Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

Otimização na Alocação de Disciplinas a Professores

Saulo Martinho Alves dos Santos

TRABALHO DE CONCLUSÃO DE CURSO

ORIENTAÇÃO:

George Henrique Godim da Fonseca

COORIENTAÇÃO:

Janniele Aparecida Soares Araújo

Julho, 2019
João Monlevade–MG

Saulo Martinho Alves dos Santos

Otimização na Alocação de Disciplinas a Professores

Orientador: George Henrique Godim da Fonseca

Coorientador: Janniele Aparecida Soares Araújo

Monografia apresentada ao curso de Sistemas de Informação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina "Trabalho de Conclusão de Curso II".

Universidade Federal de Ouro Preto

João Monlevade

Julho de 2019

S237o Santos, Saulo Martinho Alves dos.
Otimização na alocação de disciplinas a professores [manuscrito] / Saulo Martinho Alves dos Santos. - 2019.

68f.: il.: color; tabs.

Orientador: Prof. Dr. George Fonseca.
Coorientadora: Profª. MScª. Janniele Araújo.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Pesquisa operacional. 2. Otimização matemática . 3. Algoritmos. I. Fonseca, George. II. Araújo, Janniele. III. Universidade Federal de Ouro Preto. IV. Titulo.

CDU: 519.8

Catalogação: ficha.sisbin@ufop.edu.br

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

Curso de Sistemas de Informação

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Otimização na Alocação de Disciplinas a Professores

Saulo Martinho Alves dos Santos

Monografia apresentada ao Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI499 – Trabalho de Conclusão de Curso II do curso de Bacharelado em Sistemas de Informação e aprovada pela Banca Examinadora abaixo assinada:

Prof. Dr. George Henrique Godim da Fonseca
DECSI - UFOP
Professor Orientador

Profa. Me. Janniele Aparecida Soares Araújo
DECSI - UFOP
Professor Coorientador

Prof. Dr. Darlan Nunes de Brito
DECSI - UFOP
Professor Convidado

Prof. Samuel Souza Brito
DECSI - UFOP
Professor Convidado

João Monlevade, 02 de julho de 2019

Dedico este trabalho primeiramente à minha mãe Márcia Marisa, que sempre me incentivou a estudar e me apoiou na minha caminhada. E a meu pai Geraldo dos Santos que sempre esteve comigo nos momentos que precisei.

Agradecimentos

Agradeço aos meus pais Márcia e Geraldo pelos sacrifícios feitos para me educar durante toda minha vida.

Agradeço a Paloma por me acompanhar nessa jornada ao longo desses anos.

Agradeço aos professores do curso de graduação em Sistema de Informação da Universidade Federal de Ouro Preto, em especial ao meu orientador George Henrique Godim da Fonseca pela orientação e a Janniele Aparecida Soares Araújo pela coorientação.

Aos meus amigos que me acompanharam e me apoiaram ao longo do curso.

"Enquanto estiver vivo, sinta-se vivo."

—Madre Teresa de Calcutá (1910– 1997)

Resumo

O problema da Atribuição de Tarefas, ou *Assignment Problem*, consiste na distribuição de um conjunto de tarefas a um conjunto determinado de recursos, maximizando a compatibilidade entre eles. Esse problema é o objeto de pesquisa e pertence à área de otimização combinatória. O presente trabalho propõe uma solução para a alocação dos professores às respectivas disciplinas a serem lecionadas durante um semestre letivo no Departamento de Computação e Sistemas (DECSI), através de um modelo de emparelhamento em grafos. Para a montagem da matriz de custo foram definidos os seguintes requisitos: área de concurso, experiência do docente na(s) disciplina(s) em questão e a preferência do docente na(s) disciplina(s) que deseja lecionar . A matriz de custo foi utilizada no método Húngaro para encontrar a solução para o problema de Atribuição de Tarefas. A solução gerada atendeu aos objetivos propostos pelo trabalho.

Palavras-chaves: Atribuição de Tarefas. *Assignment Problem*. Método Húngaro.

Abstract

The task assignment problem, or Assignment Problem, is the distribution of a set of tasks to a given set of resources, maximizing the compatibility between them. This problem is the search object and belongs to the combinatorial optimization area. The present work proposes a solution for the allocation of the professors to the respective disciplines to be taught during a semester in the Departamento de Computação e Sistemas (DECSI), through a model of pairing in graphs. In order to set up the cost matrix, the following requirements were defined: the area of the competition, the teacher's experience in the subject (s) in question and professors preference in the discipline (s) he wishes to teach. The cost matrix was used in the Hungarian method to find the solution to the Task Assignment problem. The solution generated met the objectives proposed by the work.

Key-words: Assignment Problem. Hungarian algorithm.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

DEELT Departamento de Engenharia Elétrica

DECSI Departamento de Computação e Sistemas

DECEA Departamento de Ciências Exatas e Aplicadas

DEENP Departamento de Engenharia de Produção

ICEA Instituto de Ciências Exatas e Aplicadas

UFOP Universidade Federal de Ouro Preto

Lista de símbolos

G	Grafo
V	Conjuntos de vértices
E	Conjunto de aretas
$\in$	Pertence

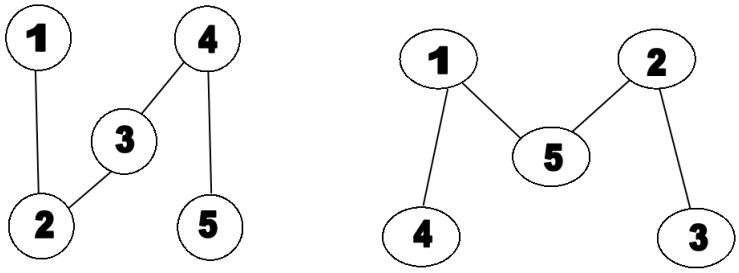
Sumário

1 Introdução

O presente trabalho contém um estudo de um problema e a criação de um método para solucioná-lo. Foi escolhido o problema de Atribuição de Tarefas ou *Assignment Problem*, que consiste em atribuir um conjunto de tarefas a um conjunto de recursos maximizando a compatibilidade entre os mesmos (CELA, 2013).

Um grafo, G = (V, E), é uma estrutura matemática composta por um conjunto de vértices V = $\{v_1, v_2, ..., v_n\}$ e um conjunto de ligações entre esses vértices, denominado arestas E = $\{e_1, e_2, ..., e_m\}$, conforme apresentado na Figura 1.

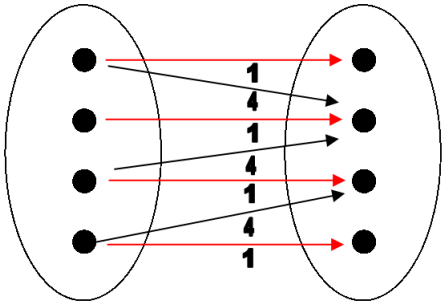
Figura 1 – Grafo



Fonte: Elaborado pelo autor

Um grafo bipartido é um grafo onde o conjunto de vértices é separado em dois conjuntos disjuntos V_1 e V_2, sendo que cada aresta e $\in$ E liga um elemento de V_1 a um elemento em V_2. Cada aresta possui ainda um peso, que representa o custo de se ligar os vértices que conecta, conforme apresentado na Figura 2.

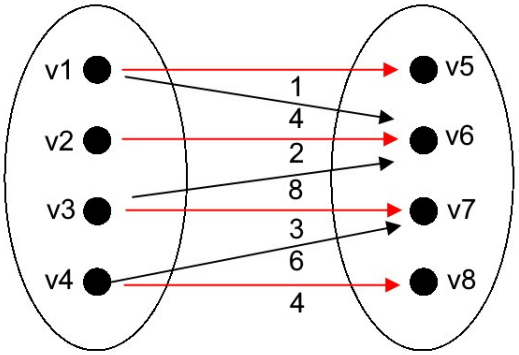
Figura 2 – Grafo Bipartido



Fonte: Elaborado pelo autor

O problema do emparelhamento pode ser de custo mínimo ou máximo. Este problema consiste em selecionar um subconjunto de arestas de menor ou maior custo possível, que por sua vez, deve ligar todos os vértices em um grafo bipartido, conforme apresentado na Figura 3. As arestas em vermelho indicam um menor custo. Exemplo: o vértice V_1 liga-se ao vértice V_5 com um custo 1.

Figura 3 – Emparelhamento de custo mínimo



Fonte: Elaborado pelo autor

Alguns problemas do nosso cotidiano podem ser mapeados como problema de atribuição e devem ser resolvidos com emparelhamento em grafos. Com o avanço da tecnologia, um leque de possibilidades para resoluções de problemas foram exploradas. Porém, ainda existem problemas do nosso cotidiano que não podem ser resolvidos em suas completas particularidades.

É possível dizer que o problema de atribuição, no meio científico, é de certa forma recorrente, se encaixando em diversas situações no que se refere à resolução de problemas.

Há uma extensa bibliografia a respeito, incluindo livros, revistas, periódicos e artigos. Estes contêm abordagens e soluções sobre os problemas que são análogos ao problema supracitado. Referenciais de estudo e soluções para este tema são encontrados em (SOARES, 2011), (DASGUPTA et al., 2008), (KUHN, 1955b), (CELA, 2013).

O problema do emparelhamento pode ser resolvido através de algoritmos polinomiais, como o algoritmo Húngaro (KUHN, 1955b); entretanto, a adição de restrições específicas aumenta a complexidade do problema, tornando-se inviável sua simples representação como um grafo. De fato, a maioria das aplicações práticas desse problema são classificadas como NP-Difícil (GAREY; JOHNSON, 1979).

1.1 Objetivos

O presente trabalho consiste em resolver o problema de alocação de professores às disciplinas, implementando o algoritmo Húngaro de modo a automatizar o processo, que até então era feito de forma manual. Informações de entrada serão os requisitos do algoritmo para gerar a matriz de custos. A saída será uma alocação de um professor para cada disciplin,a levando-se em conta o melhor benefício de combiná-los e as restrições impostas por eles.

1.1.1 Objetivos específicos

Este trabalho possui os seguintes objetivos específicos:

- Encontrar soluções otimizadas de alocações de disciplinas à professores para utilizar no DECSI da Universidade Federal de Ouro Preto (UFOP);
- Atender aos requisitos de restrição especificados na entrada;
- Facilitar e agilizar a criação do horário DECSI, do modo manual para o automatizado;
- Atender a certas preferências de modo a fazer uma alocação de cada disciplina a pelo menos um professor.

1.2 Organização do trabalho

O restante deste trabalho é organizado como se segue. O Capítulo 2 apresenta uma revisão bibliográfica dos conceitos relacionados ao desenvolvimento do trabalho e sistemas correlatos. O Capítulo 3 descreve todo o processo da implementação do algoritmo, desde as etapas iniciais, com o levantamento de requisitos e levantamento de dados. No Capítulo 4 são apresentadas as apreciações sobre o resultado do estudo e os tipos de

implementações abordadas. No Capítulo 5 são apresentadas as considerações finais e propostas para trabalhos futuros.

2 Revisão bibliográfica

O presente capítulo apresentará os principais conceitos que abrangem o problema de alocação de tarefas e suas áreas de interesse de estudo. A Seção 2.1 faz um apanhado de conhecimentos que se relacionam com o problema estudado. As seções de 2.1.1 à 2.1.3 apresentam sobre qual campo de pesquisa estuda este tipo de problema. O problema de alocação de tarefas é apresentado na Seção 2.1.4. A Seção 2.1.5 trará a formalização matemática do problema. A Seção 2.1.6 trará descrição do algortimo utilizado. A Seção 1 trará pseudocódigo e resolução de problema como exemplo. A Seção 2.2 irá expor trabalhos correlatos com objetivo de contextualizar melhor sobre o problema tratado nesta monografia.

2.1 Referencial Teórico

2.1.1 Pesquisa Operacional e Programação Linear

Pesquisa Operacional é a aplicação de métodos científicos a problemas complexos para auxiliar no processo de tomadas de decisão, tais como projetar, planejar e operar sistemas em situações que requerem alocações eficientes de recursos escassos (ARENALES et al., 2017). Um problema básico em pesquisa operacional é distribuir univocamente tarefas para instalações de um modo otimizado. Tarefas como encontrar a melhor distribuição de componentes em chips (KHOO; CONG, 1992), trabalhadores em empregos (SHEN; TZENG; LIU, 2003), maquinário em locais de construção (FURTADO; LORENA, 1997) e professores para disciplinas (SOARES, 2011).

O matemático e físico Jean-Baptiste durante o século XVIII causou uma reviravolta propondo inovações nos métodos de resolução de inequações lineares. Posteriormente ao longo da história outros grandes estudiosos proporão vários métodos de utilização e resolução de problemas utilizando inequações lineares. Estas proposições auxiliam ainda hoje na formulação de problemas complexos que só podem ser resolvidos através da computação.

A formulação do problema a ser resolvido segue três pontos básicos:

- Definição do objetivo (maximar ou minimar o resultado).
- Definição das variáveis (características tangíveis que podem ser quantificadas).
- Definição das restrições (regras que fazem com que o problema seja possível).

2.1.2 Características do Modelo de Programação Linear

Segundo Goldbarg e Luna (2005) um problema de programação linear deve atender aos seguintes requisitos:

- Proporcionalidade: a quantidade de recursos consumidos por uma dada atividade deve ser proporcional ao nível dessa atividade na solução final do problema. Além disso, o custo de cada atividade é proporcional ao nível de operação da atividade.
- Não Negatividade: deve ser sempre possível desenvolver dada atividade em qualquer nível não negativo, e qualquer proporção de um dado recurso deve sempre poder ser utilizado.
- Aditividade: o custo total é a soma das parcelas associadas a cada atividade.
- Separabilidade: pode-se identificar de forma separada o custo (ou consumo de recursos) específico das operações de cada atividade.

Os problemas de otimização são definidos como de Programação Linear quando sua função objetivo e restrições atendam aos conceitos de linearidade (GOLDBARG; LUNA, 2005). Assim também problemas onde variáveis apenas assumem valores pertencentes ao conjunto de números naturais há uma subclasse de Programação Linear denominada de Programação Inteira ou Programação Linear Inteira. Este tipo de problema é considerado NP-difícil. O seguinte estudo atende a essas peculiaridades, sendo assim um caso de Programação Linear Inteira.

2.1.3 Programação Linear Inteira

A programação inteira linear é uma subclasse da Programação Linear onde as variáveis assumem apenas valores inteiros. A fórmula abaixo apresenta a definição matemática Walser (1999).

$$\begin{aligned} max &= c'x \\ Ax &\leq b \\ x &\geq 0 \end{aligned}$$

O que essa formulação representa:

- A é uma matriz m por n;
- c é um vetor linha n-dimensional;
- b é um vetor coluna m-dimensional;
- e x é um vetor coluna n-dimensional de variáveis ou valores desconhecidos.

Sendo todas as variáveis inteiras, haverá um Programa Inteiro. Enfim se fazem necessárias variáveis discretas sem possuir valores contínuos.

2.1.4 Problema de Alocação

No problema de alocação de tarefas existe um número de agentes e um número de tarefas. Podemos alocar qualquer agente a qualquer uma das tarefas, porém, cada alocação tem um custo que pode variar dependendo cada tarefa e agentes específicos. É necessário que todas as tarefas sejam feitas, designando exatamente um agente para cada tarefa de modo que o custo total a soma de todas as alocações seja minimizado (DASGUPTA et al., 2008).

Este tipo de problema (que pode ser nomeado também como designação, *matching*, emparelhamento, dentre outros nomes) constitui uma parte importante da Ciência da Computação e Matemática, com aplicação prática direta. Em um problema deste tipo, tem se dois conjuntos (agentes-tarefas, trabalhadores-empregos, entre outros exemplos) e deve-se encontrar uma função que ligue elementos destes dois conjuntos. Pode haver (e na maioria dos casos há) restrições e requisitos para a ligação de um par de elementos, constituindo um custo para a designação. O problema está em encontrar a função que minimiza o custo somado de todas as alocações, respeitando as restrições existentes (SOARES, 2011).

O desafio está em como planejar as entradas corretas e todas as restrições de modo matemático. O que muitas vezes não reflete perfeitamente o mundo em que vivemos.

Utilizando-se dos Grafos e da Programação Linear pode-se modelar o problema de forma científica de modo a quantificar o problema e buscar uma solução mais eficiente em casos de minimizar custos (emparelhamento mínimo) ou maximizar lucros (emparelhamento máximo).

Segundo o Teorema da Alocação Ótima, a soma das n entradas de uma alocação é chamada de custo da alocação. Uma alocação com o menor custo possível é denominada uma alocação ótima de tarefas. O problema da alocação de tarefas consiste em encontrar uma alocação ótima a partir de uma matriz custo dada. Teorema: um número real é somado ou subtraído de todas as entradas de uma linha ou coluna de uma matriz custo, então uma alocação ótima para a matriz custo resultante é também uma alocação de tarefas ótima para a matriz custo original (RODRIGUES; VIEIRA; AGUSTINI, 2005).

2.1.5 Formulação Matemática

Uma formulação matemática apresentada no trabalho Soares (2011) é definida a seguir.

O conjunto A representa os Agentes.

O conjunto T representa as Tarefas.

A variável x_{ij} foi modelada da seguinte maneira:

$$x_{i,j} = \begin{cases} 1 & se \quad f(i) = j \\ 0 & caso\, contrario \end{cases}$$

Assim sendo define-se a função objetivo por:

$$min \sum_{i \in A} \sum_{j \in T} c(i,j) x_{ij} \tag{2.1}$$

A fim de garantir que apenas um elemento de A associe-se apenas um elemento de T:

$$\sum_{j \in T} x_{i,j} = 1 \ \forall i \in A \tag{2.2}$$

A fim de garantir que apenas um elemento de T associe-se apenas um elemento de A:

$$\sum_{i \in A} x_{i,j} \; = \; 1 \; \forall j \; \in T \tag{2.3}$$

A fim de garantir que a variável x_{ij} assuma valores binários:

$$x_{i,j} \; \in \{0,1\} \, , \; i \; \in \; A, \; j \; \in \; T \tag{2.4}$$

x_{ij} assume o valor 1 quando i foi associado a j, e 0 caso contrário.

Temos a formulação completa:

$$min \sum_{i \in A} \sum_{j \in T} c(i,j) x_{ij} \tag{2.5}$$

$$\sum_{i \in A} x_{i,j} \; = \; 1 \; \forall j \; \in T \tag{2.6}$$

$$\sum_{j \in T} x_{i,j} \; = \; 1 \; \forall i \; \in A \tag{2.7}$$

$$x_{i,j} \; \in \{0,1\} \, , \; i \; \in \; A, \; j \; \in \; T \tag{2.8}$$

O problema estudado neste artigo pode-se dizer ser uma subclasse do problema do transporte onde definimos $k = 1$. Isto implica que cada recurso será ligado a apenas uma tarefa. Caso seja necessário pode-se atribuir uma tarefa n vezes ou um recurso n vezes. Estes devem ter entradas n vezes, pois representão elementos distintos mesmo tendo valores iguais. Exemplo: uma máquina trabalha em três turnos durante um dia assim ela necessitará de três operadores devida à troca de turnos.

Temos a formulação para o Problema do transporte:

$$\sum_{j \in T} x_{i,j} \; \leq \; k, \; \forall i \; \in A$$

2.1.6 Método Húngaro

Kuhn (1955b) divulgou o método Húngaro, porém, este havia sido inventado em 1931, pelos húngaros E. Egerváry e D. König. O método propõe a resolução de problemas de otimização combinatória como os de atribuição linear, a resolução é tempo polinomial, normalmente, $O(n^3)$. Diz-se que um algoritmo é de tempo polinomial se seu tempo de execução é limitado por uma expressão polinomial no tamanho da entrada para o algoritmo, isto é, $T(n) = O(n^k)$.

Esse nome teve origem em 1955 devido a H. W. Kuhn, pesquisador na área de programação linear, que em um de seus trabalhos (KUHN, 1955b), fez homenagem aos

descobridores do algoritmo em 1931, os húngaros E. Egerváry (KUHN, 1955a) e D. König, sendo que este último demonstrou um teorema combinatório em 1916 que serviu de base para o algoritmo (Teorema de König). O método Húngaro pode ser aplicado em diversos problemas práticos de alocação de tarefas desde que se construa, de forma conveniente, a matriz custo com as informações de que dispomos do problema.

Teorema de König 1 *Se o número mínimo de traços ou riscos que atravessam todos os zeros for n, temos uma alocação possível para cada linha ou coluna.*

Estes riscos ou traços são marcações em uma determina linha ou coluna. Esta marcação indica que esta coluna ou linha possui um ou mais zeros. O algoritmo tenta fazer essas marcações de modo a gastar a menor quantidade de riscos ou traços possível.

O algoritmo modela um problema de alocação como uma matriz de custo $n\ x\ m$, onde cada item representa o custo de atribuir a enésima tarefa ao m-ésimo recurso. Por padrão, o algoritmo realiza a minimização dos elementos da matriz; Portanto, se é um problema de minimizar custos, é suficiente começar a eliminar Gauss-Jordan para fazer zeros (pelo menos um zero por linha e por coluna). No entanto, no caso de um problema de maximização do lucro, o custo da matriz precisa ser modificado para que a minimização de seus elementos leve a uma maximização dos valores de custo originais.

2.1.7 Pseudocódigo Algoritmo Húngaro

O Algoritmo 1 representa um pseudocódigo para o algoritmo Húngaro.

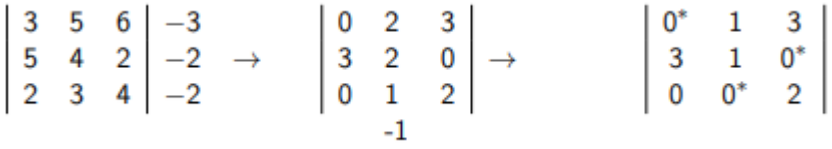
AlgoritimoHungaro($M[\ ]\ [\ ]$, linhas, colunas)
n ← linhas
while *!solução viável (número de riscos < n)* **do**
  Identifique o valor mínimo de cada linha e o subtraia de cada elemento da linha;
  Identifique o valor mínimo de cada coluna e o subtraia de cada elemento da coluna;
  Identifique o número mínimo de riscos que cubra todos os zeros da matriz;
  if *!solução viável (número de riscos < n)* **then**
    Identifique o o valor mínimo dos elementos não riscados;;
    subtraia desses mesmos elementos;;
    Para elementos cobertos por dois riscos, adicione esse valor;
  else
    Identifique a solução ótima na solução viável encontrada.;
  end
end
return *solução ótima*

Algoritmo 1: Algoritmo Húngaro

Pode concluir através do Teorema de König (KUHN, 1955b) que após feito o escalonamento de linhas e colunas, cobre-se com riscos ou traços todos zeros em determinada linha ou coluna. Se o número de traços ou riscos for igual a n contem uma solução viável ao problema. Caso o número de traços ou riscos seja diferente de n aplica-se a operação de viabilização e esta consiste em selecionar o menor número entre todos os números que não foram marcados com linha ou traço. Este número deve ser subtraído onde não existe linha ou traço marcando-os e adicionado onde há o encontro de duas linhas ou traços.

Na Figura 4 há uma exemplificação da execução do algoritmo Húngaro, onde não foi necessária a viabilização devido ao Teorema de König encontrar solução viável. Feito o escalonamento de linhas e colunas e identificação dos riscos ou traços para cobrir todos zeros em determinada linha ou coluna. Fica evidente a solução linha 1 coluna 1, linha 2 coluna 3 e linha 3 coluna 2.

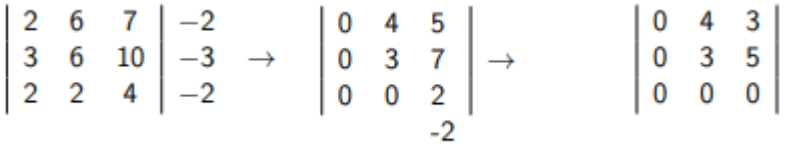
Figura 4 – Exemplo 1 da aplicação do algoritmo Húngaro



Fonte: (CARVALHO, 2018)

Na Figura 5 há uma exemplificação da execução do algoritmo Húngaro, onde é necessária a viabilização devido ao Teorema de König não encontrar solução viável. São feitas bonificações aos pesos sem riscos ou traços e penalidades aos pesos com dois riscos ou traços.

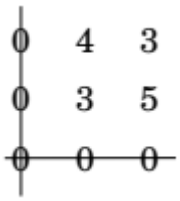
Figura 5 – Exemplo 2 da aplicação do algoritmo Húngaro



Fonte: (CARVALHO, 2018)

A Figura 6 é a demonstração da utilização do Teorema de König.

Figura 6 – Teorema de König



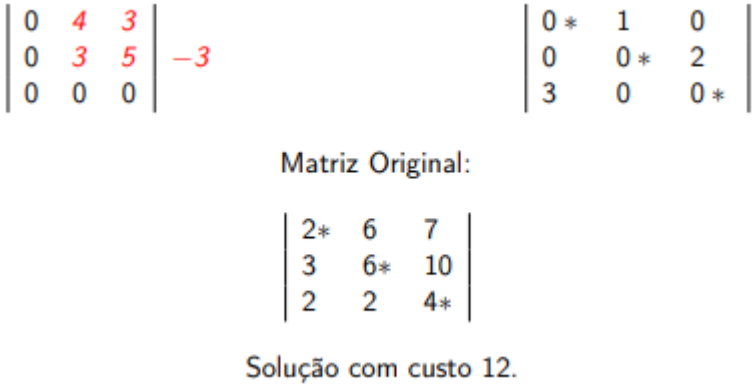
Fonte: (CARVALHO, 2018)

Nas Figura 7 e Figura 8 são expostas as execuções da operação de viabilização. Após identificar ue traços ou linhas foram diferentes de n seleciona-se o menor número entre todos os números que não foram marcados com risco ou traço. Este número deve ser subtraído onde não existe riscos ou traço marcando-os e adicionado onde há o encontro de duas linhas ou traços. Fica evidente a solução linha 1 coluna 1, linha 2 coluna s e linha 3 coluna 3.

Figura 7 – Operação de viabilização.

Fonte: (CARVALHO, 2018)

Figura 8 – Solução do algoritimo Húngaro



Fonte: (CARVALHO, 2018)

2.2 Trabalhos correlatos

Em Soares (2011) há um estudo detalhado sobre formas de resolução do problema de alocação de tarefas através da análise do problema, definição de abordagens na literatura e escolha de algoritmos para solução do problema, havendo implementação computacional. O estudo foi feito na Universidade Federal de São Paulo no Campus São José dos Campos havendo um problema bem semelhante ao encontrado no ICEA. Devido às afinidades pessoais e áreas de especialidade de cada profissional há um aumento na complexidade em atribuir qual professor irá lecionar determinada disciplina no semestre letivo. O trabalho faz estudo literário sobre Programação Linear, Programação Inteira e diversos algoritmos. Soares (2011) implementados computacionalmente os seguintes algoritmos: Algoritmo Guloso, Simulated Annealing, Branch and Bound e Força Bruta.

Rodrigues, Vieira e Agustini (2005) afirmam que: em nossa sociedade, é muito frequente depararmos com problemas que requerem tomadas de decisões visando a melhoria da relação custo-benefício por meio da maximização ou minimização de elementos do

problema. Esse tipo de problema forma uma classe especial de problemas de otimização, ou seja, problemas cuja solução consiste em maximizar ou minimizar uma função numérica de um determinado número de variáveis (ou funções), estando estas sujeitas a certas restrições. O trabalho faz uma abordagem do algoritmo Húngaro de um caso particular de problema de transporte em programação linear: o problema da alocação de tarefas. O artigo abordou o assunto de forma teórica fazendo a proposição de definições e teoremas, optou-se por uma forma expositiva de resolução de exemplos não havendo uma implementação computacional.

Segundo SANTOS et al. (2015) as pessoas se deparam cada vez com situações em que precisa-se tomar decisões a fim de obter melhores resultados na relação custo-benefício, como também obter ferramentas que facilitem estas escolhas. O objetivo do trabalho é criar um método que tem por finalidade a resolução de problemas de alocação de tarefas, o qual é um caso particular do problema de transporte.O trabalho demostra aplicações do algoritmo, através de exemplificações sobre a temática levantada, objetivando uma melhor compreensão de suas etapas e buscando aproximar os exemplos de problemas do cotidiano. Este método foi apresentado à Educação básica de ensino como um atrativo para despertar o envolvimento dos alunos da educação básica em situações problemas que envolvam a matemática de forma contextualizada. O trabalho mostra um interessante apêndice, o problema de alocação de tarefas é resolvido usando a função Solver do Excel através do método Simplex. Como foi descrito, SANTOS et al. (2015) optou por uma forma expositiva de resolução de exemplos e uma implementação computacional através da função Solver do Excel.

A Figura 9 é uma demonstração da implementação computacional através da função Solver do Excel foi feita no trabalho (SANTOS et al., 2015):

Em uma empresa deve alocar quatro operários A, B, C e D para as máquinas X, Y, Z e K. Queremos saber qual é o tempo ótimo para que cada operário seja efetivo em sua respectiva máquina e também saber quanto dura um ciclo de produção total.

- Cria-se uma tabela no Excel com os dados do problema;
- Faz-se uma nova tabela, igual à primeira, porém com seus valores nulos e no campo Total, utiliza-se a auto-soma tanto em linha como em coluna;
- Cria-se um item de oferta e demanda todos iguais a um, uma vez que cada operário irá para um único destino;
- Depois uma célula é criada com a função soma-produto, que no nosso exemplo será na célula H19 com produto das matrizes dos valores da primeira e segunda tabela que foi criada.

Figura 9 – Utilização da Ferramenta Solver - entrada dados planilha Excel

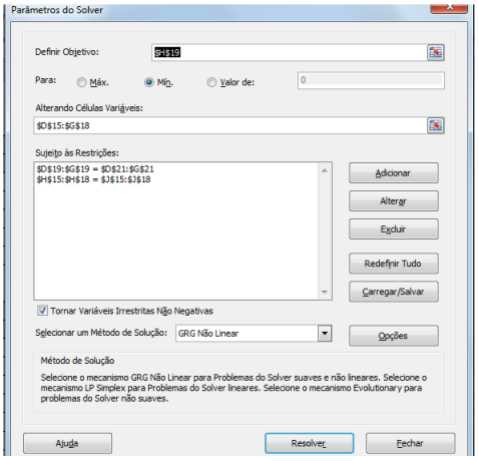
SEGUNDOS	MAQUINA				OFERTA
OPERARIOS	X	Y	Z	K	1
A	5	24	13	7	1
B	10	25	3	23	1
C	28	9	8	5	1
D	10	17	15	3	1
DEMANDA	1	1	1	1	

SEGUNDOS	MAQUINA				TOTAL	SINAL	OFERTA
OPERARIOS	X	Y	Z	K		=	1
A	1	0	0	0	1	=	1
B	0	0	1	0	1	=	1
C	0	1	0	0	1	=	1
D	0	0	0	1	1	=	1
TOTAL	1	1	1	1	20		
SINAL	=	=	=	=			
DEMANDA	1	1	1	1			

Fonte: (SANTOS et al., 2015)

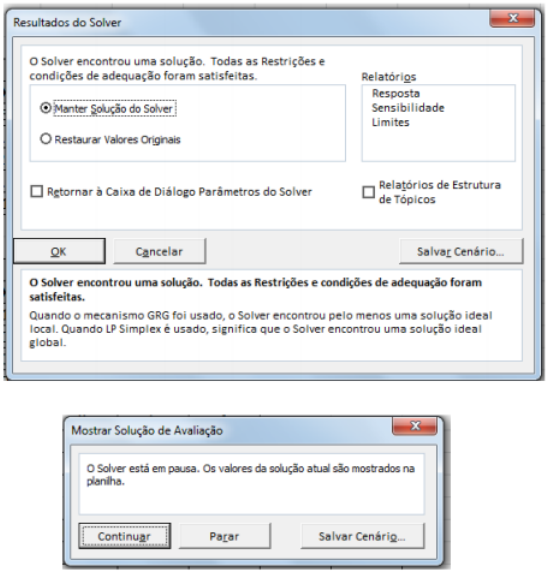
- Como demonstrado na Figura 10 será utilizada a função Solver, que faz parte dos complementos do Excel para colocar as restrições e comandos para o funcionamento do programa;
  - Como trata-se de uma minimização seleciona-se, o programa para opção minimização. Coloca-se também a célula da soma-produto H19 em definir objetivo e no campo células variáveis, as células da segunda tabela, ou seja, D15 a G18.
  - Adiciona as restrições, como o total na coluna igual à oferta e o total na linha igual à demanda.
- Após este passo, em Opções, seleciona-se o item Mostrar resultados de iterações e clica em Ok. A página retornará para a anterior onde clica em Resolver e, em seguida, Ok e em Continuar até chegar ao resultado final, conforme demonstrado na Figura 11;

Figura 10 – Utilização da Ferramenta Solver - definição das restrições e função objetivo



Fonte: (SANTOS et al., 2015)

Figura 11 – Utilização da Ferramenta Solver - cálculo



Fonte: (SANTOS et al., 2015)

De acordo com a Figura 12 obtém-se a seguinte solução com o operário A na máquina X, operário B na máquina Z, operário C na máquina Y e operário D na máquina K, com tempo mínimo de 20 segundos com esta distribuição, conforme figura abaixo:

Figura 12 – Utilização da Ferramenta Solver - resultado

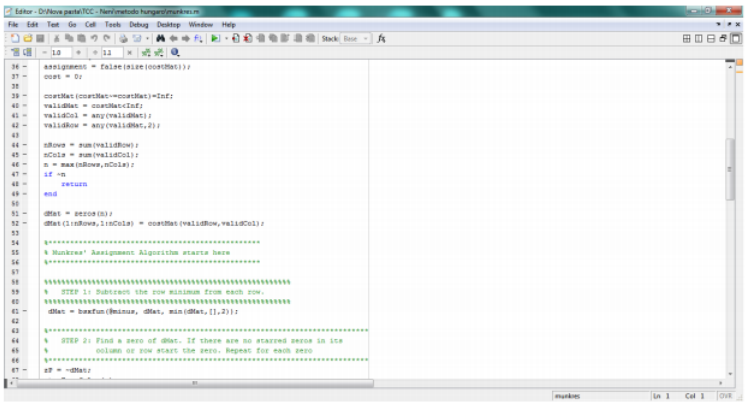
SEGUNDOS	MAQUINA				OFERTA
OPERARIOS	X	Y	Z	K	1
A	5	24	13	7	1
B	10	25	3	23	1
C	28	9	8	5	1
D	10	17	15	3	1
DEMANDA	1	1	1	1	

SEGUNDOS	MAQUINA				TOTAL	SINAL	OFERTA
OPERARIOS	X	Y	Z	K		=	1
A	1	0	0	0	1	=	1
B	0	0	1	0	1	=	1
C	0	1	0	0	1	=	1
D	0	0	0	1	1	=	1
TOTAL	1	1	1	1	20		
SINAL	=	=	=	=			
DEMANDA	1	1	1	1			

Fonte: (SANTOS et al., 2015)

No trabalho, foi utilizado o método Húngaro e o Matlab em problemas de alocação de tarefas, realizado por Brito (2015). O autor focou em abordar a parte teórica da alocação de tarefas voltada para a matemática. Possuindo uma extensa revisão bibliográfica e uma exposição de resolução de exemplos dentro do texto do trabalho. Foi feita uma pesquisa sobre a ferramenta Matlab e na implementação computacional foi utilizado código disponibilizado no site da *Mathworks*.[1] A Figura 13 é um exemplo de utilização desta ferramenta.

Figura 13 – Implementação Algoritmo Húngaro no Matlab



Fonte: (BRITO, 2015)

[1] Disponível em: https://www.mathworks.com/

3 Desenvolvimento

O Instituto de Ciências Exatas e Aplicadas (ICEA) é composto pelos seguintes departamentos: Departamento de Ciências Exatas e Aplicadas (DECEA), Departamento de Computação e Sistemas (DECSI), Departamento de Engenharia de Produção (DEENP), Departamento de Engenharia Elétrica (DEELT) e Possuindo o total de noventa e seis professores para ofertar cento e oitenta e três disciplinas obrigatórias em cada semestre. É tarefa de cada departamento do instituto alocar a um ou mais professores as disciplinas obrigatórias de sua grade curricular e ofertar certa quantidade de disciplinas eletivas. Cada professor, em média, se responsabilizará por duas ou três disciplinas.

Para o trabalho de conclusão de curso, foi proposto analisar DECSI que necessita de uma solução automatizada para o problema de alocação de encargos (disciplinas). A cada semestre, o professor responsável precisa fazer este trabalho manualmente, contando apenas com o auxílio de planilhas eletrônicas. Para o problema da atribuição de tarefas do DECSI foi necessário definir os dados a serem utilizados para calcular a compatibilidade entre recursos(professores) e tarefas(disciplinas), possibilitando a criação de uma matriz de custos. Para a montagem da matriz de custos foram definidos os seguintes requisitos: área de concurso, experiência docente na(s) disciplina(s) em questão e a preferência entre elas.

Para o DECSI os parâmetros foram definidos e recolhidos através de:

- Pesquisa de preferências (um arquivo com nome e as afinidades demonstradas por cada professor em relação a cada disciplina)
- Ter lecionado a matéria em algum momento.
- Matéria pertence à sua área de atuação.

Ao modelar a função de custo é necessário verificar a necessidade de adaptações. Esta "afinidade", que define a satisfação do docente ao lecionar cada disciplina será um bom indicador de custo, já que, ao maximizar este custo, em teoria obtemos uma associação que melhor agrada os professores em relação às designações.

Foi planejada a utilização do algoritmo Húngaro em conjunto aos dados conseguidos com o professor para gerar a matriz de custos. Foi feita a implementação do algoritmo e seus devidos testes para os dados. Esta implementação foi utilizada para o experimento de designação do semestre 2018/1 da Universidade.

3.1 Matriz de Custo

Para matriz de custo foi definida a entrada dos seguintes dados:

- Cada professor receberia um vetor de pesos para as disciplinas que tem preferência. Foi elaborado um questionário, onde cada professor deveria responder em ordem decrescente de afinidade sua preferência por cinco disciplinas, o que definiria os custos de acordo com a seguinte escala: 10, 7, 5, 2 e 1 de custo. Esta escala foi gerada aleatoriamente. Desta forma, os professores poderiam escolher as disciplinas mais coerentes, de acordo com sua área de atuação.
- Área do concurso que o professor fez ao entrar na faculdade: 5 de custo. O professor ao ingressar na faculdade é submetido à prova e testes de aptidão para determinadas áreas. É importante que a experimentação levasse em conta este item para evitar que professores fossem alocados para áreas diferentes a que estão aptos a lecionar.
- Histórico de disciplinas lecionadas na instituição (2 de custo). Dados históricos de cinco anos foram utilizados para analisar as disciplinas em que o professor possui experiência em lecionar.

3.1.1 Normalização

Os dados coletados e transformados na matriz de custo modelarão um problema de maximização, porém o algoritmo resolve problemas de minimização. Devido a esta característica, foi necessário fazer uma normalização na matriz de custo. A normalização foi feita através da multiplicação de cada elemento por -1. A matriz deve ser percorrida em busca do maior elemento e somando-se esse elemento selecionado aos demais elementos da matriz.

Existem disciplinas com carga horária de 60 horas (4 créditos) e de 30 horas (2 crédito). O modelo foi proposto para todas as disciplinas serem de 4 créditos. Disciplinas de 2 crédito como CSI 491 e CSI 427 são lecionadas em duas turmas, assim é possível unir duas delas e transformar em uma disciplina de 2 créditos. Porém, a disciplina CSI 201 é um caso onde essa transformação não é possível, uma vez que a referida disciplina é de um crédito e é lecionada em apenas uma turma. Por não atender aos requisitos exigidos para a transformação, esta teve de ser retirada do modelo.

3.1.2 Casos Especiais

Houveram casos especiais como professores que trocaram de universidade após o término do semestre e nesses casos professores substitutos são contratados. As disciplinas CSI433, CSI472, CSI476, CSI488, CSI506 e CSI548 precisaram ser retiradas do modelo,

devido ao fatos dos professores que as lecionaram não participaram da pesquisa feita para gerar a matriz de custo.

3.1.3 Resolvendo um Exemplo de Matriz de Custo

A Tabela 1 é um exemplo de matriz de custos 10×10. A ferramenta *Solve* [1], consegue resolver problemas com tamanho até 10.

Tabela 1 – Exemplo de matriz de custo Professor x Disciplina

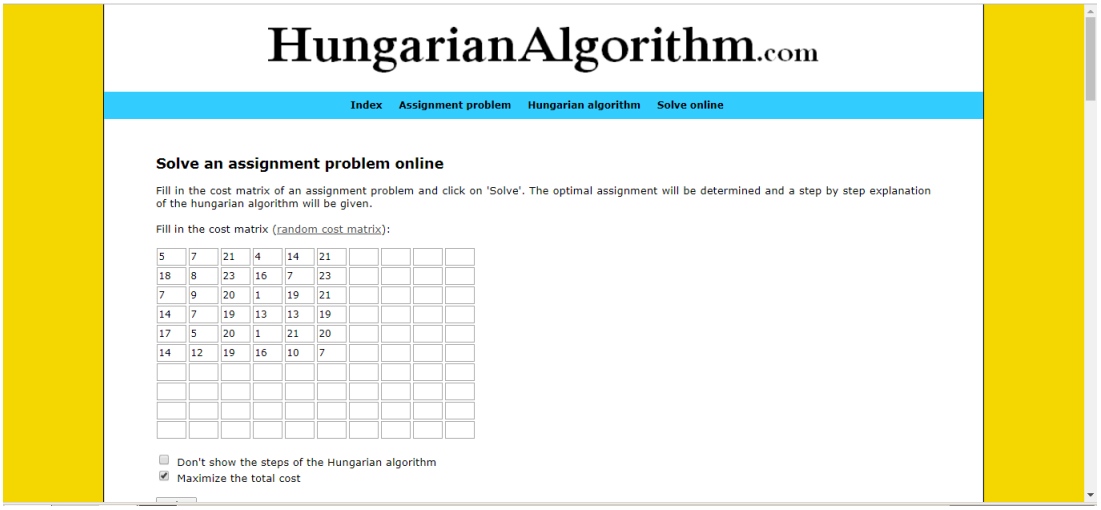
	Cod1	Cod2	Cod3	Cod4	Cod5	Cod6	Cod7	Cod8	v9	Cod10
Professor1	15	7	21	14	14	21	15	7	21	14
Professor2	18	8	23	16	7	23	18	8	23	16
Professor3	14	9	20	14	19	21	14	9	20	14
Professor4	14	7	19	13	13	19	14	7	19	13
Professor5	17	5	20	16	21	20	17	5	20	16
Professor6	0	0	0	0	0	0	0	0	0	0
Professor7	15	7	21	14	14	21	15	7	21	14
Professor8	18	8	23	16	7	23	18	8	23	16
Professor9	14	9	20	14	19	21	14	9	20	14
Professor10	14	7	19	13	13	19	14	7	19	13

Fonte: Elaborado pelo autor

A Figura 14 representa o início da resolução do exemplo da Tabela 1 na ferramenta *Solve*. A ferramenta é preenchida com a matriz de custos.

[1] Disponível em: http://www.hungarianalgorithm.com/solve.php

Figura 14 – Ferramenta Solve



Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 15, é a representação da Tabela 1 através da ferramenta *Solve*.

Figura 15 – Matriz de Custos Original

This is the original cost matrix:

5	7	21	4	14	21
18	8	23	16	7	23
7	9	20	1	19	21
14	7	19	13	13	19
17	5	20	1	21	20
14	12	19	16	10	7

Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 16, essa matriz passa pelo processo de transformação de um problema de máximização para minimização. Cada linha é multiplicada por -1 e a cada elemento é somado o valor do maior elemento da matriz.

Figura 16 – Normalização da Matriz

Negate all values

Because the objective is to maximize the total cost we negate all elements:

-5	-7	-21	-4	-14	-21
-18	-8	-23	-16	-7	-23
-7	-9	-20	-1	-19	-21
-14	-7	-19	-13	-13	-19
-17	-5	-20	-1	-21	-20
-14	-12	-19	-16	-10	-7

Make the matrix nonnegative

The cost matrix contains negative elements,
we add 23 to each entry to make the cost matrix nonnegative:

18	16	2	19	9	2
5	15	0	7	16	0
16	14	3	22	4	2
9	16	4	10	10	4
6	18	3	22	2	3
9	11	4	7	13	16

Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 17, essa matriz passa pelo processo de escalonamento de linhas e de colunas. Essse processo consiste em subtrair o menor elemento da linha ou coluna.

Figura 17 – Primeiro Escalonamento de Linhas e Colunas

Subtract row minima

We subtract the row minimum from each row:

16	14	0	17	7	0	(-2)
5	15	0	7	16	0	
14	12	1	20	2	0	(-2)
5	12	0	6	6	0	(-4)
4	16	1	20	0	1	(-2)
5	7	0	3	9	12	(-4)

Subtract column minima

We subtract the column minimum from each column:

12	7	0	14	7	0
1	8	0	4	16	0
10	5	1	17	2	0
1	5	0	3	6	0
0	9	1	17	0	1
1	0	0	0	9	12
(-4)	(-7)		(-3)		

Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 18, essa matriz passa por um teste utilizando Teorema de Köning para saber se a solução ótima foi encontrada. O teorema testa se a quantidade de riscos traçados é igual a quantidade de linhas da matriz.O riscos são traçados a fim de se cobrir todos os zeros possíveis. No caso demonstrado a solução ótima ainda não foi encontrada.

Figura 18 – Primeiro Teste do Teorema de Köning

Cover all zeros with a minimum number of lines

There are 4 lines required to cover all zeros:

12	7	**0**	14	7	**0**	
1	8	**0**	4	16	**0**	
10	5	**1**	17	2	**0**	
1	5	**0**	3	6	**0**	
0	**9**	**1**	**17**	**0**	**1**	**x**
1	**0**	**0**	**0**	**9**	**12**	**x**
		x			**x**	

Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 19, essa matriz passou pelo processo de viabilização, este consiste em criar novos zeros na matriz através de bonificações e penalidades. As bonificações são dadas aos elementos não cobertos pelas linhas traçadas e as penalidades são geradas a elemntos que posuam uma linha horizontal e vertical sobre eles. Dentre os elementos não cobertos pelas linhas selecionamos o menor para ser aquela a bonificar e penalizar. O elemento escolhido foi 1.

Figura 19 – Primeira Execução da Operação de Viabilização

Create additional zeros

The number of lines is smaller than 6. The smallest uncovered number is 1.
We subtract this number from all uncovered elements and add it to
all elements that are covered twice:

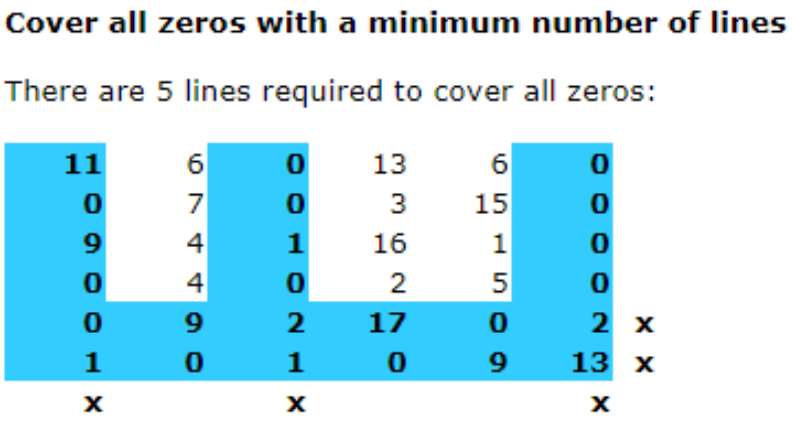
11	6	0	13	6	0
0	7	0	3	15	0
9	4	1	16	1	0
0	4	0	2	5	0
0	9	2	17	0	2
1	0	1	0	9	13

Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 20, essa matriz passa por um teste utilizando

Teorema de Köning para saber se a solução ótima foi encontrada. No caso demonstrado a solução ótima ainda não foi encontrada.

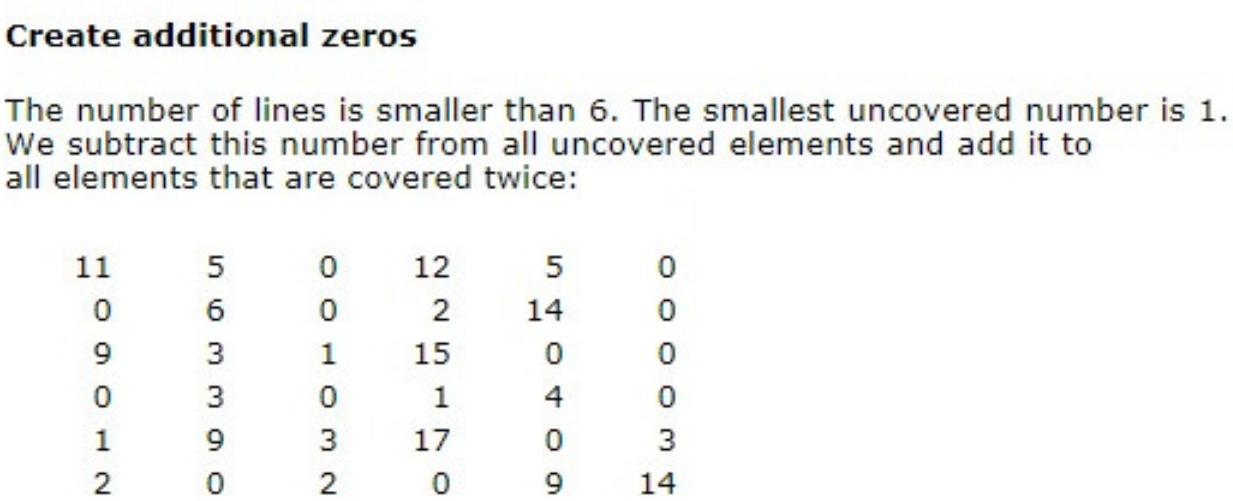
Figura 20 – Segundo Teste do Teorema de Köning



Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 21, essa matriz passou pelo processo de viabilização. O elemento escolhido foi 1.

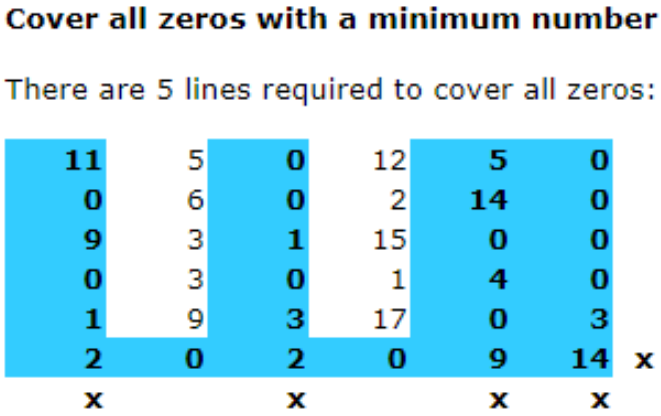
Figura 21 – Segunda Execução da Operação de Viabilização



Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 22, essa matriz passa por um teste utilizando Teorema de Köning para saber se a solução ótima foi encontrada. No caso demonstrado a solução ótima ainda não foi encontrada.

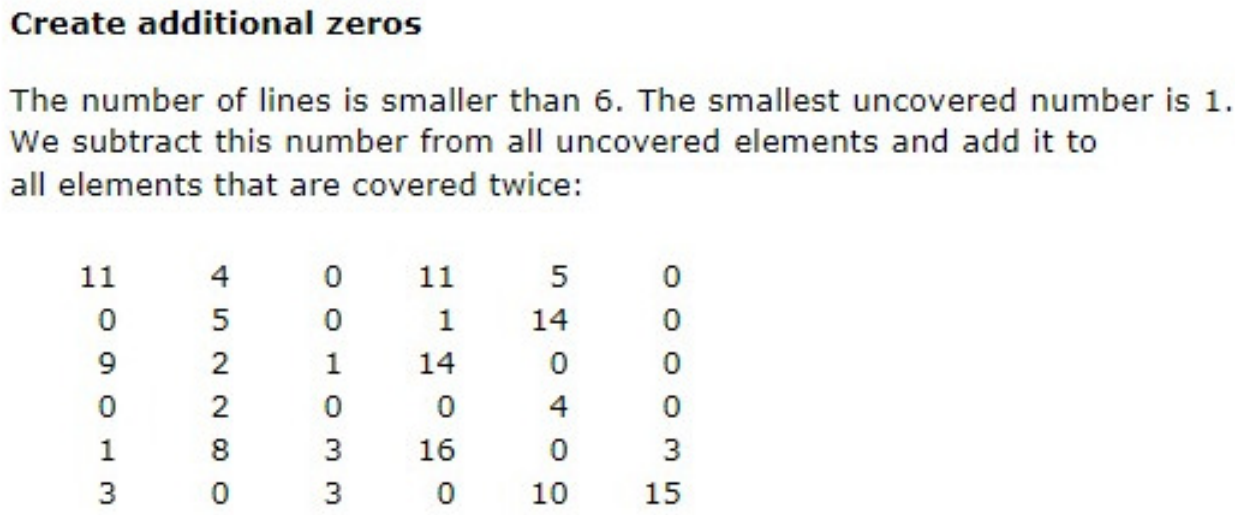
Figura 22 – Terceiro Teste do Teorema de Köning



Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 23, essa matriz passou pelo processo de viabilização. O elemento escolhido foi 1.

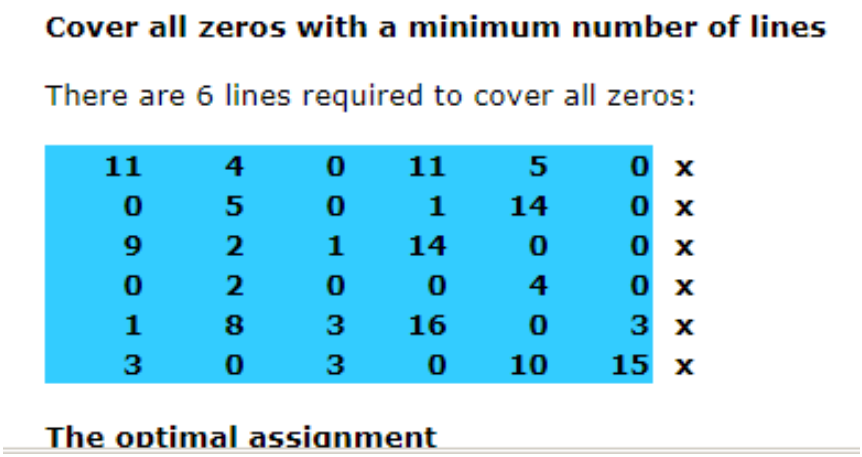
Figura 23 – Terceira Execução da Operação de Viabilização



Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 24, essa matriz passa por um teste utilizando Teorema de Köning para saber se a solução ótima foi encontrada. No caso demonstrado a solução ótima foi encontrada.

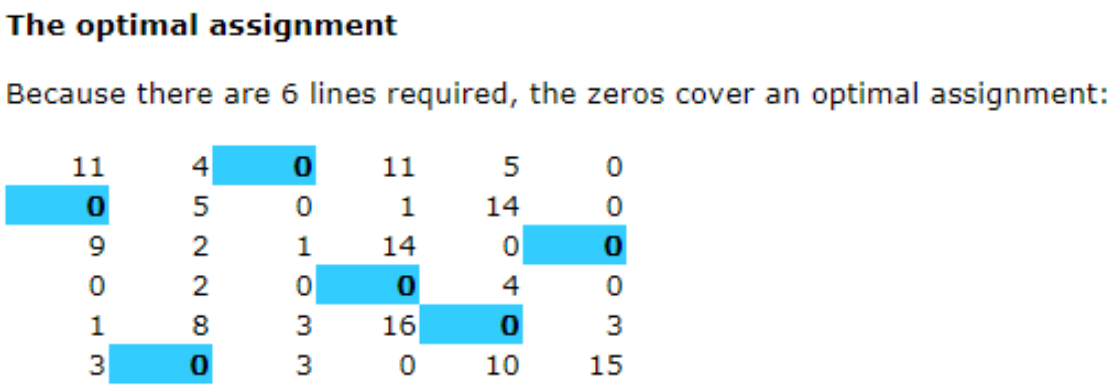
Figura 24 – Quarto Teste do Teorema de Köning



Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 25, essa matriz exibe as posições selecionadas que representam a solução ótima encontrada. No caso a matriz ainda esta em sua forma de minimização.

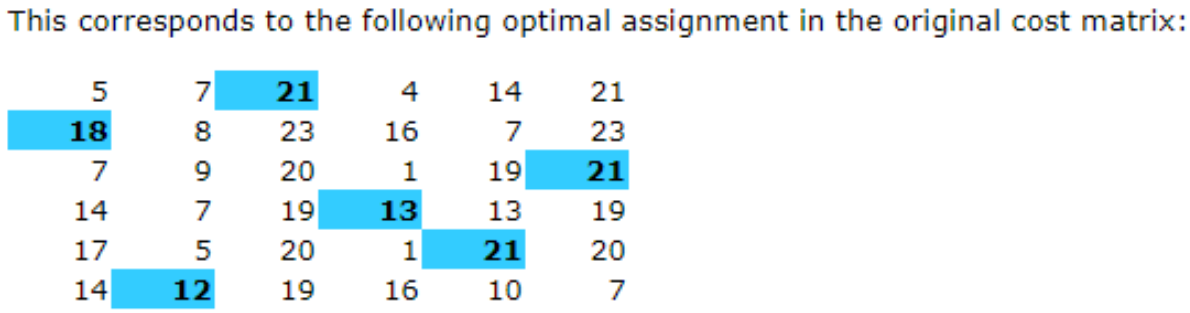
Figura 25 – Solução Matriz de Minimização



Fonte: Elaborado pelo autor

A matriz de custos, exibida na Figura 26, é exebida a matriz original e as posições selecionadas que representam a solução ótima encontrada.

Figura 26 – Solução Matriz de Mazimização



Fonte: Elaborado pelo autor

Através da saída final gerada pela ferramenta *Solve*, foi montada a Tabela 2 e esta representa a solução para o exemplo contido na Tabela 1.

Tabela 2 – Resultado do Exemplo de matriz de custo Professor x Disciplina

	Cod
Professor 1	3
Professor 2	1
Professor 3	6
Professor 4	4
Professor 5	5
Professor 6	2

Fonte: Elaborado pelo autor

3.2 Algoritmo Húngaro

Segundo o Teorema de König descrito em Kuhn (1955b), se o número mínimo de traços ou riscos que atravessam todos os zeros for exatamente n, ao possuir um número n de tarefas, será possível atribuir um recurso a cada tarefa de modo único. Um traço ou risco é criado para cobrir todos zeros de uma linha ou coluna. Há uma alocação possível para cada linha ou coluna da matriz de custos e uma solução ótima. Esta conterá a melhor solução possível para o problema. Caso contrário, existirão linhas ou colunas que não possuirão elementos zero, impedindo que haja uma atribuição ótima.

Para a operação de viabilização, identificamos o valor do menor elemento não riscado ou traçado e o subtraímos em todos os elementos não riscados ou que não estejam traçados. Para elementos riscados ou que estejam traçados duas vezes, adicionamos esse mesmo valor.(CARVALHO, 2018).

Segue algoritmo Húngaro:

1. Os Algoritmos 2, 3, 4, 5, 6 e 7 representam funções auxiliares;

```
MenordaLinha(M[ ], n)
menor ← 9999;
for coluna = 0 → n do
    if (menor > M[ coluna ]) then
        menor ← M[ coluna ] ;
    end
end
return menor
```

Algoritmo 2: Encontra o Menor da Linha

```
SubtraidaLinha(M[ ], n)
sub ← MenordaLinha(M[ ], n);
for coluna = 0 → n do
    if (menor > M[ coluna ]) then
        M[ coluna ] − sub;
    end
end
```

Algoritmo 3: Subtrai o Menor da Linha

```
MenordaColuna(M[ ], n, coluna)
menor ← 9999;
for linha = 0 → n do
    if (menor > M[ linha ] [ coluna ]) then
        menor ← M[ linha ] [ coluna ];
    end
end
return menor
```

Algoritmo 4: Econtra o Menor da Coluna

```
SubtraidaColuna(M[ ], n, coluna)
sub ← MenordaColuna(M[ ], n, coluna) ;
for linha = 0 → n do
    if (menor > M[ linha ] [ coluna ]) then
        M[ linha ] [ coluna ] ← M[ linha ] [ coluna ] − sub;
    end
end
```

Algoritmo 5: Subtrai o Menor da Coluna

```
QTD0(M[ ] , n)
Qtd0 ← 0;
for linha = 0 → n do
    if (0 == M[ linha ] ) then
        Qtd0 ← Qtd0 + 1;
    end
end
return Qtd0
```

Algoritmo 6: Calculo da quantidade de 0 da Linha

```
Auxiliares(Array , M[ ] [ ] , n, zeros[ ], Array2,NaoAlocado)
for indice = 0 → n do
    Array.Insere(indice);
    Array2.Insere(indice);
    NaoAlocado.Insere(indice);
    zeros[ indice ] ← QTD0(M[ indice ] , n);
end
```

Algoritmo 7: Gera o Aray de Tarefas Livres, o Vetor com Quantidade 0 de Cada Linha e Indica que a Linha Não Foi Alocada

2. O Algoritmo 8 representa a normalização da matriz de custos;

```
Normalização(M[ ] [ ], n)
menor ← 9999;
for linha = 0 → n do
    for coluna = 0 → n do
        if (menor > M[ linha ] [ coluna ]) then
            menor ← M[ linha ] [ coluna ];
        end
        M[ linha ] [ coluna ] ← M[ linha ] [ coluna ] ∗ − 1 ;
    end
end
for linha = 0 → n do
    for coluna = 0 → n do
        M[ linha ] [ coluna ] ← M[ linha ] [ coluna ] + menor ;
    end
end
```

Algoritmo 8: Normalização

3. O algoritmo 9 faz o escalonamento de linha. Para todas as linhas, encontrar menor elemento da linha e subtrair de todo elemento dela

```
Escalonalinha(M[ ] [ ], n)
for linha = 0 → n do
    SubtraidaLinha(M[ linha ], n);
end
```

Algoritmo 9: Escalonamento de linha

4. O Algoritmo 10 faz o escalonamento de coluna. Para todas as colunas, Encontrar menor elemento da coluna e subtrair de todo elemento dela

```
Escalonacoluna(M[ ] [ ], n, coluna)
for coluna = 0 → n do
    SubtraidaColuna(M[ ], n, coluna);
end
```

Algoritmo 10: Escalonamento de coluna

5. O Algoritmo 11 gera todas as alocações de disciplinas a professores que sejam possíveis. Para cada linha onde a coluna que possua 0 e esta coluna não foi feita

uma atribuição faça esta atribuição;

```
Atribuição(M[ ] [ ], n ,MC[ ] [ ] , Alocacoes[ ] [ ], NaoAlocado)
Auxiliares(LinhaLivre , M[ ] [ ] , n, zeros[ ], ColunaLivre, NaoAlocado) ;
while (LinhaLivre.Size ( ) > 0) do
    menor ← 9999;
    qtdalocacoes ← 0;
    for indice = 0 → LinhaLivre.Size() do
        if ( zeros[ LinhaLivre.Get(indice) ] < menor ) then
            menor ← zeros[ LinhaLivre.Get(indice) ];
            indicemenor ← LinhaLivre.Get(indice) ];
        end
        LinhaLivre.Remove(indicemenor);
        for coluna = 0 → n do
            if ( M[ indicemenor ] [ coluna ] == 0 ) then
                if ( ColunaLivre.Contains(coluna) ) then
                    ColunaLivre.Remove(coluna);
                    Alocacoes[ indicemenor ] [ 0 ] ← coluna ;
                    Alocacoes[ indicemenor ] [ 1 ] ← MC[ indicemenor ] [
                     coluna ] ;
                    qtdalocacoes ← qtdalocacoes + 1 ;
                    NaoAlocado.Remove(indicemenor);
                end
            end
        end
    end
end
return qtdalocacoes
```

Algoritmo 11: Gerando as atribuições possíveis

6. O Algoritmo 12 representa o Teorema de Köning. Trace ou risque os zeros, de forma a gerar menor quantidade de riscos ou traços:

   a) Risque ou trace todas as linhas da matriz sem atribuições;

   b) Risque ou trace todas as colunas (não riscadas ou traçadas) com zeros nas linhas riscadas ou traçadas no passo anterior;

   c) Risque ou trace todas as linhas com atribuições em colunas do passo anterior;

   d) Faça o mesmo processo para as demais linhas sem atribuições;

   e) Inverta as linhas da matriz não traçadas ou não riscadas para linhas riscadas ou traçadas e vice-versa.

```
Teorema de Köning(M[ ] [ ], n, qtdalocacoes, LinhaNaoAlocado,
 ColunaNaoAlocado)
if ( qtdalocacoes != n ) then
    for linha = 0 → LinhaNaoAlocado.Size() do
        for coluna = 0 → n do
            if ( M[ LinhaNaoAlocado.Get(linha) ] [ coluna ] == 0 ) then
                if ( !ColunaNaoAlocado.Contains(coluna) ) then
                    ColunaNaoAlocado.Insere(coluna);
                end
            end
        end
    end
    for linha = 0 → n do
        for indice = 0 → ColunaNaoAlocado.Size() do
            if ( linha != LinhaNaoAlocado.Get(indice) ) then
                if ( ColunaNaoAlocado.Contains( linha[ ] [ 0 ] ) ) then
                    LinhaNaoAlocado.Insere(linha);
                end
            end
        end
    end
end
```

Algoritmo 12: Teorema de Koning

7. O Algoritmo 13 demonstra a operação de viabilização. Se a quantidade de riscos ou traços for igual a quantidade n da entrada, o resultado encontrado foi ótimo. Senão, ainda existe uma solução ótima, siga os passos abaixo:

   a) Entre todos os custos riscados ou traçados, verifique o menor;

   b) Subtraia o menor encontrado acima de todos os custos riscados ou traçados;

   c) Faça a adição do menor encontrado acima a todos os custos com dois riscos ou traçoss;

   d) Volte ao passo número 2.

```
Viabilização(M[ ] [ ], n, qtdalocacoes, LinhaNaoAlocado, ColunaNaoAlocado)
if ( qtdalocacoes != n ) then
    menor ← 9999;
    for linha = 0 → n do
        for coluna = 0 → n do
            if ( !ColunaNaoAlocado.Contains(coluna) &&
             !LinhaNaoAlocado.Contains(linha) ) then
                if ( menor > M[ linha ] [ coluna ]) then
                    menor ← M[linha ] [ coluna ] ;
                end
            end
        end
    end
    for linha = 0 → n do
        for coluna = 0 → n do
            if ( ColunaNaoAlocado.Contains(coluna) &&
             LinhaNaoAlocado.Contains(linha) ) then
                M[linha ] [ coluna ] ← M[linha ] [ coluna ] + menor ;
            end
            if ( !ColunaNaoAlocado.Contains(coluna) &&
             !LinhaNaoAlocado.Contains(linha) ) then
                M[linha ] [ coluna ] ← M[linha ] [ coluna ] − menor ;
            end
        end
    end
end
Recomece o algoritmo usando essa nova matriz;
```

Algoritmo 13: Viabilização

4 Resultados

O presente capítulo irá abordar como o experimento foi conduzido e a validação da implementação do algoritmo. Foi utilizado algoritmo Húngaro para o problema de alocação de disciplinas DECSI. Após a implementação desse algoirtmo foi necessário verificar os resultados obtidos com o método desenvolvido. O experimento visa analisar se o resultado final alcançado cumpre com o objetivo de gerar uma solução viável e automatizada que seja melhor que a solução manual, definindo assim uma proporção de quanto esta solução gerada pelo algoritmo é melhor. A Seção 4.1 aborda o ambiente em que os experimentos foram feitos. A Seção 4.2 aborda a comparação feita entre alocação realizada de forma manual e a alocação gerada pela implementação Java e sobre o resultado desta comparação.

4.1 Ambiente Computacional

A máquina utilizada para o desenvolvimento desse trabalho foi um Desktop e este possui processador Intel Pentium Dual Core CPU E2180, @2.0GHz. Sua RAM possui 2 GB. O sistema operacional utilizado é o Microsoft Windows 10 Home. O Ambiente de Desenvolvimento Integrado (IDE) utilizado para implementação dos algoritmos e realização dos testes foi o Eclipse Oxygen.3a Release (4.7.3a). O algoritmo foi implementados na linguagem de programação Java. o tempo que a implementação apresentada demora parar gerar o resultado é de 107 milissegundos.

4.2 Comparativo entre solução manual e metodologia proposta

Foi projetada uma forma de utilização da matriz de custos através do algoritmo Húngaro em Java.

Cada professor possui uma carga horária semanal que tem disponível para lecionar as disciplinas. Por sua vez, cada disciplina tem uma carga horária. Os dados do DECSI possuem essas definições das cargas horárias. A carga horária dos professores define a quantidade de matérias que cada professor pode lecionar (quantidade de entradas do mesmo professor). Portanto, quando a quantidade de professores é igual à quantidade de matérias, o problema é quadrático $n \times n$, sendo possível executar o algoritmo em busca de um emparelhamento. O Peso da matriz de custo é calculado com a seguinte fórmula: Custo = Preferência (10 ou 7 ou 5 ou 2 ou 1) + Área de concurso (5) + Ter lecionado a disciplina (2). O problema estudado é um cálculo de custo máximo, porém o algoritmo Húngaro utiliza a matriz de custo para cálculo de mínimo. Foi necessário, portanto transformar a matriz para o cálculo de custo mínimo. O encarregado pela alocação deverá gerar e utilizar

o experimento como ponto de partida em busca de novas soluções a serem utilizadas pelo DECSI.

O problema de alocação finalmente, irá obter a solução utilizando algoritmo Húngaro, esta será avaliada focando principalmente em critérios de qualidade de solução através de uma comparação com solução manual gerada pelo departamento.

4.2.1 Comparativo entre solução manual e metodologia proposta

As Tabelas 3 e 4 representam todas as alocações feitas de forma manual e todas as alocações feitas de forma automatizada e otimizada pela implementação Java para o semestre letivo 2018/1 para o DECSI. As alocações são compostas por um professor(carga horária) e código da disciplina (carga horária) e o peso atribuído na matriz de custos. Devido a carga horária de cada pofessor variar, ele pode aparecer mais ou menos vezes que outros professores. As linhas são compostas pela alocação manual e a alocação da implementação Java. As linhas com o nome formatado em negrito possuem sugestões de trocas pela implementação Java.

Tabela 3 – Alocação Manual X Alocação Implementação Java Parte 1

Professor	Disciplina Manual	Peso Manual	Disciplina Algoritmo	Peso Algoritmo
GEORGE HENRIQUE GODIM	CSI436	12	CSI436	12
GEORGE HENRIQUE GODIM	CSI466	12	CSI466	12
ALEXANDRE MAGNO DE SOUSA	CSI428	7	CSI428	7
ALEXANDRE MAGNO DE SOUSA	CSI428	7	CSI428	7
ALEXANDRE MAGNO DE SOUSA	CSI693	17	CSI693	17
ALVARO ANTONIO FONSECA	CSI030	17	CSI030	17
ALVARO ANTONIO FONSECA	CSI148	14	CSI030	17
ALVARO ANTONIO FONSECA	CSI148	14	CSI030	17
BRUNO CERQUEIRA HOTT	CSI443	7	CSI428	9
BRUNO CERQUEIRA HOTT	CSI443	7	CSI428	9
BRUNO CERQUEIRA HOTT	CSI428	9	CSI428	9
BRUNO CERQUEIRA HOTT	CSI463	12	CSI463	12
BRUNO RABELLO MONTEIRO	CSI440	17	CSI440	17
BRUNO RABELLO MONTEIRO	CSI440	17	CSI440	17
BRUNO RABELLO MONTEIRO	CSI442	12	CSI442	12
CAMILO DE LELES GARCA	CSI419	17	CSI419	17
CAMILO DE LELES GARCA	CSI419	17	CSI419	17
CAMILO DE LELES GARCA	CSI575	9	CSI575	9
DARLAN NUNES DE BRITO	CSI009	0	CSI437	7
DARLAN NUNES DE BRITO	CSI145	0	CSI437	7
DARLAN NUNES DE BRITO	CSI491	0	CSI514	0
DARLAN NUNES DE BRITO	CSI509	12	CSI509	12
DARLAN NUNES DE BRITO	CSI509	12	CSI509	12
DIEGO ZUQUIM GUIMARAES	CSI450	17	CSI450	17
DIEGO ZUQUIM GUIMARAES	CSI450	17	CSI450	17
DIEGO ZUQUIM GUIMARAES	CSI735	5	CSI735	5
GILDA APARECIDA DE ASSIS	CSI429	14	CSI429	14
GILDA APARECIDA DE ASSIS	CSI429	14	CSI429	14
GILDA APARECIDA DE ASSIS	CSI508	12	CSI508	12
EULER HORTA MARINHO	CSI485	17	CSI485	17
EULER HORTA MARINHO	CSI485	17	CSI485	17
EULER HORTA MARINHO	CSI486	14	CSI486	14

Fonte: Elaborado pelo autor

Tabela 4 – Alocação Manual X Alocação Implementação Java Parte 2

Professor	Disciplina Manual	Peso Manual	Disciplina Algoritmo	Peso Algoritmo
ELTON MAXIMO CARDOSO	CSI428	2	CSI546	14
ELTON MAXIMO CARDOSO	CSI437	0	CSI515	7
ELTON MAXIMO CARDOSO	CSI437	0	CSI443	7
ELTON MAXIMO CARDOSO	CSI428	2	CSI443	7
FERNANDO BERNARDES	CSI030	7	CSI427	4
FERNANDO BERNARDES DE	CSI477	14	CSI477	14
FERNANDO BERNARDES	CSI557	12	CSI557	12
HARLEI MIGUEL DE ARRUDA	CSI030	4	CSI148	7
HARLEI MIGUEL DE ARRUDA	CSI203	8	CSI148	7
HARLEI MIGUEL DE ARRUDA	CSI501	9	CSI501	9
MARIA GABRIELA DE CASSIA	CSI427	0	CSI460	17
MARIA GABRIELA DE CASSIA	CSI462	12	CSI462	12
MARIA GABRIELA DE CASSIA	CSI514	0	CSI009	2
MATEUS FERREIRA SATLER	CSI030	9	CSI030	9
MATEUS FERREIRA SATLER	CSI030	9	CSI457	17
MATEUS FERREIRA SATLER	CSI546	3	CSI457	17
MARLON PAOLO LIMA	CSI426	0	CSI426	0
MARLON PAOLO LIMA	CSI510	7	CS510	0
LUCINEIA SOUZA MAIA	CSI439	17	CSI439	17
LUCINEIA SOUZA MAIA	CSI460	14	CSI145	10
LUCINEIA SOUZA MAIA	CSI729	1	CSI729	1
RAFAEL FREDERICO ALEXANDRE	CSI030	12	CSI030	12
RAFAEL FREDERICO ALEXANDRE	CSI032	9	CSI032	9
RAFAEL FREDERICO ALEXANDRE	CSI466	0	CSI030	12
SAMIRA SANTOS DA SILVA	CSI203	14	CSI203	14
SAMIRA SANTOS DA SILVA	CSI203	14	CSI203	14
SAMIRA SANTOS DA SILVA	CSI515	0	CSI203	14
TALLES HENRIQUE	CSI457	14	CSI491	2
TALLES HENRIQUE	CSI457	14	CSI466	5
TALLES HENRIQUE	CSI733	12	CSI733	12
TATIANA ALVES COSTA	CSI032	12	CSI032	12
TATIANA ALVES COSTA	CSI567	7	CSI567	7

Fonte: Elaborado pelo autor

As Tabelas 3 e 4 representam todas as alocações feitas de forma manual e todas as alocações feitas de forma automatizada e otimizada pela implementação Java para o semestre letivo 2018/1 para o DECSI. O valor da função objetivo referente a resolução gerada de forma manual foi de 618. O valor da função objetivo referente a resolução gerada pela implementação Java foi de 707. A solução obtida pelo algoritmo implementado neste trabalho possui uma qualidade 14% melhor do que a alocação realizada manualmente para os dados DECSI 2018/1.

5 Conclusão

O trabalho utilizou o método Húngaro para solução do problema de alocações de diciplinas para o DECSI 2018/1. Foram coletados dados de preferência com os professores e histórico das disciplinas lecionadas por estes. O experimento visou automatizar e otimizar a alocação de disciplinas. O método proposto encontrou solução viável, sendo possível sua utilização no DECSI. O algoritmo é um ótimo ponto de partida na análise, compreensão, entendimento e discussão do problema de alocação. A resolução sugerida pela aplicação Java é 14% mais otimizada que a alocação gerada manualmente para os dados do experimento .

5.1 Trabalhos Futuros

Para abordar a necessidade de um novo requisito, como um mesmo professor lecionar a mesma disciplina em cursos diferentes, será necessário implementar algoritmos heurísticos para busca local (RODRIGUES et al., 2000). É recomendável a implementação de leitura de arquivo no Aplicativo Android. Recomenda-se, também, posterior implementação nos demais departamentos do ICEA, utilização de um banco de dados histórico das alocações Professor/Disciplina, a possibilidade do ajuste dos pesos e o tratamento de casos onde não há informação necessária para alocação (professores ou disciplinas novas, etc).

Referências

ARENALES, M. et al. *Pesquisa operacional: para cursos de engenharia*. [S.l.]: Elsevier Brasil, 2017. Citado na página 20.

BRITO, S. J. Método húngaro e aplicações. Instituto Federal de Educação, Ciência e Tecnologia de São Paulo , SP, 2015. Citado na página 32.

CARVALHO, M. A. M. de. *Aula 11 - BCC204 - Teoria dos Grafos* . 2018. <http://www.decom.ufop.br/marco/site_media/uploads/bcc204/11_aula_11.pdf>. [Acesso em: 10 de Dezembro de 2018]. Citado 3 vezes nas páginas 27, 28 e 43.

CELA, E. *The quadratic assignment problem: theory and algorithms*. [S.l.]: Springer Science & Business Media, 2013. v. 1. Citado 2 vezes nas páginas 16 e 18.

DASGUPTA, D. et al. A comparison of multiobjective evolutionary algorithms with informed initialization and kuhn-munkres algorithm for the sailor assignment problem. In: *Proceedings of the 10th Annual Conference Companion on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2008. (GECCO '08), p. 2129–2134. ISBN 978-1-60558-131-6. Disponível em: <http://doi.acm.org/10.1145/1388969.1389035>. Citado 2 vezes nas páginas 18 e 22.

FURTADO, J. C.; LORENA, L. A. N. Otimização em problemas de leiaute. In: *XX Congresso Nacional de Matemática Aplicada e Computacional e 2ª Oficina Nacional de PCE*. [S.l.: s.n.], 1997. p. 129–146. Citado na página 20.

GAREY, M.; JOHNSON, D. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, 1979. (Books in mathematical series). ISBN 9780716710448. Disponível em: <https://books.google.com.br/books?id=fjxGAQAAIAAJ>. Citado na página 18.

GOLDBARG, M. C.; LUNA, H. P. L. *Otimização combinatória e programação linear: modelos e algoritmos*. [S.l.]: Elsevier, 2005. Citado na página 21.

KHOO, K.-Y.; CONG, J. A fast multilayer general area router for mcm designs. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, IEEE, v. 39, n. 11, p. 841–851, 1992. Citado na página 20.

KUHN, H. On combinatorial properties of matrices. *George Washington University Logistics Papers*, v. 11, p. 1–11, 1955. Citado na página 25.

KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, Wiley Online Library, v. 2, n. 1-2, p. 83–97, 1955. Citado 4 vezes nas páginas 18, 24, 26 e 43.

RODRIGUES, L. B.; VIEIRA, F. B. P.; AGUSTINI, E. O método húngaro de otimização para o problema da alocação de tarefas. *FAMAT em Revista, Uberlândia*, n. 4, 2005. Citado 2 vezes nas páginas 23 e 28.

RODRIGUES, M. A. P. et al. Problema do caixeiro viajante: Um algoritmo para resoluçao de problemas de grande porte baseado em busca local dirigida. Florianópolis, SC, 2000. Citado na página 55.

SANTOS, C. E. S. d. et al. Utilizando o método húngaro e o matlab em problemas de alocação de tarefas. Universidade Federal Rural de Pernambuco, 2015. Citado 4 vezes nas páginas 29, 30, 31 e 32.

SHEN, M.; TZENG, G.-H.; LIU, D.-R. Multi-criteria task assignment in workflow management systems. In: IEEE. *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*. [S.l.], 2003. p. 9–pp. Citado na página 20.

SOARES, H. C. de A. Um estudo sobre o problema de alocação. *UNIFESP - Universidade Federal de São Paulo*, 2011. Citado 5 vezes nas páginas 18, 20, 22, 23 e 28.

WALSER, J. P. *Integer optimization by local search: a domain-independent approach*. [S.l.]: Springer-Verlag, 1999. Citado na página 22.

Apêndices

APÊNDICE A – Aplicação Android Desenvolvida

Apresentação da implementação feita no Android Studio, ambiente de desenvolvimento integrado para a plataforma Android. O ambiente Android Studio foi anunciado em 16 de Maio de 2013 na conferência Google I/O e é disponibilizado gratuitamente sob a Licença Apache 2.0. Versão Android utilizada foi a 4.0.

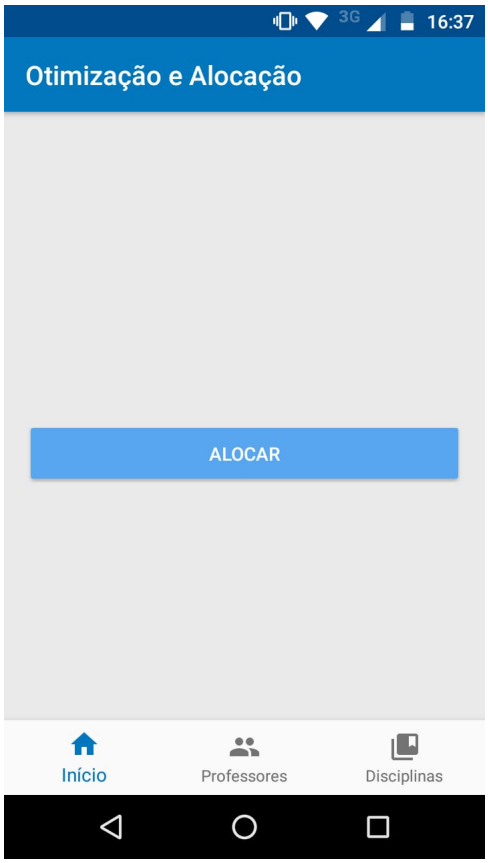
O aplicativo pode alternar entre as telas e elas possuem os seguintes recursos:

- Inicial: gerar resultado;
- Alocar: cálculo do algoritmo Húngaro e resultado;
- Disciplinas(tarefas): CRUD disciplinas;
- Professores(recursos): tarefas e CRUD professores;
- Tarefas: adicionar, exibir e excluir.

CRUD (acrónimo do inglês Create, Read, Update and Delete) são as quatro operações básicas (criação, consulta, atualização e destruição de dados).

Na Figura 27, tem-se a exibição da primeira tela visualizada pelo usuário após iniciar o aplicativo. O recurso oferecido pelo botão alocar precisa que a matriz de custos seja quadrada. Para isso, é feita uma verificação da soma da quantidade das matérias e a soma da quantidade de professores, cujos valores deverão ser iguais.

Figura 27 – Tela Inical

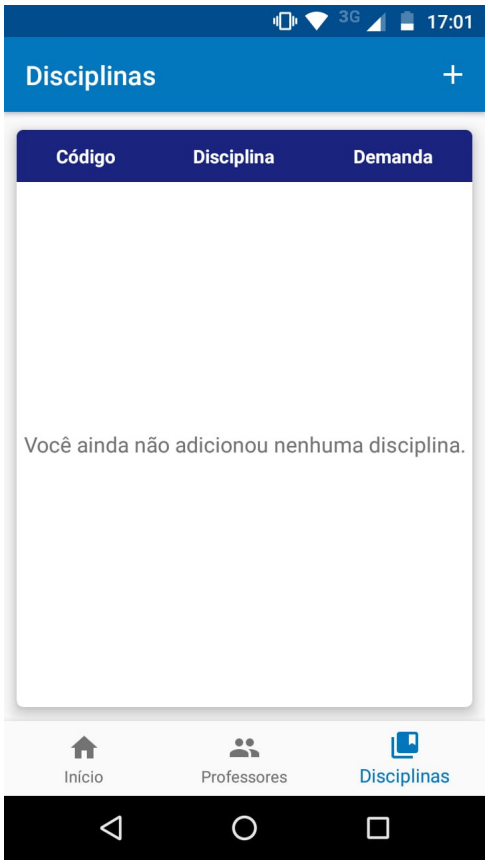


Fonte: Elaborado pelo autor

Na Figura 28, é exibida a tela onde os recursos pertinentes às tarefas(disciplinas) estão disponíveis. Estes recursos são:

- Cadastrar: botão mais;
- Editar: clicando na linha da disciplina desejada.

Figura 28 – Tela das Tarefas(Disciplinas)

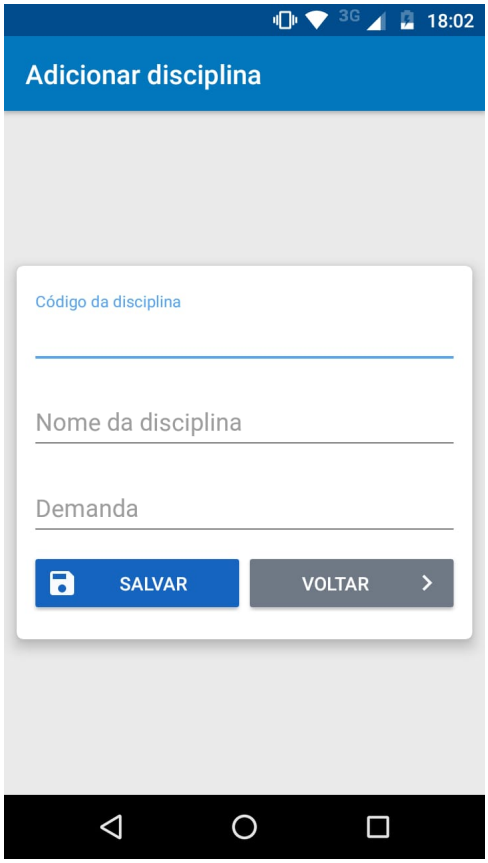


Fonte: Elaborado pelo autor

No exemplo a seguir, foi feito o cadastro de cinco disciplinas. Botão + no canto superior direito.

A Figura 29 é utilizada para o cadastro de cinco disciplinas como exemplo de utilização do recurso de cadastro de tarefas.

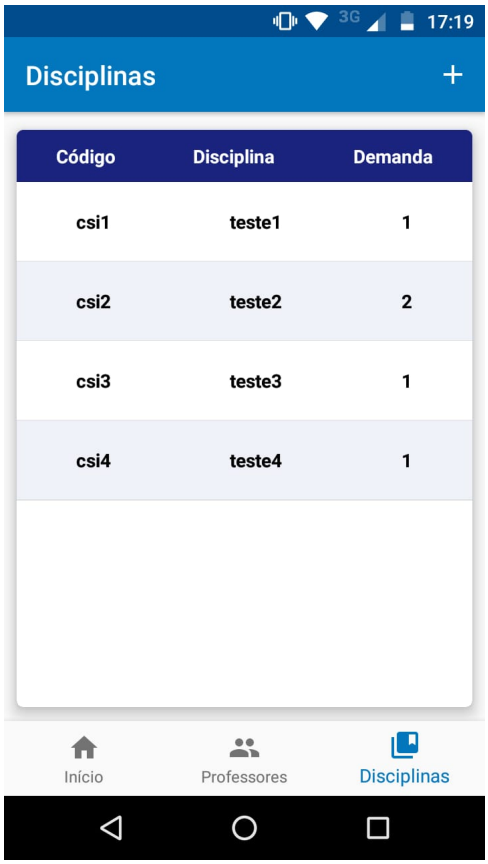
Figura 29 – Tela de cadastro das Tarefas(Disciplinas)



Fonte: Elaborado pelo autor

A Figura 30 exibe o aplicativo após as cinco disciplinas de exemplo serem cadastradas com sucesso.

Figura 30 – Exibição das Tarefas(Disciplinas) Cadastradas



Fonte: Elaborado pelo autor

A Figura 31 é um exemplo da edição ou exclusão da Disciplina csi1.

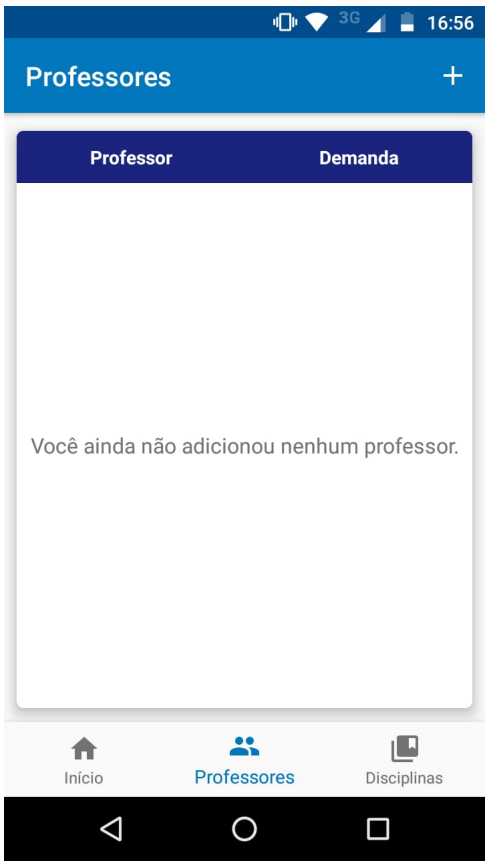
Figura 31 – Tela de alteração de dados ou exclusão de uma Tarefa(Disciplina)

Fonte: Elaborado pelo autor

A Figura 32 demonstra a tela onde os recursos pertinentes aos professores estão disponíveis. O recursos disponíveis são:

- Cadastrar: botão mais;
- Editar: clicando na linha do professor desejado ou no botão no final da linha do professor. No sub menu selecionar editar professor;
- Adicionar preferência: clicando na linha do professor desejado ou no botão no final da linha do professor. No sub menu selecionar adicionar preferência;
- Listar e excluir preferências: clicando na linha do professor desejado ou no botão no final da linha do professor. No sub menu selecionar editar preferências.

Figura 32 – Tela dos Recursos(Professores)

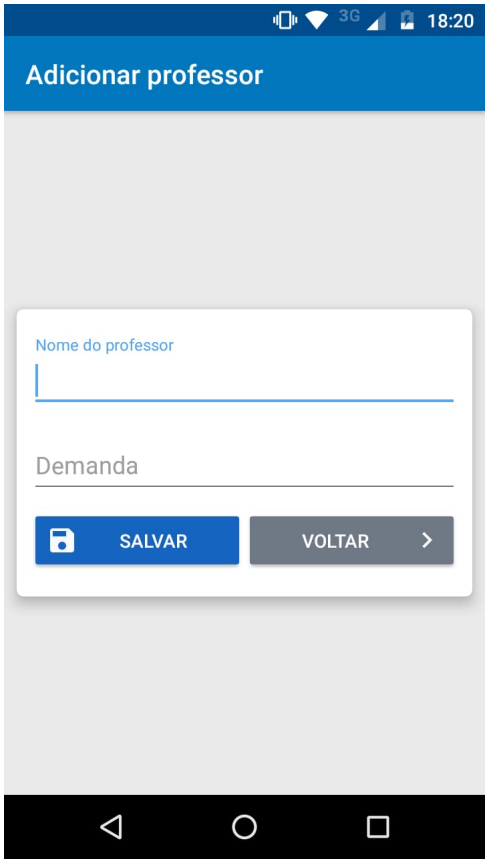


Fonte: Elaborado pelo autor

No exemplo a seguir foi feito o cadastro de três professores. Botão + no canto superior direito

A Figura 33 é utilizada para o cadastro de três professores como exemplo de utilização do função de cadastro de recursos.

Figura 33 – Tela de cadastro das Recurso(Professores)



Fonte: Elaborado pelo autor

A Figura 34 exibe aplicativo após três professores de exemplo serem cadastrados com sucesso:

Figura 34 – Exibição dos Recursos(Professores) Cadastrados

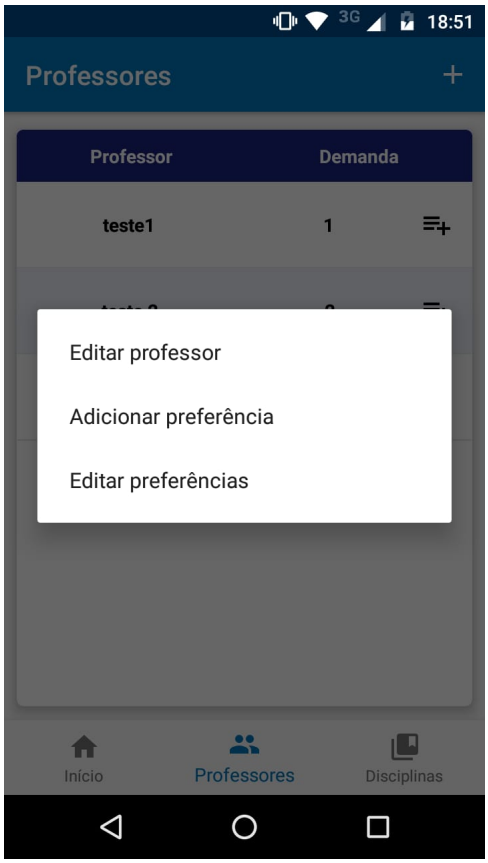
Fonte: Elaborado pelo autor

Sub menu como acessar: clicar na linha do professor desejado ou no botão no final da linha do professor. Cada professor cadastrado vai possuir um sub menu com as opções e recursos:

- Editar professor;
- Adicionar preferência;
- Listar e excluir preferências.

A Figura 35 mostra como este recurso é no aplicativo.

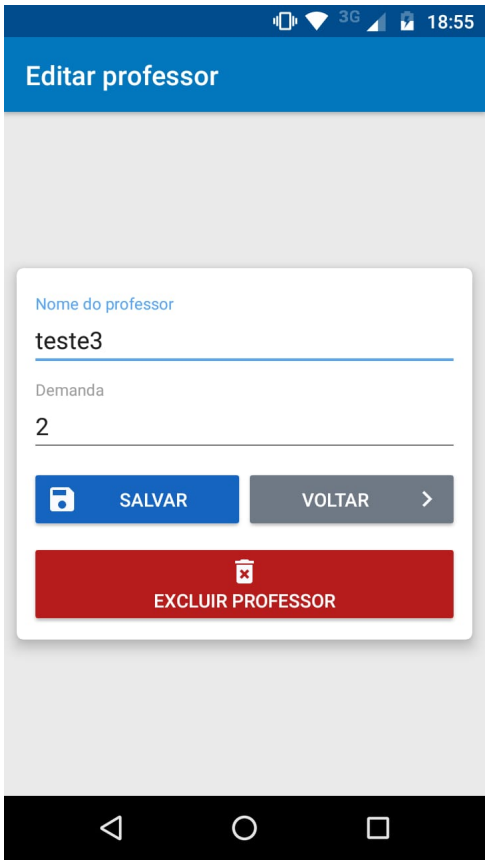
Figura 35 – Sub menu de funções de cada Recurso(Professores)



Fonte: Elaborado pelo autor

A Figura 36 mostra edição ou exclusão do Professor teste3 como o exemplo dos recursos.

Figura 36 – Tela de alteração dados ou exclusão de um Recurso(Professor)



Fonte: Elaborado pelo autor

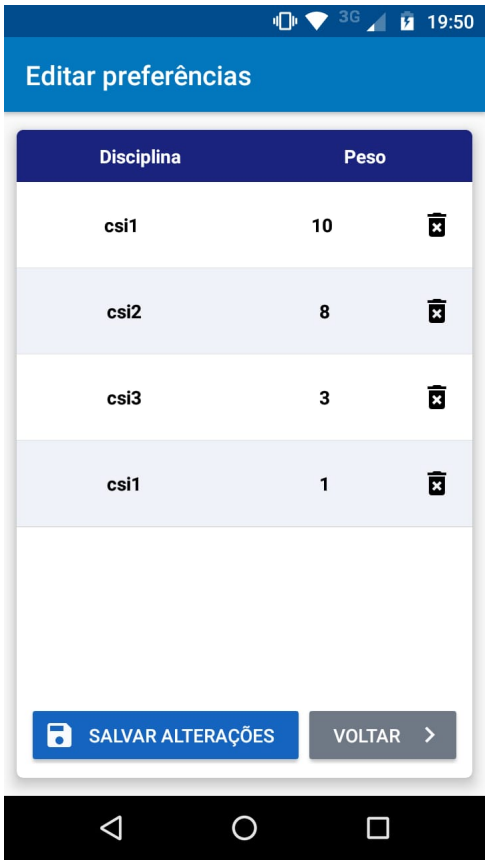
No levantamento dos dados foi feita uma pesquisa das afinidades do professor com cada uma das disciplinas através de um questionário. Uma preferência é a quantificação em valor numérico da afinidade do professor com a disciplina(exemplo: professor Teste1 tem afinidade 10 com a disciplina1). A Figura 37 mostra a adição de uma preferência do Pofessor teste1.

Figura 37 – Sub menu de funções de cada Recurso(Professores)

Fonte: Elaborado pelo autor

A Figura 38 demonstra um exemplo da exclusão das Preferências de um professor.

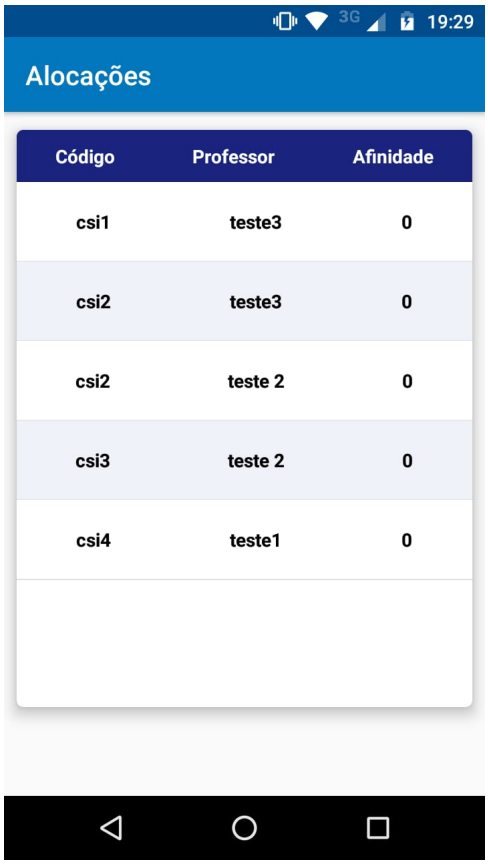
Figura 38 – Tela de exclusão de um Professor(Recurso)



Fonte: Elaborado pelo autor

A Figura 39 demonstra um exemplo de teste onde não foram adicionadas preferências. A matriz de Custo fica vazia.

Figura 39 – Teste onde não foram preenchidas as preferências(matriz de custo vazia)



Fonte: Elaborado pelo autor

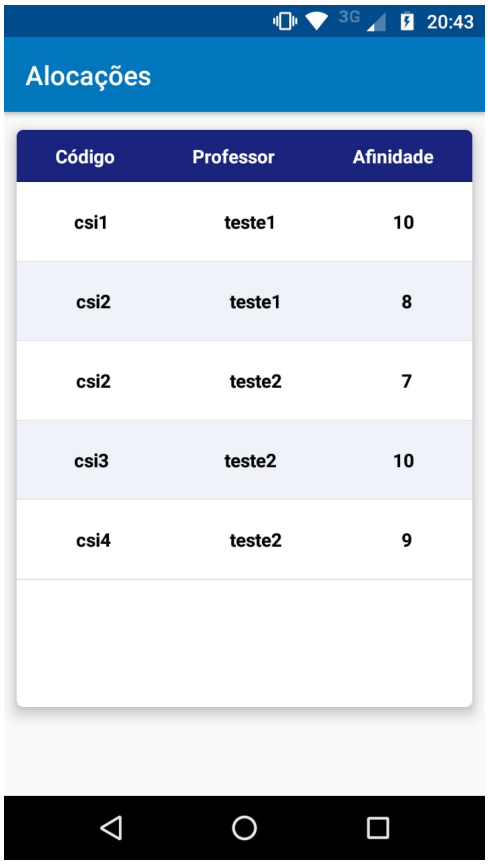
A Figura 40 exibe a soulução de um exemplo com os dados oferecidos pela Tabela 5.

Tabela 5 – Instância de exemplo

	Csi1	Csi2	Csi3	Csi4	Csi5
Teste1	10	8	8	3	1
Teste1	10	8	8	3	1
Teste2	2	7	7	10	9
Teste2	2	7	7	10	9
Teste2	2	7	7	10	9

Fonte: Elaborado pelo autor

Figura 40 – Teste para os dados da Tabela 1



Fonte: Elaborado pelo autor

Anexos

ANEXO A – Termo de Responsabilidade

TERMO DE RESPONSABILIDADE

Eu, **Saulo Martinho Alves dos Santos** declaro que o texto do trabalho de conclusão de curso intitulado "***Otimização na Alocação de Disciplinas a Professores***" é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas,referências ou consentimento dos respectivos autores.

João Monlevade, 2 de julho de 2019

Saulo Martinho Alves dos Santos

ANEXO B – Declaração de Conformidade

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DO CURSO DE SISTEMAS DE INFORMAÇÃO

DECLARAÇÃO DE CONFORMIDADE

Certifico que o(a) aluno(a) Saulo Martinho Alves dos Santos, autor do trabalho de conclusão de curso intitulado "Otimização na Alocação de Disciplinas a Professores" efetuou as correções sugeridas pela banca examinadora e que estou de acordo com a versão final do trabalho.

João Monlevade, 23 de julho de 2019.

Professor (a) Orientador (a)