Modelo de Gestão de uma Companhia Aérea

Gabriel Estácio de Souza Passos

*Instituto Metrópole Digital - IMD / Universidade Federal do Rio*

*Grande do Norte - UFRN*

**Resumo**

Este relatório tem por objetivo descrever um problema e a aplicação de grafos como solução, sendo este problema a gestão de linhas aéreas de uma companhia aérea localizada em Natal, Rio Grande do Norte.

*Palavras-chave:* grafos, gestão, companhia aérea



1. **Introdução**

A construção de uma malha aérea não é uma tarefa simples. Envolve muitos estudos, muitas análises e diversas observações para tomar decisões estratégicas sobre logísticas que auxiliem o bom funcionamento da empresa e seu lucro.

Uma abordagem possível para a análise de melhores trajetos para vôos é utilizando grafos, mais especificamente, grafos ponderados, que levem em consideração fatores importantíssimos para o desenvolvimento econômico da companhia, a fim de que ela também ofereça ótimos serviços.

A partir disso, apresento esse estudo de caso, uma simulação de construção da malha aérea de uma companhia fictícia, sediada em Natal e que está buscando ter uma cobertura nacional, para se tornar referência em voos domésticos.

1. **Descrição do problema real**

O espaço aéreo brasileiro é bastante restrito a algumas companhias específicas para voos domésticos. Considerando um contexto onde uma nova companhia está sendo criada para adentrar neste mercado, é necessário um estudo estratégico e logístico que possibilite à empresa definir as rotas das linhas áereas que irá operar.

Para isso, é importante levar em consideração alguns fatores: a distância entre dois aeroportos que constituem uma linha aérea, o custo de operação envolvido e, principalmente, a demanda de passagens para aquela linha aérea. A partir destes fatores, a proposta é utilizar grafos ponderados para auxiliar na definição de rotas vantajosas para a companhia.

1. **Abstrações necessárias**

Dada a complexidade do problema e de desconhecimentos técnicos profundos em relação à logística e aviação, além do tempo hábil para realização do projeto, algumas abstrações serão necessárias:

1. A companhia possui uma quantidade específica de aeronaves e todas são do mesmo modelo. Por conta da variedade de informações disponíveis online, consideraremos todas as aeronaves do modelo Boeing 737-800, uma aeronave tradicional no meio da viação e muito utilizada no Brasil para voos domésticos;
2. O tempo de decolagem e aterrissagem em todos os voos é padrão, assim como o gasto de combustível nos dois processos. Essa abstração é necessária devida a complexidade de se fazer medições precisas desses processos, não sendo essa a finalidade desta atividade, mas a aplicação de grafos para solução de um problema;
3. Todas as aeronaves partem, desde o início da sua circulação, com o tanque de combustível cheio. Logo, o valor de custo com combustível considerado aqui é para repor o combustível gasto na viagem;
4. O tempo de voo apresentado é o tempo de voo de cruzeiro, adicionando o tempo padrão de 20 minutos para aterrissagem e decolagem;
5. O gasto envolvendo o staff de cada voo envolve apenas o tempo efetivo do trajeto, desconsiderando o tempo de preparação da aeronave, de revista, manutenção, recepção, entre outros. Portanto, só foram considerados gastos envolvendo 1 comandante, 1 co-piloto e 3 comissários de bordo (equipe necessária para atender uma aeronave do modelo Boeing 737-800). Apenas a fim de registro, não há voos entre capitais, partindo de Natal/RN, que ultrapassem 6h seguidas, sendo desnecessário levar em consideração troca de pessoal, horas extras, entre outros, no cálculo de custos de pessoal;
6. Por fim, em casos de rotas com conexões/escalas, é desconsiderado o tempo de transição entre uma aeronave e outra. Além disso, não consideraremos que há trajetos onde duas companhias diferentes se unem para realizar uma linha aérea. Considere que todas as linhas são cobertas pela nossa companhia do início ao fim. Além disso, nenhuma rota faz mais de duas paradas (escala ou conexão). Ultrapassar esse limite aumentaria consideravelmente os custos de deslocamento, além de ser desconfortável para o cliente da companhia aérea.
7. **Algumas informações importantes**

Esta é uma listagem de todos os fatores levados em consideração no cálculo dos pesos de cada aresta do grafo ponderado:

1. Quanto à distância, foi realizado um levantamento da distância aproximada entre cada parada de cada trajeto. Portanto, num trajeto Natal -> Brasília, por exemplo, seguindo uma rota Natal -> Recife -> Salvador -> Brasília, foi verificada a distância entre Natal e Recife, entre Recife e Salvador, e entre Salvador e Brasília.
2. Quanto ao custo de operações de um trajeto, foi feito um cálculo que soma valores do custo de pagamento da equipe e do gasto com combustível.
   1. Para o custo de pagamento da equipe, foi calculado o tempo total de vôo, calculado através da fórmula:

(km/850)\*60+20,

onde 850 corresponde a velocidade média da aeronave em km/h, o produto por 60 realiza a conversão para minutos e a soma de 20 adiciona o tempo padrão de decolagem e aterrissagem ao tempo total do voo. O resultado desse cálculo é dado em minutos, e o resultado final é feito numa conversão simples de minutos para horas.

O custo com staff envolve apenas o valor de trabalho por horas, que, somando toda a equipe, dá R$247,50/h, ou, aproximadamente, R$4,13/min. Para o valor total de cada trajeto, foi feito o cálculo do produto do tempo de voo em minutos pelo custo do minuto trabalhado pela equipe.

* 1. Além disso, Já os custos com combustível envolvem alguns fatores: o tipo de combustível (querosene para aviação (QAV)), fixado num valor de R$4,69/litro (valor registrado no último mês de abril); e o consumo de combustível da aeronave, que é de, aproximadamente, 3,6L/km. Para o valor total de cada trajeto, foi feito o cálculo do produto da distância em quilômetros pelo consumo de combustível da aeronave, resultando no total de litros gastos na rota. Após isso, foi realizado um novo produto entre o total de litros gastos pelo preço atual de cada litro do QAV, resultando, por fim, no valor total de gasto com combustível.

A soma dos resultados desses dois cálculos é o custo total de operação em um trajeto, e foi calculado de forma direta, através da fórmula:

(km/850)\*60+20 + (km\*3,6)\*4,64

1. Quanto à demanda de passagens:, não foram encontradas muitas informações concretas tendo Natal como partida, portanto, a determinação dos níveis de demanda aqui são arbitrárias e não correspondem à realidade. Para organizar melhor essa informação, foram determinados 6 níveis de demanda, onde cada capital possui uma classificação específica em relação à Natal, sendo:

1: Demanda alta

2: Demanda média/alta

3: Demanda média

4: Demanda média/baixa

5: Demanda baixa

6: Demanda baixíssima

Portanto, o peso de cada aresta será definido pela seguinte fórmula:

((distância\*100) + custo total)/(10000) + nível de demanda

A multiplicação da distância por 100 é usada para trazer o valor da distância mais próximo para a casa do custo total, que é de dezenas de milhares, em média (por isso, a divisão por 10000). Isso resultaria num valor mais "afetado" pelo nível de demanda, possibilitando que este último parâmetro tivesse também impacto no peso de uma aresta.

1. **Modelagem em grafos**

Antes de falarmos de soluções algorítmicas, apresento aqui uma simulação mais “manual” do processo de determinação das rotas. Nessa simulação, a forma de pesar as arestas é diferente: elas são pesadas dentro de um intervalo de 1 a 28, em forma decrescente, através de uma análise humana dos fatores nível de demanda, custos de deslocamento e distância, nessa ordem de prioridade.

Todas as informações resultantes dos cálculos dessa simulação estão listadas na planilha presente no seguinte link: [Análise de Caso - Gestão de Companhia Aéreas - Grafos 2022.1](https://docs.google.com/spreadsheets/d/1rb_vkxT0aZ-00NwzS2UABL2NatOGWxms1nYwBWGCoF4/edit?usp=sharing)

A modelagem do problema utilizando grafos ponderados ficou da seguinte maneira:



**Figura 5.1 (Clique** [**aqui**](https://drive.google.com/file/d/1U3V-Oc4qNtqslvW-fkxqj0nFIqTffDWt/view?usp=sharing) **para vê-la em um tamanho maior):** A ordem dos pesos é decrescente.

A representação gráfica de uma malha aérea, como a representada na figura 5.1, possui o formato de um grafo. Na aplicação do caso escolhido, que é de gestão de linhas aéreas, é interessante usar grafos ponderados para guiar a tomada de decisões de acordo com os pesos das arestas, podendo assim determinar qual linha deve ser priorizada, qual traz mais lucros para a companhia, qual representa mais gastos, quais são os melhores trajetos para que nenhum lugar fique sem cobertura da companhia, entre outras aplicações.

Para efeito de exemplo, fica o modelo do grafo que representa as linhas aéreas caso todas as capitais tivessem um voo direto de Natal:



**Figura 5.2:** Representação da malha aérea caso todas as capitais possuíssem voos diretos de Natal.

Como é possível perceber, caso essa fosse a realidade, o espaço aéreo de Natal seria um completo caos, desconsiderando ainda os voos das outras companhias aéreas, fora os custos envolvidos, os horários de operação da empresa, que seriam muito prejudicados, o próprio uso do modelo específico de aeronave que foi selecionado, entre outros problemas.

Abaixo, algumas visualizações interessantes do grafo resultante do nosso problema:



**Figura 5.3:** Grafo subjacente do grafo ponderado utilizado na resolução do problema.



**Figura 5.4:** Vértices de maior grau.

Esses vértices de maior grau indicam capitais chaves para o bom funcionamento das rotas escolhidas pela companhia aérea. Este caso inclui Manaus, Belém, Recife e Brasília, além, claro, de Natal.



**Figura 5.5:** Ciclo Natal-João Pessoa-Recife

A presença do ciclo Natal-João Pessoa-Recife, causado pela política da companhia de apenas realizar rotas com, no máximo, duas paradas, pode acarretar na necessidade de uma decisão estratégica a curto prazo. Um questionamento que pode surgir é: está valendo a pena manter duas rotas com mesmo destino? Será que vale a pena, para quem faz o voo Natal-Recife com escala em João Pessoa, perder tempo com uma parada num voo tão curto?



**Figura 5.6:** Grafo que representa algumas possibilidades de rotas entre Natal e Curitiba.

Através da imagem acima, podemos perceber que, por mais que existam muitas possibilidades de trajeto entre duas capitais, poucas delas serão realmente efetivas. A um primeiro olhar, qualquer caminho parece bom, desde que se chegue no destino. Porém, quando se leva em consideração os fatores usados nesse caso de estudo para determinar uma rota viável (demanda, distância e custo), nós podemos observar que não só temos poucas rotas viáveis, mas também uma rota mais efetiva, que, neste caso, é um voo direto Natal/Curitiba.

Todas as imagens presentes nessa seção e ainda outras representações podem ser acessadas para melhor visualização através [deste link](https://drive.google.com/drive/folders/1xBE5GgvBmqNmLyJg9BxjlwQ-D2KFnhOh?usp=sharing).

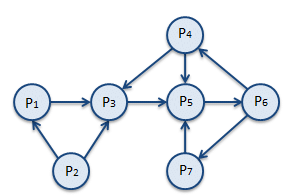
1. **Estado-da-arte**

Apesar da literatura para este problema em específico não ser muito vasta, foi possível encontrar algumas abordagens diferentes para a construção e otimização de malhas aéreas, e alguns problemas relacionados muito interessantes.

**6.1: Encontrando possíveis rotas com matrizes**

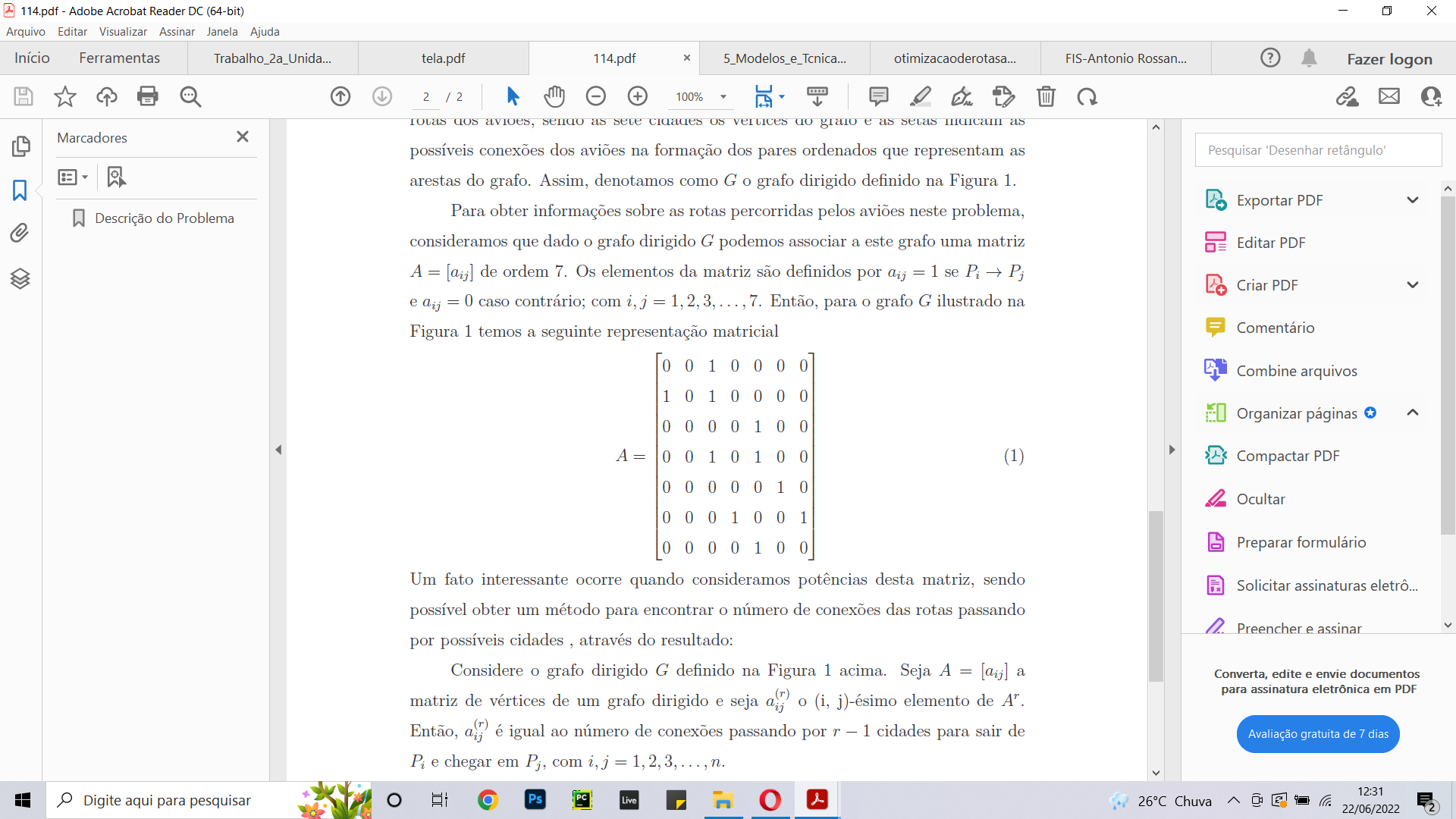
A proposta deste artigo é abordar a utilização de matrizes para representar grafos. No caso específico, utilizar matrizes para representar as possíveis rotas entre aeroportos de uma malha aérea. Apesar desse problema ser diferente do abordado aqui, que pretende encontrar os menores caminhos, como a análise de possíveis rotas é parte do problema, é interessante trazer essa abordagem.

A descrição do problema abordado no artigo é a seguinte: “Uma pequena companhia aérea atende sete cidades, digamos P1; P2; P3; P4; P5; P6; e P7. O proprietário da companhia aérea tem interesse em estabelecer a melhor rota para os aviões, de modo a obter o menor tempo de viagem, gastando a menor quantidade possível de combustível.”



Grafo G = (N,M). Direitos de Imagem: autores do artigo [7]

Para termos as informações das possíveis rotas dos aviões, é associado ao grafo G acima uma matriz A7x7 = aij, onde aij = 1 se existe Pi-> Pj, e aij = 0 caso contrário. Portanto, considerando G, teremos como resultado a seguinte matriz:



Direitos de Imagem: autores do artigo [7]

A partir dessa matriz, podemos obter informações valiosas, como o número de conexões possíveis entre cidades, utilizando potências de matrizes. Considere aij(r) o (i,j)-ésimo elemento de Ar. Então, aij(r), segundo os próprios autores, “é igual ao número de conexões passando por r - 1 cidades para sair de Pi e chegar em Pj, com i,j = 1, 2, 3, …, n”.

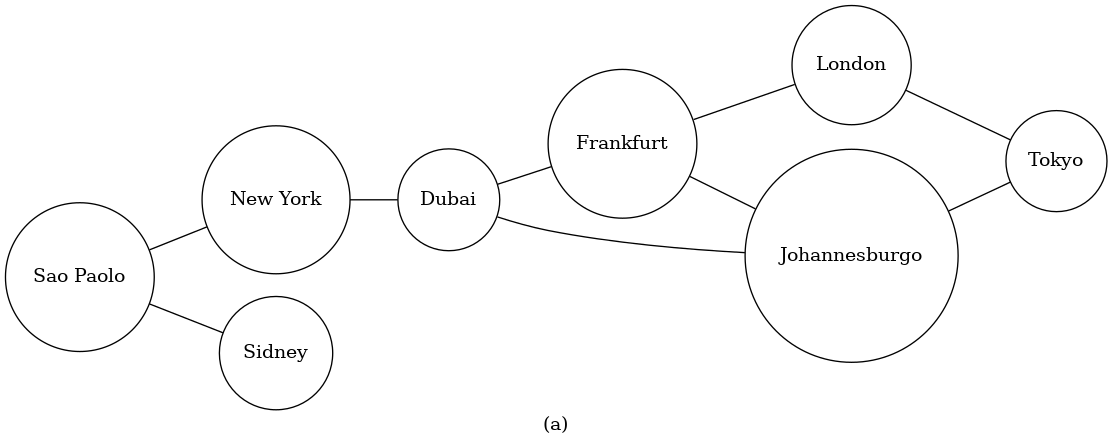
Todas as imagens estão disponíveis para melhor visualização através [deste link](https://drive.google.com/drive/folders/1xBE5GgvBmqNmLyJg9BxjlwQ-D2KFnhOh?usp=sharing). Além disso, o link para este artigo está referenciado na seção de Referências, com índice [7].

**6.2: Caminho mais curto com busca em largura e árvore de extensão**

A primeira abordagem encontrada é como um problema de caminho mais curto. No caso específico do artigo estudado, o problema é considerado para uma malha aérea global, desconsiderando que existam vôos diretos entre todas as cidades, ou seja, não é possível sair de qualquer aeroporto e chegar ao destino sem passar por escalas. Isso não é, no entanto, obrigatório: há sim casos de vôos diretos sem escalas.

Outras abstrações necessárias para essa abordagem são: o preço é único para um destino, independente da existência de escalas na rota e da quantidade de paradas; não há tempo entre escalas; os voos sempre respeitam os horários marcados.

Dito isso, a pergunta que move esse problema é: qual o menor caminho entre um aeroporta A e um aeroporto B com menos escalas, considerando a malha aérea abaixo:

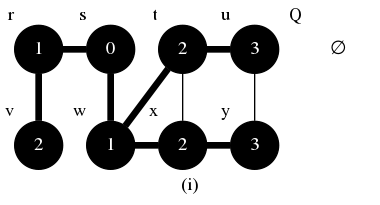


Direitos de imagem: Eduardo Souza, autor do artigo [8].

Para resolver essa questão algoritmicamente, o autor utiliza a ideia da busca em largura e árvore de extensão, com código descrito [neste documento](https://drive.google.com/file/d/1UNwGol8Yhrxt3dUQ8Au6IKRk4d-k_xHh/view?usp=sharing).

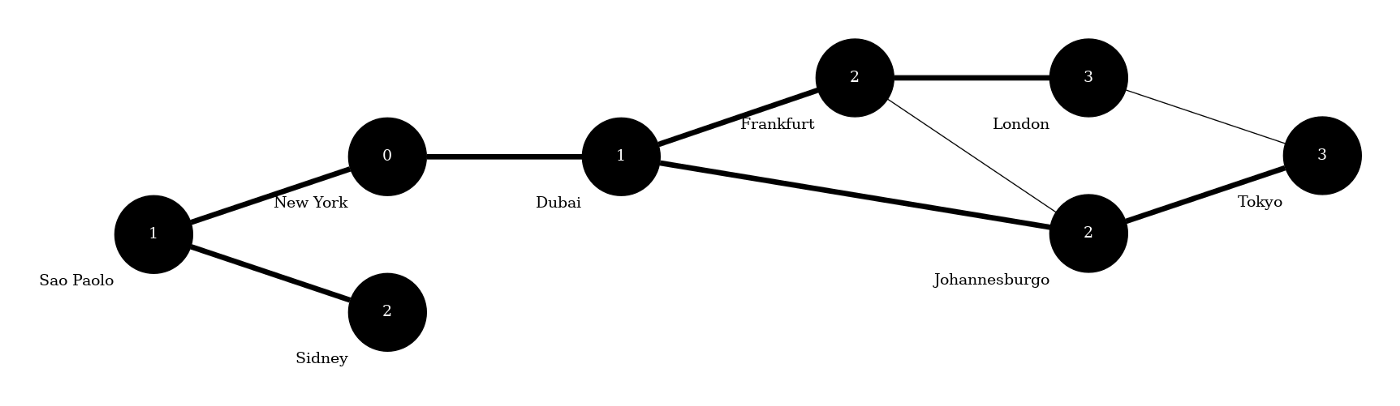
O algoritmo funciona recursivamente, partindo do destino, encontrando os nós ancestrais de cada ancestral do destino, até que o nó ancestral encontrado seja a origem. Há também a possibilidade de se encontrar nós sem ancestral e diferentes da origem, o que indica que não há caminho possível entre a origem e o destino.

O grafo abaixo mostra uma resolução através da aplicação do algoritmo, onde o caminho mais curto entre os vértices s e u é s - w - t - u:



Direitos de imagem: Eduardo Souza, autor do artigo [8].

Aplicando este algoritmo ao contexto do exemplo, o grafo resultante seria:



Direitos de imagem: Eduardo Souza, autor do artigo [8].

Nesse caso, as arestas correspondem aos voos entre duas cidades e o número dentro dos vértices corresponde à menor distância entre a cidade e a origem (considerada como Nova York no exemplo).

Todas as imagens estão disponíveis para melhor visualização através [deste link](https://drive.google.com/drive/folders/1xBE5GgvBmqNmLyJg9BxjlwQ-D2KFnhOh?usp=sharing). Além disso, o link para este artigo está referenciado na seção de Referências, com índice [8].

**6.3: Utilizando redes e matrizes para análise de uma malha aérea real**

A proposta deste artigo é de estudar “características estruturais e dinâmicas de redes aéreas, (...) visando diminuir sua vulnerabilidade e otimizar a eficiência dos fluxos. Os dados utilizados neste estudo são dados de voos domésticos dos Estados Unidos, de diversas companhias aéreas, fornecidos pela Research and Innovate Technology Administration, do Departamento de Transportes dos Estados Unidos”, nas palavras dos próprios autores.

Para representar a existência de uma rede aérea, pode-se utilizar matrizes de adjacência, para expressar a conexão entre vértices, e a matriz de pesos, para expressar a intensidade dessas conexões. Para caracterizar esta rede, são utilizados parâmetros como a conectividade e o caminho mais curto entre um par de vértices.

Um dos objetivos desse artigo também é visualizar o impacto do tamanho da companhia no uso da malha aérea.

Considere uma matriz de adjacência A= aij, onde aij = 1 se existe i -> j, e aij = 0 caso contrário. Essa matriz será útil para determinar a conectividade dos vértices.

A outra medida relevante de uma rede é a proximidade dos nós, que não está relacionada necessariamente à distância geográfica, mas a distância percorrida nas arestas. Uma aplicação possível para resolver isso é a solução que utilizarei pra resolver o problema proposto no trabalho: o algoritmo de Dijkstra.

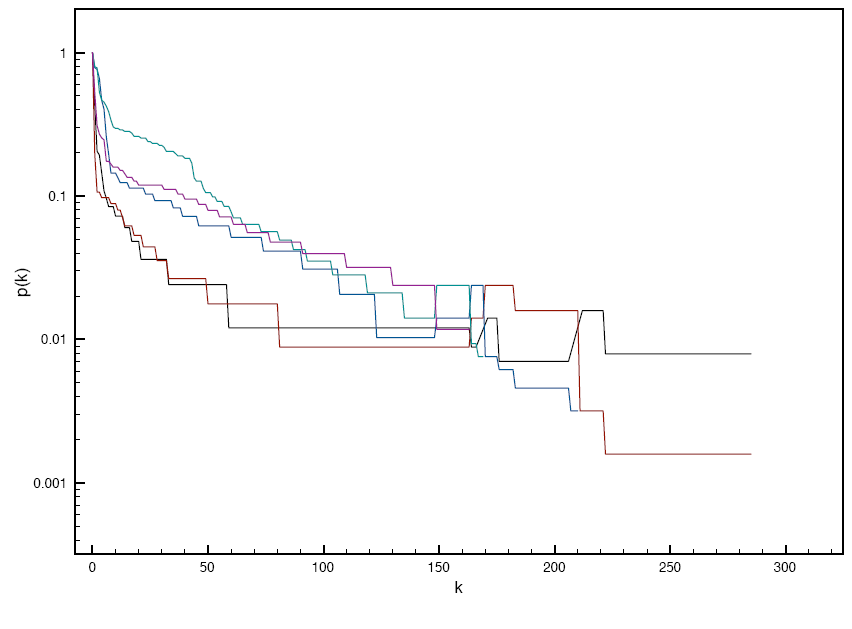
Alguns resultados do estudo são relatados abaixo.

* Crescimentos relacionados aos aeroportos:

Houve um suave aumento no número de aeroportos operantes, principalmente no ano de 1995. Também houve um aumento no número de voos por aeroportos por mês.

* Distribuição de conectividade:

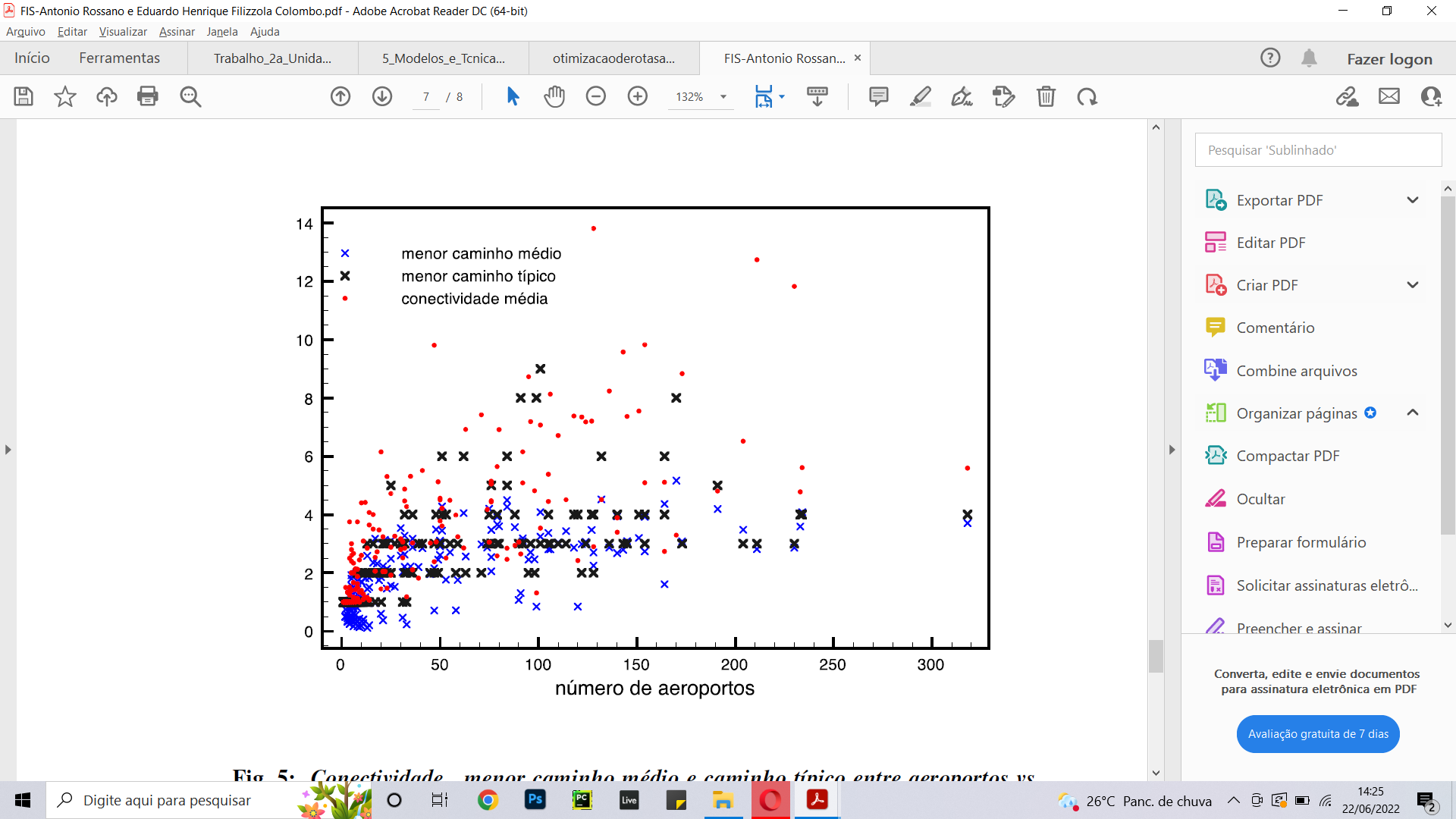
Observando a conectividade das 5 maiores companhias aéreas, foi possível concluir que as 3 maiores companhias apresentaram uma distribuição exponencial, enquanto a 4ºe a 5º apresentam distribuições próximas à uma “lei de potência”, conforme o gráfico abaixo. (k representa a conectividade e p(k) pode ser definida como a “distribuição de probabilidade de um vértice escolhido aleatoriamente conectividade k”).



Direitos de Imagem: autores do artigo [9]

* Distribuição de menores caminhos:

Para todas as companhias, existe um limite de caminhos mais curtos bem definido. A partir do gráfico abaixo, é possível inferir que a conectividade, caminhos mais curtos entre os caminhos médios e o caminho típico entre os aeroportos cresce à medida que o número de aeroportos operantes da companhia também cresce. Isso mostra que redes com mais aeroportos possuem uma estrutura que possibilita a criação de aglomerados, onde, segundo os autores, “voos até o destino final efetuam uma única escala”.



Direitos de Imagem: autores do artigo [9]

Através destes resultados é possível concluir que o tamanho de uma companhia implica no aumento do caminho mais curto e da conectividade. De forma geral, companhias maiores criam centros de conexão, ao passo que prejudicam a coesão da rede.

Todas as imagens estão disponíveis para melhor visualização através [deste link](https://drive.google.com/drive/folders/1xBE5GgvBmqNmLyJg9BxjlwQ-D2KFnhOh?usp=sharing). Além disso, o link para este artigo está referenciado na seção de Referências, com índice [9].

**6.4: Otimização de rotas aéreas utilizando o Problema do Caixeiro Viajante**

A proposta deste artigo é, segundo os próprios autores, “otimizar rotas aéreas de ponto a ponto (aeroportos), para um melhor aproveitamento da malha aérea das companhias, concebendo assim uma maior lucratividade pela empresa e consequentemente um serviço mais prático e barato para os usuários. Para isso, foram demonstradas rotas através do Problema do Caixeiro Viajante (PCV).”

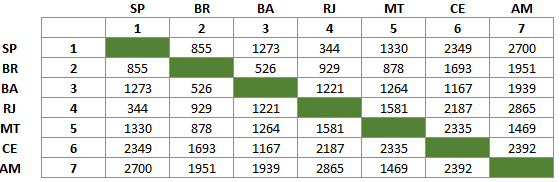
A forma utilizada para solucionar o problema foi o Microsoft Excel, juntamente com a ferramenta Solver, utilizada para resolução de rotas através do PCV. A ideia é encontrar a menor rota possível (caminho mais curto) com menor custo benefício para a empresa e seus clientes.

Antes de partirmos para o problema em si, algumas definições são necessárias:

* Malha Aérea: conjunto de rotas realizadas por empresas de aviação diversas em períodos regulares;
* Problema do Caixeiro Viajante (PCV): problema logístico e matemático cujo objetivo é identificar caminhos mais curtos em caminhos fechados.

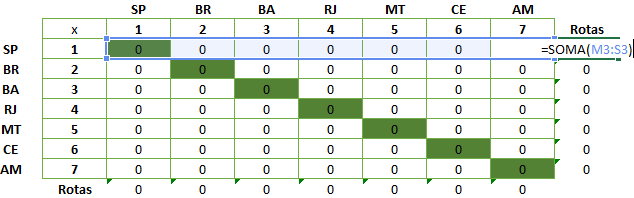
Para exemplificar a solução do problema, a rota que queremos otimizar tem origem em São Paulo e passa por aeroportos em Mato Grosso, Rio de Janeiro, Brasília, Bahia, Ceará e Amazonas, não necessariamente nessa ordem, antes de retornar para São Paulo.

Para resolver o problema, foram criadas quatro matrizes: a primeira matriz armazenará os valores das distâncias entre os 7 aeroportos.



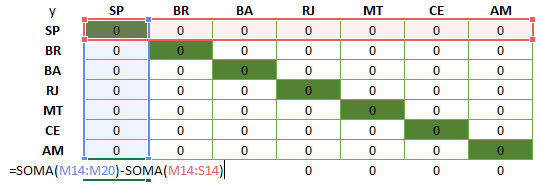
Direitos de Imagem: autores do artigo [10]

Na segunda matriz, foram aplicadas tanto fórmulas quanto restrições na ferramenta Solver que forçassem o sistema a passar por somente uma rota em cada linha e coluna da matriz. Isso foi feito através da soma de linhas e colunas e de restrições que fizessem 1) com que todas as rotas, tanto as verticais quanto as horizontais, fossem igual a 1 onde a rota já tinha passado; e 2) que as células da Matriz 2 (vértice x) apresentassem apenas valores binários; fazendo assim com que uma rota jamais fosse repetida. A aplicação da fórmula de soma nas linhas da matriz está abaixo. Perceba que, como há a restrição de que todos os valores da matriz sejam binários, o resultado da soma na célula “Rota” também será binário (1 para caso a rota já tenha passado por aquele aeroporto e 0 caso contrário).



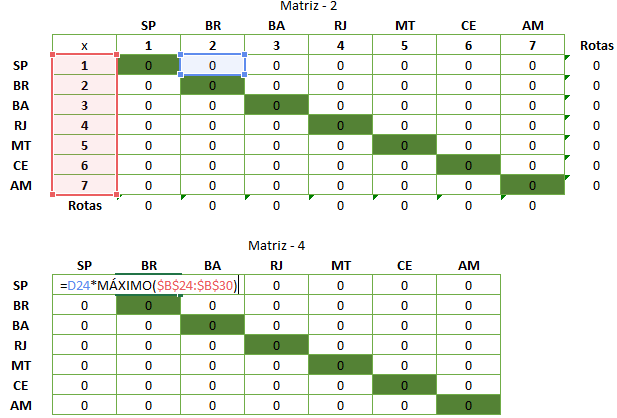
Direitos de Imagem: autores do artigo [10]

A terceira matriz (vértice y) define qual caminho deve ser percorrido saindo de São Paulo, percorrendo todos os aeroportos, e retornando para São Paulo, atendendo os critérios do PCV. Nessa matriz, foi aplicada a soma da coluna SP e subtração da linha SP para definir o valor da célula “Rotas” na coluna SP:



Direitos de Imagem: autores do artigo [10]

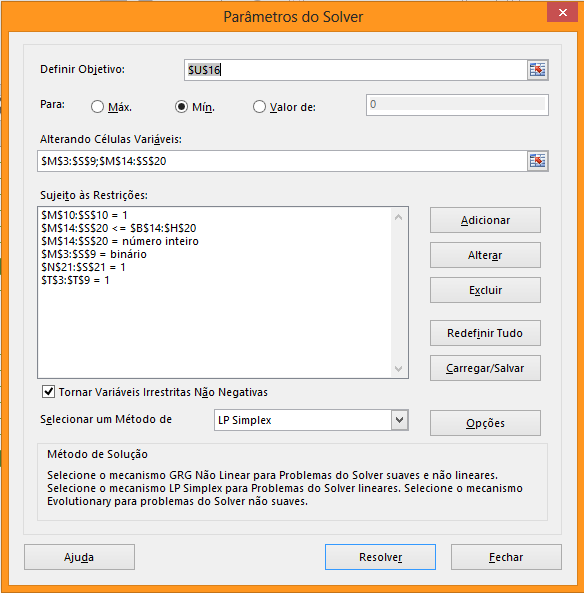
Na quarta matriz, é verificado o valor máximo da primeira matriz. Para isso, foi aplicada a fórmula do produto do valor máximo para cada célula correspondente na matriz 2 pela coluna.



Direitos de Imagem: autores do artigo [10]

Para conseguir o resultado, foi utilizada uma fórmula que soma o produto da matriz 1 pela matriz 4.

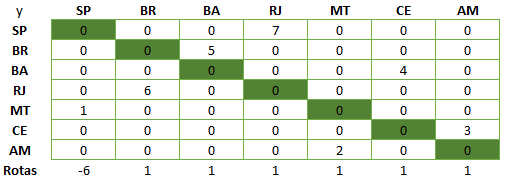
Abaixo, a configuração da ferramenta Solver com as restrições:



Direitos de Imagem: autores do artigo [10]

O valor de “Definir Objetivo” é a célula de menor distância. No parâmetro “Para”, é selecionado o valor mínimo, pois o objetivo é encontrar a menor rota. Na seção de restrições, temos 1) que a coluna e a linha “Rotas”, da matriz 2 tem que ter valor 1; 2) que a matriz 3 seja menor ou igual que a matriz 4; 3) que os valores da matriz 3 sejam inteiros; 4) os valores das linhas e colunas da matriz 2 sejam binários; e 5) que somente a linha “Rotas” da matriz 3 tenha valor 1 em todas as células.

Calculadas as distâncias entre os aeroportos, a solução do problema foi o caminho SP - MT - AM - CE - BA - BR - RJ - SP.



Direitos de Imagem: autores do artigo [10]

A distância da menor rota calculada é de 8.157 km. Essa otimização diminui os custos para a companhia aérea, possibilitando a economia de combustível, trazendo lucros para empresas e mais clientes, já que as rotas foram otimizadas. Para efeito de comparação, a economia de combustível média, considerando que as aeronaves dessa companhia são Boeing 777 é de mais de R$150 por vôo, comparando a rota encontrada com a segunda rota mais eficiente, que percorreria o trajeto SP - RJ - BR - MT - AM - CE - BA - SP, num total de 8.452 km.

Todas as imagens estão disponíveis para melhor visualização através [deste link](https://drive.google.com/drive/folders/1xBE5GgvBmqNmLyJg9BxjlwQ-D2KFnhOh?usp=sharing). Além disso, o link para este artigo está referenciado na seção de Referências, com índice [10].

**6.5: Links relevantes**

Alguns links complementares relevantes estão listados abaixo.

* [Como funciona a malha aérea Brasileira](https://memoria.ebc.com.br/noticias/brasil/2013/06/entenda-como-funciona-a-malha-aerea-do-brasil)
* [[VÍDEO] Projeto e Análise de Algoritmos - Aula 13 - Problema do caminho mais curto](https://www.youtube.com/watch?v=02fSc7kJ_NE)
* [Algoritmo de caminho de custo mínimo de Dijkstra - uma introdução detalhada e visual](https://www.freecodecamp.org/portuguese/news/algoritmo-de-caminho-de-custo-minimo-de-dijkstra-uma-introducao-detalhada-e-visual/)
* [Algoritmo de Dijkstra](https://www.ime.usp.br/~pf/algoritmos_para_grafos/aulas/dijkstra.html)

1. **Proposta de solução algorítmica**

Este é um problema que se caracteriza como um problema do caminho mais curto em um grafo ponderado e sem arestas de valores negativos. Por conta da natureza do problema e da complexidade de implementação, a solução algorítmica utilizada será o Algoritmo de Dijkstra, que será implementado na linguagem Java para a resolução do problema.

O algoritmo foi implementado para todos os destinos. Então, tendo Natal como origem de todos os trajetos, para cada um dos destinos, foram geradas redes com vértices pré-determinados e, a partir disso, com a implementação do Algoritmo de Dijkstra, descobrimos qual a rota mais vantajosa para cada destino partindo de Natal, segundo os parâmetros informados, formando assim, a melhor malha aérea possível para a companhia aérea.

**7.1: Descrição da implementação**

O primeiro passo “extra-Dijkstra” da nossa solução algorítmica é importar os dados da nossa base de dados ([Base de Dados Completa](https://docs.google.com/spreadsheets/d/1DKsnh1rO6Hjoj7eX5Yhhws-rdxSXM7NnxoIxMS1g7Z0/edit?usp=sharing)) numa versão resumida, contando apenas com as colunas Trajeto, Distâncias e Nível de Demanda ([Data](https://docs.google.com/spreadsheets/d/1x8B1pl_EHIqHVYaaQ0CKrON4mUbL3_dcS7KDLAEKK74/edit?usp=sharing)). Essa importação foi feita obtendo essa base de dados em formato .csv, importados pro projeto através da ferramenta OpenCSV. Cada destino tem um arquivo .csv associado a si, com seus dados singulares.

Após a importação, são construídos os objetos do tipo City, cidades da rede, e os objetos do tipo Route, rotas da rede. Após isso, são calculados os custos de operação em cada rota e seu peso é calculado utilizando-se a fórmula informada na seção 4 deste relatório. Todas essas informações de cada destino são armazenadas num objeto tipo “Matrice”, que armazena uma lista de “Routes” (arestas), que são as rotas entre cada parada de um trajeto, e uma lista de cidades “Cities” (vértices).

Cada objeto do tipo City, possui três campos: distance e last\_stop (distância e rótulo), que serão atualizados pelo algoritmo de Dijkstra; e o campo nome, que armazena uma string para identificar a cidade. Além dos getters e setters, possui um método chamado “isIn” que recebe uma lista de cidades e retorna se o objeto pertence a essa lista. Esse método tem complexidade O(n).

Já os objetos do tipo Route, tem três campos: source, destination e weight (origem, destino e peso), informações que serão importante para o algoritmo de Dijkstra. Além dos getters e setters, possui um método chamado “isIn” que recebe uma lista de rotas e retorna se o objeto pertence a essa lista. Esse método tem complexidade O(n).

Tanto para cidades, quanto para rotas, um objeto é buscado na lista do objeto Matrice correspondente antes de ser adicionado, para não termos ocorrências de objetos repetidos.

Como esses métodos são chamados no construtor do tipo Matrice, a complexidade deste construtor é O(n). Outros métodos do tipo Matrice, além dos getters e setters, como listCities e listRoutes, que constróem as listas de cidades e rotas, respectivamente, verifyRoute, searchRoute (dois métodos de busca que retornam valores de tipo distintos), verifyCity e searchCity, que verificam se objetos dos tipos Route e City já se encontram na lista de rotas e cidades, respectivamente, também possuem complexidade O(n). Já os métodos weighter, que calcula o peso de cada rota, e countCosts, que calcula os gastos de cada rota para o cálculo do peso, possuem complexidade O(1).

Tudo isso só acontece quando a construção de uma nova matriz é solicitada pelo objeto do tipo Dijkstra, passando o destino selecionado pelo usuário como parâmetro pro construtor de Matrice.

Um objeto do tipo Dijkstra tem seis campos: weights, que receberá nosso tipo Matrice, a matriz de pesos; open, a lista de vértices (cidades) abertos; closed, a lista de vértices (cidades) fechados; smallest\_path, a lista de vértices (cidades) que compõem o caminho mais curto; neighbourhood, uma lista de vértices (cidades) que compõem a vizinhança de cada vértice verificado pelo algoritmo; e index, uma lista de inteiros que vai receber os índices das rotas que têm o vértice consultado como origem na lista de rotas. Essa lista complementa a lista neighbourhood no cálculo da vizinhança de um vértice.

Como a nossa classe principal Menu só constrói um único objeto Dijkstra durante toda a sua execução, foi criado o método initialize, que cria a matriz de pesos do destino solicitado pelo usuário, cria uma nova lista de nós abertos recebendo a lista de cidades da matriz recém-criada, e limpa as listas de nós fechados, do caminho mais curto, de vizinhança e de índices. A complexidade deste método é O(1).

Há também o método getNeighbours, que calcula a vizinhança de um vértice passado. Ele faz isso percorrendo a lista de rotas, procurando as rotas que possuem a cidade origem com o mesmo nome do vértice passado. Caso ele encontre, percorre a lista de nós abertos se o destino dessa rota encontrada está na lista de nós abertos. Caso esteja, isso significa que este nó destino pode ser incluído na vizinhança do nó passado. Ele é então adicionado à lista neighbourhood e o índice da rota encontrada na lista de rotas é adicionado na lista de índices, de modo que o elemento i da lista de vizinhos seja da rota com o índice na posição i da lista de índices. Esse método tem complexidade O(n²).

Por fim, temos a implementação do algoritmo de Dijkstra, presente no método algorithm. O método inicia chamando o método initialize, para preparar as listas que serão utilizadas e construir a matriz de pesos necessária.

Foi criado um nó do tipo City chamado current, que controlará qual o nó que está sendo verificado no momento (equivalente ao i do pseudocódigo do algoritmo de Dijkstra). Em um primeiro momento, ele recebe a primeira cidade da lista de cidades, isto é, o nó fonte. Sua distância recebe o valor 0 e seu rótulo (last\_stop) recebe o próprio valor do nó fonte. Após isso, este nó é removido da lista de nós abertos e inserido na lista de nós fechados, e tem sua vizinhança calculada pelo método getNeighbours.

Nosso próximo passo são dois laços de repetição aninhados que inicializarão todos os outros nós da rede: o primeiro for percorre a lista de cidades, e o segundo, a lista de vizinhos do nó fonte. Para cada vizinho j do nó fonte, o valor da distância é atualizado com o peso da rota que liga o nó fonte a j, e seu rótulo recebe o nó fonte. Para os vértices que não estão na vizinhança do nó fonte, sua distância é atualizada para “infinito” (na prática, é o valor máximo que uma variável do tipo inteiro pode assumir), e seu rótulo recebe o valor NULL.

Após essa inicialização, o programa entra num laço de repetição, num primeiro momento, percorre a lista de nós abertos procurando qual de seus elementos possui a menor distância acumulada do nó fonte. Após isso, o nó selecionado é removido da lista de nós abertos e inserido na lista de nós fechados. Removemos os vizinhos do vértice anteriormente consultado da lista neighbourhood e calculamos a vizinhança do nó selecionado. O último passo da repetição é verificar, para os vizinhos do nó selecionado, caso existam, se algum deles possui a distância acumulada maior que a soma da distância acumulada do nó selecionado e do peso da rota (aresta) que liga o nó selecionado à este vizinho.

Depois que todos os nós são verificados e o laço de repetição se encerra, o nó current recebe o conteúdo do último elemento da lista de cidades da matriz de pesos, que, por causa da forma que construímos essa lista, é sempre o destino final do trajeto, ou seja, o nosso sumidouro. Agora atualizado com o valor do nosso destino, current é adicionado à lista de cidades que compõem o menor caminho.

Após isso, o programa entra num laço de repetição, onde o nó current receberá o valor do seu rótulo, ou seja, do nó ancestral a ele, reconstruindo o menor caminho na ordem inversa, do destino para a origem, repetindo-se enquanto current tiver um rótulo diferente do nó fonte.

Por fim, simplesmente invertemos a ordem de todos os elementos da lista do menor caminho através do método Collections.reverse, fazendo com esta lista agora esteja ordenada da forma correta, da fonte para o sumidouro. Para finalizar, retornamos a lista de vértices que correspondem ao melhor caminho. Essa lista vai ser processada na interface para que o nome de cada uma das cidades que a compõem seja imprimida na tela para o usuário e salva também em um arquivo output.txt.

Este método, que contém a implementação do algoritmo de Dijkstra, possui complexidade O(n²). A complexidade geral da solução é, portanto, O(n²). Os resultados informados pelo programa estão listados na planilha [Resultados](https://docs.google.com/spreadsheets/d/1jk5oPMdb1WqnX9iZ6rG1rypRX-eZagUpzCqVQvbRVzY/edit?usp=sharing).

**7.2: Detalhes técnicos da implementação**

O programa foi desenvolvido em linguagem Java, através da IDE Eclipse (Eclipse for Java Developers), com o auxílio das bibliotecas OpenCSV e Apache Common Lang 3, que estão já adicionadas às dependências do projeto e disponibilizadas na pasta “lib” do projeto em formato .jar. Foi desenvolvido utilizando Java 16.

O programa pode ser executado através de uma IDE (recomenda-se o uso do Eclipse).

1. **Conclusão**

Através deste estudo de caso, foi possível analisar e construir conhecimento para uma decisão estratégica das mais importantes para uma companhia aérea, tornando essa decisão prática não uma escolha “às cegas”, mas embasada. Para tanto, o uso de grafos foi fundamental, pois direcionou a escolha por rotas específicas dentro de um mar de possibilidades, além de indicar para a companhia quais linhas são as mais efetivas e com maior potencial lucrativo para a empresa. É importante frisar que, no entanto, este estudo tem algumas abstrações, não sendo um modelo totalmente realístico de execução do cálculo de rotas por um sistema comercial.

Diante da literatura disponível para o problema, vimos que existem diferentes abordagens para resolvê-lo: utilizando algoritmos de caminhos mais curtos,, utilizando busca em largura e árvores de extensão, combinando matrizes de adjacência e de pesos, e, por fim, utilizando o Problema do Caixeiro Viajante.

Para nosso problema especificamente, utilizamos o algoritmo de Dijkstra para encontrar o caminho mais curto, consequentemente, a melhor rota entre Natal e os outros estados do Brasil.

Os recursos necessários para execução da implementação aqui descrita podem ser baixados em [MalhaAerea.zip](https://drive.google.com/file/d/17HsZTH0Y4ISP4KfcGXDE0lhakT_k6OhT/view?usp=sharing).

**Referências**

[1] Goldbarg, Marco Cesar. Grafos: conceitos, algoritmos e aplicações / Marco Goldbarg, Elizabeth Goldbarg. - Rio de Janeiro : Elsevier, 2012.

[2] “[Boeing 737](https://pt.wikipedia.org/wiki/Boeing_737)”, Wikipedia. Consultado em 3 de Maio de 2022.

[3] “[Especificações e mapa do Boeing 737-800](https://www.klm.com.br/information/travel-class-extra-options/aircraft-types/boeing-737-800)”, KLM. Consultado em 3 de Maio de 2022.

[4] “[Família Boeing 737 Next-Generation](https://www.boeing.com.br/produtos-e-servicos/avioes-comerciais/737-ng.page)”, Boeing Brasil. Consultado em 4 de Maio de 2022.

[5] [Skyscanner](https://www.skyscanner.com.br). Consultado em 3 de Maio de 2022.

[6] “[Petrobras eleva preço do querosene de aviação em mais de 18% em vários polos](https://g1.globo.com/economia/noticia/2022/04/01/petrobras-eleva-preco-do-querosene-de-aviacao-em-mais-de-18-em-varios-polos.ghtml)”, G1 Notícias. Consultado em 3 de Maio de 2022.

[7] “[UM PROBLEMA DE GRAFOS APLICADO EM ROTAS DE AVIÕES](https://editora.pucrs.br/edipucrs/acessolivre/anais/1501/assets/edicoes/2020/arquivos/114.pdf)”, Casagrande, Guilherme Marin; Bazão, Vanderléa Rodrigues. Consultado em 19 de Junho de 2022.

[8] “[Ciência da Computação na prática — Teoria dos Grafos: Caminho mais curto](https://inside.contabilizei.com.br/ci%C3%AAncia-da-computa%C3%A7%C3%A3o-na-pr%C3%A1tica-teoria-dos-grafos-caminho-mais-curto-665c01099d29)”, Souza, Eduardo. Consultado em 19 de Junho de 2022.

[9] “[ESTRUTURA E DINÂMICA DAS REDES AÉREAS](https://www.puc-rio.br/ensinopesq/ccpg/pibic/relatorio_resumo2011/Relatorios/CTC/FIS/FIS-Antonio%20Rossano%20e%20Eduardo%20Henrique%20Filizzola%20Colombo.pdf)”, Rossano, Antonio; Colombo, Eduardo Henrique Filizzola; Anteneodo, Celia. Consultado em 19 de Junho de 2022.

[10] “[Otimização de rotas aéreas pelo PCV](https://semanaacademica.org.br/system/files/artigos/otimizacaoderotasaereas.pdf)”, Lima, Guilherme Bispo; Silva, Luan de Freitas; Vargas, Samuel Fernandes; Quintino, Luis Fernando; Oliveira, Rafael Rodrigues; Oliveira, Wesley Barbosa. Consultado em 19 de Junho de 2022.