
64 #se a disciplina não for da area do professor, ele não
       pode ser alocado nela
65 for p in P:
66     for d in D.keys():
67         SB = set(PrefA.get(p))
68         if D[d][0] not in SB:
69             m.addConstr(X[p][d][D[d][2]][D[d][1]] <= Y[p]
                   )
70
71
72 """RESTRIÇOES FORTES"""
73
74
75 #segqua e terça e quinta não pode
76 for p in P:
77     for d in W:
78         for t in T:
```

```
79                m.addConstr((X[p][d][D[W[0]][2]][D[d][1]]) +
                      (X[p][t][D[T[6]][2]][D[t][1]]) <= 1)
80
81  #ter e quinta não pode com seg quar e sexta
82  for p in P:
83      for d in Z:
84          for t in T:
85              m.addConstr((X[p][d][D[Z[0]][2]][D[d][1]]) +
                    (X[p][t][D[T[6]][2]][D[t][1]]) <= 1)
86
87  #dado um dia da semana, um professor e um horário, este
        só pode dar uma aula ou nenhuma
88  for p in P:
89      for s in S:
90          for h in H:
91              A = getKeysByValues4(D, h);
92              B = getKeysByValues3(D, s);
93              C = set(A).intersection(B);
94              m.addConstr(quicksum([X[p][d][s][h] for d in
                    C]) <= 1)
95
96
97
98  #dada uma disciplina, esta só pode ser dada por um
        professor
99  for d in D.keys():
100   m.addConstr(quicksum([X[p][d][D[d][2]][D[d][1]] for p
          in P]) == 1)
101
```

```
102
103
104 #dado um mesmo horário, não pode cair quarta e sexta
       junto com seg,qua,sex
105 for p in P:
106     for h in H:
107         A = getKeysByValues4(D, h);
108         B = set(A).intersection(I);
109         C = set(A).intersection(Z);
110         for d in C:
111             for z in B:
112              m.addConstr((X[p][z][D[I[0]][2]][h]) + (X[p]
                    [d][D[Z[0]][2]][h]) <= 1)
113
114
115
116 #dado um mesmo horário, não pode cair seg e quarta junto
       com quarta e sexta
117 for p in P:
118     for h in H:
119         A = getKeysByValues4(D, h);
120         B = set(A).intersection(I);
121         C = set(A).intersection(W);
122         for d in C:
123             for z in B:
124              m.addConstr((X[p][z][D[I[0]][2]][h]) + (X[p]
                    [d][D[W[0]][2]][h]) <= 1)
125
```

```
126 #dado um mesmo horário, não pode cair seg quarta junto
      com seg,qua,sex
127 for p in P:
128     for h in H:
129         A = getKeysByValues4(D, h);
130         B = set(A).intersection(I);
131         C = set(A).intersection(Z);
132         for d in C:
133             for z in B:
134                 m.addConstr((X[p][z][D[I[0]][2]][h]) + (X
                      [p][d][D[Z[0]][2]][h]) <= 1)
135
136
137
138
139
140 #créditos
141
142 for p in P:
143   m.addConstr(quicksum([X[p][d][D[d][2]][D[d][1]]*D[d][3]
        for d in D.keys()]) >= 8)
144
145
146 for p in P:
147     m.addConstr(quicksum([X[p][d][D[d][2]][D[d][1]]*D[d][
        3] for d in D.keys()]) <= 12)
148
149
150
```

```
151 #turnos não conseguintes
152
153 for p in P:
154     for d in M:
155         for o in O:
156             m.addConstr(X[p][d][D[d][2]][D[M[0]][1]]+X[p]
                    [o][D[o][2]][D[O[0]][1]] <= 1)
157
158
159 for p in P:
160     for d in Q:
161         for o in O:
162             m.addConstr(X[p][d][D[d][2]][D[Q[0]][1]]+X[p]
                    [o][D[o][2]][D[O[0]][1]] <= 1)
163
164 for p in P:
165     for d in Q:
166         for n in N:
167             m.addConstr(X[p][d][D[d][2]][D[Q[0]][1]]+X[p]
                    [n][D[n][2]][D[N[0]][1]] <= 1)
168
169 for p in P:
170     for d in M:
171         for n in N:
172             m.addConstr((X[p][d][D[d][2]][D[M[0]][1]]) +
                    (X[p][n][D[n][2]][D[N[0]][1]]) <= 1)
173
174 m.update()
175
```

```
176
177 m.optimize()
178
179 for v in m.getVars():
180     if v.x == 1:
181         print(v.varName, v.x)
```

Professores	Disciplinas/Turma	Dias na Semana	Horário
ALINE	IC251T02	SEG,QUA	15:00-17:00
ALINE	IC251T03	SEG,QUA	10:00-12:00
ALINE	IC251T07	SEG,QUA	13:00-15:00
ANDRÉMARTINS	IC276T01	SEG,QUA	8:00-10:00
ANDRÉMARTINS	IC863T01	SEG,QUA	10:00-12:00
ANDRÉSMAURÍCIO	IC243T01	TER,QUI	10:00-12:00
ANDRÉSMAURÍCIO	IC855T01	TER,QUI	13:00-15:00
ANGEL	IC251T06	TER,QUI	15:00-17:00
ANGEL	IC252T01	TER,QUI	13:00-15:00
ANGEL	IC252T02	TER,QUI	8:00-10:00
CARLOSANDRÉS	IC241T02	SEG,QUA,SEX	8:00-10:00
CARLOSANDRÉS	IC297T01	SEG,QUA	15:00-17:00
CLÁUDIO	IC261T01	SEG,QUA,SEX	8:00-10:00
CLÁUDIO	IC862T01	SEG,QUA	13:00-15:00
DANIEL	IC239T03	TER,QUI	13:00-15:00
DANIEL	IC267T01	TER,QUI	10:00-12:00
DOUGLAS	IC241T07	SEG,QUA,SEX	8:00-10:00
DOUGLAS	IC269T01	SEG,QUA	15:00-17:00
DUILIO	IC239T02	TER,QUI	8:00-10:00
DUILIO	IC239T04	TER,QUI	15:00-17:00
DUILIO	IC243T02	TER,QUI	10:00-12:00
EDIVALDO	IC239T64	QUA,SEX	18:00-20:00
EDIVALDO	IC242T64	SEG,QUA,SEX	20:00-22:00
EULINA	IC242T02	SEG,QUA,SEX	8:00-10:00
EULINA	IC270T01	SEG,QUA	15:00-17:00
GABRIEL	IC241T04	SEG,QUA,SEX	13:00-15:00
GABRIEL	IC241T05	SEG,QUA,SEX	8:00-10:00
GISELA	IC577T01	TER,QUI	8:00-10:00
GISELA	IC801T01	TER,QUI	13:00-15:00
GISELA	IC861T01	TER,QUI	10:00-12:00
LEANDRO	IC251T01	SEG,QUA	15:00-17:00
LEANDRO	IC251T05	SEG,QUA	10:00-12:00
LEANDRO	IC252T03	SEG,QUA	8:00-10:00
LUCIANO	IC241T03	SEG,QUA,SEX	15:00-17:00
LUCIANO	IC242T01	SEG,QUA,SEX	10:00-12:00

MARCIA	IC571T01	TER,QUI	13:00-15:00
MARCIA	IC579T01	TER,QUI	10:00-12:00
MARCIA	IC852T01	TER,QUI	8:00-10:00
MONTAUBAN	IC242T04	SEG,QUA,SEX	13:00-15:00
MONTAUBAN	IC242T05	SEG,QUA,SEX	15:00-17:00
ORLANDO	IC244T02	SEG,QUA	8:00-10:00
ORLANDO	IC244T03	SEG,QUA	13:00-15:00
ORLANDO	IC268T01	SEG,QUA	10:00-12:00
PAULOPARGA	IC241T01	SEG,QUA,SEX	10:00-12:00
PAULOPARGA	IC260T01	SEG,QUA	8:00-10:00
PEDRO	IC251T08	SEG,QUA	18:00-20:00
PEDRO	IC275T01	SEG,QUA	13:00-15:00
RENANTEIXEIRA	IC241T06	SEG,QUA,SEX	20:00-22:00
RENANTEIXEIRA	IC243T03	SEG,QUA	18:00-20:00
RENANVICENTE	IC244T01	TER,QUI	8:00-10:00
RENANVICENTE	IC294T01	TER,QUI	13:00-15:00
RENATOAQUINO	IC576T01	SEG,QUA	8:00-10:00
RENATOAQUINO	IC802T01	SEG,QUA	13:00-15:00
ROSANE	IC272T01	TER,QUI	8:00-10:00
ROSANE	IC851T01	TER,QUI	13:00-15:00
SÉRGIOVENTURA	IC815T01	TER,QUI	15:00-17:00
SÉRGIOVENTURA	IC815T02	TER,QUI	10:00-12:00
VALDOMIRO	IC251T04	TER,QUI	15:00-17:00
VALDOMIRO	IC251T09	TER,QUI	18:00-20:00
VINICIUS	IC277T01	SEG,QUA	15:00-17:00
VINICIUS	IC279T01	QUA,SEX	8:00-10:00
WILIAN	IC251T10	TER,QUI	15:00-17:00
WILIAN	IC252T06	TER,QUI	18:00-20:00
MARCIA Yp = 1			

A Anexo 1: Disciplinas ofertadas no segundo semestre

Disciplinas	Nome
IC239	ALGEBRA LINEAR II
IC241	CÁLCULO 1
IC242	CÁLCULO 2
IC243	CÁLCULO 3
IC244	CÁLCULO 4
IC251	MAT 1
IC252	MAT 2
IC260	FUNÇÕES DE VARIÁVEL COMPLEXA
IC261	ALGEBRA 1
IC267	ANÁLISE REAL 1
IC268	ANÁLISE REAL 2
IC269	ANÁLISE REAL 3
IC270	TOPOLOGIA
IC272	MÉTODOS DE MATEMÁTICA APLICADA
IC275	EVOLUÇÃO DA MATEMÁTICA
IC276	MATEMÁTICA PARA ECONOMIA
IC277	PROGRAMAÇÃO MATEMÁTICA 1
IC279	CÁLCULO NUMÉRICO
IC294	MATEMÁTICA COMBINATÓRIA
IC297	ANÁLISE NUMÉRICA
IC571	LABORATÓRIOS DE ESTUDOS MATEMÁTICOS 1
IC576	TÓPICOS DE GEOMETRIA ESPACIAL
IC577	TÓPICOS DE GEOMETRIA PLANA
IC579	FUNDAMENTOS DA MATEMÁTICA
IC801	LAB. DE ENSINO DA MATEMÁTICA DA EDUC. BÁSICA 1
IC802	LAB. DE ENSINO DA MATEMÁTICA DA EDUC. BÁSICA 2
IC815	GEOMETRIA ANALÍTICA
IC851	ALGEBRA LINEAR A
IC852	ALGEBRA LINEAR B
IC855	INTRODUÇÃO À ANÁLISE FUNCIONAL
IC861	FUNDAMENTOS DE MATEMÁTICA ELEMENTAR
IC862	TEORIA DOS GRUPOS

B Anexo 2: Professores no Departamento de Matemática

Professores
Aline Mauricio Barbosa
André Luiz Martins Pereira
Andrés Mauricio López Barragan
Angel Ramon Sanchez Delgado
Carlos Andres Reyna Vera-Tudela
Claudio Cesar Saccomori Júnior
Daniel Reis de Oliveira
Douglas Monsôres de Melo Santos
Duilio Tadeu da Conceição Junior
Edivaldo Figueiredo Fontes Júnior
Eulina Coutinho Silva do Nascimento
Gabriel
Gisela Maria da Fonseca Pinto
Leandro Tomaz Araújo
Luciano Vianna Félix
Marcia Costa Chaves
Montauban Moreira de Oliveira Júnior
Orlando dos Santos Pereira
Paulo César Parga Rodrigues
Pedro Carlos Pereira
Renan de Souza Teixeira
Renan Vicente Pinto
Renato Machado Aquino
Rosane Ferreira de Oliveira
Sérgio Drumond Ventura
Valdomiro Neves Lima
Vinícius Leal do Forte
Wilian Jeronimo dos Santos