UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS QUIXADÁ

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JONAS COSTA FERREIRA DA SILVA

ALOCAÇÃO DE DISCIPLINAS EM HORÁRIOS MAXIMIZANDO AS OPÇÕES DE MATRÍCULA

QUIXADÁ – CEARÁ

2016

JONAS COSTA FERREIRA DA SILVA

ALOCAÇÃO DE DISCIPLINAS EM HORÁRIOS MAXIMIZANDO AS OPÇÕES DE MATRÍCULA

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Computação.

Orientador: Prof. Dr. Críston Pereira Souza

Co-Orientador: Prof. Msc Lucas Ismaily Bezerra Freitas

QUIXADÁ – CEARÁ

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58a Silva, Jonas Costa Ferreira da.
Alocação de disciplinas em horários maximizando as opções de matrícula / Jonas Costa Ferreira da Silva. – 2016.
42 f. : il.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2016.
Orientação: Prof. Dr. Críston Pereira de Souza.
Coorientação: Prof. Me. Lucas Ismaily Bezerra Freitas.

1. Universidade Federal do Ceará - Campus Quixadá. 2. Alocação. 3. Disciplina. 4. Horários de trabalho. 5. Programação inteira. I. Título.

CDD 004

JONAS COSTA FERREIRA DA SILVA

ALOCAÇÃO DE DISCIPLINAS EM HORÁRIOS MAXIMIZANDO AS OPÇÕES DE MATRÍCULA

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Computação.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Críston Pereira Souza (Orientador)
Campus Quixadá
Universidade Federal do Ceará – UFC

Prof. Msc Lucas Ismaily Bezerra Freitas (Co-Orientador)
Campus Quixadá
Universidade Federal do Ceará - UFC

Prof. Dr. Fábio Carlos Sousa Dias
Campus Quixadá
Universidade Federal do Ceará - UFC

Prof. Dr. Wladimir Araujo Tavares
Campus Quixadá
Universidade Federal do Ceará - UFC

A Deus.

Aos meus pais, João Mendonça e Cássia.

A minhas irmãs, Jocássia e Jéssica.

Aos meus sobrinhos, Jônatas e Camille.

Aos meus amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pois eu sei que absolutamente tudo que conquistei foi por graça Dele.

A minha família, principalmente meus pais e minhas irmãs, por todo o apoio e pela confiança que depositaram em mim ao longo desses anos de graduação, sem pressão nem interesses apenas amor!

À Micaele, minha belíssima namorada, pelo seu carinho, atenção e por acreditar em mim. Amo você!

Ao Prof. Críston, pelo convite para participar do projeto que culminou nesse trabalho, pela sua excelente orientação e por todo o conhecimento que me transmitiu como professor.

Ao Prof. Lucas Ismaily, pela co-orientação e também pelo seu empenho como professor.

Aos professores participantes da banca examinadora Prof. Fábio e Prof. Wladimir pelo tempo, pelas valiosas colaborações e sugestões.

Aos meus colegas de turma, em especial: Ana Paula, aluna exemplar; André Davys, que apesar da baixa estatura é um grande homem; Jhonata, referência ambulante do c++; Wallinson, grande construtor de AVL's; Sergio, amigo dos números e Rômulo, alegria da turma. Sem vocês eu ainda estaria no primeiro semestre.

Aos meus amigos, e de maneira especial, Joéliton, pelas suas valorosas sugestões e João Paulo, que teve que ouvir tudo sobre esse trabalho algumas centenas de vezes.

A todos que direta ou indiretamente fizeram parte da minha formação.

"O teu lugar no Céu parecerá como se ele tivesse sido feito para ti, só para ti, porque foste feito para ele."

(C.S. Lewis)

RESUMO

A atividade de organizar disciplinas em horários semanais é uma tarefa comum em instituições de ensino superior que devem considerar como os horários afetam o andamento do curso dos alunos. No ato da matrícula, por exemplo, o ideal é que todas as disciplinas nas quais um aluno necessite matricular-se estejam em horários compatíveis, ou seja, não choquem horários entre si. No entanto, devido as diferentes necessidades dos alunos, fornecer uma grade de horários que seja compatível com todos os alunos pode não ser possível. Tendo em vista essa problemática, este trabalho tem como objetivo encontrar uma alocação de disciplinas em horários que maximize as opções de matrícula dos alunos da Universidade Federal do Ceará - Campus Quixadá. Para tal, foi criado um modelo de programação linear inteira com base nas restrições de alocação do campus. O modelo foi testado com dados dos alunos, disciplinas e cursos do campus e a solução proposta pelo modelo foi comparada com a alocação aplicada no semestre 2016.2. A alocação gerada pelo modelo mostrou-se mais eficiente que a referência adotada, considerando apenas as opções de matrícula dos alunos.

Palavras-chave: Alocação de horários. Alocação de disciplinas. Programação linear inteira. UFC-Quixadá.

ABSTRACT

The weekly classes schedule planning is a common task on Universities, on this planning, is important consider how it affects the students major progress. For example, ideally all courses that a student needs to attend should have compatible schedules. However, this is basically impossible since each student have different needs. This study aims to give a class schedule that maximizes the Universidade Ferderal do Ceará - Campus Quixadá students' inscription options. To do so, was created an integer program based on the campus timetabling constraints. The integer program was tested with an instance created from UFC-Quixadá data. The solution given by the integer programing was compared with schedule used on the campus in 2016.2 and the proposed solution showed to be better, considering only the students' inscription options.

Keywords: Timetabling. Course timetabling. Integer programming. UFC-Quixadá.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

CSV	Comma-Separeted values
PAHD	Problema de Alocar Professores em Disciplinas
POHCU	Problema de Organização de Horários de Cursos Universitários
PL	Programação Linear
PLI	Programação Linear Inteira
UFC-Quixadá	Universidade Federal do Ceará - Campus Quixadá

SUMÁRIO

1 INTRODUÇÃO

Instituições de ensino superior precisam regularmente organizar os horários semanais das aulas. É preciso definir quais professores irão ministrar cada disciplina, e ainda, a cada aula atribuir uma sala. Entretanto, essa pode não ser uma tarefa simples de ser realizada manualmente pois usualmente existem algumas restrições atreladas a esse problema, por exemplo, para determinar quais disciplinas um determinado professor irá ministrar é necessário assegurar que nenhuma delas tenha sido alocada no mesmo horário. Quando há um grande número de recursos envolvidos (disciplinas, professores, salas, turmas) essa tarefa se torna muito difícil.

Problemas deste tipo são agrupados em uma classe de problemas chamada de Problema de Horários Universitários (*University Timetabling Problem*) (KRIPKA; KRIPKA; SILVA, 2011) e podem ser muito complexos computacionalmente. Por exemplo, o Problema de Organização de Horários de Cursos Universitários (*University Course Timetabling Problem*) faz parte dessa classe e é NP-completo. A menos que P=NP, a solução ótima para esse tipo de problema não pode ser obtida em tempo hábil (polinomial), então a maioria das abordagens são baseadas em meta-heurísticas que buscam soluções aproximadas (LACH; LÜBBECKE, 2008).

Decompor o problema geral em subproblemas que possam ser resolvidos separadamente e combinar suas soluções é uma estratégia que pode fornecer boas soluções. No caso da **Universidade Federal do Ceará - Campus Quixadá** (UFC-Quixadá), o problema pode ser dividido em: alocar os horários das disciplinas e atribuir professores a esses horários. Esse trabalho trata do primeiro subproblema: o Problema de Alocar Horários de Disciplinas (PAHD).

Os horários das disciplinas não podem ser escolhidos arbitrariamente, pois algumas disciplinas não podem ter aulas no mesmo horário. A razão disso é que uma alocação de horários impacta no número de opções de matrículas dos alunos. Por exemplo, se um aluno está interessado em cursar duas disciplinas e estas são ofertadas no mesmo horário, ele perde a opção de cursar ambas. Essa situação deve ser minimizada pois dificulta o andamento do curso para os alunos. E além disso, os horários devem respeitar o turno, carga horária e outras restrições envolvendo disciplinas que são impostas pela UFC-Quixadá.

O objetivo desse trabalho é determinar uma alocação de disciplinas em horários de modo a maximizar as opções de matrícula dos alunos da UFC-Quixadá, ou seja, minimizando o número de alunos que podem cursar disciplinas alocadas em um mesmo horário. Para isso foi desenvolvido um modelo de programação linear inteira que modela as restrições de alocação

que são empregadas no Campus. Por fim, a solução proposta pelo modelo foi comparada com a alocação de horários utilizada na UFC-Quixadá em 2016.2.

O restante desse trabalho é divido da seguinte maneira: o Capítulo 2 apresenta os conceitos de programação linear e programação linear inteira; o Capítulo 3 apresenta alguns trabalhos relacionados aos Problemas de Horários Universitários; no Capítulo 4 é realizado uma breve contextualização sobre a alocação de horários na UFC-Quixadá e é apresentado o modelo desenvolvido para esse problema; o Capítulo 5 detalha os procedimentos que foram realizados com os dados da UFC-Quixadá e os resultados obtidos; por fim, o Capítulo 6 apresenta as considerações finais sobre este trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Programação Linear

Um modelo de **programação linear** (PL) é um problema de otimização com uma função objetivo linear, um conjunto de restrições lineares e um conjunto de restrições de intervalo impostas sobre as variáveis de decisão envolvidas nessas funções, isto é, é um modelo de otimização na seguinte forma:

Função Objetivo:

$$\min \sum_{j=0}^{n} c_j x_j$$

Restrições:

$$\sum_{j=1}^{n} a_{ij} x_j = b(i), \forall i \in \{1,\ldots,m\} \tag{2.1}$$

$$x_j \geq 0, \forall j \in \{1,\ldots,n\} \tag{2.2}$$

Este PL possui n variáveis de decisão não negativas x_j e m restrições de igualdade 2.1. Existem várias maneiras alternativas de se modelar um PL, por exemplo, a função objetivo pode ser colocada na forma de maximização ou as restrições podem ser desigualdades. Este tipo de problema com apenas restrições de igualdade, variáveis de decisão não negativas e na forma de minimização da função objetivo é dito estar na **forma padrão**.

Cormen (2009) define a forma normal de maneira similar, porém, em sua terminologia as restrições precisam ser desigualdades do tipo menor-ou-igual. Nesse trabalho é considerada a forma padrão definida por Ahuja, Magnanti e Orlin (1993) onde todas as variáveis de decisão são não negativas, a função objetivo está na forma de minimização e as restrições não possuem desigualdades (equações lineares).

Um modelo de programação linear na notação matricial tem a seguinte forma:

Função Objetivo:

$$\min cx$$

Sujeito a:

$$Ax = b,$$

$$x \geq 0$$

Nesta formulação a **matriz de coeficientes** $A = (a_{ij})$ é uma matriz $m \times n$, c é um vetor linha n-dimensional e $x = (x_i)$ e $b = (b_i)$ são vetores coluna com n e m linhas respectivamente. A j-ésima coluna de A que corresponde a variável x_j é denotada por A_j. Esse modelo possui uma notação mais curta $\min\{cx : Ax = b, x \in \mathbb{R}^n_+\}$.

O método simplex para a resolução de programas lineares foi desenvolvido em 1947 e foi considerado como uma das mais importantes descobertas algorítmicas do século. Sendo ainda largamente utilizado até hoje (AHUJA; MAGNANTI; ORLIN, 1993).

Uma **solução viável** de um PL é uma configuração de valores para as variáveis $x_1, x_2, \ldots, x_n$ de modo que todas as restrições sejam satisfeitas. Uma solução viável que forneça um valor ótimo para a função objetivo é chamada **solução ótima**. Um valor ótimo é o valor máximo possível que a função objetivo pode assumir sujeito ao conjunto de restrições, ou ainda o valor mínimo, no caso da função objetivo estar na forma de minimização (CORMEN, 2009).

2.1.1 Solução básica viável

O método simplex tem como requisito que o PL esteja na forma padrão. Qualquer modelo de programação linear que não esteja na forma padrão pode ser convertido em um modelo equivalente na forma padrão através de algumas transformações. Por exemplo, para converter um problema de maximização $\max \sum_{j=1}^{n} c_j x_j$ em um problema de minimização equivalente basta multiplicar toda a função objetivo por -1 o que resulta em $\min \sum_{j=1}^{n} -c_j x_j$. Para converter uma restrição de desigualdade $a_{11}x_1 + a_{12}x_2 \leq b_1$ em uma restrição de igualdade é preciso adicionar uma **variável de folga** y_1 com custo 0 na função objetivo, resultando em $a_{11}x_1 + a_{12}x_2 + y_1 = b_1$. A forma padrão é útil pois com as restrições na forma de igualdade as operações de linha da álgebra linear (multiplicar uma linha por uma constante e adicionar uma linha a outra) podem ser realizadas sobre a matriz de coeficientes sem alterar o conjunto de soluções viáveis (desde que as mesmas operações sejam replicadas no vetor b), ou seja, gerando formulações equivalentes.

Propriedade Canônica: A formulação possui uma variável de decisão isolada em cada uma das restrições; a variável que está isolada em uma restrição possui o coeficiente $+1$ nesta restrição e coeficiente 0 nas demais, inclusive na função objetivo.

Função Objetivo:

$$\min 0x_1 + 0x_2 + 3x_3 - x_4 + 10$$

Sujeito a:

$$x_1 + 0x_2 - 3x_3 + 3x_4 = 6 \tag{2.3}$$

$$0x_1 + x_2 - 8x_3 + 4x_4 = 4 \tag{2.4}$$

$$x_1, x_2, x_3, x_4 \geq 0 \tag{2.5}$$

O modelo acima está na forma canônica pois a variável x_1 está isolada em (2.3) e a variável x_2 em (2.4). Essa forma é importante pois permite que a atribuição de valores a x_3 e x_4 determine unicamente valores para x_1 e x_2. De fato, ao tornar $x_3 = x_4 = 0$ imediatamente se obtém $x_1 = 6$ e $x_2 = 4$. Esse tipo de solução é chamada de **solução básica viável** e tem um papel muito importante no simplex pois ele consegue iterar entre essas soluções até alcançar uma que seja ótima. De modo geral, dado um PL na forma canônica, obtém-se uma solução básica viável atribuindo a variável isolada na restrição i, i-ésima **variável básica**, o valor do lado direito desta restrição (b_i) e atribuindo 0 às variáveis restantes, que são as variáveis não básicas. Todas as variáveis básicas formam uma base.

Geralmente uma solução básica viável pode ser encontrada da seguinte maneira. Isola-se uma variável em cada restrição. Para simplificar, x_i é a variável básica da restrição i. Sejam $B = \{1, 2, \ldots, m\}$ e $N = \{m+1, m+2, \ldots, n\}$ os índices das variáveis básicas e não básicas respectivamente. Com o par (B, N) é possível particionar as colunas da matriz de coeficientes A em: $\mathscr{B} = \{A_1, A_2, \ldots, A_m\}$ e $\mathscr{N} = \{A_{1+m}, A_{2+m}, \ldots, A_n\}$, onde $\mathscr{B}$ é uma matriz $m \times m$ chamada **matriz de base**. Definindo ainda $x_B = [x_i : i \in B]$ e $x_N = [x_j : j \in N]$ como sendo uma partição das variáveis em sub-vetores correspondentes aos conjuntos de índices B e N, é possível reescrever $Ax = b$ como

$$\mathscr{B}x_B + \mathscr{N}x_N = b. \tag{2.6}$$

Pode-se converter (2.6) para a forma canônica pré-multiplicando todos os termos por $\mathscr{B}^{-1}$, a inversa da matriz de base $\mathscr{B}^{-1}$, resultando em

$$x_B + \mathscr{B}^{-1}\mathscr{N}x_N = \mathscr{B}^{-1}b. \tag{2.7}$$

E para se obter uma **solução básica** a partir de 2.7 basta atribuir 0 a toda variável não básica$(X_N = 0)$, obtendo

$$x_B = \mathscr{B}^{-1}b. \tag{2.8}$$

Essa solução será básica viável apenas se todas as variáveis básicas forem não negativas ($x_B \geq 0$). Para algumas matrizes de base a solução será viável e para outras não. É importante notar que a matriz de base deve ser inversível(**não singular**), isto será possível apenas se as colunas associadas a ela forem linearmente independentes. Quando uma matriz não possui inversa ela é chamada de **singular**(AHUJA; MAGNANTI; ORLIN, 1993).

Um estudo mais aprofundado sobre programação linear e o método simplex pode ser encontrado em (AHUJA; MAGNANTI; ORLIN, 1993; CORMEN, 2009; SCHRIJVER, 1998).

2.1.2 Programação Inteira

Um problema de **programação inteira** PLI é um programa linear que restringe ao menos uma das variáveis a assumir apenas valores inteiros. Os programas lineares inteiros (puros) possuem todas as variáveis inteiras como no exemplo abaixo:

Função Objetivo:

$$\max z = c_1 x_1 + c_2 x_2$$

Sujeito a:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 &\leq b_2 \\ x_1, x_2 &\geq 0 \\ x_1, x_2 &\in \mathbb{Z} \end{aligned}$$

Se nem todas as variáveis são inteiras o PLI é chamado **misto**:

Função Objetivo:

$$\max z = c_1 x_1 + c_2 x_2$$

Sujeito a:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 &\leq b_2 \\ x_1, x_2 &\geq 0 \\ x_1 &\in \mathbb{Z} \end{aligned}$$

Existe ainda o PLI **binário**, que restringe todas as variáveis a possuírem valor 0 ou 1 (WOLSEY, 1998). Embora um PLI seja sintaticamente semelhante a um PL, saber se um PLI qualquer tem uma solução ótima é NP-completo (SCHRIJVER, 1998).

Para um PLI na forma $\max\{cx : Ax \leq b, x \in \mathbb{Z}_+^n\}$ quando os elementos de A e b são inteiros é importante saber se é possível remover as restrições de integralidade, de forma que, o PL resultante $\max\{cx : Ax \leq b, x \in \mathbb{R}_+^n\}$ tenha solução ótima inteira e que esta também seja a solução do PLI original.

Sabendo que soluções básicas viáveis são da forma: $x = (x_B, x_N) = (B^{-1}b, 0_n)$ onde B é uma submatriz $m \times m$ não singular de (A, I) e I é uma matriz identidade $m \times m$, então a seguinte observação é uma condição suficiente.

Lema 1 (Condição suficiente)**.** *Se a base ótima B possui* $det(B) = \pm 1$*, então o PL obtido através da relaxação das restrições de integralidade resolve o PLI.*

Prova. Pela regra de Cramer $B^{-1} = adj(B) * \frac{1}{det(B)}$ onde $adj(B)$ é a matriz adjunta de B. Dado que $adj(B)$ é uma matriz inteira, já que ela é obtida através de produtos de termos de B, como o $det(B) = \pm 1, B^{-1}$ também é inteira, assim como $B^{-1}b$. □

Quando A for totalmente unimodular é possível relaxar as restrições de integralidade do PLI. Uma matriz A é **totalmente unimodular** (TU) se toda submatriz quadrada de A possui determinante $+1, -1$ ou 0.

Proposição 2.1.1 (Condição suficiente)**.** *Uma matriz A é TU se:*

(i) $a_{ij} \in \{+1, -1, 0\}$ *para todo i,j.*

(ii) Toda coluna contém no máximo dois coeficientes não nulos $(\sum_{i=1}^{m} |a_{ij}| \leq 2)$.

(iii) Existe uma partição (M_1, M_2) *do conjunto de linhas M, de forma que cada coluna j que contenha dois coeficientes não nulos satisfaz* $\sum_{i \in M_1} a_{ij} - \sum_{i \in M_2} a_{ij} = 0$.

Prova. Suponha que A não é TU. Seja B a menor submatriz quadrada de A cujo $det(B) \notin \{+1, -1, 0\}$. A matriz B não possui nenhuma coluna com apenas um elemento não nulo, pois B não seria mínima. B contém dois elementos não nulos em cada coluna. Pela condição (iii) adicionando as linhas em M_1 e subtraindo das linhas em M_2 produz o vetor nulo e, neste caso, $det(B) = 0$ o que é uma contradição(WOLSEY, 1998). □

2.1.3 Problema de atribuição com programação inteira

Dado um grupo de n pessoas e n tarefas para serem executadas e assumindo que é possível fornecer uma pontuação para a capacidade da pessoa i em realizar tarefa j. O problema de atribuição consiste em atribuir cada pessoa a uma tarefa e a cada tarefa um trabalhador de modo que a soma das n pontuações seja a maior possível (KUHN, 1955). Segue abaixo o modelo de PLI para o problema de atribuição:

$$z = \max \sum_{i=1}^{n} \sum_{j=1}^{n} p_{ij} x_{ij} \tag{2.9}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j \in \{1, \ldots, n\} \tag{2.10}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \forall i \in \{1, \ldots, n\} \tag{2.11}$$

$$x_{ij} \geq 0, x_{ij} \in \mathbb{Z} \tag{2.12}$$

No modelo acima a variável x_{ij} terá o valor 1 quando o a pessoa i for escalada para a tarefa j e terá o valor 0 caso contrário. A pontuação da pessoa i na tarefa j é expressa no modelo por p_{ij}. A função objetivo (2.9) visa maximizar a soma das pontuações. A restrição 2.10 garante que um trabalho será realizado por apenas uma pessoa. De forma similar, a restrição 2.11 restringe uma pessoa a realizar apenas um trabalho.

O problema de atribuição é um exemplo de PLI em que é possível relaxar as restrições de integralidade. Basta observar que todas as condições da Proposição 2.1.1 se aplicam. Todos os coeficientes são 0 ou 1 satisfazendo (i). Toda variável x_{ij} aparece em exatamente duas restrições, o que satisfaz (ii). E para satisfazer (iii) basta tornar M_1 e M_2 as linhas correspondentes ao primeiro e segundo tipo de restrição respectivamente.

2.1.4 Métodos exatos para programação linear inteira

Nem todos os problemas de programação linear inteira são como o problema de atribuição que podem ser convertidos em programas lineares equivalentes que são mais fáceis de resolver. Para esses casos não são conhecidos algoritmos exatos que sejam polinomiais. O desenvolvimento de métodos exatos para programação linear inteira foi bem sucedidos nos últimos 50 anos. Existem pelo menos três tipos de abordagens exatas para resolver esses problemas, sendo elas (GENOVA; GULIASHKI, 2011):

- Métodos de plano de corte;

- *Branch and Bound*;
- Técnicas de relaxação e decomposição.

Um estudo mais aprofundado sobre esses métodos pode ser encontrado em (WOLSEY, 1998; GENOVA; GULIASHKI, 2011).

3 TRABALHOS RELACIONADOS

3.1 Problema de organização de horários de cursos universitários

O problema de alocação de horários e professores do Campus de Quixadá possui algumas restrições em comum com o **Problema de Organização de Horários de Cursos Universitários** (POHCU) que é apresentado por Babaei, Karimpour e Hadidi (2015). Sendo elas:

- Um professor não pode dar duas aulas ao mesmo tempo;
- Um professor ensina apenas uma disciplina em um mesmo horário;
- Um professor pode sugerir certas prioridades há alguns horários de sua preferência;

As duas primeiras são classificadas como restrições *hard* e a última como restrição *soft*, as restrições *hard* precisam ser todas satisfeitas e as restrições *soft* estão relacionadas com a função objetivo e não necessariamente precisam ser satisfeitas. A última restrição é um pouco diferente na UFC-Quixadá, onde os professores sugerem preferências a disciplinas e não a horários. A maioria das outras restrições identificadas por (BABAEI; KARIMPOUR; HADIDI, 2015) são relacionadas com a alocação de salas de aula. Na UFC-Quixadá é assumido que as salas são suficientes e não são levadas em consideração na alocação, o que facilita bastante o processo.

Neste trabalho os autores revisam algumas das abordagens já existentes no estudo do POHCU como pesquisa operacional, meta-heurísticas, análise multi-objetivo e multi-critério e introduzem uma nova abordagem baseada em multi-agentes distribuídos.

3.2 Formulação para o problema de alocação de salas de aula com minimização de deslocamentos

O problema apresentado por Kripka, Kripka e Silva (2011) visa minimizar os deslocamentos que terão que ser realizados pelos alunos fora da sede do curso. Em seu trabalho, Kripka, Kripka e Silva (2011) buscam atribuir disciplinas à salas de aula de acordo com as seguintes restrições:

- duas disciplinas não podem ter aulas simultâneas em uma mesma sala;
- uma disciplina não pode ter uma aula em duas salas ao mesmo tempo;
- a capacidade de alunos de uma sala deve ser maior que o número de alunos inscritos para

aquela aula;

Essas restrições são chamadas pelos autores de impeditivas, ou seja, que não podem ser violadas para que uma alocação seja considerada viável. Além dessas três, eles consideram uma restrição não impeditiva, que é relacionada à sobra de lugares vagos na sala. As restrições relativas à carga horária das disciplinas não foram incluídas, pois, segundo os autores, em cada turno, para que uma disciplina seja alocada é preciso apenas de uma sala com capacidade suficiente.

Além de alocar as salas é importante para a Universidade de Passos Fundos que as aulas de um determinado curso não fiquem muito distantes da sua sede própria. Os autores usam uma técnica de têmpera simulada, que é uma meta-heurística, para resolver o problema. Esse problema é um exemplo de como o conjunto de restrições do problema de pode variar de acordo com as prioridades da Universidade.

Diferentemente dos trabalhos que foram apresentados neste Capítulo, que buscam resolver todo o conjunto de restrições dos seus respectivos problemas de alocação, o presente trabalho se propõe a encontrar uma solução para apenas uma parte das restrições da alocação da UFC-Quixadá, deixando espaço para que as demais restrições possam ser resolvidas utilizando outra abordagem.

4 PROBLEMA DE ALOCAR HORÁRIOS DE DISCIPLINAS

Este capítulo apresenta o Problema de Alocar Horários de Disciplinas, bem como o solução que este trabalho propõe para resolvê-lo. O PAHD foi definido a partir das necessidades da UFC-Quixadá, por isso, a Seção 4.1 deste capítulo apresenta algumas informações sobre a estrutura do campus no que diz repeito à alocação de disciplinas. A Seção 4.2 apresenta a formulação em programação linear inteira do PAHD.

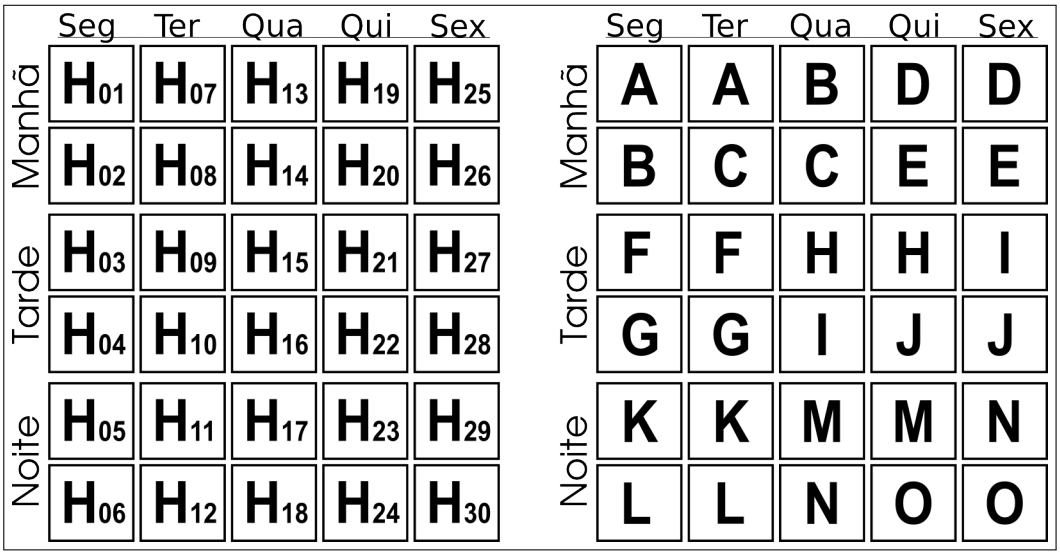
4.1 Contextualização

Atualmente na UFC-Quixadá existem seis cursos em funcionamento: os cursos de Ciência da Computação e Sistemas de Informação funcionam, preferencialmente, no turno da manhã; Design Digital, Engenharia de Computação e Engenharia de Software, preferencialmente, no turno da tarde e Redes de Computadores no turno da noite. A grade de horários semanais do campus prevê 10 horários de aula em cada turno, ou seja, 2 horários por dia e totalizando 30 horários nos três turnos. Cada disciplina tem um número de aulas semanais e cada uma dessas aulas deve ser atribuída a um horário.

A maioria das disciplinas que são ofertadas no campus possuem duas aulas por semana, por exemplo, das 106 disciplinas que foram ofertadas no semestre letivo de 2016.2, 94 delas possuíam duas aulas semanais. Por essa razão, os horários foram organizados em pares fixos chamados de **slots**. Os *slots* foram selecionados objetivando evitar duas aulas em um mesmo dia da mesma disciplina, e ao mesmo tempo assegurar que elas fiquem em dias próximos. O ideal seria que todos os *slots* fossem compostos por horários de dias consecutivos, porém tal organização não é possível. A organização de horários e *slots* adotada no campus é apresentada na Figura 1. A utilização de *slots* é importante pois reduz o número de alocações possíveis. Por exemplo, para uma disciplina existem 5 possibilidades de alocação em cada turno contra 45 possibilidades sem a utilização de *slots* (considerando apenas disciplinas de duas aulas). Disciplinas com três aulas semanais devem ocupar no máximo dois *slots* com as mesmas restrições de não haver aulas em um mesmo dia, nem ocupando mais de três dias consecutivos. Disciplinas com uma aula semanal podem ocupar qualquer horário do turno.

Antes de realizar a alocação de disciplinas é preciso definir quais disciplinas serão ofertadas. Para isso, as coordenações de curso são acionadas para fornecer a demanda de disciplinas de acordo com as necessidades das turmas existentes. Na prática uma turma é um

Figura 1 – Distribuição de horários semanais (esquerda) e distribuição de *slots* (direita) onde cada par de letras corresponde a um *slot*.



Fonte: Desenvolvida pelo autor

grupo de alunos que cursam um determinado semestre de um curso. Esses alunos precisam cursar, em cada semestre, as disciplinas que estão previstas na matriz curricular do curso. Algumas dessas disciplinas são obrigatórias e devem preferencialmente ser cursadas naquele semestre, portanto elas não podem chocar horário entre si para que os alunos da turma possam se matricular em todas elas. A matriz curricular pode prever também uma carga horária de disciplinas optativas, de maneira que o aluno possa escolher qual disciplina cursar de acordo com o regulamento do curso. A partir desse ponto, uma **turma** será vista apenas como um conjunto de disciplinas obrigatórias e optativas associadas a um semestre de um curso.

As disciplinas obrigatórias de uma turma devem ser ofertadas no turno do seu respectivo curso, porém existem turmas em que a soma da carga horária de suas disciplinas é superior à carga horária de um turno. Em casos como este é necessária a alocação de horários em outros turnos, por exemplo, o primeiro semestre do curso de Ciência da Computação possui seis disciplinas que somam 12 aulas e duas delas são ofertadas no turno da tarde.

A partir do cenário que foi descrito nessa seção, foram formuladas algumas restrições que devem ser consideradas na alocação de horários das disciplinas. Essas restrições são:

1. Disciplinas obrigatórias de uma mesma turma não podem chocar horário;
2. Uma disciplina não pode ter duas aulas no mesmo dia;
3. O intervalo entre a primeira e última aula de uma disciplina deve ter no máximo três dias;
4. As disciplinas de duas aulas semanais devem ser alocadas de acordo com os *slots*

predefinidos;

5. Disciplinas de três aulas semanais devem ocupar no máximo dois *slots*;
6. Deve ser alocado o número máximo possível de disciplinas obrigatórias no turno da turma.

O PAHD consiste em encontrar uma alocação de horários que satisfaça todas as restrições acima e ainda maximize as opções de matrícula dos alunos. Isso significa que dada uma possível alocação de horários é preciso saber como essa alocação afeta as opções de matrícula dos alunos, pois este será o critério de permitirá comparar duas alocações e saber qual delas é a melhor. Maximizar as opções de matrícula de um aluno significa diminuir os choques de horário entre as disciplinas que ele precisa cursar, permitindo que ele possa se matricular na maioria ou em todas elas.

4.2 Formulação

O PAHD foi modelado como sendo um PLI binário. Nesse modelo, o impacto de uma alocação sobre as opções de matrículas dos alunos foi estimado como sendo a soma do número de alunos que podem cursar cada par de disciplinas alocadas no mesmo horário. O modelo recebe como entrada uma série de parâmetros que formam uma instância do PAHD. Esses parâmetros estão listados abaixo:

- D é o conjunto de todas as disciplinas da demanda;
- H é o conjunto de horários semanais, $H = \{01, 02, 03, \ldots, 30\}$;
- S é o conjunto dos *slots*, $S \subset H \times H$;
- $hor : D \longmapsto 2^{|H|}$ é uma função que determina os horários em que uma disciplina pode ser alocada de acordo com o seu turno;
- $slots : D \longmapsto 2^{|S|}$ é uma função que fornece os *slots* em que uma disciplina de duas aulas semanais pode ser alocada de acordo com o seu turno;
- $turno : D \longmapsto \{0, 1, 2\}$ é uma função que retorna o turno da disciplina com base em seu conjunto de horários. Os turnos manhã, tarde e noite são representados, respectivamente, por 0, 1 e 2.
- T é o conjunto de turmas da demanda. Uma turma $t \in T$ possui dois conjuntos de disciplinas $t_{obg}, t_{opt} \subset D$, que representam as disciplinas de t que não podem chocar horário, e as que podem chocar horário, respectivamente. Em outras palavras, o conjunto T consiste na união dos subconjuntos de disciplinas obrigatórias e optativas;
- $c_{dd'}$ é o número de alunos que podem cursar o par de disciplinas $d, d' \in D$;

- n_d é o número de aulas semanais da disciplina $d \in D$, ou seja, o número de horários que devem ser atribuídos a essa disciplina.

Além dos parâmetros de entrada, o PLI possui dois grupos de variáveis de decisão:

- $y_{dd'}$ é uma variável binária que indica se as disciplinas $d, d' \in D$ compartilham algum horário;
- x_{dh} é uma variável binária que indica que uma aula da disciplina $d \in D$ está alocada para o horário $h \in H$.

Formalmente o PLI desenvolvido para o PAHD é descrito abaixo:

Função Objetivo:

$$\min \sum_{d \in D} \sum_{d' \in D} c_{dd'} y_{dd'}$$

Restrições:

$$x_{dh} + x_{d'h} \leq y_{dd'} + 1, \forall d, d' \in D, \forall s \in hor(d) \cap hor(d') \tag{4.1}$$

$$\sum_{h \in hor(d)} x_{dh} = n_d, \forall d \in d \tag{4.2}$$

$$\sum_{d \in t_{obg}} x_{dh} \leq 1, \forall t \in T, \forall h \in hor(T) \tag{4.3}$$

$$x_{dh} = x_{dh'}, \forall d \in \{d \in D : n_d = 2\}, \forall (h, h') \in slots(d) \tag{4.4}$$

$$(x_{d19+i} + x_{d20+i}) + (x_{d25+i} + x_{d26+i}) \leq 2 - 2(x_{d01+i} + x_{d02+i}),$$
$$\forall d \in \{d \in D : n_d = 3\}, i = 2 * turno(d) \tag{4.5}$$

$$(x_{d01+i} + x_{d02+i}) + (x_{d07+i} + x_{d08+i}) \leq 2 - 2(x_{d25+i} + x_{d26+i}),$$
$$\forall d \in \{d \in D : n_d = 3\}, i = 2 * turno(d) \tag{4.6}$$

$$(x_{d19+i} + x_{d20+i}) + (x_{d25+i} + x_{d26+i}) \leq 2 - 2(x_{d07+i} + x_{d08+i}),$$
$$\forall d \in \{d \in D : n_d = 3\}, i = 2 * turno(d) \tag{4.7}$$

$$(x_{d01+i} + x_{d02+i}) + (x_{d07+i} + x_{d08+i}) \leq 2 - 2(x_{d19+i} + x_{d20+i}),$$
$$\forall d \in \{d \in D : n_d = 3\}, i = 2 * turno(d) \tag{4.8}$$

$$y_{dd'} \in \{0, 1\}, \forall d, d' \in D \tag{4.9}$$

$$x_{dh} \in \{0, 1\}, \forall d \in D, \forall h \in H \tag{4.10}$$

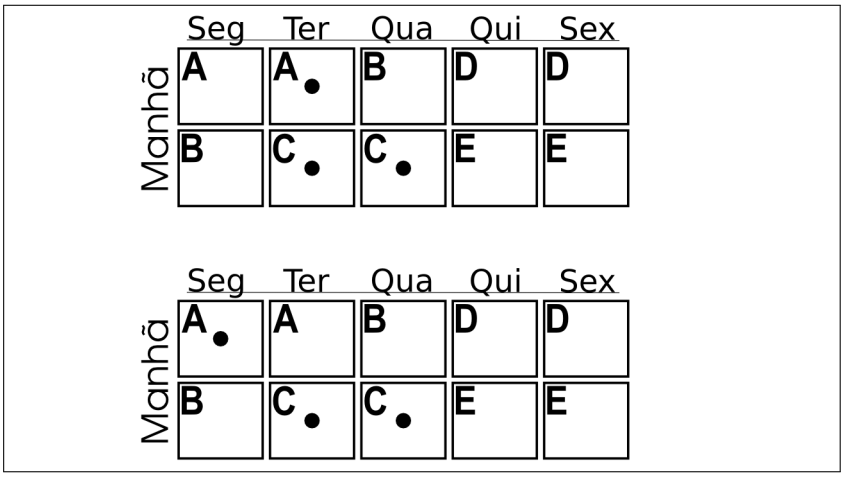
A função objetivo visa minimizar a soma do número de alunos que poderiam fazer cada par de disciplinas com choque de horário, visto que, se um aluno pode cursar d e d' e estas são alocadas no mesmo horário, isso gera um conflito pois ele terá que optar por uma delas apenas.

A restrição (4.1) atribui $y_{dd'} = 1$ quando d e d' compartilham algum horário h. É importante observar que a restrição (4.1) não garante $y_{dd'} = 0$ para $x_{dh} + x_{d'h} \leq 1$, porém como a função objetivo é de minimização, não ocorrerá o caso de $y_{dd'} = 1$ e $x_{dh} + x_{d'h} \leq 1$ desde que $c_{dd'} > 0$. A restrição (4.2) garante que o número de horários atribuídos para a disciplina d seja exatamente igual ao número de aulas desta disciplina. A restrição (4.3) impede que disciplinas obrigatórias de uma mesma turma compartilhem horário. Na restrição (4.4) as duas aulas da disciplina d são forçadas a ficarem em algum dos *slots* predefinidos, sendo que, esta restrição só se aplica a disciplinas de duas aulas semanais.

As restrições (4.5), (4.6), (4.7) e (4.8) são aplicadas às disciplinas que possuem três aulas semanais, impedindo que essas aulas fiquem "espaçadas" ao longo da semana. Nessas restrições o parâmetro i tem o papel de deslocar os horários nos quais elas se aplicam de acordo com o turno de d, para uma melhor compreensão consulte a Figura 1. Se alguma aula de d for alocada na segunda feira, a restrição 4.5 impede hajam outras aulas de d na quinta ou sexta feira. A restrição 4.6 funciona de maneira semelhante, porém no sentido oposto e para os dias sexta, segunda e terça feira. Como efeito colateral, essas duas restrições impedem que as três aulas de d sejam alocadas nos dois primeiros ou nos dois últimos dias da semana. Similarmente, as restrições (4.7) e (4.8) adicionam ao modelo o impedimento de aulas terças e quintas de uma mesma disciplina. É importante frisar que ao adicionar estas restrições serão sacrificadas as soluções em que as aulas de d são distribuídas nos dias de terça, quarta e quinta feira. Entretanto, (4.5), (4.6), (4.7) e (4.8) reduzem o número de possíveis soluções com duas aulas seguidas, de modo que, se ocorrerem é possível removê-las sem muito prejuízo, como no exemplo da Figura 2.

Na Figura 2 os círculo pretos representam as aulas de uma disciplina. Na alocação superior uma aula é transferida para evitar que hajam duas aulas em um mesmo dia. Neste exemplo, ao trocar essa aula de posição não deverão acontecer novos choques de horário, a menos que haja uma disciplina de uma aula semanal alocada na primeira aula da segunda. Não exitem restrições para forçar que as disciplinas de três aulas semanais ocupem apenas dois *slots*, porém a função objetivo irá minimizar os *slots* alocados para estas disciplinas, pois quanto mais *slots* mais choques de horário. As restrições (4.9) e (4.10) são de domínio das variáveis.

Figura 2 – Possível alocação com aulas seguidas(acima) e possível ajuste(abaixo).

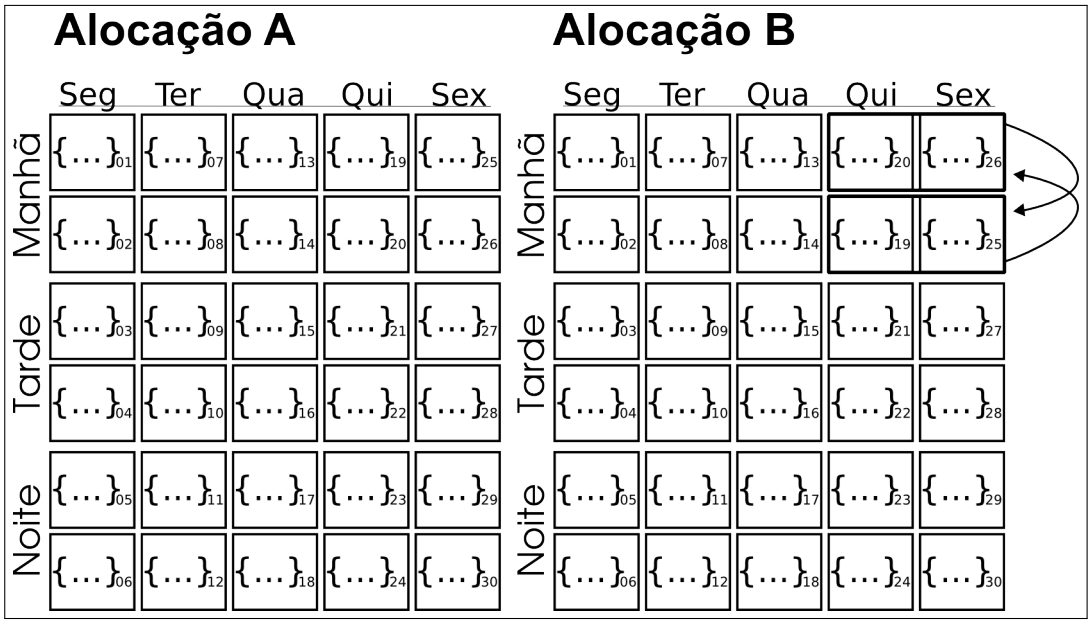


Fonte: Desenvolvida pelo autor

4.2.1 Soluções simétricas

Várias alocações possíveis são simétricas. Duas alocações são **simétricas** quando os grupos de disciplina que chocam horário são os mesmos nas duas alocações, mas em horários diferentes. Isso significa que alocações simétricas são idênticas se considerado apenas o impacto que causam nas opções de matrícula dos alunos. Um exemplo de alocações simétricas é ilustrado na Figura 3.

Figura 3 – Exemplo de alocações simétricas A e B.



Fonte: Desenvolvida pelo autor

Na Figura 3, $\{\dots\}_i$ representa o conjunto de disciplinas que possuem aulas alocadas no horário *i* da alocação A. Na alocação B os conjuntos $\{\dots\}_{19}$ e $\{\dots\}_{20}$, $\{\dots\}_{25}$ e $\{\dots\}_{26}$ foram permutados de horários. É importante ressaltar que nem todas as possíveis alocações simétricas são soluções viáveis do modelo.

5 RESULTADOS EXPERIMENTAIS

A primeira etapa deste trabalho foi desenvolver um modelo de programação linear inteira para o PAHD. O modelo desenvolvido é apresentado no Capítulo 4. As demais etapas são apresentadas ao longo desse capítulo.

5.1 Implementação do modelo

O PLI apresentado na Seção 4.2 foi implementado utilizando a linguagem C++ [1] juntamente com a biblioteca CPLEX [2], que permite a modelagem e resolução de programas lineares e inteiros. Os parâmetros de entrada do modelo são gerados por um *script* desenvolvido na linguagem Python [3] em sua versão 2.7.6 e armazenados em um arquivo que é lido pelo programa que executa o modelo. Primeiramente são lidas as disciplinas, representadas por números inteiros, seguidas dos horários em que são alocáveis. Em seguida são lidos os custos entre os pares de disciplinas $c_{dd'}$ e as turmas. As funções tur, hor e $slots$ são calculadas internamente a partir dos dados de entrada.

Para diminuir o número de variáveis de decisão do modelo as variáveis simétricas foram descartadas. Por exemplo, $x_{dd'}$ e $x_{d'd}$ são simétricas e possuem exatamente o mesmo significado, ou seja, a remoção de uma delas não prejudica o modelo. Apenas as variáveis $x_{dd'}$ e $y_{dd'}$, no qual $d < d'$, foram mantidas. Observe que dessa maneira nenhum par de disciplinas é perdido.

Ao final da execução o programa imprime o valor da função objetivo e consulta o valor das variáveis x_{dh} para gerar a lista de disciplinas com seus respectivos horários, como no exemplo da Figura 4.

---

1 <http://www.cplusplus.com/reference/>
2 <http://www.ibm.com/support/knowledgecenter/SSSA5P_12.6.1/>
3 <https://docs.python.org/2/reference/>

Figura 4 – Exemplo de saída do modelo.

```
CC-1    ARQUITETURA DE COMPUTADORES 19  25
CC-1    FUNDAMENTOS DE PROGRAMAÇÃO  21  27  16
CC-2    PROGRAMAÇÃO ORIENTADA A OBJETOS 1   7
CC-2    MATEMATICA DISCRETA 2   13
CC-2    ESTRUTURA DE DADOS  8   14
CC-2    CÁLCULO DIFERENCIAL E INTEGRAL I    19  25
CC-2    SISTEMAS OPERACIONAIS   20  26
CC-4    ANALISE E PROJETO DE SISTEMAS   19  25
CC-4    PROJETO E ANALISE DE ALGORITMOS 8   14
```

Fonte: Desenvolvida pelo autor

5.2 Obtenção e formatação dos dados da UFC-Quixadá

5.2.1 Dados das disciplinas

O modelo apresentado na Seção 4.2 recebe como parâmetro um conjunto de disciplinas D e um conjunto de turmas T, que foram criados a partir de um arquivo CSV (*Comma-Separeted values*) contendo os dados das disciplinas ofertadas no semestre 2016.2 na UFC-Quixadá, conforme apresentado na Figura 5. Cada linha desse arquivo corresponde a uma disciplina e contém as colunas:

- **ofertada_por** que representa a turma para o qual a disciplina é ofertada;
- **nome** que contém o nome da disciplina;
- **compartilha1** que pode conter uma turma para o qual essa disciplina também será ofertada (disciplina compartilhada);
- **disjunto1** que possuirá o valor 1 para indicar que essa disciplina não poderá chocar horário com as disciplinas obrigatórias da turma indicada por *compartilha1;*
- **compartilha2, disjunto2, compartilha3, disjunto3** são similares a *compartilha1* e *disjunto1*.
- **tipo_slot** indica o turno e o número de aulas da disciplina.

Figura 5 – Demonstração do formato do arquivo de oferta de disciplinas de 2016.2.

	ofertada_por	nome	compartilha1	disjunto1					tipo_slot
2	CC-1	ARQUITETURA DE COMPUTADORES	NULL	0	...	...	...	...	Manhã 2 aulas
3	CC-1	FUNDAMENTOS DE PROGRAMAÇÃO	SI-1	0	...	...	...	...	Tarde 3 aulas
4	CC-2	PROGRAMAÇÃO ORIENTADA A OBJETOS	NULL	0	...	...	...	...	Manhã 2 aulas
5	CC-2	MATEMATICA DISCRETA	NULL	0	...	...	...	...	Manhã 2 aulas
6	CC-2	ESTRUTURA DE DADOS	ES-3	0	...	...	...	...	Manhã 2 aulas
7	CC-2	CÁLCULO DIFERENCIAL E INTEGRAL I	NULL	0	...	...	...	...	Manhã 2 aulas
8	CC-2	SISTEMAS OPERACIONAIS	NULL	0	...	...	...	...	Manhã 2 aulas
9	CC-4	ANALISE E PROJETO DE SISTEMAS	ES-3	0	...	...	...	...	Manhã 2 aulas
10	CC-4	PROJETO E ANALISE DE ALGORITMOS	ES-5	0	...	...	...	...	Manhã 2 aulas

Fonte: Desenvolvida pelo autor

Uma linha do arquivo determina uma disciplina $d \in D$, e ainda $d \in t_{obg}$, onde $t \in T$ é a turma indicada pela coluna *ofertada_por*. Sendo t' uma turma indicada por uma das colunas *compartilha1, compartilha2 ou compartilha3*, se sua respectiva coluna *disjunta1, disjunta2 ou disjunta3* tiver o valor 1, então $d \in t'_{obg}$, caso contrário $d \in t'_{opt}$.

5.2.2 Dados dos alunos

Para obter o valor de custo entre as disciplinas foi utilizado um banco de dados contendo as informações de matrículas de alunos em disciplinas do campus no período de 2007 até o fim de 2015. O banco de dados foi construído a partir de um arquivo CSV que foi obtido através da coordenação do curso de Ciência da Computação. Cada linha desse arquivo representa a matrícula de um aluno em uma disciplina e contém: nome e código da disciplina, curso que ofertou a disciplina, matrícula do aluno e o resultado final (APROVADO ou REPROVADO). Para facilitar a manipulação esses dados foram reestruturados nas tabelas a seguir:

- ALUNO (**mat**, **curso**, **ingresso**) as colunas dessa tabela representam a matrícula, curso e o período de ingresso de um aluno;
- DISCIPLINA (**cod**, **nome**, **ch**) essas colunas representam o código, nome e a carga horária (em horas aula) de uma disciplina;
- APROVAÇÕES (**mat**, **cod**, **periodo**) as colunas dessa tabela referenciam um aluno aprovado em uma disciplina, indicando ainda o período de aprovação.

Posteriormente foi necessário incluir ao banco de dados as informações de pré-requisitos das disciplinas. Uma mesma disciplina pode possuir diferentes pré-requisitos dependendo do curso que a oferta. Por esta razão, a matriz curricular de cada curso foi consultada para construir a tabela pré-requisitos que é descrita a seguir:

- PRÉ-REQUISITOS (**cod_requisito**, **cod**, **curso**) nessa tabela a coluna **cod_requisto** representa o código da disciplina que é pré-requisito da disciplina referenciada por **cod** e a coluna **curso** indica o curso em que esse pré-requisito é válido.

Antes de calcular o custo entre duas disciplinas é preciso definir para quais disciplinas isso será necessário, por exemplo, o custo entre disciplinas de turnos diferentes é irrelevante para uma alocação. No caso de disciplinas do mesmo turno, o custo deve sempre ser considerado se elas forem do mesmo curso. Se as disciplinas, d e d', são de cursos diferentes do mesmo turno, o custo entre elas só deve ser considerado no seguinte cenário: d é compartilhada com alguma turma do mesmo curso de d' ou o contrário.

Idealmente o valor de custo $c_{dd'}$ deveria ser o número de alunos que podem ou precisam cursar d e d'. No entanto isso não é possível, pois a alocação de horários de um semestre letivo é realizada durante o semestre anterior e ainda não se tem os resultados de aprovações dos alunos. Isso é um problema pois um aluno pode reprovar uma disciplina que é pré-requisito de outra que será ofertada no próximo semestre, ou ainda, por causa dessa reprovação poderá cursar novamente a disciplina reprovada caso esta seja ofertada. Uma outra maneira de estimar $c_{dd'}$ seria utilizar uma média histórica de quantos alunos cursaram esse par de disciplinas. O problema dessa abordagem é que média poderia ser tendenciosa, pois o número de alunos que cursam um par de disciplinas é diretamente afetado pela alocação de horários que foi empregada naquele semestre.

Neste trabalho, o custo foi estimado como sendo o número de alunos que podiam cursar d e d' no mesmo período do ano anterior. Essa abordagem tem como vantagens não ser influenciada pelas alocações anteriores e poder ser estimada sem os resultados de aprovações do semestre atual, porém pode não ser muito precisa. Um aluno contribui para o custo $c_{dd'}$ quando ele possui todos os pré-requisitos para cursar d e d' e ainda não foi aprovado em nenhuma delas. Usando o banco de dados descrito anteriormente o custo foi calculado seguindo o Algoritmo 1.

Nas linhas 1 e 2 a função **pre_requisitos** é utilizada para criar os conjuntos $P_d, P_{d'}$ contendo os pré-requisitos das disciplinas d e d'. Se a disciplina d não possui pré-requisitos, os alunos que podem cursá-la são todos aqueles que ainda não foram aprovados em d, linhas 3 e 4. Para encontrar os alunos que não foram aprovados em uma disciplina é necessário acessar o banco de dados, esse acesso é feito pela função **naoaprovados**. De forma similar a função **aprovados**, acessa o banco de dados para retornar os alunos que já foram aprovados em uma dada disciplina. Essas duas funções recebem ainda um curso e um semestre letivo (ex. 2016.1)

Algoritmo 1: *Custo entre disciplinas*

Entrada: Duas disciplinas d e d', um curso *curso* e um semestre letivo *periodo*
Saída: Número de alunos de um curso que podiam cursar as disciplinas d e d em um determinado período

```
início
1   | P_d ← pre_requisitos(d, curso)
2   | P_d' ← pre_requisitos(d', curso)
3   | se P_d = ∅ então
4   |   | aptos_d ← naoaprovados(d, curso, periodo)
    | senão
5   |   | para cada requisito ∈ P_d faça
6   |   |   | aptos_d ← aptos_d ∪ aprovados(requisito, curso, periodo)
    |   | fim
7   |   | aptos_d ← aptos_d ∩ naoaprovados(d, curso, periodo)
    | fim
8   | se P_d' = ∅ então
9   |   | aptos_d' ← naoaprovados(d', curso, periodo)
    | senão
10  |   | para cada requisito ∈ P_d' faça
11  |   |   | aptos_d' ← aptos_d' ∪ aprovados(requisito, curso, periodo)
    |   | fim
12  |   | aptos_d' ← aptos_d' ∩ naoaprovados(d', curso, periodo)
    | fim
13  | retorne |aptos_d ∩ aptos_d'|
fim
```

para refinar a busca. Parâmetro *curso* é usado para filtrar apenas alunos daquele curso e o *periodo* é um limite superior para o período de aprovação. No caso de d possuir pré-requisitos os alunos aptos a cursá-la são aqueles que foram aprovados em todos os pré-requisitos e não foram aprovados em d, linhas 5 a 7. A segunda metade do algoritmo, linhas 8 a 12, encontra os alunos aptos a cursar d' de maneira similar. O algoritmo termina retornando o número de alunos que são aptos a cursar d e d' na linha 13. Uma observação importante é que esse algoritmo recebe apenas um curso como parâmetro, entretanto d e d' não necessariamente são do mesmo curso. Se d e d' são de cursos diferentes, necessariamente uma delas é compartilhada com o curso da outra, neste caso o curso que deve ser passado para o algoritmo deve ser o da disciplina é compartilhada.

Como mencionado anteriormente o valor retornado pelo Algoritmo 1 pode ser impreciso. Isso se deve primeiramente ao fato de que os alunos do ano anterior podem ter um perfil diferente dos alunos do ano corrente. Outro problema são disciplinas sem pré-requisito, pois usualmente o número de alunos que podem se matricular nelas é bem maior que os que

de fato se matriculam. Quanto maior o semestre de oferta da disciplina, maior o erro. Como uma forma de contornar esse problema todos os custos foram ajustados da seguinte maneira: $c'_{dd'} = c_{dd'} * \alpha^i$, onde i é o módulo da diferença entre os semestres de oferta de d e d' e $0 < \alpha < 1$ é um parâmetro de entrada.

5.3 Análise do custo real de uma alocação

Como o custo entre duas disciplinas é representado por um valor numérico, a função objetivo não leva em conta alunos repetidos. Por exemplo, sejam d_1, d_2 e d_3 um grupo de disciplinas que chocam horário e seja a um aluno que possa cursar as três disciplinas desse grupo. Desconsiderando o ajuste realizado entre disciplinas de semestres diferentes, o aluno a vai contribuir com o valor 1 no custo entre d_1 e d_2; d_1 e d_3; e d_2 e d_3. Dessa maneira, a vai ser somado 3 vezes na função objetivo, porém ele perdeu apenas duas matrículas nesse grupo. Se outro aluno b puder cursar apenas duas disciplinas desse grupo, ele será somado apenas uma vez na função objetivo, o que seria o valor correto. Por causa dessas peculiaridades a função objetivo não calcula corretamente o número de matrículas perdidas. O número de matrículas perdidas é o que é chamado aqui de **custo real**. Dada uma alocação de horários, o custo real pode ser calculado da seguindo o Algoritmo 2.

Dados um grupo de n disciplinas que chocam horários e um aluno a que possa fazer i ($2 \leq i \leq n$) disciplinas desse grupo, o Algoritmo 2 vai contabilizar a $i-1$ vezes no valor final, o que corresponde ao número de matrículas que ele perdeu nesse grupo de disciplinas. A razão disso é que embora a possa ser adicionado várias vezes no conjunto *alunos* para uma mesma disciplina d, ele irá contribuir com apenas 1 na cardinalidade do conjunto. Dessa maneira, dada a sequência ordenada de disciplinas que a pode cursar e que chocam horário, a será contabilizado uma vez para cada disciplina d dessa sequência, exceto quando d for a última, pois para que a seja adicionado em *alunos* é preciso que haja outra disciplina $d' > d$ na sequência, o que só não acontece para a última disciplina.

O Algoritmo 2 é aplicado a uma alocação de horários de algum semestre letivo A. A partir de uma alocação A é possível extrair a lista de disciplinas que foram alocadas e o semestre letivo em que A foi empregada através das funções **disciplinas** e **periodo**, linhas 1 e 2, respectivamente. O laço da linha 5 seleciona uma disciplina d enquanto o laço da linha 9 irá selecionar toda disciplina d', tal que $d' > d$ (neste abuso de notação, $d' > d$ quando d' vier depois na lista de disciplinas). Na linha 11, a função **horarios** retorna o conjunto de horários

Algoritmo 2: *Custo real de uma alocação*

```
   Entrada: Alocação de horários de disciplinas de um semestre letivo: A
   Saída: Número de matrículas perdidas em uma alocação
   início
1      D ← disciplinas(A)
2      periodo ← periodo(A)
3      total ← 0
4      i ← 1
5      enquanto i < |D| faça
6          d ← D[i]
7          j ← i+1
8          alunos ← ∅
9          enquanto j ≤ |D| faça
10             d' ← D[j]
11             se horarios(d,A) ∩ horarios(d',A) ≠ ∅ E compartilhadas(d,d',A) então
12                 curso ← cursooferta(d,d',A)
13                 alunos ← alunos∪ custo*(d,d',curso,periodo)
               fim
14             j ← j+1
15
           fim
16         total ← total + |alunos|
17         i ← i+1
       fim
18     retorne total
   fim
```

de uma disciplina em uma alocação. A função **compartilhadas** recebe duas disciplinas d, d' e uma alocação e retorna *VERDADE* quando d e d' são do mesmo curso ou quando são de cursos diferentes mas suas turmas possuem alguma disciplina compartilhada, em caso contrário retorna *FALSO*. O teste da linha 11 verifica se d e d' chocam horário e se o choque de horário entre elas é relevante.

A função **cursooferta**, na linha 12, retorna o curso da disciplina d se d possui vagas para alguma turma (se possui vagas reservadas para alunos de uma turma) do curso de d'. No caso de d' ser compartilhada com alguma turma do curso de d, o curso da disciplina d' será retornado. Se ambas as situações forem verdadeiras o retorno é qualquer um dos cursos. A ideia dessa função é priorizar os alunos do curso que ofertou a disciplina com vagas compartilhadas que tornou relevante o custo entre essas disciplinas.

A função **custo*** é o Algoritmo 1 modificado para retornar o conjunto de alunos aptos a cursar d e d' no lugar de retornar apenas a cardinalidade do conjunto. Dito isto, ao fim

do laço mais interno o conjunto *alunos* irá conter os alunos que podem fazer d e d', para toda disciplina $d' > d$ que choque horário com d. No fim do laço mais externo o valor total é somado ao número de alunos presentes no conjunto *alunos*.

5.4 Execução do modelo com os dados da UFC-Quixadá

O modelo foi executado para uma instância com 78 disciplinas dos cursos de Ciência da Computação, Engenharia de Software, Redes de Computadores e Sistemas de Informação. As disciplinas dos Cursos de Engenharia de Computação e Design Digital foram desconsideradas nesta instância, pois não existem dados suficientes para utilizar na computação dos custos, já que em 2015.2 algumas disciplinas destes cursos ainda não haviam sido cursadas. Uma outra instância contendo essas disciplinas foi criada para verificar o tempo de execução do modelo. O Quadro 1 mostra como as disciplinas ficaram distribuídas nas instâncias de teste.

Quadro 1 – Distribuição de disciplinas nas instâncias de teste.

Instância	1 aula semanal	2 aulas semanais	3 aulas semanais	5 aulas semanais
78 disciplinas	6	67	3	2
106 disciplinas	6	94	4	2

Fonte: Elaborado pelo autor

As disciplinas de 5 aulas que são apresentadas no Quatro 1 são referentes ao estágio supervisionado e possuem horários pré-determinados que foram devidamente fixados no modelo.

Uma dificuldade encontrada na execução do modelo foi o grande número de alocações simétricas. O modelo encontrava muitas soluções que eram "idênticas" e estourava a memória sem encontrar uma solução ótima. Para reduzir as alocações simétricas foi necessário fixar os horários de algumas disciplinas. Como as disciplinas fixas não podem permutar horário, situações como a que foi mostrada na Figura 3 não podem mais ocorrer. Foi fixada uma turma por turno, e estas turmas fixadas foram passadas como parâmetros para o modelo.

A execução do modelo foi realizada em um computador executando o sistema operacional Ubuntu 14.04.5 LTS e com as seguintes configurações de hardware: processador Intel(R) Core(TM) i5-4210U @ 1.70GHz, 8GB de memória RAM e com 8GB de espaço em disco reservado para *swap*.

Para a instância de 78 disciplinas o modelo encontrou uma solução ótima em 257.69 segundos e utilizou aproximadamente 694.68 MB de memória. Para realizar uma comparação com a alocação sugerida pelo modelo, foram calculados os custos da função objetivo e o custo

real da alocação que foi empregada no campus nesse semestre letivo (2016.2). Os valores das duas alocações são mostrados na Tabela 1.

Tabela 1 – Custos das alocações.

Alocação	custo objetivo	custo real
Alocação sugerida pelo modelo	1648.05	1647
Alocação empregada no campus em 2016.2	2744.0	2932

Fonte: Produzido pelo autor
custo objetivo: valor da função objetivo.
custo real: valor calculado seguindo o Algoritmo 2.

A alocação sugerida pelo modelo obteve uma redução de aproximadamente 44% no número de opções de matrícula perdidas, com relação a alocação que foi empregada no campus. Porém, a alocação de 2016.2 leva em consideração várias outras restrições não consideradas neste trabalho.

Nos testes com ambas instâncias os custos entre os pares de disciplinas foram ajustados com o fator $\alpha = 0.9$, conforme descrito na Seção 5.2.2. Esse ajuste foi o que aproximou o custo obtido pela função objetivo com o custo real. O fator $\alpha = 0.9$ foi estimado empiricamente, uma melhor estimativa pode requerer testes mais rigorosos.

Para a instância com 106 disciplinas, o modelo encontrou uma solução ótima em 680.24 segundos, e utilizou cerca de 1155.82 MB de memória. O valor da função objetivo foi 1813.72, porém essa não é uma informação muito significativa pois os custos envolvendo disciplinas dos cursos Engenharia de Computação e Design Digital foram gerados aleatoriamente. O tempo de resposta e o uso de memória sugerem que esse modelo poderia ser aplicado para realizar a alocação do semestre 2016.2, considerando o tamanho da instância. Contudo, como o número de disciplinas ofertadas a cada semestre vai aumentar conforme as turmas iniciais dos cursos mais novos vão avançando, talvez seja necessário uma reformulação no modelo para seja possível obter alocações para instâncias maiores em tempo hábil.

6 CONSIDERAÇÕES FINAIS

Esse trabalho tinha como objetivo encontrar uma alocação de horários para as disciplinas da UFC-Quixadá que maximizasse as opções de matrícula dos alunos do campus. Esse objetivo foi alcançado, pois foi desenvolvido um modelo de programação linear inteira para o problema e esse modelo foi executado com instâncias geradas a partir de dados dos alunos e disciplinas do campus.

De maneira geral, o modelo encontrou uma solução melhor (considerando apenas as opções de matrícula dos alunos) que a que foi utilizada no campus no semestre de 2016.2. Apesar da comparação ter sido realizada com um número reduzido de disciplinas, também foram executados testes com um número maior de disciplinas. E estes testes sugerem que o modelo poderia ter fornecido uma alocação completa para o semestre 2016.2.

A necessidade de fixar uma turma em cada turno tira um pouco a flexibilidade do modelo, pois as disciplinas serão alocadas com base naquelas que foram fixadas. Porém, como as disciplinas que serão fixadas são recebidas como entrada, o modelo pode ser testado com diferentes combinações dessas entradas. Além disso, a alocação final pode ser uma alocação simétrica daquela que foi encontrada pelo modelo, dessa forma as opções de matrículas dos alunos ainda serão valorizadas.

Outro problema que pode inviabilizar o uso desse modelo de forma efetiva no campus é que a alocação sugerida pelo modelo pode não ser viável para a alocação de professores, visto que os horários alocados para as disciplinas podem não ser compatíveis com os horários disponíveis dos professores. Soluções simétricas também podem ajudar nesse problema, mas só se os ajustes forem pequenos.

A qualidade da solução fornecida pelo modelo depende diretamente de como os custos entre as disciplinas são calculados e quão próximos são dos valores reais. Se a alocação acontecesse após os resultados finais das disciplinas, seria possível calcular mais precisamente esses custos e consecutivamente o modelo forneceria melhores soluções.

REFERÊNCIAS

AHUJA, R.; MAGNANTI, T.; ORLIN, J. **Network flows: theory, algorithms, and applications**. [S.l.]: Prentice Hall, 1993.

BABAEI, H.; KARIMPOUR, J.; HADIDI, A. A survey of approaches for university course timetabling problem. **Computers & Industrial Engineering**, Elsevier, v. 86, p. 43–59, 2015.

CORMEN, T. H. **Introduction to algorithms**. [S.l.]: MIT press, 2009.

GENOVA, K.; GULIASHKI, V. Linear integer programming methods and approaches–a survey. **Journal of Cybernetics and Information Technologies**, v. 11, n. 1, 2011.

KRIPKA, R. M. L.; KRIPKA, M.; SILVA, M. C. da. Formulação para o problema de alocação de salas de aula com minimização de deslocamentos.". **XLIII Simpósio Brasileiro de Pesquisa Operacional**, 2011.

KUHN, H. W. The hungarian method for the assignment problem. **Naval research logistics quarterly**, Wiley Online Library, v. 2, n. 1-2, p. 83–97, 1955.

LACH, G.; LÜBBECKE, M. E. Optimal university course timetables and the partial transversal polytope. In: **Experimental algorithms**. [S.l.]: Springer, 2008. p. 235–248.

SCHRIJVER, A. **Theory of linear and integer programming**. [S.l.]: John Wiley & Sons, 1998.

WOLSEY, L. A. **Integer programming**. [S.l.]: Wiley New York, 1998.