UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
DEPARTAMENTO DE INFORMÁTICA

PROGRAMAÇÃO LINEAR E INTRODUÇÃO À OTIMIZAÇÃO
OTIMIZAÇÃO NA ALOCAÇÃO DE DISCIPLINAS A PROFESSORES

Alunos: Felipe Daniel Dias dos Santos
Matheus Batista Silva
Paulo Roberto de Jesus Gonçalves

VITÓRIA / SÃO MATEUS - ES
OUTUBRO DE 2021

1. INTRODUÇÃO

Este relatório é parte do trabalho realizado para a disciplina de Programação Linear e Introdução à Otimização do Departamento de Informática da Universidade do Espírito Santo, desenvolvido com intuito de aprimorar um aprendizado específico sobre modelagem de dados lineares e um problema existente no meio de estudo em questão: a otimização de custo.

O trabalho tem como objetivo apresentar o enunciado do problema selecionado, assim como sua definição matemática, e três exemplos de instâncias do problema, contendo os dados selecionados e suas fontes. Todas as instâncias apresentadas serão utilizadas para resolução através da ferramenta Gurobi, especializada em problemas de otimização.

2. ENUNCIADO DO PROBLEMA

O presente trabalho consiste em resolver o problema de alocação de professores às disciplinas, utilizando a linguagem Python e a ferramenta Gurobi de modo a automatizar o processo, tanto de entrada e saída de dados, quanto de resolução de instâncias do problema. O algoritmo requisita a matriz de custos como entrada, que pode ser gerada automaticamente, de maneira aleatória, ou confeccionada manualmente, utilizando os dados desejados. A saída será uma tabela de alocação, informando qual professor será responsável por cada disciplina, levando-se em conta o melhor benefício de combiná-los (menor custo) e as restrições impostas por eles.

Em um problema deste tipo, têm-se dois conjuntos e deve-se encontrar uma função que ligue elementos destes dois conjuntos. Pode haver 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 a soma de todas as alocações, respeitando as restrições existentes. 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). Uma alocação com o menor custo possível é denominada uma alocação ótima. O problema da alocação consiste em encontrar uma alocação ótima a partir de uma matriz custo dada.

No problema em questão, existe um número de professores e um número de disciplinas. Podemos alocar qualquer professor a qualquer uma das disciplinas, porém, cada alocação tem um custo que pode variar dependendo de cada disciplina e professor específicos. É necessário que todas as disciplinas sejam ofertadas, isto é, possuam algum professor responsável, designando exatamente um professor para cada disciplina, de modo que o custo total da soma de todas as alocações seja minimizado. O problema é modelado como uma instância de grafo bipartido, no qual cada vértice é uma entidade (professor ou disciplina) e cada aresta conecta um professor a uma disciplina.

3. DESCRIÇÃO DO MODELO

O modelo da alocação de disciplinas a professores possui uma variável de decisão, que chamaremos de ***fluxo_na_aresta***, que determina se uma aresta específica será selecionada para a alocação ótima. Em outras palavras, a variável ***fluxo_na_aresta*** de uma aresta ***(prof, disc)*** determina se o professor ***prof*** será responsável pela disciplina ***disc***. Assim, temos que:

$$fluxo\_na\_aresta = \{1\ se\ o\ professor\ leciona\ a\ disciplina\ e\ 0\ caso\ contrário\}$$

O modelo possui um parâmetro que chamaremos de ***custo_da_aresta***, que define o peso de uma aresta qualquer. Assim, ***custo_da_aresta*** de uma aresta ***(prof, disc)*** determina qual é o custo do professor ***prof*** lecionar a disciplina ***disc***. Os valores dos custos de todas as arestas do modelo estão contidos na matriz de custo, que é uma das entradas do algoritmo. Esse custo leva alguns fatores em consideração, como a afinidade do professor em relação à disciplina, sua especialidade, etc.

Com a definição da variável de decisão e parâmetro de custo, podemos caracterizar a função objetivo, isto é, aquela da qual se deseja obter o custo mínimo. A função objetivo nada mais é que a soma dos custos das arestas selecionadas na alocação, ou seja, a soma do produto entre as variáveis ***fluxo_na_aresta*** e ***custo_da_aresta***, para todas as arestas. Matematicamente:

$$soma_{(para\ todo\ vértice\_professor,\ para\ todo\ vértice\_disciplina)}\ fluxo\_na\_aresta\ *\ custo\_da\_aresta$$

Para formalizar a alocação de disciplinas a professores como um problema de Programação Linear completo, é necessário definir as restrições do problema. A primeira restrição clara é a domínio da variável de decisão, que precisa ser binária:

$$fluxo\_na\_aresta \in \{0, 1\}\ para\ todo\ vértice\_professor\ e\ para\ todo\ vértice\_disciplina$$

Além disso, cada disciplina possui somente um professor responsável pela mesma e cada professor é responsável por somente uma disciplina. Isto é, cada professor está associado a somente uma disciplina e vice-versa. Assim sendo, o fluxo total de todos os vértices, tanto professor quanto disciplina, são iguais a 1. Matematicamente:

$$para\ todo\ vértice\_professor,\ a\ soma\ do\ fluxo\_na\_aresta\ originados\ naquele\ vértice\ = 1$$
$$para\ todo\ vértice\_disciplina,\ a\ soma\ do\ fluxo\_na\_aresta\ originados\ naquele\ vértice\ = 1$$

Logo, o problema de Programação Linear da alocação de disciplinas a professores é matematicamente descrito como:

Minimize:

$$soma_{(para\ todo\ vértice\_professor,\ para\ todo\ vértice\_disciplina)}\ fluxo\_na\_aresta\ *\ custo\_da\_aresta$$

Sujeito a:

$$para\ todo\ vértice\_professor,\ a\ soma\ dos\ fluxo\_na\_aresta\ originados\ naquele\ vértice\ = 1$$
$$para\ todo\ vértice\_disciplina,\ a\ soma\ dos\ fluxo\_na\_aresta\ originados\ naquele\ vértice\ = 1$$
$$fluxo\_na\_aresta \in \{0, 1\}\ para\ todo\ vértice\_professor\ e\ para\ todo\ vértice\_disciplina$$

4. IMPLEMENTAÇÃO

4.1. GERAÇÃO DE DADOS

```python
#Importando as bibliotecas necessárias
import xlsxwriter
from random import randint

#Função de geração de dados referentes às demandas, ofertas e custos, todos aleatórios
def geracaoDeDados(qnt_professores, qnt_disciplinas):

    #Criando o arquivo de nome "Dados", que conterá as planilhas de custos
    #oferta de professores e demanda de disciplinas
    dados = xlsxwriter.Workbook("Dados.xlsx")
    planilha_custos = dados.add_worksheet("Custos")
    planilha_professores = dados.add_worksheet("Professores")
    planilha_disciplinas = dados.add_worksheet("Disciplinas")

    #Nomeando as colunas das planilhas
    planilha_professores.write(0, 0, "Professor")
    planilha_professores.write(0, 1, "Oferta")
    planilha_disciplinas.write(0, 0, "Disciplina")
    planilha_disciplinas.write(0, 1, "Demanda")

    #Definindo o nome dos professores e a quantidade de turmas que cada um pode assumir
    for prof in range(qnt_professores):

        linha = prof + 1
        nome = "Professor " + str(linha)
        planilha_custos.write(linha, 0, nome)
        planilha_professores.write(linha, 0, nome)

        #O professor x pode assumir de 1 a 3 turmas
        #planilha_professores.write(linha, 1, randint(1, 3))

        #O professor, nesse modelo, assume somente uma turma
        planilha_professores.write(linha, 1, 1)

    #Definindo o nome das disciplinas e a quantidade de professores demandada
    for disc in range(qnt_disciplinas):
```

```python

        coluna = disc + 1
        nome = "Disciplina " + str(coluna)
        planilha_custos.write(0, coluna, nome)
        planilha_disciplinas.write(coluna, 0, nome)

        #Cada disciplina, nesse modelo, necessita somente de um professor
        planilha_disciplinas.write(coluna, 1, 1)

    #Preenchendo a planilha de custos: a matriz de custos conterá valores aleatórios
    for prof in range(qnt_professores):

        #Adicionando custos aleatórios à planilha de custos
        for disc in range(qnt_disciplinas):

            #O custo de um professor assumir uma disciplina varia de 1 a 100,
            #determinado aleatoriamente
            planilha_custos.write(prof + 1, disc + 1, randint(1, 100))

    dados.close()

#Chamando a função principal com dois argumentos: a quantidade de professores e disciplinas
geracaoDeDados(10, 10)
```

4.2. DESCRIÇÃO DO MODELO

```python
#Importando as bibliotecas necessÃ¡rias
import gurobipy as gp
import xlsxwriter

#Imprimindo soluÃ§Ã£o e salvando em um arquivo
def imprimirSalvarSolucao(modelo, arestas, custos, fluxo_na_aresta):

    if modelo.status == gp.GRB.OPTIMAL:

        print("\nSoluÃ§Ã£o existente para essa instÃ¢ncia do problema. Imprimindo resultados:")

        #Criando o arquivo de nome "Resultados", que conterÃ¡ os dados resultantes
        resultados = xlsxwriter.Workbook("Resultados.xlsx")
        planilha_alocacao = resultados.add_worksheet("AlocaÃ§Ã£o")
        planilha_alocacao.write(0, 0, "Professor")
        planilha_alocacao.write(0, 1, "Disciplina")
        planilha_alocacao.write(0, 2, "Custo")

        linha = 1
        custo_total = 0

        print("\nCusto mÃnimo:", modelo.objVal)
        print("\nAlocaÃ§Ã£o para atingir o custo mÃnimo:\n")

        for aresta in arestas:

            #Para cada aresta alocada, imprimi-la e armazenÃ¡-la na planilha, assim como seu
            #custo
            if fluxo_na_aresta[aresta].x > 0.0001:

                print("AlocaÃ§Ã£o:", aresta, "Custo:", custos[aresta])

                planilha_alocacao.write(linha, 0, aresta[0])
                planilha_alocacao.write(linha, 1, aresta[1])
                planilha_alocacao.write(linha, 2, custos[aresta])
                linha += 1
                custo_total += custos[aresta]

        planilha_alocacao.write(linha, 2, custo_total)
        resultados.close()

        print("\nSoluÃ§Ã£o armazenada com sucesso no arquivo Resultados.xlsx\n")

    else:

        print("NÃ£o existe soluÃ§Ã£o para essa instÃ¢ncia do problema. Tente com outros dados.")

#FunÃ§Ã£o de resoluÃ§Ã£o do problema de programaÃ§Ã£o linear de alocaÃ§Ã£o de disciplinas a professores
def resolver(professores, qnt_professores, disciplinas, qnt_disciplinas, arestas, custos):

    #DefiniÃ§Ã£o do modelo
    modelo = gp.Model("AlocaÃ§Ã£o de Disciplinas")

    #AdiÃ§Ã£o de variÃ¡vel de decisÃ£o no modelo (o fluxo em cada aresta)
    fluxo_na_aresta = modelo.addVars(arestas, ub = 1, name = 'x')

    #FunÃ§Ã£o objetivo: minimizar a soma dos custos das arestas selecionadas
    modelo.setObjective(gp.quicksum(custos[(prof, disc)] * fluxo_na_aresta[prof, disc] for prof, disc in aresta

    #RestriÃ§Ãµes: cada disciplina e professor sÃ£o alocados tantas vezes quanto requisitado
    #Nesse caso, deve existir um professor em cada disciplina e cada professor leciona
    #somente uma disciplina (ou conforme as informaÃ§Ãµes disponÃveis nas planilhas)
    modelo.addConstrs((fluxo_na_aresta.sum('*', disc) == qnt_disciplinas[disc] for disc in disciplinas), '_')
    modelo.addConstrs((fluxo_na_aresta.sum(prof, '*') == qnt_professores[prof] for prof in professores), '_')

    #Resolvendo e imprimindo o modelo
    modelo.optimize()
    imprimirSalvarSolucao(modelo, arestas, custos, fluxo_na_aresta)
```

4.3. OBTENÇÃO DE DADOS

```python
#Importando as bibliotecas necessárias
import os
import xlrd
import gurobipy as gp

#Extraindo e formatando os dados da planilha de professores
def obterProfessores(planilha_professores):

    professores = {}
    linha = 1

    #A planilha é percorrida uma excessão de índice inválido ocorrer
    while True:

        try:

            professores[planilha_professores.cell_value(linha, 0)] = planilha_professores.cell_value(linha, 1)
            linha += 1

        except IndexError:

            break

    #Transformando o dicionário em um multidict
    professores, qnt_professores = gp.multidict(professores)

    return professores, qnt_professores

#Extraindo e formatando os dados da planilha de disciplinas
def obterDisciplinas(planilha_disciplinas):

    disciplinas = {}
    linha = 1

    #A planilha é percorrida uma excessão de índice inválido ocorrer
    while True:
```

```python
        try:

            disciplinas[planilha_disciplinas.cell_value(linha, 0)] = planilha_disciplinas.cell_value(linha, 1)
            linha += 1

        except IndexError:

            break

    #Transformando o dicionário em um multidict
    disciplinas, qnt_disciplinas = gp.multidict(disciplinas)

    return disciplinas, qnt_disciplinas

#Extraindo e formatando os dados da planilha de custos
def obterCustos(planilha_custos, professores, disciplinas):

    arestas = []
    custos = {}

    for linha in range(len(professores)):

        for coluna in range(len(disciplinas)):

            #Extraindo as tuplas (professor, disciplina) disponíveis e armazenando em um
            #dicionário cuja chave é a tupla e valor o custo associado
            tupla = (planilha_custos.cell_value(linha + 1, 0), planilha_custos.cell_value(0, coluna + 1))
            custos[tupla] = planilha_custos.cell_value(linha + 1, coluna + 1)

            #Armazenando as arestas em uma lista
            arestas.append(tupla)

    #Transformando o dicionário em um multidict e a lista em um tuplelist
    arestas = gp.tuplelist(arestas)
    _, custos = gp.multidict(custos)
```

```
    return arestas, custos

#Função para extrair todos os dados das planilhas, armazenando-os em estruturas adequadas
#para utilização no solucionador do problema de programação linear
def obterDados():

    #Abrindo o arquivo e obtendo as planilhas
    dados = xlrd.open_workbook("Dados.xlsx")
    planilha_professores = dados.sheet_by_name("Professores")
    planilha_disciplinas = dados.sheet_by_name("Disciplinas")
    planilha_custos = dados.sheet_by_name("Custos")

    #Obtendo os dados de cada planilha
    professores, qnt_professores = obterProfessores(planilha_professores)
    disciplinas, qnt_disciplinas = obterDisciplinas(planilha_disciplinas)
    arestas, custos = obterCustos(planilha_custos, professores, disciplinas)

    #Retornando todo o conteúdo necessário
    return professores, qnt_professores, disciplinas, qnt_disciplinas, arestas, custos
```

4.4. INSTALAÇÃO DE BIBLIOTECAS

```
#Importando as bibliotecas necessárias
import sys
import subprocess
import pkg_resources

#Verificando os pacotes necessários que estão faltando e instalando-os na máquina
def setup():

    requeridos = {"xlrd == 1.2.0", "gurobipy", "xlsxwriter"}
    instalados = {pkg.key for pkg in pkg_resources.working_set}
    faltando = requeridos - instalados

    if faltando:

        subprocess.check_call([sys.executable, "-m", "pip", "install", *faltando])
```

4.5. FUNÇÃO PRINCIPAL

```
#Importanto as bibliotecas necessÃ¡rias
from setup import setup
from modelo import resolver
from obtencaoDeDados import obterDados

#FunÃ§Ã£o principal: instalaÃ§Ã£o de pacotes, obtenÃ§Ã£o de dados e resoluÃ§Ã£o do problema
def main():

  #Instalando os pacotes necessÃ¡rios, caso estejam faltando
  setup()

  #Obtendo os dados necessÃ¡rios, gerados e armazenados nas planilhas
  professores, qnt_professores, disciplinas, qnt_disciplinas, arestas, custos = obterDados()

  #Resolvendo o problema com os dados obtidos
  resolver(professores, qnt_professores, disciplinas, qnt_disciplinas, arestas, custos)

if __name__ == "__main__":

  main()
```

5. EXEMPLOS DE APLICAÇÃO

5.1. CASO DE TESTE 5x5

Matriz de custos:

Custos	Disciplina 1	Disciplina 2	Disciplina 3	Disciplina 4	Disciplina 5
Professor 1	92	15	87	60	66
Professor 2	37	3	92	50	78
Professor 3	61	5	30	50	56
Professor 4	88	56	91	68	50
Professor 5	67	85	17	67	31

Resultados:

Professor	**Disciplina**	**Custo**
Professor 1	Disciplina 2	15
Professor 2	Disciplina 1	37
Professor 3	Disciplina 4	50
Professor 4	Disciplina 5	50
Professor 5	Disciplina 3	17
		169

5.2. CASO DE TESTE 10x10

Matriz de custos:

Custos	Disciplina 1	Disciplina 2	Disciplina 3	Disciplina 4	Disciplina 5	Disciplina 6	Disciplina 7	Disciplina 8	Disciplina 9	Disciplina 10
Professor 1	70	54	39	38	29	63	78	18	100	1
Professor 2	70	70	99	60	51	74	9	79	13	3
Professor 3	74	61	37	47	69	96	40	86	77	57
Professor 4	18	41	48	74	33	66	60	20	36	13
Professor 5	55	74	38	55	13	52	69	64	50	77

Professor 6	63	7	94	35	41	48	70	11	4	73
Professor 7	35	91	83	73	74	15	71	32	98	45
Professor 8	9	94	28	22	35	22	10	59	86	51
Professor 9	83	5	21	98	83	23	88	78	43	30
Professor 10	83	27	66	50	71	8	29	2	4	59

Resultados:

Professor	**Disciplina**	**Custo**
Professor 1	Disciplina 10	1
Professor 2	Disciplina 7	9
Professor 3	Disciplina 3	37
Professor 4	Disciplina 1	18
Professor 5	Disciplina 5	13
Professor 6	Disciplina 9	4
Professor 7	Disciplina 6	15
Professor 8	Disciplina 4	22
Professor 9	Disciplina 2	5
Professor 10	Disciplina 8	2
		126

5.3. CASO DE TESTE 15x15

Matriz de custos:

Custos	Disciplina 1	Disciplina 2	Disciplina 3	Disciplina 4	Disciplina 5	Disciplina 6	Disciplina 7	Disciplina 8	Disciplina 9	Disciplina 10	Disciplina 11	Disciplina 12	Disciplina 13	Disciplina 14	Disciplina 15
Professor 1	58	27	16	50	16	51	39	69	55	41	48	13	91	84	51
Professor 2	46	12	41	74	83	20	56	6	69	20	38	49	91	8	70
Professor 3	58	17	45	74	16	31	63	15	76	20	55	83	71	61	86
Professor 4	92	1	75	6	30	60	4	86	8	97	48	25	95	97	58
Profes	100	31	53	34	30	26	15	61	40	6	40	71	76	37	46

sor 5															
Professor 6	77	60	58	9	74	82	2	68	46	92	85	85	97	10	83
Professor 7	48	77	49	83	88	89	85	70	68	12	53	77	65	10	22
Professor 8	83	47	58	86	30	15	44	38	94	30	41	32	14	25	4
Professor 9	85	29	98	77	94	15	96	16	73	65	22	56	21	97	71
Professor 10	93	24	37	65	35	80	85	6	31	96	16	96	90	7	91
Professor 11	77	91	76	32	37	37	76	86	9	84	29	29	27	81	9
Professor 12	38	6	49	43	85	54	70	6	26	62	43	61	96	2	100
Professor 13	69	11	100	73	57	54	33	75	69	46	23	23	97	68	95
Professor 14	10	69	65	41	89	97	23	83	4	68	80	3	22	31	87
Professor 15	28	50	18	38	83	17	19	13	78	39	74	28	12	54	43

Resultados:

Professor	**Disciplina**	**Custo**
Professor 1	Disciplina 3	16
Professor 2	Disciplina 8	6
Professor 3	Disciplina 5	16
Professor 4	Disciplina 4	6
Professor 5	Disciplina 10	6
Professor 6	Disciplina 7	2
Professor 7	Disciplina 14	10
Professor 8	Disciplina 15	4
Professor 9	Disciplina 6	15
Professor 10	Disciplina 11	16
Professor 11	Disciplina 9	9
Professor 12	Disciplina 2	6
Professor 13	Disciplina 12	23
Professor 14	Disciplina 1	10
Professor 15	Disciplina 13	12
		157