PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
PROGRAMA DE PÓS-GRADUAÇÃO STRICTO SENSU EM ENGENHARIA DE PRODUÇÃO E SISTEMAS

SOBRE O PROBLEMA DE DESIGNAÇÃO DE SALAS DE AULA PARA A PUC GOIÁS: UM ESTUDO DE CASO PARA A ÁREA 3, CAMPUS I

Geovane Reges de Jesus Campos

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica de Goiás, como parte dos requisitos para obtenção do título de Mestre em Engenharia de Produção e Sistemas.

Orientador: Professor Marco Antonio F. Menezes, Dr.

GOIÂNIA – GO
SETEMBRO – 2012

SOBRE O PROBLEMA DE DESIGNAÇÃO DE SALAS DE AULA PARA A PUC GOIÁS: UM ESTUDO DE CASO PARA A ÁREA 3, CAMPUS I

Geovane Reges de Jesus Campos

Esta Dissertação foi julgada adequada para obtenção do título de Mestre em Engenharia de Produção e Sistemas, e aprovada em sua forma final pelo Programa de Pós-graduação em Engenharia de Produção e Sistemas da Pontifícia Universidade Católica de Goiás em Setembro de 2012.

_____

Prof. Ricardo Luiz Machado, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia de Produção e Sistemas

Banca Examinadora:

_____

Prof. Marco Antonio Figueiredo Menezes, Dr.
Pontifícia Universidade Católica de Goiás
Orientador

_____

Prof. Leizer de Lima Pinto, Dr.
Universidade Federal de Goiás

_____

Prof. Sibélius Lellis Vieira, Dr.
Pontifícia Universidade Católica de Goiás

GOIÂNIA – GO
SETEMBRO – 2012

Campos, Geovane Reges de Jesus

Sobre o problema de designação de salas de aula para a PUC Goiás. Um estudo de caso para a área 3, campus I /Geovane Reges de Jesus Campos,– Goiânia: PUC Goiás/MEPROS, 2012.

IX p.:39 il.; 29,7 cm.

Orientador: Marco Antonio Figueiredo Menezes

Dissertação (mestrado) – PUC Goiás/ MEPROS/ Programa de Pós-Graduação em Engenharia de Produção e Sistemas, 2012.

Referências Bibliográficas: p.40 – 42

1. Problema de designação de salas de aula. 1.2 Modelagem. 1.3 Otimização Linear. I. Menezes, Marco Antonio Figueiredo. II. Pontifícia Universidade Católica de Goiás, MEPROS, Programa de Mestrado em Engenharia de Produção e Sistemas. III. Sobre o problema de designação de salas de aula para a PUC Goiás. Um estudo de caso para a área 3, campus I.

A minha esposa Aracelly.
As minhas filhas Eduarda, Geovana e Manuela.
Aos meus familiares e amigos.

AGRADECIMENTOS

A Deus na pessoa de Jesus Cristo por ter dado essa oportunidade.

A minha esposa essa companheira amada que sempre esteve ao meu lado.

As minhas filhas Geovana, Eduarda e Manuela por fazer-me sentir a melhor pessoa.

Aos meus pais pelo amor e carinho que souberam-me transmitir.

Ao Professor e orientador Marco Antônio F. Menezes, pelos conhecimentos transmitidos, e todos os outros professores, do Programa de Mestrado em Engenharia de Produção e Sistema.

Aos colegas da Universidade Estadual de Goiás - UEG e ao programa Educando e Valorizando a Vida – EVV.

Aos colegas do MEPROS, em especial ao Robson Iwamoto, pelo companheirismo, amizade e pelas palavras de incentivo e motivação.

Ao coordenador do MEPROS Professor Ricardo Luiz Machado.

Aos que participaram desta dissertação de forma indireta ou direta, Professor Ivon Rodrigues Canedo, Brenno Marks Neves Silva, Kelligton Fabrício de Souza Neves e os coordenadores dos cursos do Centro Técnico Cientifico.

Aos servidores do MEPROS, em especial a Kênia e o Ernani.

E finalmente, a todas as pessoas que contribuíram para a realização deste.

Resumo da Dissertação apresentada ao MEPROS/ PUC Goiás como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia de Produção e Sistemas (M.Sc.)

SOBRE O PROBLEMA DE DESIGNAÇÃO DE SALAS DE AULA PARA A PUC GOIÁS: UM ESTUDO DE CASO PARA A ÁREA 3, CAMPUS I

Geovane Reges de Jesus Campos

SETEMBRO/2012

Orientador: Prof. Marco Antonio Figueiredo Menezes, Dr.

Este trabalho apresenta um estudo para o problema de designação de salas de aula para a PUC Goiás, área 3, Campus I, baseado em um sistema de programação (SAPA), no algoritmo Húngaro e na idéia de resolver o problema horário por horário. O problema é resolvido, com dados reais, em não mais do que 6 segundos.

PALAVRAS–CHAVE: problema de designação de salas de aula, modelagem, otimização linear.

Abstract of Dissertation presented to MEPROS/ PUC Goiás as part of the requirements for obtaining a degree in Production Engineering and Systems (M.Sc.).

THE PROBLEM OF CLASSROOM ASSIGNMENT PROBLEM FOR THE PUC GOIÁS: A CASE STUDY FOR AREA 3, CAMPUS I

Geovane Reges de Jesus Campos

SETEMBRO/2012

Advisor: Prof. Marco Antonio Figueiredo Menezes, Dr.

This paper presents a study for the classroom assignment problem for the PUC Goiás, Area 3, Campus I, based on a programming system (SAPA), the Hungarian algorithm, and on the idea of solving the problem time by time. The problem is solved with real data no more than 6 seconds.

KEYWORDS: classroom assignment problem, modeling, linear optimization.

SUMÁRIO

LISTA DE FIGURAS

INTRODUÇÃO

A designação de salas de aula é um problema que atende as características da área 3, campus I, na PUC Goiás. Este problema consiste em, dado um conjunto de turmas com professores e horários já definidos e um conjunto de salas, designar uma sala para cada turma de acordo com um conjunto de restrições. Segundo Souza (2000), é difícil para a comunidade científica generalizá-lo, principalmente pela diversidade de regimes educacionais e as características de cada instituição de ensino.

A solução manual desse problema pode ser um trabalho dispendioso, devido à quantidade de turmas e salas, e a solução obtida pode ser insatisfatória, por não atender aos interesses dos envolvidos. Além disso, a maioria dos problemas encontrados na literatura procuram resolvê-lo com heurística e metaheurística, encontrando soluções satisfatórias mas que ainda necessitam da intervenção humana, conforme Carter & Tovey (1992).

O trabalho de alocação das turmas-salas de aula da área 3, campus I, na PUC Goiás, é feito de forma manual pela Coordenação de Programação Acadêmica (CPA) e leva um mês e uma semana na primeira etapa. O objetivo é automatizar essa designação implementando o algoritmo Húngaro, testado por Neves (2010), o qual mostra sua eficiência em tempo de execução que não chega a 3 segundos para o número de turmas igual a 685 (285 disciplinas dispostas em 43 horários diferentes durante a semana) e o número total de salas igual a 34. Neste semestre, 2012/01, o problema é constituído de 1254 turmas dispostas em 43 horários diferentes durante a semana, nos turnos matutino, vespertino e noturno, que devem ser alocadas para 81 salas dispostas em 6 blocos

diferentes e, também, duas áreas diferentes. Aqui trabalharemos com um refinamento melhor dos dados e pesos sugerido por Silva (2011).

O problema de designação de salas de aula, área 3, campus I, da PUC Goiás, é um problema de otimização que procura encontrar uma solução tendo que considerar um conjunto de restrições essenciais e não essenciais, conforme Burke et. al. (1997). Segundo a Coordenadora de Programação Acadêmica, no momento, existem seis restrições importantes para a gerência do problema de designação de salas de aula. As duas primeiras restrições são requisitos essenciais, enquanto que as quatro últimas são requisitos não essenciais: R_1: uma sala terá no máximo uma turma no mesmo horário; R_2: cada turma que necessita de sala com destinação própria deverá ser alocada para exatamente uma sala desse tipo, com sua capacidade mínima e máxima respeitada; R_3: cada turma poderá ter a necessidade de salas diferenciadas; R_4 : cada turma deverá ser alocada para uma sala próxima ao bloco de seu curso; R_5: turmas de mesmo curso deverão ser alocadas preferencialmente para salas no mesmo bloco na semana; R_6: turmas de mesmo período da grade curricular de um curso deverão ser alocadas preferencialmente para a mesma sala na semana.

Este trabalho apresenta um estudo para o problema de designação de salas de aula para a PUC Goiás, área 3, Campus I, baseado em um sistema de programação (SAPA), no algoritmo Húngaro e na idéia de resolver o problema horário por horário. O problema é resolvido, com dados reais, em não mais do que 6 segundos.

Este trabalho será composto por três capítulos assim distribuídos: no capítulo 1 apresentamos um estudo sobre a otimização linear (programação linear (PL) e programação linear inteira (PLI)) e uma introdução à teoria dos grafos. No capítulo 2

apresentamos os problemas de designação de salas de aula e uma formulação, no processo de modelagem, para o problema de designação de salas de aula da área 3, campus I, na PUC Goiás. No capítulo 3 apresentamos os resultados computacionais obtidos com a aplicação do algoritmo Húngaro ao problema de designação de salas de aula da área 3, campus I, na PUC Goiás. Finalmente, concluímos nosso trabalho apresentando as considerações finais desta dissertação.

Neste trabalho, utilizaremos a primeira pessoa do plural por uma questão de estilo de acordo com o orientador.

CAPÍTULO I - OTIMIZAÇÃO LINEAR E GRAFOS

Nesse capítulo será apresentado um estudo sobre a otimização linear (programação linear (PL) e programação linear inteira (PLI)) e uma introdução à teoria dos grafos. O texto aqui apresentado está fortemente baseado em Alves & Menezes (2010), Menezes (2011) e Bondy & Murty (1976). Sugerimos, também, os livros Arenales, Armentano, Morabito & Yanasse (2007), Maculan & Fampa (2006), Bondy & Murty (2008), Cormen, Leiserson, Rivest & Stein (2002) e Toscani &Veloso (2005).

Iniciamos esse capítulo com o estudo do problema de programação linear (PPL).

1.1 O problema de programação linear

O problema de otimização é o problema de encontrar possíveis minimizadores ou maximizadores de uma função definida em uma determinada região.

Denotamos $N = \{1,2,...\}$ o conjunto dos números naturais, Z o conjunto dos números inteiros, Z_+ o conjunto dos números inteiros não negativos, Z^n o conjunto cartesiano de n conjuntos dos números inteiros, $\Re$ o conjunto dos números reais, $\Re^n$ o conjunto cartesiano de n conjuntos dos números reais, $\{0,1\}$ o conjunto de zeros e uns, e $\{0,1\}^n$ o conjunto cartesiano de n conjuntos de zeros e uns.

Consideremos os números inteiros m e n tais que $0 < m < n$. Dados uma matriz numérica com coeficientes reais A, $m \times n$, e vetores $b \in \Re^m$ e $c \in \Re^n$, o problema de

programação linear no formato padrão é o seguinte problema de otimização, usualmente denominado problema primal:

$$(P_1) \quad \text{minimizar} \quad c^T x$$

$$\text{sujeito a:} \quad Ax = b$$

$$x \geq 0.$$

Seguem-se algumas definições acerca do problema (P_1).

Definição 1.1 Considere o problema de PL (P_1).

(a) A função linear $x \mapsto c^T x$ é chamada função objetivo, e $c^T x$ é chamado o valor da função objetivo.

(b) O conjunto

$$X = \{x \in \Re^n ; Ax = b, x \geq 0\}$$

é chamado conjunto viável e um ponto $x \in X$ é denominado ponto viável.

(c) O conjunto

$$X(P_1) = \{x^* \in X ; c^T x^* \leq c^T x \text{, para todo } x \in X\}$$

é chamado conjunto de soluções ótimas e um ponto $x \in X(P_1)$ é denominado solução ótima.

(d) O problema (P_1) chama-se problema ilimitado quando existe uma sequência (x^k) tal que $x^k \in X$ e $c^T x^k \to -\infty$, quando $k \to \infty$.

(e) O problema (P_1) chama-se problema inviável quando X é vazio.

O problema de PL (P_1) pode ser interpretado da seguinte maneira: dada uma matriz tecnológica com números reais $A,\ m \times n,$ dados um vetor do lado direto $b \in \Re^m$ e um vetor custo $c \in \Re^n,$ encontrar, se existir, um vetor de variáveis de decisão $x^* \in X$ tal que, $c^T x^* = \min\{c^T x;\ x \in X\}$. Caso não exista, certificar que o problema (P_1) é um problema ilimitado ou um problema inviável.

O problema de otimização é um problema de PL quando as funções envolvidas – a função objetivo e as restrições do problema – são afins (lineares) e contínuas e, além disso, as variáveis do problema são contínuas.

A seguir, definimos poliedro como um poliedro convexo fechado.

Definição 1.2 Sejam dados um vetor não nulo $a \in \Re^n,$ denominado vetor normal, e um escalar $\delta \in \Re$.

(a) O conjunto

$$H = \{x \in \Re^n; a^T x = \delta\}$$

é denominado um hiperplano.

(b) Os conjuntos

$$H_l = \{x \in \Re^n; a^T x \leq \delta\} \text{ e } H_u = \{x \in \Re^n; a^T x \geq \delta\}$$

são denominados semiespaços fechados.

(c) Um poliedro é um conjunto formado pela interseção de um número finito de semiespaços fechados.

Agora, definiremos ponto extremo para, por exemplo, tratarmos do teorema fundamental da PL adiante.

Definição 1.3 Sejam dados q vetores $x^1, x^2, ..., x^q \in \Re^n$.

(a) Dizemos que $x \in \Re^n$ é uma combinação linear de $x^1, x^2, ..., x^q \in \Re^n$, quando existirem q escalares $\lambda_1, \lambda_2, \ldots, \lambda_q \in \Re$, tais que

$$x = \lambda_1 x^1 + \lambda_2 x^2 + ... + \lambda_q x^q.$$

(b) Dizemos que $x \in \Re^n$ é uma combinação convexa de $x^1, x^2, ..., x^q \in \Re^n$, quando x for uma combinação linear,

$$\lambda_1 + \lambda_2 + ... + \lambda_q = 1 \text{ e } \lambda_1, \lambda_2, ..., \lambda_q \in [0,1].$$

(c) Seja S um subconjunto do $\Re^n$. Dizemos que S é um conjunto convexo, quando todas as combinações convexas de quaisquer dois pontos de S pertencerem a S.

(d) Seja S um subconjunto convexo do $\Re^n$. Um ponto x em S é denominado ponto extremo de S, quando não for uma combinação convexa de quaisquer dois outros pontos distintos em S.

A seguir definimos solução básica viável através de outras definições importantes.

Definição 1.4 Sejam dados uma matriz A, $m \times n$, $0 < m < n$, e um vetor b em $\Re^m$. Considere um sistema de equações lineares $Ax = b$, tal que $posto(A) = m$.

(a) Uma matriz quadrada B, $m \times n$, obtida de A, com m vetores coluna linearmente independentes denomina-se matriz base de A. Uma matriz N, $m \times (n-m)$, obtida de A, com os $n-m$ vetores coluna restantes denomina-se matriz não base.

(b) Considere uma matriz base B, $m \times m$. O conjunto de índices correspondentes a esta matriz base B, no sistema $Ax = b$, chama-se conjunto de índices base. O conjunto

com os demais $n-m$ índices chama-se conjunto de índices não base. Denotamos o conjunto de índices base por I_B e o conjunto de índices não base por I_N.

(c) Considere uma matriz base B, $m \times m$. As variáveis correspondentes a esta matriz base B, no sistema $Ax = b$, chamam-se variáveis básicas. As demais $n-m$ variáveis chamam-se variáveis não básicas. Denotamos o vetor de variáveis básicas por x^B e o vetor de variáveis não básicas por x^N.

(d) Anulando as $n-m$ variáveis não básicas, obtemos um sistema compatível determinado, constituído de m equações e m incógnitas. Determinando o valor das variáveis básicas, obtemos uma solução básica. Ou seja, $x \in \Re^n$ é uma solução básica, quando $x^N = 0$ e x^B é a solução do sistema linear $Bx^B = b$.

(e) Uma solução básica em que as variáveis básicas são não negativas denomina-se solução básica viável.

(f) Uma solução básica viável onde existe ao menos uma variável básica nula denomina-se solução básica viável degenerada.

No teorema a seguir, apresentaremos ponto extremo de uma maneira mais operacional.

Teorema 1.1 Considere o problema de PL (P_1). Um ponto viável $x \in X$ é um ponto extremo se, e somente se, x for uma solução básica viável.

O próximo teorema caracteriza o conjunto viável de um PPL, formalizando a sua geometria.

Teorema 1.2 Considere o problema de PL (P_1). Todo conjunto viável X é um poliedro com um número finito de pontos extremos e, quando não vazio, possui ao menos um ponto extremo.

O teorema fundamental da programação linear será enunciado agora.

Teorema 1.3 Considere o problema de PL (P_1). Se (P_1) admite solução ótima, então uma solução ótima é atingida em ao menos um ponto extremo do conjunto viável.

Finalmente, vamos caracterizar o conjunto de soluções ótimas $X(P_1)$.

Teorema 1.4 Considere o problema de PL (P_1). O conjunto de soluções ótimas é um poliedro e, quando não vazio, possui uma única ou uma infinidade de soluções ótimas. Além disso, quando existir, ao menos uma solução ótima é um ponto extremo.

Complexidade de algoritmo define uma medida de custo necessário para executar um algoritmo para um problema de tamanho da cardinalidade do conjunto de entradas. A complexidade no pior caso dá o máximo custo. A complexidade no caso médio dá a média dos custos, levando em conta a probabilidade de ocorrência de cada entrada. A principal medida de complexidade é a medida em tempo, relacionada à velocidade. Complexidade assintótica é definida pelo crescimento da complexidade para entradas suficientemente grandes. Considere $f, g : N \to \Re_+$. Dizemos que $f(n)$ é $O(g(n))$ quando existem $c > 0$ e $n_0 \in N$ tais que, para todo $n \geq n_0$, $f(n) \leq cg(n)$. Dizemos Classe P à classe de problemas que são resolvidos em tempo polinomial por um algoritmo (determinístico). Por exemplo, o problema de PL pertence a classe P.

Na próxima seção estudaremos sobre problemas de programação linear inteira (PLI).

1.2. Problemas de programação linear inteira

Consideremos os números inteiros m e n tais que $0 < m < n$. Dados uma matriz numérica com coeficientes reais A, m x n, e vetores $b \in \Re^m$ e $c \in \Re^n$, uma formulação para o problema de programação linear inteira é o seguinte problema de otimização:

$$(P) \quad \text{minimizar} \quad c^T x$$

$$\text{sujeito a:} \quad Ax = b$$

$$x \in Z_+^n.$$

Por outro lado, uma formulação para o problema de programação linear inteira 0-1 (binário) é o seguinte problema de otimização:

$$(B) \quad \text{minimizar} \quad c^T x$$

$$\text{sujeito a:} \quad Ax = b$$

$$x \in \{0,1\}^n.$$

Em ambos os problemas (P) e (B), a função objetivo é definida pela função linear $x \mapsto c^T x$; o valor da função objetivo definido pelo número $c^T x$; o conjunto viável definido pelo conjunto

$$XI = \{x \in Z^n; Ax = b, x \geq 0\},$$

ou

$$XB = \{x \in \{0,1\}^n; Ax = b\};$$

um ponto viável definido como um elemento do conjunto viável; o conjunto de soluções ótimas definido pelo conjunto

$$X(P) = \{x^* \in XI; c^T x^* \leq c^T x, \text{ para todo } x \in XI\};$$

ou

$$X(B) = \{x^* \in XB; c^T x^* \leq c^T x, \text{para todo } x \in XB\};$$

uma solução ótima definida como um elemento do conjunto de soluções ótimas; o problema (P) é inviável quando XI é vazio, e o problema (B) é inviável quando XB é vazio; e o problema (P) é ilimitado quando existe uma sequência (x^k) tal que $x^k \in XI$ e, quando $k \to \infty$, $c^T x^k \to -\infty$.

Seguem-se as definições de minimizador global e minimizador local.

Definição 1.5 Sejam dados um conjunto $D \subset Z^n$ e uma função $f : D \to \Re$.

(a) Dizemos que um ponto $\bar{x} \in D$ é um minimizador global de f em D quando, para todo $x \in D$,

$$f(\bar{x}) \leq f(x).$$

(b) Dizemos que um ponto $\bar{x} \in D$ é um minimizador local de f em D quando, existe um $\varepsilon \geq 1$ tal que, para todo $x \in \{x \in D; \| x - \bar{x} \|_\infty \leq \varepsilon\}$,

$$f(\bar{x}) \leq f(x).$$

A proposição a seguir mostra que o problema de PLI (P) pode ser reduzido ao problema (B), conforme Salkin (1975).

Proposição 1.1 Suponha que no problema de PLI (P) cada $x_j \leq u_j$, com $u_j > 0$, $j = 1, \ldots, n$. Então, este novo problema de PLI pode ser reduzido ao problema de PLI 0-1 (B).

A definição a seguir nos auxilia a olhar para a resolução de um problema de PLI através de métodos de programação linear (contínua).

Definição 1.6 Uma matriz quadrada com coeficientes inteiros B é chamada matriz unimodular, quando seu determinante é igual a -1 ou igual a 1. Uma matriz com coeficientes inteiros A é chamada matriz totalmente unimodular, quando qualquer submatriz quadrada não singular de A é unimodular.

O teorema a seguir mostra que em um problema de PLI pode ser melhor resolvê-lo como um problema de programação linear, dependendo da matriz A satisfazer a condição de unimodularidade total. Veja Papadimitriou & Steiglitz (1998).

Teorema 1.5 Se A é uma matriz totalmente unimodular, então todos os pontos extremos de

$$X = \left\{ x \in \Re^n ; Ax = b, x \geq 0 \right\}$$

são vetores com coordenadas inteiras para qualquer vetor de inteiros b.

A recíproca deste teorema é falsa. Por exemplo, defina

$$X = \left\{ x \in \Re^n ; Ax = b, x \geq 0 \right\}$$

com

$$A = \begin{bmatrix} 2 & 3 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix} \text{ e } b = \begin{bmatrix} 5 \\ 1 \\ 1 \end{bmatrix}.$$

Com efeito, os pontos extremos

$$\begin{bmatrix} 0 \\ 0 \\ 5 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 2 \\ 1 \\ 0 \end{bmatrix}$$

são todos vetores com coordenadas inteiras, todavia a submatriz quadrada não singular

$$B = \begin{bmatrix} 2 & 3 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

possui $\det(B) = -2$; que é diferente tanto de -1 quanto de 1.

A seguir veremos porque o arredondamento nem sempre funciona. Esse exemplo pode ser encontrado em Goldbarg & Luna (2005).

Considere o problema

$$\text{maximizar } x_1 + 21x_2$$

$$\text{sujeito a: } x_1 + 20x_2 \leq 50$$

$$x_1 + x_2 \geq 20$$

$$x_1, x_2 \in Z_+,$$

cuja solução ótima aproximada do problema relaxado é $\hat{x}_1 = 18{,}42$, $\hat{x}_2 = 1{,}57$ com $c^T \hat{x} = 51{,}39$. Aplicando uma estratégia de arredondamento, obteríamos $x_1 = 18$ e $x_2 = 1$, ponto inviável; $x_1 = 18$ e $x_2 = 2$, ponto inviável; $x_1 = 19$ e $x_2 = 1$, com $c^T x = 40$; e $x_1 = 19$ e $x_2 = 2$, ponto inviável; ou seja, o erro é aproximadamente 22% no arredondamento para $x_1 = 19$ e $x_2 = 1$, pois a solução ótima inteira é $x_1^* = 30$, $x_2^* = 1$ com $c^T x^* = 51$. Com um número maior de variáveis e com essa margem de erro a técnica de arredondamento pode resultar em uma derrocada completa no esforço de modelagem e solução, impondo-se outros métodos de solução.

Dizemos classe NP à classe de problemas que são resolvidos em tempo polinomial por um algoritmo não determinístico. Dizemos que um problema B está na classe NPC, digo NP-completo, quando B está na classe NP e, para todo problema A em NP, A é redutível em tempo polinomial a B. Dizemos que um problema B está na classe NP-Difícil, quando para todo

problema A em NP, A e redutível em tempo polinomial a B. Por exemplo, o problema (geral) de PLI pertence a classe NPC.

Na próxima seção estudaremos sobre grafos.

1.3 Grafos

Nessa seção estaremos interessados na teoria básica de grafos. Iniciamos o nosso propósito com a definição de grafos e grafos simples.

Definição 1.7 Um grafo G é um par ordenado $(V(G), E(G))$ que consiste de um conjunto não vazio e finito $V(G)$ de vértices de G e um conjunto finito $E(G)$, disjunto de $V(G)$, de arestas de G, junto com uma função de incidência ψ_G que associa a cada aresta de G um par não ordenado de vértices de G.

Considere um grafo G. Se e é uma aresta e u e v são vértices tais que $\psi_G(e) = uv$, então e é dito unir os vértices u e v, e os vértices u e v são chamados as extremidades de e. As extremidades de uma aresta são ditas incidentes com a aresta e vice-versa. Dois vértices que são incidentes com uma aresta comum são ditos adjacentes e, da mesma forma, duas arestas que são incidentes com um vértice comum. Uma aresta com extremidades idênticas é chamada laço. Uma aresta com extremidades distintas é chamada ligação . Duas ou mais ligações com o mesmo par de extremidades são chamadas arestas paralelas. Qualquer grafo com um único vértice é chamado grafo trivial. Podemos encontrar o termo multigrafo para a Definição 1.7.

Definição 1.8 Dizemos grafo simples ao grafo sem laços e sem arestas paralelas.

Podemos encontrar o termo grafo; simplesmente, para a Definição 1.8.

Agora, definiremos alguns grafos usuais e grau de vértices.

Definição 1.9

(a) Um grafo simples em que quaisquer dois vértices são adjacentes é chamado grafo completo. O grafo completo com n vértices é denotado por K_n.

(b) Um grafo bipartido é aquele cujo conjunto de vértices pode ser particionado em dois subconjuntos X e Y, tais que cada aresta tem uma extremidade em X e outra em Y. Uma tal partição (X,Y) é chamada uma bipartição do grafo.

(c) Um grafo bipartido completo é um grafo simples bipartido com bipartição (X,Y) em que cada vértice de X é unido a cada vértice de Y. O grafo bipartido completo com $|X| = p$ e $|Y| = q$ é denotado por $K_{p,q}$.

(d) Considere um grafo G. Chamamos grau de um vértice v em G, denotado por $d(G) = d_G(v)$, o número de arestas de G incidentes a v, tal que cada laço é contado como duas arestas.

A seguir apresentaremos as matrizes de incidência e adjacência, as quais são uma maneira de armazenar um grafo no computador.

Definição 1.10 Seja G um grafo com conjunto de vértices V, conjunto de arestas E e função de incidência ψ_G.

(a) Considere $i = 1,...,n$ e $j = 1,...,m$. Chamamos matriz de incidência de G a matriz $n \times m$, denotada por $M(G) = [m_{ij}]$, em que m_{ij} é o número de vezes que o vértice v_i e a aresta e_j são incidentes.

(b) Considere $i = 1,...,n$ e $j = 1,...,n.$ Chamamos matriz de adjacência de G a matriz $n \times n,$ denotada por $A(G) = [a_{ij}],$ em que a_{ij} é o número de arestas unindo o vértice v_i com o vértice $v_j,$ com cada laço contando como duas arestas.

Definições acerca de subgrafos serão apresentadas a seguir.

Definição 1.11

(a) Dizemos que um grafo H é subgrafo de um grafo $G,$ denotado por $H \subseteq G,$ quando $V(H) \subseteq V(G),$ $E(H) \subseteq E(G)$ e ψ_H é a restrição de ψ_G para $E(H).$

(b) Dizemos que um grafo H é subgrafo gerador de G quando $H \subseteq G$ com $V(H) = V(G).$

(c) Suponha V' um subconjunto não vazio de $V(G).$ O subgrafo de $G,$ cujo conjunto de vértices é V' e cujo conjunto de arestas é o conjunto de arestas de G que tem ambas as extremidades em $V',$ é chamado subgrafo de G induzido por $V',$ denotado por $G[V'].$

(d) Suponha E' um subconjunto não vazio de $E(G).$ O subgrafo de $G,$ cujo conjunto de vértices é o conjunto de extremidades de arestas de E' e cujo conjunto de arestas é $E',$ é chamado subgrafo de G aresta-induzido por $E',$ denotado por $G[E'].$

A seguir, definiremos caminho e ciclo em um grafo.

Definição 1.12 Sejam G um grafo e k um inteiro positivo.

(a) Um passeio de G é uma sequência finita e não nula

$$W = v_0 e_1 v_1 e_2 v_2 ... e_k v_k ,$$

cujos termos são alternadamente vértices e arestas, tais que, para $1 \leq i \leq k$, as extremidades da aresta e_i são os vértices v_{i-1} e v_i. Os vértices v_0 e v_k são as extremidades do passeio. O inteiro positivo k é o comprimento de W. Dizemos passeio fechado de G quando as extremidades do passeio são os mesmos vértices.

(b) Quando as arestas $e_1, e_2, ..., e_k$ de um passeio W são distintas, W é chamado uma trilha de G. Quando as arestas de um passeio fechado W são distintas, W é chamado uma trilha fechada de G.

(c) Quando os vértices $v_0, v_1, ..., v_k$ de um passeio W são distintos, W é chamado um caminho de G.

(d) Uma trilha fechada cuja origem e vértices internos são distintos é chamada de ciclo de G.

Agora, vamos definir conexidade em grafos.

Definição 1.13 Considere G um grafo. Dizemos que G é um grafo conexo quando existir um caminho entre quaisquer par de seus vértices. Caso contrário, o grafo é chamado grafo desconexo. Dois vértices u e v de G é dito estarem conectados quando existir um caminho de u a v em G.

O próximo resultado caracteriza grafos bipartidos.

Teorema 1.6 Um grafo G é um grafo bipartido se, e somente se, todos os ciclos de G possuem comprimento par.

Agora definiremos grafos direcionados.

Definição 1.14 Um grafo direcionado (ou digrafo) D é um par ordenado $(V(D), A(D))$ que consiste de um conjunto $V(D)$ de vértices (ou nós) e um conjunto $A(D)$,

disjunto de $V(D)$, de arestas (ou arcos), junto com uma função de incidência ψ_D que associa a cada aresta de D um par ordenado, não necessariamente distinto, de vértices de D.

A seguir apresentaremos a idéia do algoritmo Húngaro. Esse exemplo pode ser encontrado em Boaventura & Jurkiewicz (2009).

Considere o problema onde temos que alocar 3 turmas para 3 salas. Pelas características de cada turma e de cada sala são atribuídos custos de alocação. Estes custos são dados na tabela a seguir:

Matriz de custos

Turma→ Sala↓	**1**	**2**	**3**
1	3	5	6
2	5	4	2
3	2	3	4

Podemos perceber que, ao atribuir uma turma a cada sala, estamos tomando três elementos da matriz tais que:

(a) cada elemento está em uma linha diferente;

(b) cada elemento está em uma coluna diferente;

(c) cada linha e cada coluna contém exatamente um elemento.

Uma solução com estas características é chamada uma solução viável. Queremos que o custo desta solução seja mínimo.

Do ponto de vista da teoria dos grafos temos um grafo bipartido valorado e estamos procurando uma solução viável com custo mínimo. Veja figura 1.

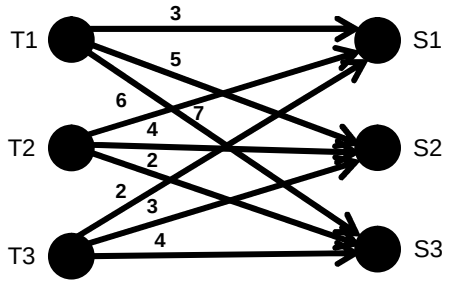


Figura 1: Grafo bipartido valorado

Podemos olhar também este problema como um problema de programação linear inteira 0-1 (binário):

$$\textit{mininizar}\ 3x_{11}+5x_{12}+6x_{13}+5x_{21}+4x_{22}+2x_{23}+2x_{31}+3x_{32}+4x_{33}$$

sujeito a: $x_{11}+x_{12}+x_{13}=1$

$x_{21}+x_{22}+x_{23}=1$ (A cada sala corresponde apenas uma turma)

$x_{31}+x_{32}+x_{33}=1$

$x_{11}+x_{21}+x_{31}=1$

$x_{12}+x_{22}+x_{32}=1$ (A cada turma corresponde apenas uma sala)

$x_{13}+x_{23}+x_{33}=1$

$x_{ij}\in\{0,1\},\ i=1,2,3\ e\ \ j=1,2,3.$

Um exemplo simples de solução viável é $\{x_{11},x_{22},x_{33}\}$, com custo igual a $3+4+4=11.$ O número de soluções viáveis é igual a $3!=6$ e por inspeção podemos verificar que a solução de custo mínimo é $\{x_{11},x_{23},x_{32}\}$, com custo 8.

Se nossa matriz for de ordem maior, a resolução por inspeção torna-se inviável. Vamos desenvolver então uma idéia bem simples do algoritmo Húngaro.

Primeiramente, observamos que o valor de nossa solução não se altera se somarmos ou subtrairmos um mesmo valor de todos os elementos de uma linha (ou coluna). De fato, só um

dos elementos afetados estará na solução mínima. A solução da nova matriz terá o seu valor diminuído no número subtraído de unidades, mas os elementos da solução serão os mesmos. Por exemplo, diminuindo 3 na primeira linha e colocando asterisco nos menores elementos das linhas, obtemos

$$\begin{vmatrix} 3 & 5 & 6 \\ 5 & 4 & 2 \\ 2 & 3 & 4 \end{vmatrix} \begin{matrix} -3 \\ \\ \\ \end{matrix} \rightarrow \begin{vmatrix} 0 & 2 & 3 \\ 5 & 4 & 2 \\ 2 & 3 & 4 \end{vmatrix} \rightarrow \begin{vmatrix} 0^{*} & 2 & 3 \\ 5 & 4 & 2^{*} \\ 2 & 3^{*} & 4 \end{vmatrix}.$$

A solução é a mesma, isto é, a mesma permutação, mas o valor ficou diminuído em 3 unidades. Vamos completar o trabalho com as linhas e depois aplicar o mesmo princípio às colunas, mas não ao mesmo tempo. Usando esta idéia no nosso exemplo temos:

$$\begin{vmatrix} 3 & 5 & 6 \\ 5 & 4 & 2 \\ 2 & 3 & 4 \end{vmatrix} \begin{matrix} -3 \\ -2 \\ -2 \end{matrix} \rightarrow \begin{vmatrix} 0 & 2 & 3 \\ 3 & 2 & 0 \\ 0 & 1 & 2 \end{vmatrix} \rightarrow \begin{vmatrix} 0^{*} & 1 & 3 \\ 3 & 1 & 0^{*} \\ 0 & 0^{*} & 1 \end{vmatrix}$$

$$-1$$

A solução $\{x_{11}, x_{23}, x_{32}\}$ está agora bastante evidente; basta procurar os zeros. Observemos que o custo 8 é dado pela soma dos valores subtraídos.

O enunciado do algoritmo Húngaro pode ser encontrado em Kuhn (1995).

No próximo capítulo desenvolveremos uma formulação específica para o problema de designação de salas de aula.

CAPÍTULO II - UMA FORMULAÇÃO ESPECÍFICA PARA O PROBLEMA DE DESIGNAÇÃO DE SALAS DE AULA

Nesse capítulo será apresentada uma formulação, no processo de modelagem, para o problema de designação de salas de aula da área 3, campus I, na PUC Goiás.

Iniciamos nosso propósito apresentando uma introdução ao problema de designação de salas de aula.

2.1 Problemas de designação de salas de aula

Schaerf (1999) discute o problema de programação de horários (*timetabling problem*) automático e observa que das variantes desse problema propostas na literatura, o que difere umas das outras baseia-se no tipo de instituição envolvida e no tipo das restrições. A partir daí, classifica-se o problema de programação de horários em três classes*: school timetabling, course timetabling* e *examination timetabling*. Aqui estamos interessados em *course timetabling problem*. Em particular, *classroom assignment problem.*

Segundo Carter & Laporte (1998), os termos importantes que compõem o entendimento do problema que pretendemos estudar são *a program*, *a course*, *a class* (ou *a course section*) os quais são traduzidos e definidos, a saber:

- *a program*: curso, isto é, consiste de um conjunto de disciplinas agrupadas por grade curricular que representam projetos pedagógicos que cada aluno deve cumprir para se colar grau;

- *a course*: disciplina, isto é, consiste de uma ementa para ser cumprida geralmente em um semestre.

- *a class* (ou *a course section*): turma, isto é, consiste de subdivisões de uma disciplina em grupos distintos de estudantes podendo ser ministradas por professores diferentes e , também, em dias, horários e salas diferentes.

Assim, traduzimos *course timetabling problem* como problema de programação da grade de horários para disciplinas, isto é, um problema de designação multi-dimensional em que alunos e professores são designados para disciplinas, turmas ou períodos com encontros realizados em salas de aula em determinados horários; e *classroom assignment problem* como problema de designação de salas de aula, isto é, um problema que consiste em encontrar, se possível, uma sala de aula aceitável para cada turma nos dias e horários especificados considerando características de cada turma e de cada sala de aula.

Carter & Laporte (1998) escreveram sobre *course timetabling* abordando artigos publicados entre 1980 e 1998, nos quais implementaram-se problemas práticos ou testaram-se dados reais em universidades. Em particular, sobre *classroom assignment* referenciaram-se duas implementações práticas, Carter (1989) e Glassey & Mizrach (1986), e um teste com dados reais, Gosselin & Truchon (1986). Em 2007, McCollum (2007) confirma, quase uma década depois, os poucos trabalhos para resolução de problemas de horários em universidades com

dados reais. Recentemente, Rudová, Müller & Murray (2011) realizaram uma implementação prática em uma universidade para o problema de *course timetabling* desenvolvendo uma nova e promissora metodologia de resolução tal que, resolvem dentre outras variantes, o problema de *classroom assignment*. Recentemente, também, Subramanian, Medeiros, Formiga & Souza (2011) resolveram um problema de *classroom assignment* para um centro de tecnologia em uma universidade. Atualmente, existem comunidades para o problema da programação de horários (*community timetabling*), conforme Schaerf (1999). E, também, competições, conforme McCollum, McMullan, Paechter, Lewis, Schaerf, Di Gaspero, Parkes, Qu & Burke (2010).

Em problemas de designação de salas de aula consideramos dois tipos de restrições, conforme Burke, Jackson, Kingston & Weare (1997): *hard constraint* denominadas requisitos essenciais, isto é, não satisfazer essas restrições inviabiliza o problema, e *soft constraint* denominadas requisitos não essenciais, isto é, não satisfazer essas restrições não inviabiliza o problema.

Em virtude do problema geral de designação de salas de aula pertencer à classe NP-completo (NPC), conforme Carter & Tovey (1992), estes problemas geralmente são resolvidos através de algoritmos que fazem uso de heurísticas ou metaheurísticas, os quais necessitam, após a sua resolução, da interferência humana, quer dizer, o administrador responsável deve retirar as inviabilidades da solução obtida. Todavia, o problema que estudaremos aqui será da classe P, uma vez que resolveremos o problema de designação de salas de aula como um problema de designação (*assignment problem*), isto é, horário por horário (Carter e Tovey 1992).

Na próxima seção apresentaremos a área 3, campus I, na PUC Goiás.

2.2 O centro técnico e científico

A área 3, campus I, na PUC Goiás, consiste em uma das áreas do campus I, onde se localiza o centro técnico e científico com quatro departamentos, a saber: Artes e Arquitetura (ARQ), com dois cursos (Arquitetura e Urbanismo, e Design); Computação (CMP), com três cursos (Ciência da Computação, Engenharia de Computação e Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas); Engenharia (ENG), com cinco cursos (Engenharia Ambiental, Engenharia Civil, Engenharia de Controle e Automação, Engenharia de Produção e Engenharia Elétrica); e Matemática e Física (MAF), com quatro cursos (Engenharia de Alimentos, Física, Matemática e Química). São treze blocos com aulas de preleção (aulas teóricas) apenas em salas com carteiras em três blocos (C, E e F). Nestes blocos, algumas salas têm carteiras e outras pranchetas. Salas com pranchetas são designadas pelo responsável do departamento de ARQ. As aulas são ministradas de segunda-feira a sexta-feira nos três períodos (matutino, vespertino e noturno) e sábado no período matutino.

Neste semestre, 2012/01, existem 609 disciplinas de graduação (e um tecnológico) totalizando 1254 turmas para serem alocados em 32 salas de aula na área 3, campus I, com oito horários diários sem sobreposição, exceto sábado com três. Aqui, os cursos de pós-graduação em mestrado e doutorado são ministrados em salas específicas. Já disciplinas com aulas em laboratórios e ateliês são designadas pelos responsáveis de cada departamento. Práticas, estágios e orientações não necessitam de salas de aula necessariamente. Ainda, em virtude do elevado número de turmas em relação ao número de salas de aula, estão disponíveis ainda mais 49 salas de aula nos blocos B, C e D da área 2, campus I. Desta forma, o problema é constituído de 1254 turmas dispostas em 43 horários diferentes durante a semana, nos turnos matutino, vespertino e noturno. Que devem ser alocadas para 81 salas dispostas em 6 blocos diferentes e, também, duas áreas diferentes.

Para o nosso problema de designação de salas de aula, uma turma será especificada como um grupo de alunos e professores de uma disciplina de um curso que se localiza na área 3, campus I, em um período do dia (matutino, vespertino ou noturno), início e término do horário de aula, dia da semana, número de créditos e quantidade de hora-aula. Por exemplo, a turma t_1

é composta de: curso Engenharia de Computação, campus I, área 3, disciplina Teoria da Computação (código CMP1150), período vespertino (código B01 – em particular, na PUC Goiás, denominamos turma B01), horário das 15h10min às 16h40min, dia da semana segunda-feira, número de créditos igual a 6, hora-aula igual a 1h30 min, e professor Geovane. É importante afirmar que, existem turmas que não necessitam de salas de aula, por exemplo, disciplinas de trabalho de conclusão de curso geralmente são desenvolvidas entre o aluno e o professor em um ambiente com uma mesa e duas cadeiras: não necessariamente em uma sala de aula.

Ainda, para o nosso problema de designação de salas de aula, uma sala de aula (ou simplesmente sala) será especificada como um espaço físico localizado na área 3, campus I, para a realização de aulas, em um bloco com número de sala. Por exemplo, a sala s_1 é composta de: número da sala 303, bloco E, área 3 e campus I. Aqui, considera-se sala o local de aulas de preleção.

A PUC Goiás possui um sistema de banco de dados de registro de alunos e alocação de horários desenvolvido pelo Centro de Processamento de Dados (CPD). As principais fases do sistema consistem na designação de turmas-horários e de turmas-professores realizados manualmente pelos Departamentos (de cursos), fornecendo os dados para o sistema e, em seguida, a Coordenação de Programação Acadêmica (CPA) realiza manualmente a designação de turmas-salas de aula. Laboratórios, práticas, estágios e orientações são alocados, manualmente, pelos Departamentos.

Segundo a Coordenadora da CPA, no momento existem seis restrições importantes para a gerência do problema de designação de salas de aula. As duas primeiras restrições abaixo são requisitos essenciais (*hard constraint*), enquanto que as quatro últimas são requisitos não essenciais (*soft constraint*):

R_1: uma sala terá no máximo uma turma no mesmo horário;

R_2: cada turma que necessita de sala com destinação própria deverá ser alocada para exatamente uma sala desse tipo, com sua capacidade mínima e máxima respeitada.

R_3: cada turma poderá ter a necessidade de salas diferenciadas. Por exemplo, em alguns períodos algumas salas são impraticáveis pelo ruído que vem da rua, da praça, ou a temperatura da sala em determinados horários é bastante elevada, ou professores que utilizam com frequência o datashow têm preocupação com a luminosidade;

R_4: cada turma deverá ser alocada para uma sala próxima ao bloco de seu curso.

R_5: turmas de mesmo curso deverão ser alocadas preferencialmente para salas no mesmo bloco na semana.

R_6: turmas de mesmo período da grade curricular de um curso deverão ser alocadas preferencialmente para a mesma sala na semana.

Na próxima seção faremos uso dessa particular descrição da área 3, campus I, na PUC Goiás, para apresentar uma formulação, através de um modelo matemático, em uma tentativa de traduzir a realidade do problema de designação de salas de aula do centro técnico e científico.

2.3 Formulação

Nossa primeira hipótese é a de que as turmas já estão designadas para um determinado horário do dia. Nossa segunda hipótese é a de que existem salas suficientes de tamanho apropriado para acomodar todas as turmas em todos os períodos, se necessário, as quais estão disponíveis. Do ponto de vista do problema de programação da grade de horários para disciplinas (*course timetabling problem*), nossa terceira hipótese é a de que os professores já estão designados para determinadas turmas com seus respectivos horários do dia.

O problema de designação de salas de aula, então, consiste em alocar essas turmas (código da disciplina, código da turma, nome do professor, início e término do horário de aula, nome do curso, campus I, área 3, dia da semana, número de créditos, quantidade de hora-aula) para salas disponíveis (número da sala, letra do bloco, área 3, campus I) em seus respectivos horários diários.

Nesse caso, definimos x_{ij}, $i = 1,2,...,m$ e $j = 1,2,...,n$, as variáveis de decisão que pretendemos encontrar, se existir, a saber:

$$x_{ij} = \begin{cases} 1, & \textit{se a turma i é designada para a sala } j, \\ 0, & \textit{caso contrário.} \end{cases}$$

Cada sala recebe a alocação de no máximo uma turma no seu respectivo horário, isto é, deve-se prevenir de alguma sala ter mais de uma turma no mesmo horário. Assim, considere os horários k, $k = 1,2,...,t.$ Defina P_k o conjunto que representa todas as turmas que se encontram no horário $k.$ Dessa forma, obtemos

$$\sum_{i \in P_k} x_{ij} \leq 1, \ j = 1,2,...,n, \ k = 1,2,...,t.$$

Algumas turmas não necessitam de salas. Todavia, daquelas que necessitam, cada uma deve ser alocada para exatamente uma sala respeitando sua capacidade, isto é, deve-se prevenir de alguma dessas turmas não terem sala. Assim, defina um conjunto de salas S_i que respeitem a capacidade de cada turma (de preleção) i, $i = 1,2,...,m.$ Dessa forma, obtemos

$$\sum_{j \in S_i} x_{ij} = 1, \ i = 1,2,...,m.$$

Aqui, o nosso objetivo é minimizar um custo para a designação que chamaremos c_{ij}, $i = 1,2,...,m$ e $j = 1,2,...,n$, a saber:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij.}$$

Mas quem é o custo c_{ij} ? Veremos agora.

Segundo o requisito não essencial R_3, pode-se ter a necessidade de salas diferenciadas para cada turma. Considere que toda turma que não necessita de alocação de salas não é submetida ao modelo de distribuição de salas. Dessa forma, é razoável afirmarmos o seguinte:

(a) turmas para salas em cujos horários o volume de ruido é exagerado, tem custo 30 (trinta);

(b) turmas para salas em cujos horários a temperatura é bastante elevada, tem custo 20 (vinte);

(c) turmas cuja disciplina exija uma metodologia diferenciada e, portanto, salas, em determinados horários, que ofereçam essas condições, tem custo 1 (um);

(d) turmas cuja disciplina exija uma metodologia diferenciada e em salas, em determinados horários, que não ofereçam essas condições, tem custo 70 (setenta);

(e) turmas que não atendam os itens (a), (b), (c) e (d), tem custo 10 (dez).

Segundo o requisito não essencial R_4, deve-se alocar cada turma para uma sala próxima ao bloco de seu curso. A CPA fornece esses valores respeitando as particularidade de cada situação.

Segundo o requisito não essencial R_5, deve-se alocar turmas de mesmo curso para salas no mesmo bloco na semana. Esse requisito está incorporado em R_4, tomando parâmetros iguais a 0 (zero) para atender a preferência de cursos por determinado bloco ou determinados blocos.

Segundo o requisito não essencial R_6, deve-se alocar turmas de mesmo período da grade curricular de um curso para a mesma sala na semana. Isso sugere resolvermos o problema horário por horário aproveitando a solução atual para introduzir pesos no valor da função objetivo com a intenção de atender esse requisito nos próximos horários.

Então, o custo é calculado pela soma dos pesos dos requisitos não essenciais R_3, R_4, R_5 e R_6.

Portanto, uma formulação para o problema de designação de salas de aula é dado pelo problema de programação linear inteira 0-1,

$$(PDS)\ \textit{minimizar} \qquad \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}\, x_{ij}$$

$$\textit{sujeito a:} \qquad \sum_{i \in P_k} x_{ij} \leq 1,\ j = 1,2,...,n,\ k = 1,2,...,t,$$

$$\sum_{j \in S_i} x_{ij} = 1,\ i = 1,2,...,m,$$

$$x_{ij} \in \{0,1\},\ i = 1,2,...,m \text{ e } j = 1,2,...,n.$$

No próximo capítulo apresentaremos os nossos resultados.

CAPÍTULO III - RESULTADOS

Nesse capítulo apresentaremos os resultados computacionais obtidos pela aplicação do algoritmo Húngaro proposto no capitulo I, aplicando-se a formulação de programação matemática proposta no capitulo II para o problema de designação de salas de aula da área 3, campus I, na PUC Goiás.

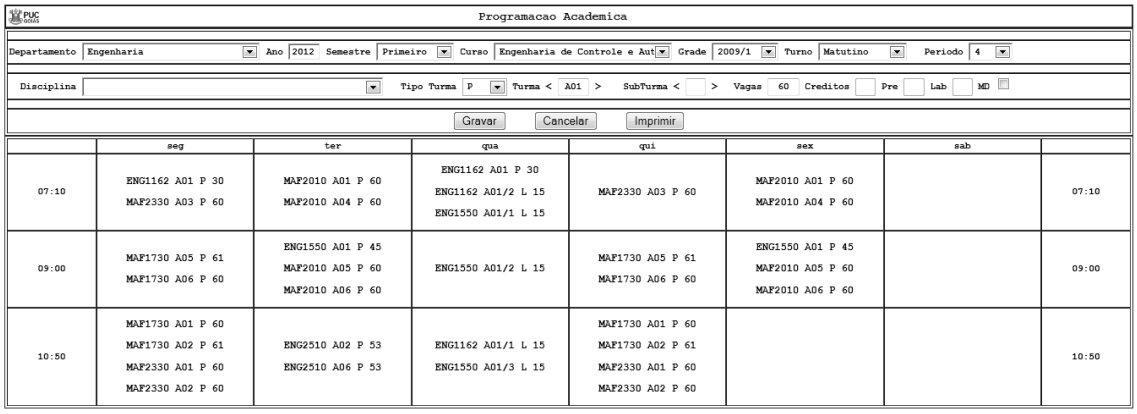
A opção pelo algoritmo Húngaro é devida ao trabalho de Neves (2010), quem resolve um problema para tamanho um tanto menor, e sugere o uso deste algoritmo para este problema.

Iniciamos nosso propósito com o problema da pesquisa.

3.1 O problema

O problema da pesquisa a ser solucionado é o problema de designação de salas de aula para o centro técnico cientifico da áreas 2 e 3, campus I da PUC Goiás.

Como problema para a viabilização da idéia de resolver o problema de designação horário por horário, foi realizada uma pesquisa *in loco* na PUC Goiás para obter os dados reais. Inicialmente foi conferido os dados das salas de aula (tamanho, capacidade, temperatura, ruidos e metodologia diferenciada (telão e quadro branco)), depois digitados esses dados em uma planilha e introduzidos os respectivos pesos. Os departamentos forneceram os relatórios “Relações de Disciplinas/Unidades Oferecidas com Professores”. Logo em seguida foi usado o programa desenvolvido pelo professor Ivon Rodrigues Canedo, para inserir os dados, que fornece um ambiente com interface web, e salva em um banco de dados. Veja a figura 2.



	seg	ter	qua	qui	sex	sab	
07:10	ENG1162 A01 P 30 MAF2330 A03 P 60	MAF2010 A01 P 60 MAF2010 A04 P 60	ENG1162 A01 P 30 ENG1162 A01/2 L 15 ENG1550 A01/1 L 15	MAF2330 A03 P 60	MAF2010 A01 P 60 MAF2010 A04 P 60		07:10
09:00	MAF1730 A05 P 61 MAF1730 A06 P 60	ENG1550 A01 P 45 MAF2010 A05 P 60 MAF2010 A06 P 60	ENG1550 A01/2 L 15	MAF1730 A05 P 61 MAF1730 A06 P 60	ENG1550 A01 P 45 MAF2010 A05 P 60 MAF2010 A06 P 60		09:00
10:50	MAF1730 A01 P 60 MAF1730 A02 P 61 MAF2330 A01 P 60 MAF2330 A02 P 60	ENG2510 A02 P 53 ENG2510 A06 P 53	ENG1162 A01/1 L 15 ENG1550 A01/3 L 15	MAF1730 A01 P 60 MAF1730 A02 P 61 MAF2330 A01 P 60 MAF2330 A02 P 60			10:50

Figura 2- Sistema de Programação Acadêmica

O banco de dados armazena informações referentes ao primeiro semestre do ano de 2012. Desta forma, o problema é constituído de 609 disciplinas dispostas em 43 horários diferentes durante a semana, nos turnos matutino, vespertino e noturno. Em um total de 1254 turmas que devem ser alocadas para 81 salas dispostas em 6 blocos diferentes. Essas turmas são dos cursos de Arquitetura e Urbanismo, Design, Ciência da Computação, Engenharia de Computação, Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Engenharia Ambiental, Engenharia Civil, Engenharia de Controle e Automação, Engenharia de Produção e Engenharia Elétrica, Matemática, Física, Engenharia de Alimentos, e Química.

Na próxima seção o ambiente de teste é apresentado.

3.2 Ambiente de testes

Os testes aqui relatados foram executados em um computador com um processador Intel Pentium Dual-Core, com clock de 2.3 GHz e 2 Gb de memória RAM disponível. O sistema operacional utilizado foi Linux 2.6.32 de 32 bits.

Para efeito de testes, foi mantida uma cópia do banco de dados em um computador com as configurações descritas. Este computador possui um servidor web

local, de forma que é possível simular toda a comunicação do algoritmo com a interface do programa com o banco de dados, coletar as informações necessárias e modelar o problema para a execução dos algoritmos.

Todo o processo passa pelas seguintes etapas:

1. Um código PYTHON consulta o banco de dados, obtém os dados das turmas e salas ordenadas por horários, salva em um arquivo para o programa executável fazer a leitura do mesmo.

2. O programa executável, denominado SATS (Sistema de Alocação de Turmas-Salas), lê do arquivo para a memória as informações das turmas e salas.

3. O programa executável calcula a quantidade de horários e a quantidade de turmas por horário.

4. O programa executável seleciona os dados do próximo horário a ser executado.

5. O programa executável calcula a matriz de custos com base nos requisitos R_3, R_4, R_5 e R_6.

6. Se não for o primeiro horário da semana, o programa verifica a solução do horário anterior e atualiza a matriz de custos de acordo com o requisito R_6.

7. O programa executa o algoritmo e salva o resultado em arquivo.

8. Se ainda houver horários a serem processados, então o programa retorna

para a etapa 4.

9. Um código PYTHON grava todos os resultados no banco de dados, finalizando a alocação.

Para facilitar a visualização dos resultados, foi implementado uma tabela no banco de dados que gera outra tabela com os dados das turmas e salas devidamente alocados.

Na próxima seção serão apresentados os resultados obtidos pela pesquisa.

3.3 Resultados

Foi utilizada uma implementação do algoritmo Húngaro em C/C++ para o Matlab. O algoritmo foi executado para toda a área 3 incluindo as salas da área 2. Os dados apresentados na figura 3 mostram que todos os requisitos essenciais são satisfeitos, isto é, não existe sala com mais de uma turma no mesmo horário e nem turmas sem salas. Essa solução viável levou menos de 6 segundos para gerar os resultados.

Hora Início Aula	Dia Semana	Turma	Nome Turma	Disciplina	Sala	Bloco	Área	Período
"07:10"	"seg"	33	"A02"	"Estrutura de Dados I"	"401"	"C"	3	3
"07:10"	"seg"	52	"A18"	"Lingua Portuguesa I"	"503"	"C"	3	6
...	...	...	...	...	...	...	...	...
"09:00"	"seg"	34	"A02"	"Calculo Diferencial e Integral II"	"204"	"E"	3	3
"09:00"	"seg"	44	"A02"	"Paradigmas de Programacao"	"302"	"E"	3	4
...	...	...	...	...	...	...	...	...
"10:50"	"seg"	45	"A03"	"Probabilidade e Estatistica"	"503"	"C"	3	4
"10:50"	"seg"	54	"A01"	"Linguagens Formais e Automatos"	"504"	"C"	3	6
...	...	...	...	...	...	...	...	...
"13:20"	"seg"	174	"B01"	"Banco de Dados"	"204"	"F"	3	7
"13:20"	"seg"	2078	"B01"	"Administracao e Financas para Engenharia"	"306"	"F"	3	8
...	...	...	...	...	...	...	...	...
"15:10"	"seg"	168	"B01"	"Redes de Computadores"	"304"	"E"	3	6
"15:10"	"seg"	2079	"B01"	"Inteligência Artificial"	"504"	"C"	3	8
...	...	...	...	...	...	...	...	...
"17:00"	"seg"	169	"B02"	"Lingua Portuguesa I"	"201"	"E"	3	6
"17:00"	"seg"	180	"B01"	"Sistemas de Software"	"202"	"E"	3	8
...	...	...	...	...	...	...	...	...
"18:45"	"seg"	197	"C01"	"Algoritmos e Estrutura de Dados II"	"304"	"F"	3	2
"18:45"	"seg"	208	"C01"	"Gerencia de Projeto de Software I"	"201"	"F"	3	3
...	...	...	...	...	...	...	...	...
"20:30"	"seg"	10239	"C01"	"Tecnologia de Desenvolvimento II"	"205"	"F"	3	4
"20:30"	"seg"	10587	"C01"	"Qualidade de Software I"	"202"	"F"	3	7
...	...	...	...	...	...	...	...	...

Figura 3: Designação de salas de aula

Os requisitos não essenciais foram atendidos na maioria dos casos, o que é verificado na figura 4 para o curso de Ciência da Computação do segundo período. Observa-se, ainda, que foi possível alocar esse período em dois blocos da área 4 e suas turmas mantiveram a sala na semana.

Hora Início	Dia Semana	Disciplina	Sala	Bloco	Área
"10:50"	"sex"	"Programacao de Computadores II"	"201"	"E"	3
"10:50"	"ter"	"Programacao de Computadores II"	"201"	"E"	3
"09:00"	"ter"	"Logica Computacional e Algoritmica II"	"202"	"E"	3
"07:10"	"qua"	"Aplicacoes da Ciência da Computacao"	"204"	"E"	3
"10:50"	"seg"	"Álgebra Abstrata"	"205"	"F"	3
"10:50"	"qui"	"Álgebra Abstrata"	"205"	"F"	3
"09:00"	"seg"	"Geometria Analitica e Calculo Vetorial"	"206"	"F"	3
"09:00"	"qui"	"Geometria Analitica e Calculo Vetorial"	"206"	"F"	3
"07:10"	"seg"	"Fisica para Computacao II"	"201"	"F"	3
"07:10"	"qui"	"Fisica para Computacao II"	"201"	"F"	3

Figura 4 – Segundo Período do Curso de Ciência da Computação

Já na figura 5 a alocação para o terceiro período do curso de Ciência da Computação foi feita em três blocos da área 3, porém suas turmas conservaram-se na mesma sala durante os dias da semana.

Hora Início	Dia Semana	Disciplina	Sala	Bloco	Área
"10:50"	"seg"	"Programacao de Computadores III"	"302"	"E"	3
"10:50"	"qui"	"Programacao de Computadores III"	"302"	"E"	3
"07:10"	"seg"	"Estrutura de Dados I"	"401"	"C"	3
"07:10"	"qui"	"Estrutura de Dados I"	"401"	"C"	3
"10:50"	"ter"	"Sistemas Digitais para Computacao"	"303"	"F"	3
"10:50"	"sex"	"Sistemas Digitais para Computacao"	"303"	"F"	3
"09:00"	"ter"	"Linguagem de Montagem"	"304"	"E"	3
"09:00"	"sex"	"Linguagem de Montagem"	"304"	"E"	3

"09:00"	"seg"	"Calculo Diferencial e Integral III"	"203"	"F"	3
"09:00"	"qui"	"Calculo Diferencial e Integral III"	"203"	"F"	3

Figura 5 – Terceiro Período do Curso de Ciência da Computação

Nota-se que tanto na figura 4 quanto na figura 5, em particular, que as turmas estão no mesmo horário em dias diferentes (a chamada “dobradinha”na PUC Goiás, por exemplo, segunda-feira com quinta-feira, terça-feira com sexta-feira); o que coincide com o cadastramento que a CPA fornece para os departamentos dos cursos.

Iniciamos esta pesquisa com Neves (2010), quem implementou o algoritmo Húngaro para o problema de designação de salas de aula para o departamento de Computação da PUC Goiás. Em seguida, Silva (2011) reescreveu o código para o algoritmo Húngaro e resolveu o problema de designação de salas de aula para o Centro Técnico Científico com dados aleatórios e, também, incorporou no modelo matemático o requisito não essencial, agora, R_6. Com o mesmo código de Silva (2011) para o algoritmo Húngaro, resolvemos o problema de designação de salas de aula para 1254 turmas por 81 salas de aula, com dados reais, com o tempo de não mais do que 6 segundos.

A seguir faremos as nossas considerações finais.

CONSIDERAÇÕES FINAIS

Este trabalho contribuiu para o desenvolvimento do sistema computacional SAPA (www.sapacmp.com.br), o qual ainda está em fase de testes. Nossa contribuição se deu ao digitarmos os dados reais para a PUC Goiás apontando problemas quando raramente ocorria.

Verificamos que não houve a necessidade das salas da área 2. Esse fato ocorreu em virtude de considerarmos as salas com pranchetas da área 3 como salas disponíveis e, também, não considerarmos turmas de preleção como aquelas turmas com aulas práticas.

Em nosso resultado mostramos que efetivamente resolver o problema de designação de salas de aula para a área 3, campus I, da PUC Goiás, horário por horário, é promissor, haja vista que resolvemos o problema em não mais do que 6 segundos enquanto a CPA resolve em um mês e uma semana, neste caso, para toda a PUC Goiás. Dessa forma, sugerimos uma tentativa de resolver o problema para toda a PUC Goiás com a mesma estratégia horário por horário.

Este trabalho foi submetido e aceito como pôster para o próximo CLAIO/SBPO em setembro deste ano.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVES, A. C. B., MENEZES, M. A. F., **Introdução à pesquisa operacional.** Goiânia: Editora da PUC Goiás, 2010, 311 p.

ARENALES, M., ARMENTANO, V., MORABITO, R., YANASSE, H., **Pesquisa operacional.** 5ª. reimpressão. Rio de Janeiro. Editora Elsevier, 2007. 542p.

BONDY, J. A., MURTY, U.S.R., **Graph Theory**. Springer, 2008.

BONDY, J. A., MURTY, U.S.R., **Graph Theory with Applications**. Macmillan, London, 1976.

BOAVENTURA, P. O., JURKIEWICZ, S., **Grafos: Introdução e prática**. São Paulo. Editora Blucher, 2009.

BURKE, E., JACKSON, K., KINGSTON, J., WEARE, R., "Automated University Timetabling: The State of the Art". **The Computer Journal**, Vol. 40, No. 9, pp. 565-571, 1997.

CARTER, M. W., "A lagrangian Relaxation Approach to the Classroom Assigment Problem". **INFOR**, Vol. 27, No 2, pp. 230-246, 1989.

CARTER, M. W., LAPORTE, G., “Recent Developments in Practical Course Timetablin”. **Practice and Theory of Automated Timetabling PATAT'97**, E. K. Burke, and M. W. Carter (Eds.) Springer Verlag Lecture Notes in Computer Science, 1408, pp. 3-19, 1998.

CARTER, M. W., TOVEY, C. A., “When Is the Classroom Assignment Problem Hard?”. **Operations Research**, Vol. 40, No. 1, p. 28-39, 1992.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., STEIN, C., **Algoritmos: teoria e prática**. Revisora técnica: Jussara Pimenta Matos. Tradutor: Vandenberg D. de Souza. Rio de Janeiro: Campus, 2002.

GLASSEY, C. R., MIZRACH, M., “A Decision Support System for Assigning Classes to Rooms”. **Interfaces**, Vol. 16, No. 5, pp. 92-100, 1986.

GOLDBARG, M. C., LUNA, H. P. L., **Otimização combinatória e programação linear:modelos e algoritmos**. Rio de Janeiro: Editora Campus, 2000.

GOSSELIN, K., TRUCHON, M., “Allocation of Classrooms by Linear Programming”. **J. Operational Research Soc.**, Vol. 37, No. 6, pp. 561-569, 1986.

KUHN, H. W., “The hungarian method for the assignment problem”. **Naval Research Logistics Quarterly**, v.2, p. 83-97, 1955.

MENEZES, M. A. F., “Uma breve introdução à programação linear inteira”. **Boletim – Informativo do Grupo de Pesquisa Matemática Computacional**, Ano 9, Número 17, março de 2011.

MACULAN, N., FAMPA M. H., **Otimização linear**. Brasília. Editora UnB, 2006, 310p.

MCCOLLUM, B., “A perspective on bridging the gap between theory and practice in university timetabling”. In E. Burke & H. Rudová (Eds.), LNCS: Vol. 3867. **Practice and theory of automated timetabling VI** (pp. 3–23). Berlin: Springer., 2007.

MCCOLLUM, B., MCMULLAN, P., PAECHTER, B., LEWIS, R., SCHAERF, A., DI GASPERO, L., PARKES, A. J., QU, R., BURKE, E. K., “Setting the research agenda in automated timetabling: the second international timetabling competition”. **INFORMS journal on Computing**, Vol. 22, p. 20-130, 2010.

NEVES, K. F. S., 2010, **Uma implementação para o problema de designação de salas de aula para a PUC Goiás: um estudo de caso no Departamento de Computação**. Monografia, Departamento de Computação da Puc Goiás.

PAPADIMITRIOU, C. H., STEIGLITZ, K., **Combinatorial Optimization: algorithms and complexity**. Mineola: Dover, p. 512, 1998.

RUDOVÁ, H., MULLER, T., MURRAY, K., "Complex University Course Timetabling". **Journal of Scheduling**, 14(2): 187-207, Springer, 2011. DOI 10.1007/s10951-010-0171-3.

SALKIN, H. M., **Integer Progamming.** Massachusetts: Addison Wesley, p. 557, 1975.

SCHAERF, A., "A Survey of Automated Timetabling". **Artificial Intelligence Review**. Vol. 13, p.87-127, 1999.

SILVA , B. M. N., 2010, **Uma implementação para o problema de designação de salas de aula para o Centro Técnico e Científico da PUC Goiás.** Monografia, Departamento de Computação da Puc Goiás.

SOUZA, M.J.F., 2000, **Programação de Horários em Escolas: Uma Aproximação por Metaheurísticas**, D.Sc. Thesis (in Portuguese), Programa de Engenharia de Sistemas e Computação, Universidade Federal do Rio de Janeiro - COPPE/UFRJ, Rio de Janeiro, Brasil.

SUBRAMANIAN, A., MEDEIROS, J. M. F., FORMIGA, L. A., SOUZA, M. J. F., “Aplicação da Metaheurística Busca Tabu ao Problema de Alocação de Aulas a Salas em uma Instituição Universitária”. **Revista Produção Online**, v.11,n.1, p.54-75, mar., 2011.

TOSCANI, L. V., VELOSO, P. A. S., **Complexidade de Algoritmos**. Porto Alegre: Instituto de Informática da UFRGS: Editora Sagra Luzzatto, 2005.