Universidade Federal da Paraíba

Centro de Informática

Relatório de Pesquisa Operacional

Grupo: Elaine Cristina Lins de Souza - Matrícula: 20170020854 - elainesouza@eng.ci.ufpb.br

Letícia Sousa Leite - Matrícula: 20170108784 - leticialeite@eng.ci.ufpb.br

Stênio Ellison Pereira Ferreira - Matrícula: 2016068763 - stenio1998@gmail.com

Professor: Teobaldo Leite Bulhões Junior.

Disciplina: Pesquisa Operacional.

JOÃO PESSOA, PB.

Maio, 2021.

ELAINE CRISTINA LINS DE SOUZA.

LETÍCIA SOUSA LEITE.

STÊNIO ELLISON PEREIRA FERREIRA.

Relatório de Pesquisa Operacional

Relatório de atividade prática da disciplina de Pesquisa Operacional, do curso de Engenharia de Computação, da Universidade Federal da Paraíba como requisito para a obtenção da primeira nota.

Orientador: Teobaldo Leite Bulhões Junior.

João Pessoa - PB

2021

Sumário

1. Introdução

Um dos grandes problemas do campo da pesquisa operacional atualmente, consiste na otimização do fluxo de redes. Essas redes, são grafos orientados que possuem arestas com capacidades e que recebem um fluxo, podendo ser diretamente associadas com problemas rotineiros em nosso cotidiano, como por exemplo as redes: de comunicação, transporte, distribuição, dentre diversas outras. Para otimizar e até mesmo resolver esses problemas, várias metodologias, algoritmos e softwares foram desenvolvidos, muitos desses, utilizaram programação linear. Devido a isso, vários outros problemas surgiram, como: problema do caminho mais curto, problema de fluxo mínimo, problema de árvore de expansão mínima, o problema de fluxo máximo, problema de custo mínimo, entre outros.

Diante de tudo o que foi exposto, essa relatório tem como principal objetivo mostrar o procedimento para a implementação de um código com o auxílio de um pacote de programação linear, a modelagem de um problema de fluxo máximo(PFM), em função de uma rede qualquer, apresentando um exemplo particular de rede e de fluxo viável, bem como a solução da letra (a) da questão 9.4-3 do livro “Introdução à Pesquisa Operacional”, dos autores Hillier e Lieberman, 9ª edição. Para o entendimento desses conceitos e para alcançar objetivos propostos, esse relatório é dividido em: definição do problema, modelagem, instruções, exercicío, resultados e discussões, considerações finais e referências bibliográficas.

2. Definição do problema

Segundo FEOFILOFF(2019), O problema do fluxo máximo (PFM) é uma poderosa ferramenta de modelagem, capaz de representar uma grande variedade de problemas de fluxo. Essa ferramenta funciona da seguinte forma: Considerando uma rede direcionada (dígrafo) conectada, com 2 nós especiais determinados de origem e destino e ainda, associado a cada arco, uma capacidade máxima não-negativa. O objetivo é encontrar o fluxo máximo na rede. Algumas aplicações do PFM são: Maximizar a rede de distribuição de suprimentos de uma determinada empresa, maximizar o fluxo de veículos de um estacionamento, ou até mesmo, maximizar o fluxo de água de um aqueduto.

Por sua vez, o problema do fluxo de custo mínimo (PFCM), é uma forma de encontrar a maneira com o menor custo possível de recursos para resolver uma certa quantidade de fluxo através de uma rede. Com o ajuste de alguns parâmetros, podemos notar que o PFM nada mais é que um subproblema do PFCM.

Tendo em conta essas informações, podemos exemplificar esse problema, considerando como exemplo uma rede de distribuição de suprimentos de uma empresa x, para seu cliente. Que resultou, na rede fluxos da imagem a seguir:

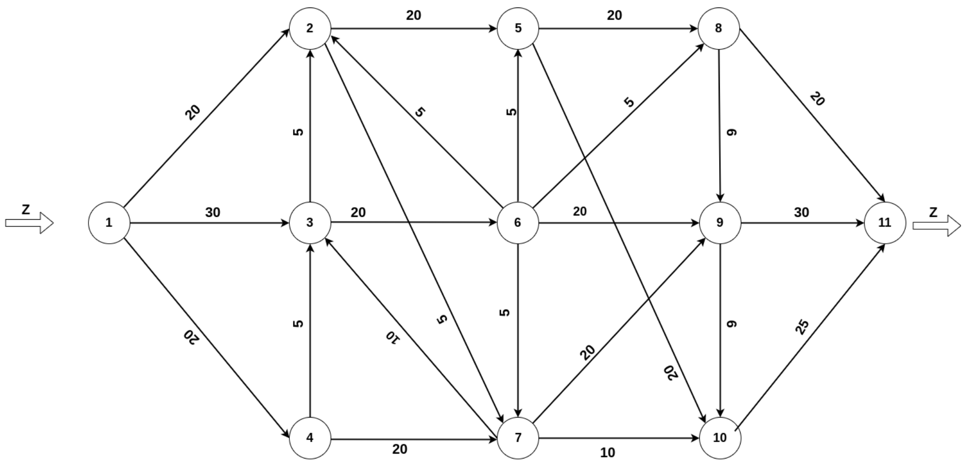

Imagem 01: Grafo exemplificativo com as capacidades máxima de cada arco e os seus nós.

Considerando que esse grafo possui 11 vértices, 24 arcos, onde o vértice 1 representa a origem e o vértice 11 o escoadouro. Deveremos escolher uma quantidade existente de nós i e j, atribuindo isso à variável X_{ij}. Em seguida, definimos a função objetivo(Z), ponderando a lei da conservação do fluxo máximo(o fluxo que entra deverá ser igual ao fluxo que sai negativamente) para realizar os cortes mínimos verticais na rede, que posteriormente irão ser maximizados até atingir toda a rede de fluxo.

A função objetivo, é portanto definida a partir dos arcos que entram no escoadouro, nesse caso:

$$\text{Max } Z = X_{811} + X_{911} + X_{1011}$$

A próxima fase para resolver esse problema é encontrar as restrições de balanceamento, desconsiderando os nós de origem e escoadouro (1 e 11).

Nó 2: $X_{12} + X_{32} + X_{62} - X_{27} - X_{25} = 0$

Nó 3: $X_{13} + X_{43} + X_{73} - X_{32} - X_{36} = 0$

Nó 4: $X_{14} - X_{43} - X_{47} = 0$

Nó 5: $X_{25} + X_{65} - X_{58} - X_{510} = 0$

Nó 6: $X_{36} - X_{62} - X_{65} - X_{67} - X_{68} = 0$

Nó 7: $X_{27} + X_{47} + X_{67} - X_{73} - X_{79} - X_{710} = 0$

Nó 8: $X_{58} + X_{68} - X_{89} - X_{811} = 0$

Nó 9: $X_{89} + X_{69} + X_{79} - X_{910} - X_{911} = 0$

Nó 10: $X_{510} + X_{710} + X_{910} - X_{1011} = 0$

As restrições para cada arco são:

Nó 01

$X_{12} \leq 20$

$X_{13} \leq 30$

$X_{14} \leq 20$

Nó 02

$X_{25} \leq 20$

$X_{27} \leq 5$

Nó 03

$X_{32} \leq 5$

$X_{36} \leq 20$

Nó 04

$X_{43} \leq 5$

$X_{47} \leq 20$

Nó 05

$X_{58} \leq 20$

$X_{510} \leq 20$

Nó 06

$X_{65} \leq 5$

$X_{67} \leq 5$

$X_{68} \leq 5$

$X_{69} \leq 20$

Nó 07

$X_{79} \leq 20$

$X_{710} \leq 10$

Nó 08

$X_{89} \leq 6$

$X_{811} \leq 20$

Nó 09

$X_{910} \leq 6$

$X_{911} \leq 30$

Nó 10

$X_{1011} \leq 25$

Portanto, para esse caso uma possibilidade de fluxo máximo viável, apesar da rede possuir arestas com capacidade maiores, é de Z = 65. A imagem a seguir irá mostrar melhor como funciona essa distribuição.

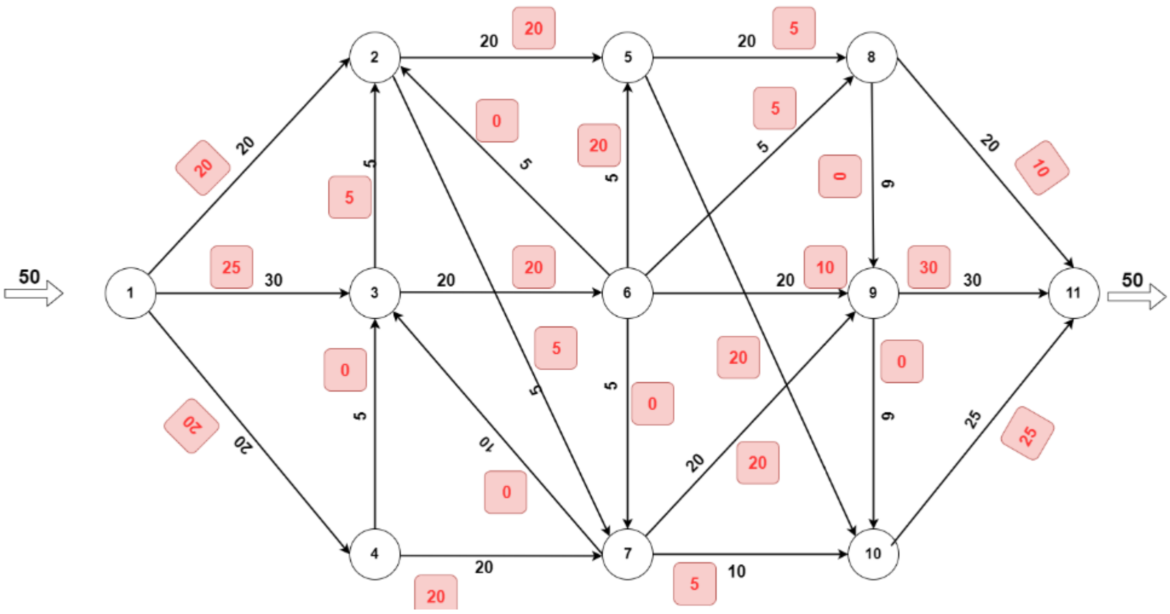

Imagem 02: Grafo de distribuição do fluxo.

Assim fica o diagrama do fluxo de rede após a transformação de PFM para PFCM:

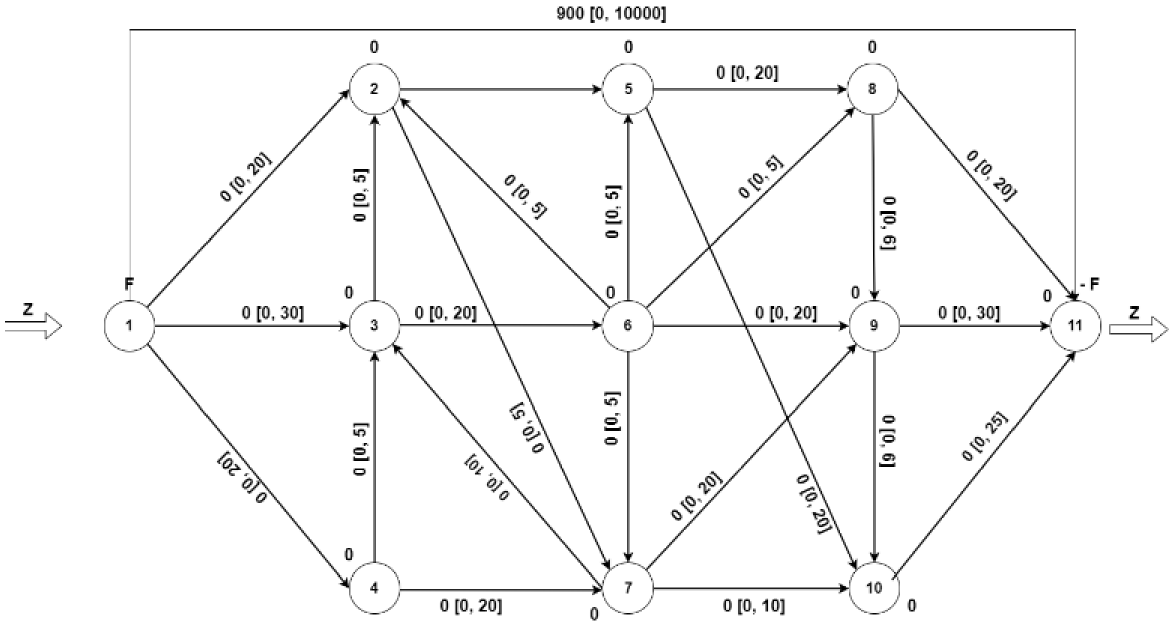

Imagem 03: Grafo que representa o problema do fluxo de custo mínimo.

Onde todos os custos e a capacidade mínima em cada arco são definidos como zero, F é o maior fluxo viável (demanda) e está relacionado apenas a origem e o escoadouro, e todos os outros nós recebem demanda igual a zero, sendo denominados nós de transbordo. Mais detalhes sobre o processo de transformação e modelagem serão descritos na próxima seção desta atividade.

3. Modelagem

3.1 Pacote de programação linear

OR-Tools é um pacote de software de código aberto para otimização da Google, ele é ajustado para solucionar problemas de roteamento de veículos, fluxos, programação inteira e linear e programação de restrição. Dessa forma, seu principal foco é em fazer otimização combinatória e pode ser utilizada com bibliotecas de programação linear(inteira e mista), por suportar bibliotecas desse tipo, e devido ao fácil acesso a conteúdos de suporte, optamos por escolher esse pacote.

3.2 Etapas do processo de modelagem

Para a transformação do problema de fluxo máximo para o problema de fluxo de custo mínimo seguimos as seguintes etapas:

1. Atribuímos o custo *Cij* igual a 0, dessa maneira, atribuímos a realidade de que os arcos não tem custo em um problema de fluxo máximo.
2. Designar um valor F com a função de limitar o fluxo máximo viável pela rede.
3. Configurar os nós de demanda e suprimento igual a F e (-F)
4. Adicionamos um único nó de suprimento até o nó de demanda
5. Atribuir a este nó um custo fixo C*ij* igual ao um custo unitário M com uma capacidade de arco ilimitada.

A definição do custo unitário para o arco que vai do nó de suprimento até o de demanda e o custo igual a 0 para os demais arcos, o PFCM envia um fluxo seguro através dos demais arcos, assim alcançando o objetivo do problema do fluxo máximo.

3.3 Implementação

★ **Etapa 01**

- Leitura do arquivo

```
#Abrindo o arquivo e salvando seus dados, linha a linha

arquivo = open("instancias/instance7.txt", 'r')
texto = arquivo.readlines()
arquivo.close()
print(texto)
```

Imagem 04: Trecho de código da leitura do arquivo.

- Indexação das variáveis

```
#Guardando dados nas variáveis

n_vertices = int(texto[0].split()[0])
n_arcos = int(texto[1].split()[0])
i_origem = int(texto[2].split()[0])
i_escoadouro = int(texto[3].split()[0])

pontoInicial = []
pontoFinal = []
capacidadeArco = []

for i in range(4, len(texto)):
    pontoInicial.append(int(texto[i].split()[0]))
    pontoFinal.append(int(texto[i].split()[1]))
    capacidadeArco.append(int(texto[i].split()[2]))

print(n_vertices, n_arcos, i_origem, i_escoadouro)
print(pontoInicial)
print(pontoFinal)
print(capacidadeArco)
```

Imagem 05: Indexação das variáveis.

★ **Etapa 02** - Início da transformação de um PFM em um PFCM

- Definindo todos os custos iguais a zero.

```
#definindo todos os custos como zero
custo = []
for i in range(4, len(texto)):
    custo.append(0)
```

Imagem 06: Configuração de custos iguais a zero.

- Definindo o valor de F, o custo unitário é uma capacidade grande.

```
#transformando o problema de fluxo máximo em fluxo de custo mínimo

#definindo todos os custos como zero
custo = []
for i in range(4, len(texto)):
    custo.append(0)

#calculando o fluxo máximo viável
soma = 0
for i in capacidadeArco:
    soma += i

F = int(soma) #Limite do fluxo máximo viável pela rede
M = 900 #Custo unitário máximo
U = 10000 #Capacidade máxima de arco

pontoInicial.append(int(i_origem))
pontoFinal.append(int(i_escoadouro))
capacidadeArco.append(int(U))
custo.append(int(M))
```

Imagem 07: Cálculo do fluxo máximo viável(F), atribuição do custo unitário e capacidade máxima.

- Configurando os nós de demanda e suprimento igual a F e (-F)

```
#definindo as demandas em cada nó
suprimentos = []
for i in range(int(n_vertices)+1):
    suprimentos.append(0)

suprimentos[i_origem] = F
suprimentos[i_escoadouro] = -F
```

Imagem 08: Nó de demanda e nó de suprimento

★ **Etapa 03**

- Uso da biblioteca Ortools

```
fluxo_custo_min = pywrapgraph.SimpleMinCostFlow()
fluxo_max = 0

for i in range(0, len(pontoInicial)):
    fluxo_custo_min.AddArcWithCapacityAndUnitCost(pontoInicial[i], pontoFinal[i], capacidadeArco[i], custo[i])

for i in range(0, len(suprimentos)):
    fluxo_custo_min.SetNodeSupply(i, suprimentos[i])

if fluxo_custo_min.Solve() == fluxo_custo_min.OPTIMAL:
    print('Custo Mínimo:', fluxo_custo_min.OptimalCost())
    print('')
    print('  Arco    Fluxo / Capacidade  Custo')
    for i in range(fluxo_custo_min.NumArcs()):
        custoArco = fluxo_custo_min.Flow(i) * fluxo_custo_min.UnitCost(i)
        print('%1s -> %1s   %3s  / %3s       %3s' % (fluxo_custo_min.Tail(i), fluxo_custo_min.Head(i), fluxo_custo_min.Flow(i), fluxo_

        #soma fluxo máximo
        if fluxo_custo_min.Tail(i) == 1 and fluxo_custo_min.Capacity(i) < U:
            fluxo_max += fluxo_custo_min.Flow(i)
else:
    print('Não foi possível encontrar a solução.')

print("")
print("Fluxo Máximo = ", fluxo_max)
```

Imagem 09: Uso da biblioteca Ortools.

★ Exemplo de resultado

Como exemplo de saída estamos apresentando o resultado final do exemplo dado na definição do problema, que corresponde ao arquivo da instância 7

```
Custo Mínimo: 258300

  Arco    Fluxo / Capacidade  Custo
1 -> 2    20  /  20         0
1 -> 3    25  /  30         0
1 -> 4    20  /  20         0
2 -> 5    20  /  20         0
2 -> 7     5  /   5         0
3 -> 2     5  /   5         0
3 -> 6    20  /  20         0
4 -> 3     0  /   5         0
4 -> 7    20  /  20         0
5 -> 8     5  /  20         0
5 -> 10    20  /  20         0
6 -> 2     0  /   5         0
6 -> 5     5  /   5         0
6 -> 7     0  /   5         0
6 -> 8     5  /   5         0
6 -> 9    10  /  20         0
7 -> 3     0  /  10         0
7 -> 9    20  /  20         0
7 -> 10     5  /  10         0
8 -> 9     0  /   6         0
8 -> 11    10  /  20         0
9 -> 11    30  /  30         0
9 -> 10     0  /   6         0
10 -> 11    25  /  25         0
1 -> 11   287  / 10000       258300

Fluxo Máximo =  65
```

Imagem 10: Saída correspondente a instância 7.

4. Instruções

Nesta seção serão apresentadas as instruções de instalação e uso do código. O arquivo compactado com todos os arquivos necessários encontram-se nesta pasta do Google Drive.

Para implementação da modelagem foi usado o *Jupyter Notebook* devido a sua praticidade na visualização dos dados. Antes de tudo deve-se instalar a versão mais recente e estável do *Python*. Para o sistema operacional Windows basta acessar o site oficial através deste link e fazer o download do executável referente às configurações de sua máquina, o mesmo vale para o sistema Mac OS X, e os downloads se encontram neste link. Para sistemas Linux basta executar os seguintes comandos no terminal:

$ sudo apt-get install python3

$ sudo apt-get install python3-pip

Em seguida deve-se instalar o software Anaconda, os instaladores para todos os sistemas operacionais encontram-se neste link. Através do Anaconda obtém-se o Jupyter Notebook, e então basta abrir a pasta por ele e executar o script. Por fim, através do terminal deve instalar o pacote utilizado utilizando o seguinte comando:

$ pip install ortools

No diretório do script também há a versão do código .py, para executar este arquivo basta instalar o python em sua máquina, fazer download da biblioteca ortools e executar o script através de uma IDE de sua preferência.

5. Exercício

O diagrama a seguir representa um sistema de aquedutos que se origina em três rios (nós R1, R2 e R3) e termina em uma cidade importante (nó T), onde os demais nós são pontos de junção nesse sistema.

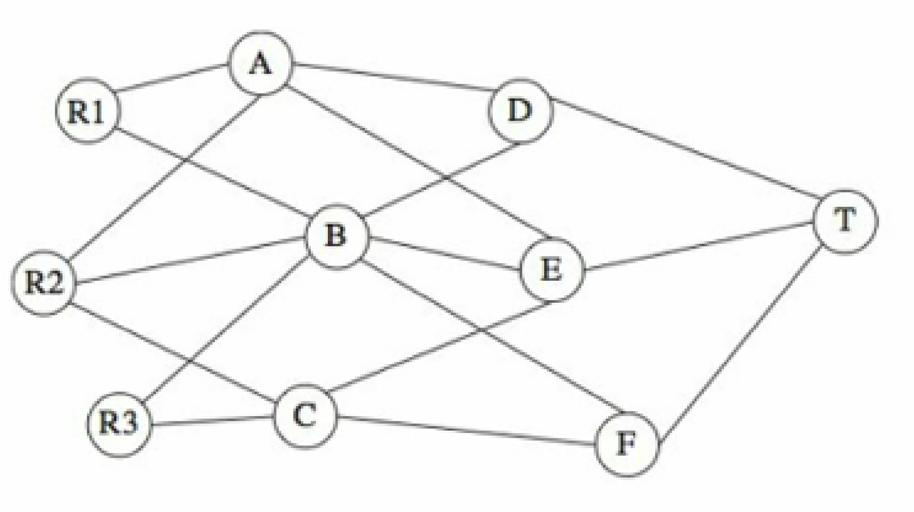

Imagem 11: Grafo do exercício da questão 9.4-3 do livro "Introdução à Pesquisa Operacional” com seus respectivos nós.

Usando unidades de milhares de pés-acre, as tabelas abaixo mostram a quantidade máxima de água que pode ser bombeada diariamente por meio de cada aqueduto.

De / Para	A	B	C	De / Para	D	E	F	De / Para	T
R1	130	115	–	A	110	85	–	D	220
R2	70	90	110	B	130	95	85	E	330
R3	–	140	120	C	–	130	160	F	240

Imagem 12: Tabela do exercício da questão 9.4-3 do livro "Introdução à Pesquisa Operacional” com a capacidade de cada arco.

(a) O gerente da companhia de águas da cidade quer estabelecer um plano de fluxo que vai maximizar o fluxo de água para a cidade.

(a) Formule esse problema como um problema do fluxo máximo identificando uma origem, um escoadouro e os nós de transbordo, e depois desenhe a rede completa que mostra a capacidade de cada arco.

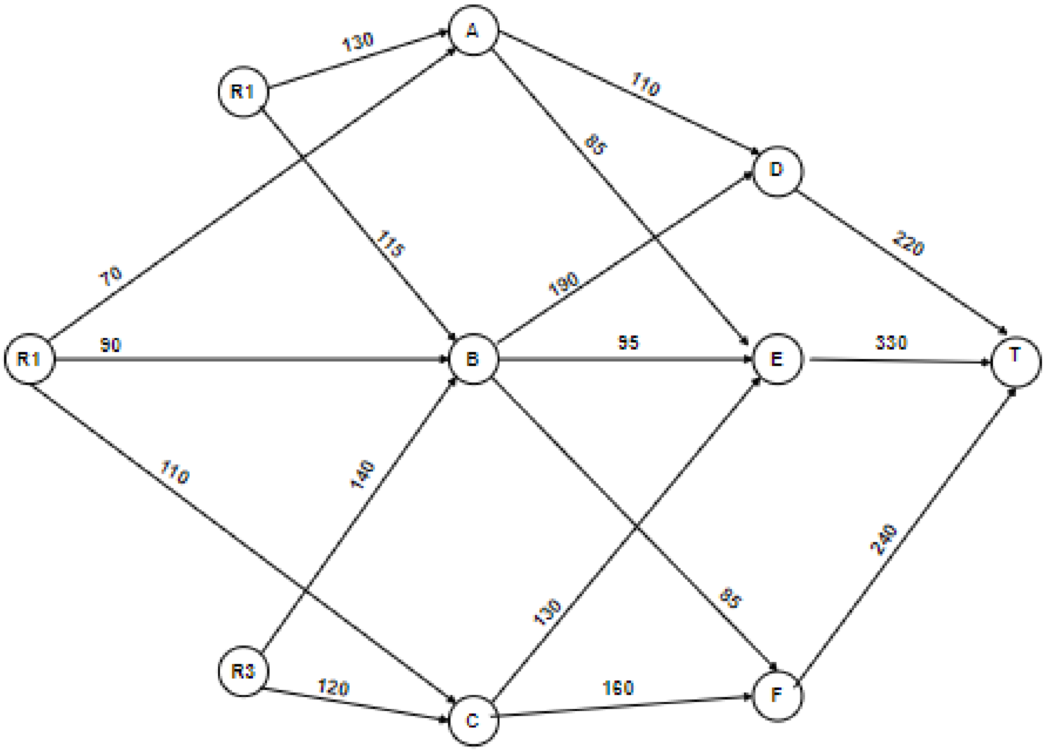

Imagem 13: Grafo exemplificativo com as capacidades máxima de cada arco e os seus nós da questão 9.4-3 do livro "Introdução à Pesquisa Operacional”.

Função Objetivo:

X_{IJ} = Quantidade de Fluxo existente entre os nós "I" e "J"

$Zmáx = X_{DT} + X_{ET} + X_{FT}$

Nós de Origem:

- *R1*
- *R2*
- *R3*

Nós de Transbordo:

- A, B, C, D, E, F.

Nó Escoadouro:

- T.

Restrição de Balanceamento

X_{IJ} = Quantidade de Fluxo existente entre os nós "I" e "J".

$Zmáx = X_{DT} + X_{ET} + X_{FT}$

Sujeito a:

Nó A: $X_{R1A} + X_{R2A} - X_{AD} - X_{AE} = 0$
Nó B: $X_{R1B} + X_{R2B} + X_{R3B} - X_{BD} - X_{BE} - X_{BF} = 0$
Nó C: $X_{R2C} + X_{R3C} - X_{CE} - X_{CF} = 0$
Nó D: $X_{AD} + X_{BD} - X_{DT} = 0$
Nó E: $X_{AE} + X_{BE} + X_{CE} - X_{ET} = 0$
Nó F: $X_{BF} + X_{CF} - X_{FT} = 0$

Restrições de Fluxo

$X_{R1A} <= 130$	$X_{BD} <= 130$
$X_{R1B} <= 115$	$X_{BE} <= 95$
$X_{R2A} <= 70$	$X_{BF} <= 85$
$X_{R2B} <= 90$	$X_{CE} <= 130$
$X_{R2C} <= 110$	$X_{CF} <= 160$
$X_{R3C} <= 120$	$X_{DT} <= 220$
$X_{AD} <= 110$	$X_{ET} <= 330$
$X_{AE} <= 85$	$X_{FT} <= 240$

Imagem 14: Modelagem do problema da questão 9.4-3 do livro "Introdução à Pesquisa Operacional".

6. Considerações finais

A partir da análise de todos os dados apresentados neste relatório foi possível alcançar os objetivos propostos e assim, verificar como funciona todos os processos para a resolução do problema do fluxo máximo convertendo-o para o custo mínimo, primeiramente através da resolução da questão 9.4-3 do livro "Introdução à Pesquisa Operacional", dos autores Hillier e Lieberman, 9ª edição, e posteriormente através da implementação do código da linguagem python. Dessa forma, o conteúdo apresentado durante as aulas pode ser posto em prática com o auxílio de pacotes de software como o OR-Tool e de bibliotecas de programação linear, o que foi essencial para discorrer o comportamento do problema aqui apresentado.

7. Referências

Google OR-Tools. Disponível em: https://developers.google.com/optimization. Acessado em 30 de abril de 2021.

Hillier, Frederick S. Introdução à pesquisa operacional [recurso eletrônico] / Frederick S. Hillier, Gerald J. Lieberman ; tradução Ariovaldo Griesi ; revisão técnica Pierre J. Ehrlich. – 9. ed. – Dados eletrônicos. – Porto Alegre : AMGH, 2013.

FEOFILOFF, Paulo. IME-USP. 2019. Disponível em: https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/flow.html. Acessado em 30 de abril de 2021.

Maximum Flows. Google OR-Tools. Disponível em: https://developers.google.com/optimization/flow/maxflow. Acessado em: 27 de abril de 2021.

Minimun Cost Flows. Google OR-Tools. Disponível em: https://developers.google.com/optimization/flow/mincostflow. Acessado em: 27 de abril de 2021.