Universidade de Brasília (UnB)
Departamento de Ciência da Computação
Pesquisa Operacional

Análise do impacto de redes intermerdiárias dinâmicas com o uso de grafos de conectividade

Danilo José Bispo Galvão

danilo.galvao@aluno.unb.br

Resumo

Esse relatório apresenta os resultados obtidos de um projeto fictício que visa avaliar o impacto em taxas de download em redes dinâmicas intermediárias, isso é, redes com alterações em suas taxas de download em nós intermediários. Para reproduzir um protocolo de rede onde os nós de origem e destino estabelecem uma conexão entre si, foi criado um problema de associação onde o custo de minimização representa o melhor custo de associação entre esses nós. Em seguida, esses nós são mapeados em dois grafos representados como matrizes de conectividade, onde seus valores são alterados em três experimentos. No problema de maximização de fluxo, o programa busca se aproximar da capacidade máxima da rede definida pelo problema anterior.

Palavras-chave: Problemas lineares; Problema de associação; Problema de maximização de fluxo.

1. Criação de uma motivação e definição de um problema

1.1. Motivação

Protocolos de rede entre nós de origem e destino estão cada vez mais robustos, mesmo em redes altamente dinâmicas. Protocolos mais modernos de QoS, por exemplo, podem garantir taxas estáveis de transferência através da diferenciação de tráfego [3]. Porém, com a alteração de capacidades de download e upload, essas taxas podem ser comprometidas. Assim, esse experimento busca compreender como essas alterações podem impactar na capacidade máxima de download/upload usando otimização linear.

1.2. Definição de problema

Para tanto, dois experimentos foram realizados: um problema de associação (*unbalanced assignment problem*) e um problema de maximização (*max-flow*). O problema de associação foi feito para definir o melhor custo de associação, ou seja, o melhor ajuste de taxas de download/upload entre os nós de origem e os nós de destino, enquanto o problema de maximização utilizará do valor de melhor associação entre dois nós para representar o limite do problema de maximização em dois grafos separadamente (representados como duas matrizes de conectividade). Ademais, os valores das matrizes de conectividade são alterados 3 vezes para representar as alterações dentro de uma rede, totalizando 6 matrizes diferentes.

2. Descrição dos modelos de otimização

Como dito anteriormente, o problema é composto de duas etapas de otimização. Portanto, dois modelos de otimização serão utilizados para expressar o resultado final.

2.1. Problema de associação

Para o problema de associação, teremos um número $M = 5$ de nós de origem (clientes) e um número $N = 6$ de nós de destino (servidores), juntamente com o vetor de custo de cada um desses dois conjuntos(VN e VM). Esses serão os nós de clientes e servidores que serão otimizados no problema de associação. Porém, a matriz de custo não foi gerada. Para gerá-la, os vetores de associação são subtraídos entre si para gerar a matriz de custo c, seguindo a seguinte fórmula:

$$\sum_{i}\sum_{j} c_{i,j} = VM_i - VN_j \tag{1}$$

De forma que a matriz c terá $M \times N$ posições ao final.

Em seguida, a matriz precisa calcular um problema de associação para minimizar os custos entre clientes e servidores. Pois a melhor associação entre esses nós será a somatória do menor valor para cada

linha da matriz c. Porém, como $M \neq N$, o problema de associação não funcionará corretamente. De forma que é adicionada uma linha extra a c com todos os valores de coluna iguais a 0, de forma que a escolha da otimização não importará. Assim, a matriz agora possui $N \times N$ linhas.

Após a criação a matriz de custo e a definição de parâmetros, o próximo passo será definir as variáveis de decisão. Portanto, uma matriz de decisão $X_{N \times N}$ estará submetida às seguintes restrições:

- cada cliente terá um servidor dentre os N, ou seja:

$$\sum_{j|(i,j)\in A} x_{ij} = 1, \ \forall i = 1, \ldots, n \tag{2}$$

- cada servidor terá um cliente dentre os M;

$$\sum_{i|(i,j)\in A} x_{ij} = 1, \ \forall j = 1, \ldots, n \tag{3}$$

- o valor de $x_{i,j}$ deve estar entre 0 e 1.

$$0 \leq x_{ij} \leq 1, \ \ \forall (i,j) \in A \tag{4}$$

A função objetivo pode ser definida como a multiplicação da matriz de decisão pela matriz de custo. Onde o objetivo é a minimização. Ou seja:

$$min(\sum_{i} \sum_{j} x_{ij} c_{ij}) \tag{5}$$

Por fim, essa matriz dará 6 resultados (um por linha, um por coluna). Porém, o resultado da última linha é ignorado. Pois o número de linhas denota o número de clientes, sendo a última linha inventada para a resolução do problema de minimização quando a matriz não é quadrada. Como o valor é uniforme para essa última linha, ela não impacta no problema de minimização e podemos assumir que ela não existe para fins de resultado.

Os outputs dessa etapa para a próxima etapa do experimento são:

- as coordenadas i, j que representam as posições dos nós cliente e servidor, respectivamente;
- o valor mínimo entre VM_i e VN_j, que define a taxa máxima de download/upload possível entre os dois nós(e.g $Min(VM_i; VN_j)$).

2.2. Problema de maximização

Antes de falar do problema em si, é importante explicar o método de geração das matrizes de conectividades. A topologia dos dois grafos(NSFNET e GEANT2) foi obtida na página[4] que continha a descrição da imagem dos grafos. Os grafos foram gerados utilizando a ferramenta online disponível em [2] e resultaram nas Figuras 1 e 2.

Os nós clientes e servidores do passo anterior foram mapeados para nós dentro do grafo através do uso de um dicionário, utilizado para mapear as entradas dos vetores de custo do primeiro problema de forma automática para os dois grafos. Os valores de dicionário serão apresentados e discutidos na seção de protótipos e resultados.

Com os grafos e os nós de clientes e servidores definidos, as matrizes de conectividade são geradas a partir de uma matriz de conectividade sem pesos, onde o loop percorre todas as posições com valor 1 e troca por valores aleatórios. Isso é feito 3 vezes seguidas para cada uma das matrizes, resultando em 6 matrizes diferentes, 3 para cada grafo.

O problema de maximização é resolvido 5 vezes por matriz, uma para cada cliente. Sendo 3 matrizes e 2 grafos diferentes, temos um total de 30 execuções. Os parâmetros de cada problema de maximização são:

- o grafo selecionado para aquela iteração (NSFNET ou GEANT2);
- a matriz de conectividade máxima $c_{i,j[t]}$ $(t = 1, \ldots, 3)$;
- nós de origem e destino (*source and target node*);

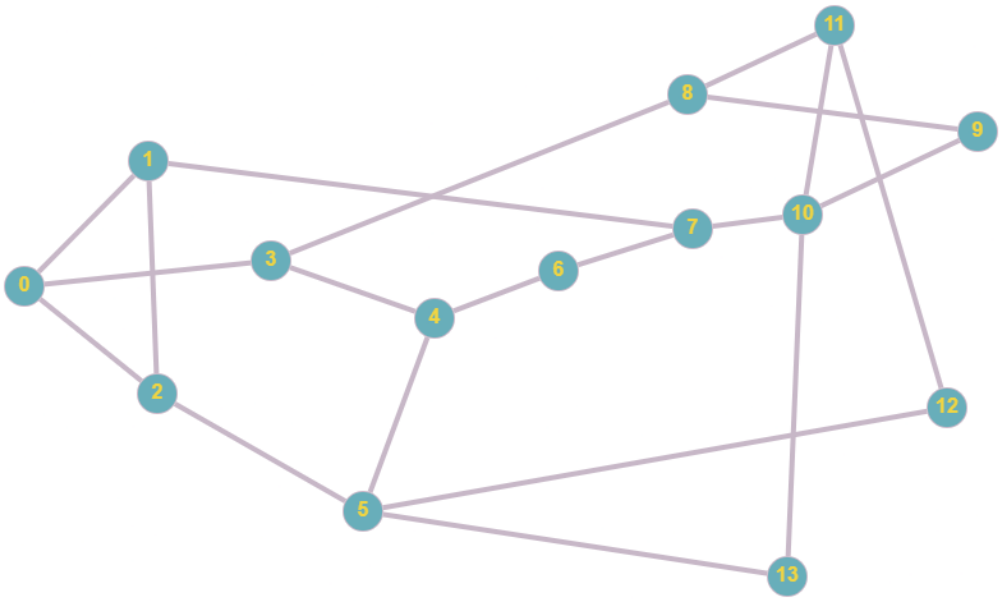

FIGURA 1 – Grafo da topologia NSFNET

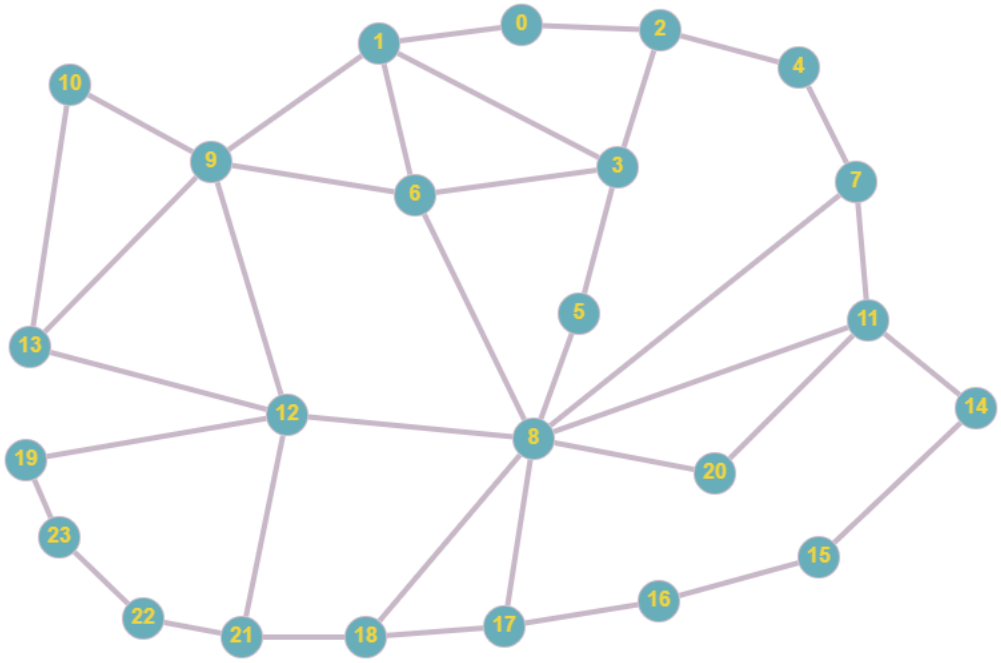

FIGURA 2 – Grafo da topologia GEANT2

- valor máximo max (obtido do $Min(VM_i; VN_j)$ para o nó;

A variável de decisão é uma matriz de decisão quadrada x_{ij} do mesmo tamanho da matriz de conectividade

As restrições para um dado problema de maximização são utilizadas para evitar ciclos na solução encontrada, utilizando restrições de divergências diferentes para os nós de origem, meio de caminho e destino. Para os nós de origem, temos que:

$$\sum_{j \in N} x_{ij} - x_{TS} = 0; \forall i = S \tag{6}$$

Os nós de destino estão sujeitos a seguinte restrição:

$$-\sum_{j \in N} x_{ij} + x_{TS} = 0; \forall i = T \tag{7}$$

Para os intermediários, temos uma outra restrição:

$$\sum_{j \in N} x_{ij} - \sum_{j \in N} x_{ji} = 0; \forall i \neq T, S \tag{8}$$

Adicionalmente, os valores de x_{ij} devem estar entre 0 e c_{ij}, assim como o valor máximo da aresta abstrata x_{TS} deve corresponder à taxa máxima de download/upload definida por max ou seja:

$$\begin{gathered} b_{ij} \leq x_{ij} \leq c_{ij} \ \forall i, j \in N \mid i \neq T \text{ e } j \neq S \\ 0 \leq x_{TS} \leq max \end{gathered} \tag{9}$$

Por fim, a função objetivo é a maximização da aresta abstrata. Assim, temos:

$$max(x_{TS}) \tag{10}$$

3. Protótipo

O projeto está disponível no GitHub[9], sendo feito em Python 3.9.6. Além do uso da biblioteca OR-Tools[8], a biblioteca Numpy[6] também foi usada por simplificar a criação de matrizes em Python.

3.1. Protótipo

O projeto possui 4 códigos-fonte em sua pasta-raiz, que compõem grande parte do código, além de pequenos fragmentos de código em outras subpastas, por critérios de modularização. O código principal se encontra em *projeto_da_disciplina.py*, que contém a resolução dos dois problemas de otimização propostos na seção anterior. Os arquivos *gerador_de_nos.py* e *gerador_de_matrizes.py*, como o nome sugere, são os geradores dos vetores de custo para nós clientes e servidores e das matrizes de conectividade para os dois grafos utilizados nesse experimento. Por fim, o código presente em *funcoes_auxiliares.py* serve para carregamento de matrizes e arquivos para estruturas de dados dentro dos programas relacionados.

Para a geração de nós, a função de gerar vetores de números inteiros aleatórios *randint* do numpy [7] foi utilizada, os valores de taxa de download/upload variam entre 1 e 20. Nenhuma restrição foi feita sobre a somatória de valores entre os dois vetores de custo serem iguais nesse experimento. Esses vetores são em seguidas salvos em dois arquivos-texto na subpasta "datasets/nodes"(*nos_clientes_array.txt* e *nos_servidores_array.txt*).

Para a geração de matrizes, um loop é feito para obter as matrizes de conectividade de dois arquivos-texto: uma para a matriz de conectividade NSFNET e outra para GEANT2. as matrizes então passam por um loop onde suas coordenadas de valor 1 (ou seja, possuem conectividade) são atribuídas valores de 1 a 5 com auxílio da função *randint*. Isso é feito para 3 matrizes separadamente, resultando em 6 matrizes diferentes, organizadas por diretório (nsfnet e geant2).

Nos dois arquivos geradores, também é possível observar o uso do código *funcoes_auxiliares.py* durante o carregamento de arquivos-texto. A organização dos arquivos foi feita de forma que os programas

geradores e o principal não tivessem definições de funções que são usadas em outros programas, isso é, um escopo global.

O arquivo principal, *projeto_da_disciplina.py*, possui a resolução do primeiro e do segundo problema de otimização descritos nas seções 2.1. e 2.2.. Para o primeiro problema, o programa realiza transformações com o auxílio do Numpy nos vetores, fazendo uma única matriz de $2 \times N$, onde uma coluna é adicionada ao fim do vetor de clientes para que ele tenha o mesmo tamanho do vetor de servidores. Em seguida, a matriz de custo é feita a partir das subtrações descritas na Fórmula 1. Finalmente, o problema de associação é resolvido com a introdução de parâmetros e variáveis de decisão.

Para o segundo problema, o maior desafio foi delinear os loops de forma correta, o loop contém 3 iterações, fora as iterações internas para estabelecer as restrições contidas variáveis de decisão, são elas:

- um loop para cada uma das topologias;
- um loop para cada uma das 3 matrizes;
- um loop para cada um dos 5 clientes e o seu problema de maximização.

O projeto possui 2 subpastas (*results* e *datasets*). Em results, um arquivo pode ser encontrado com o resultado bruto do output do programa. Os outputs avaliados na seção de resultados são dos arquivos-texto já gerados, o usuário pode criar novos vetores de custo e matrizes de conectividade, alternativamente.

4. Resultados

Essa seção será separada pelos resultados de cada etapa dos problemas de otimização

4.1. Problema de Associação

O problema de associação teve sua matriz de custo formada pelos passos descritos em 2.1.. Os vetores de custo VM e VN possuem os valores a seguir:

$$VM = \begin{bmatrix} 14 & 10 & 1 & 7 & 15 \end{bmatrix} \tag{11}$$

$$VN = \begin{bmatrix} 1 & 14 & 17 & 3 & 3 & 17 \end{bmatrix} \tag{12}$$

Após as devidas transformações, a matriz de custo virou uma matriz $C_{N \times N}$, com os valores abaixo:

$$\begin{bmatrix} 13 & 0 & 3 & 11 & 11 & 3 \\ 9 & 4 & 7 & 7 & 7 & 7 \\ 0 & 13 & 16 & 2 & 2 & 16 \\ 6 & 7 & 10 & 4 & 4 & 10 \\ 14 & 1 & 2 & 12 & 12 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{13}$$

O problema de associação retornou a matriz da variável de decisão com os seguintes valores:

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{14}$$

Por fim, os pares de nós clientes-servidores estão dispostos na Tabela 1.

Cliente	Servidor	Valor máximo de taxa de transferência
0	1	14
1	2	10
2	0	1
3	4	3
4	5	15

TABELA 1 – Resultados da associação gerada pelo OR-Tools

Diferentemente do problema de maximização, o primeiro possui um processo simplificado que permite

a visualização dos resultados com mais clareza. Com o cálculo dos valores e da matriz de custo, é possível enxergar que o programa de fato conseguiu alcançar a minimização.

Outro ponto importante é que como esse problema é um problema de associação desbalanceado [1], a última linha do resultado pode ser simplesmente descartada, pois representa um cliente inexistente para o problema, de forma que o servidor 3 não fará parte da solução do problema de otimização do *max-flow*.

Adicionalmente, foi feito um arquivo-texto separado do output dos resultados desse primeiro problema, estando na pasta de resultados com o nome de arquivo *resultados_problema_associacao.txt*.

4.2. Problema de maximização

Os dicionários mencionados na Seção 2.2. fazem um mapeamento de nós clientes e servidores para um dos grafos, nas Tabelas 2 e 3 estão listados os mapeamentos adotados, seguindo a numeração das Figuras 1 e 2.

Grafo NSFNET			
Nó de origem	**Nó de origem no grafo**	**Nó de destino**	**Nó de destino no gráfico**
0	0	0	10
1	1	1	11
2	2	2	9
3	3	4	7
4	5	5	13

TABELA 2 – Mapeamento de nós clientes e servidores para o grafo NSFNET

Grafo GEANT2			
Nó de origem	**Nó de origem no grafo**	**Nó de destino**	**Nó de destino no gráfico**
0	0	0	19
1	1	1	23
2	2	2	22
3	4	4	18
4	7	5	17

TABELA 3 – Mapeamento de nós clientes e servidores para o grafo GEANT2

O problema de maximização foi executado 30 vezes separadamente, sendo inviável colocar cada matriz que foi utilizada nesse relatório. Porém, o output também está disponível em arquivos separados. Cada cliente obteve 6 resultados separadamente. Os resultados podem ser vistos no diretório do GitHub[5], ordenados pelo número do cliente (0 a 4). No relatório, serão investigados os maiores valores obtidos na maximização para cada caso em ambos os grafos.

4.3. Resultados do grafo NFSNET

O grafo NSFNET é notoriamente menor, uma das hipóteses levantadas seria que como cada um dos nós de conectividade possuem um valor de 1 a 5, seria mais difícil aumentar muito a taxa de transferência, uma vez que os nós intermediários forneciam pouca taxa de transferência. Logo, na análise dos nós clientes, é importante levar em conta o valor máximo possível na terceira coluna da Tabela 2.

Para o cliente 0, o maior valor obtido da função objetivo é 7, sendo que 14 era o valor máximo possível, isso pode ser justificado pelos poucos nós, como ciclos não são permitidos, o valor máximo não pôde ser alcançado. De forma similar, o cliente 1 também não conseguiu obter o seu valor máximo 10, mas conseguiu 8 na mesma matriz 0 do cliente 0. Isso demonstra que os valores da primeira matriz também são maiores.

Avançando para o cliente 2, sua maximização foi ótima em ambos os grafos, pois o valor de maximização desejado é 1, tornando sua conectividade trivial, o único caso em que isso não aconteceria seria se a matriz não tivesse caminho entre os nós, evidentemente, esse cenário não acontece. O cliente 3 possui valor de maximização 3 e também obteve a maximização ótima em todas as matrizes de todos os grafos.

Para o cliente 4, o melhor resultado para o primeiro grafo foi 5 nas matrizes 1 e 2. Porém, seu valor máximo da função objetivo é 15, isso pode ser atribuído ao fato do nó 5 estar posicionado em uma borda

afastada dos nós intermediários, deixando poucos nós para que ele possa alcançar o valor desejado pela função. Além disso, alguns nós estão simplesmente inacessíveis devido às restrições de meio de caminho que previnem ciclos para alguns nós.

4.4. Resultados do grafo GEANT2

O grafo GEANT2 é um pouco maior, possuindo mais nós de conectividade (24 ao total). No entanto, a implementação adotada nesse experimento impediu o uso de ciclos para a conectividade entre nós, o que pode ter impactado negativamente para muitos dos clientes, pois eles ficam incapacitados de utilizar alguns nós extras em suas soluções. O cliente 0 já evidencia esse impacto: seu melhor desempenho foi na matriz 2 (a última) de conectividade, porém, o valor obtido pela maximização foi 6, menos que a metade. Em contraste, o cliente 1 conseguiu obter 70% do seu desempenho nas matrizes 0 e 2 com valor 7 de resultado, mas também obteve desempenho menor que no primeiro grafo.O cliente 2, como comentado anteriormente, atingiu com folga a sua maximização, tendo como valor máximo 1, mesmo com a pouca disponibilidade de nós de conectividade. Isso ocorreu novamente em todas as matrizes.

O cliente 3 também apresentou valor máximo, como comentado anteriormente. Surpreendentemente, o cliente 4 apresentou os melhores resultados dessa etapa, com o valor de maximização 8 na matriz 1 do grafo, seu desempenho foi o melhor dentre todas as observações, isso pode ser atribuído ao fato de seu nó de origem no grafo ser mais afastado dos outros nós de origem e também por precisar percorrer mais nós durante o problema de maximização para chegar ao destino.

Uma das observações mais claras do experimento é que os nós de conectividade possuíam valores notoriamente baixos para a taxa máxima de transferência, nenhuma das funções objetivo com valor máximo maior ou igual a 10 chegou a esse valor. Uma das possíveis alternativas seria aumentar o valor de cada nó de conectividade ou aumentar o tamanho do grafo, de forma que o valor máximo poderia ser alcançado mesmo sem ciclos.

5. Considerações Finais e Conclusão

O problema de associação foi bastante interessante pois eles estão presentes em muitos contextos, a montagem de uma matriz de custo a partir de dois grupos de interesse também evidenciou as diferentes aplicações desse tipo de problema. Um fator que impediu que a maximização obtivesse valores máximos durante a segunda parte do experimento foi a que os grafos utilizados como exemplo também possuíam muita interconectividade, mas baixo valor em seus nós e restrições de ciclo. Evidenciando como é importante considerar sempre a rede e todas as outras características relevantes quando se está modelando um problema de fluxo máximo.

Com o aumento de complexidade e dinamicidade das redes atuais, é difícil estimar valores de taxa de transferência somente com otimizadores lineares. No entanto, se os problemas forem corretamente modelados, muitas soluções devem agregar ferramentas de otimização linear como o OR-Tools para suporte a essas soluções, pois elas ajudam a simplificar a tomada de decisão e a identificação de pontos fortes e fracos em uma rede. Sendo uma ferramenta que merecia mais atenção de diversos especialistas em variados campos de aplicação.

6. Referências

Referências

[1] *Assignment Problem: Meaning, Methods and Variations | Operations Research*. URL: https://www.engineeringenotes.com/project-management-2/operations-research/assignment-problem-meaning-methods-and-variations-operations-research/15652 (acesso em 31/10/2021).

[2] *Create Graph online*. URL: https://graphonline.ru/en/ (acesso em 30/10/2021).

[3] Djamel Djenouri e Ilangko Balasingham. “Traffic-Differentiation-Based Modular QoS Localized Routing for Wireless Sensor Networks”. Em: *IEEE Transactions on Mobile Computing* 10.6 (2011), pp. 797–809. DOI: 10.1109/TMC.2010.212.

[4] *Knowledge-Defined Networking Training Datasets*. URL: https://knowledgedefinednetworking.org/ (acesso em 30/10/2021).

[5] *maxflow at master danilobispo/pesquisaOperacional_Projeto_da_Disciplina*. URL: https://github.com/danilobispo/pesquisaOperacional_Projeto_da_Disciplina/tree/master/projetoDaDisciplina/resultados/maxflow (acesso em 31/10/2021).

[6] *Numpy - Overview*. URL: https://numpy.org/doc/stable/index.html (acesso em 30/10/2021).

[7] *Numpy - Randint*. URL: https://numpy.org/doc/stable/reference/random/generated/numpy.random.randint.html?highlight=randint#numpy.random.randint (acesso em 30/10/2021).

[8] *OR-Tools Main Page*. URL: https://developers.google.com/optimization (acesso em 30/10/2021).

[9] *Página do projeto no GitHub*. URL: https://github.com/danilobispo/pesquisaOperacional_Projeto_da_Disciplina (acesso em 30/10/2021).