Uma aplicação de teoria dos grafos para otimização de custos de rotas aeronáuticas internacionais

Pedro S. Prieto, Fabio Lubacheski

Faculdade de Computação e Informática
Universidade Presbiteriana Mackenzie (UPM) - São Paulo, SP– Brasil

savoyprieto91@gmail.com, fabio.lubacheski@mackenzie.br

Abstract. *This research's main theme is the application of graph theory to find the best choice of international airway routes, that is analyze the variables that go into the cost of the routes, and as a final result model a graph to find the most efficient route from point A to B utilizing the Dijkstra algorithm and Floyd-Warshall's algorithm.*

Resumo. *Esta pesquisa apresenta como tema principal o estudo de teoria dos grafos aplicados para a escolha de rotas aeronáuticas internacionais, ou seja, analisar as variáveis que fazem parte do custo das rotas e então como resultado final modelar um grafo para achar a rota mais eficaz de um ponto A para B utilizando o algoritmo de Dijkstra e Floyd-Warshall.*

1. Introdução

1.1 Contextualização do problema

O sistema aeroportuário é de extrema importância para sociedade em termos de relevância econômica e social, e é necessário para o funcionamento da sociedade que este sistema deva ser operado da maneira mais eficaz possível.

Em 2019, o último ano completo antes da pandemia COVID-19, o número de vôos foi aproximadamente 38,9 milhões (STATISTA, 2021), e neste número é incluso desde pessoas que estão viajando de férias, até transporte de itens essenciais a vida, como remédios, alimentação, equipamento cirúrgico e etc.

Um dos maiores fatores de complexidade neste sistema é o custo das viagens e como fazer a melhor escolha de caminhos para essas aeronaves chegarem de um ponto A para B, logo neste estudo iremos focar nas rotas aeronáuticas internacionais (*airways)* e como escolher a rota mais eficiente segundo os critérios das companhias aéreas.

Para contextualizar melhor o tema, vamos supor um vôo que tem como ponto de partida Chicago (ORD) e destino Londres (LHR), e para simplificar diremos possíveis duas rotas aeronáuticas para escolher, ou seja, podemos escolher a rota A (preta) ou B (vermelho), como mostrado na imagem abaixo:

Figura 1: Exemplificação de duas rotas de ORD para LHR

Supondo que a Rota A é a mais curta do que a B, a lógica nos diria que a escolha correta seria a A, seria trivial. Porém há muitos fatores além da distância a serem considerados.

Fazendo algumas suposições para exemplificar a rota A: diremos que o vento está para o sentido Oeste, isso já será um fator negativo, pois estamos indo para o Leste; com o vento no sentido oposto também teremos um consumo de combustível maior; ao passar por qualquer país é necessário pagar *Overflight Fees* (x dólares por hora), e o Canadá tem uma das taxas mais caras no mundo.

Conclusão, um problema que parecia trivial acaba se tornando muito mais complexos do que primeiro imaginamos. Todos esses fatores e mais outros precisam ser levados em conta e então é por isso que a Rota B pode ser a melhor escolha, mesmo que a distância e o tempo sejam maiores.

Este cálculo é feito em média 100.000 vezes por dia (STATISTA, 2021), com mais fatores e mais rotas a serem analisadas todas as vezes, logo um sistema que coleta todos esses dados e nos retorna a melhor rota traz benefícios tanto para as companhias aéreas e passageiros, quanto para a economia local que pode tornar a importação e exportação mais baratas para determinado trajeto.

1.2 Objeto da pesquisa

A partir dos fatores determinados, como: distância, tempo, vento, *overflight fees*, será feito um cálculo que terá como resultado peso da rota, ou seja, uma rota A pode ter X de custo e a rota B tem custo X + 10.

A partir deste cálculo, estes dados serão modelados em um grafo, onde os vértices serão os aeroportos e as arestas serão as rotas aeronáuticas.

Tendo o grafo já modelado o sistema receberá o local de partida e destino e então os algoritmos de Dijkstra e Floyd-Warshall calcula qual será a melhor rota a ser tomada para sair do ponto A com destino ao ponto B.

Neste contexto, a pergunta que será respondida nesta pesquisa é:

- COMO DEFINIR O QUE SERÃO OS PESOS REPRESENTADOS NO GRAFO ?

- COMO ELABORAR UM GRAFO, TENDO COMO PESO DAS ARESTAS O CÁLCULO DOS FATORES DAS ROTAS, PARA ENTÃO ACHAR A MELHOR ROTA A PARTIR DO ALGORITMO DE DIJKSTRA E FLOYD-WARSHALL ?

1.3 Objetivos do Estudo

1.3.1 Objetivo Geral

O objetivo geral deste estudo é entender os fatores que fazem parte do cálculo de rotas aeronáuticas internacionais, a partir destas informações modelar um grafo para então fazer a análise do melhor caminho de um ponto A para B a partir do algoritmo de Dijkstra e Floyd-Warshall.

1.3.2 Objetivos Específicos

Os objetivos específicos deste estudo são três:

"É desejável ter uma ideia do padrão de lucros e prejuízos sobre a rede da BA (British Airways). Informações sobre rotas específicas são escassas." (Civil Aviation Policy and the Privatisation of British Airways, 1984, p.85, tradução nossa). Logo um dos objetivos é entender quais os fatores que são analisados para poder fazer o cálculo do custo de uma rota aeronáutica internacional, ou seja, fatores como ventos, distância, tempo, tipo de aeronave (consumo de combustível), *overflight fees* e entender também se teremos interferências de leis a levar em consideração quando fazemos os cálculos.

Estudar conhecimentos gerais de grafos e então a partir disso considerar como melhor implementar um grafo para um sistema de rotas aeronáuticas. E por último analisar como implementar os algoritmos de dijkstra e Floyd-Warshall para achar a melhor rota de um ponto A para B.

1.4 Justificativa

Este tema foi escolhido a partir do interesse sobre aviação civil e como é possível que um sistema tão completo e que liga o mundo inteiro funcione em maior parte sem problemas. E então tentar fazer uma simulação em menor escala deste sistema para conseguir adquirir o conhecimento de como, em partes, essa indústria funciona utilizando os conhecimentos aprendidos durante o curso de Ciência da Computação.

Como objetivo final para atrair atenção ao tema iremos montar um sistema que simulará o funcionamento de escolha de rotas em um aeroporto e a partir disso exemplificar para os leitores os fatores que são levados em consideração para as decisões tomadas por companhias aéreas *(que foram mencionados no primeiro parágrafo dos objetivos específicos)* e explicar o funcionamento em menor escala deste sistema de rotas tão complexo que muitos de nós tomamos por garantido.

2. Referencial Teórico

No Referencial Teórico serão apresentados 3 conceitos para servir de base para o desenvolvimento do estudo, são eles: Introdução a Teoria de Grafos, Introdução sobre rotas aeronáuticas internacionais, Algoritmo de Dijkstra e Algoritmo de Floyd-Warshall.

2.1 Introdução a Teoria de Grafos para representação de sistemas de transporte

Nesta seção do referencial teórico iremos apresentar alguns conceitos básicos sobre Teoria de Grafos, e a partir desta descrição é possível extrapolar como um grafo pode ser utilizado para representar um sistema de transporte, e para esta pesquisa, um sistema de rotas aeronáuticas.

- **Conceitos básicos de Grafos:**

"Um grafo G = (V,E) consiste de V , um conjunto não vazio de vértices (ou nós) e de E, um conjunto de arestas. Cada aresta tem um ou dois vértices associados a ela, chamados de suas extremidades. Dizemos que cada aresta liga ou conecta suas extremidades." (Audemir Lima de Souza, 2013, p. 4)

A partir desta definição a próxima etapa é entender como podemos definir melhor as arestas, ou seja, até agora as arestas simplesmente conectam até dois vértices, porém podemos adicionar o que chamamos de peso para a aresta.

Para Sedgewick e Wayne (2011, p. 638, tradução nossa): "(...) O modelo de grafos é imediato: vértices correspondem a interseções e arestas correspondem a ruas, com pesos nas arestas se modela o custo, talvez distância ou tempo de viagem."

Extrapolando esta definição para o problema e tema da pesquisa, os vértices são os aeroportos em todo o mundo e cada um deles são ligados por arestas que correspondem a nossas rotas aeronáuticas, e por último os pesos que são o custo de cada uma dessas rotas.

Para exemplificar melhor como podemos representar um grafo com custos explicado acima, como mostrado na figura 2 para chegar de A até B consideramos dois caminhos possíveis {A, B} com custo de 17 e {A, C, D, B} com custo de 58, logo teremos que o primeiro caminho será o mais eficaz, pois tem o menor custo.

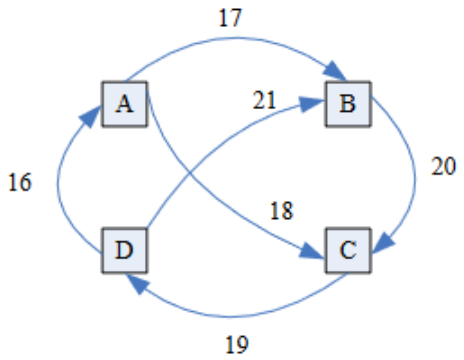

Figura 2: grafo com custo; fonte: Journal of Public Transportation, Vol. 20, No. 1, p. 75.

2.2 Introdução sobre rotas aeronáuticas internacionais

- **Introdução a Rotas Aeronáuticas:**

Segundo a *Federal Aviation Administration (Instrument Procedures Handbook, Chapter 2, 2017, p. 2, tradução nossa): "Rotas aeronáuticas podem ser pensadas como estradas de três dimensões para aeronaves. Na maioria do mundo, as aeronaves são requeridas a voar entre o local de partida e o aeroporto destino"*. Logo as rotas aeronáuticas serão as arestas do grafo a ser modelado, ou seja o caminho entre dois aeroportos, destino e partida.

- **Estrutura de Custo sobre Rotas Aeronáuticas:**

Para a escolha da rota aeronáutica é necessário balancear vários fatores para tentar chegar a um número que representa o preço para uma rota:

COOK e BILLIG (2017, p. 182, tradução nossa): "Mão de obra e combustível são dois dos maiores custos para uma companhia aérea que comprometem 50% do custo total [...] O terceiro maior custo é chamado de transporte, não é um nome que ajuda muito. Essa categoria inclui vários custos menores, porém o maior deles é a compra de serviços regionais parceiros em certas rotas".

A partir desta frase podemos extrapolar uma melhor estrutura de custos, como é dito na primeira parte há dois itens que são necessários para análise: o primeiro é a mão de obra. Para escolha de uma rota que gere uma mão de obra menor é melhor que o voo seja de um tempo menor, e quando analisamos o gasto de combustível é necessário que por exemplo tenhamos uma rota em que o vento beneficie a aeronave.

Na segunda parte da citação os *"serviços regionais parceiros em certas rotas"*, estes seriam o que é chamado na aviação de *Overflight Fees (como mostrado no exemplo da figura 1),* ou seja, é o custo de sobrevoar um território e pagar pelo serviço oferecido por aquele país, como por exemplo as torres de comunicação, logo é necessário analisar qual é a melhor rota baseada no preço de sobrevoar certos locais (países).

Um dos objetivos deste estudo é então chegar a um número calculado a partir dos fatores apresentados neste capítulo levando estes e outros mais fatores em consideração, para que estes valores sejam modelados no grafo como pesos nas arestas.

2.3 Algoritmo de Dijkstra

Tendo o grafo já modelado com os fatores das rotas aeronáuticas calculados como peso das arestas é necessário agora um algoritmo para achar o caminho de custo mínimo entre ponto A até B. Para isso escolhemos o algoritmo de Dijkstra, criado pelo cientista da computação Edsger Dijkstra em 1956 que soluciona o caminho mais curto com arestas de peso não negativo.

FEOFILOFF (2020) define o algoritmo de Dijkstra da seguinte forma:

"Dado um grafo G com custos positivos nos arcos e um vértice s, o algoritmo de Dijkstra faz crescer uma subárvore radicada em G, a partir do vértice s, até que ela englobe todos os vértices que estão ao alcance de s. Para simplificar a discussão, vamos supor que todos os vértices de G estão ao alcance de s. Assim, ao final da execução do algoritmo, a subárvore torna-se geradora."

SEDGWICK (2011, p. 652, tradução nossa) define a implementação do algoritmo de Dijkstra da seguinte forma:

"Começamos inicializando dist[s] para 0 e todos os outros distTo[ ] como infinito positivo, então relaxamos e adicionamos a árvore um vértice de não-árvore com o menor valor de distTo[ ], e continuamos até que todos os vértices estejam na árvore ou nenhum vértice não-árvore tem valor finito de distTo[ ]."

- **Implementação de Dijkstra (Pseudocódigo):**

```
1  para todo v ∈ V[G]
2        d[v] ← ∞
3        π[v] ← -1
4  d[s] ← 0
5  Q ← V[G]
6  enquanto Q ≠ ∅
7        u ← extrair-mín(Q)
8        para cada v adjacente a u
9              se d[v] > d[u] + peso (u, v)
10                   então d[v] ← d[u] + peso (u, v)
11                         π[v] ← u
```

2.4 Algoritmo de Floyd-Warshall

O Floyd-Warshall é o outro algoritmo para resolver o problema de achar o caminho mais curto entre todos os pares de vértices em um grafo orientado e valorado. Este algoritmo foi publicado por Floyd em 1962.

"[..] para resolver o problema de caminhos mínimos para todos os pares em um grafo dirigido G = (V, E). O algoritmo resultante, conhecido como algoritmo de Floyd-Warshall, é executado no tempo Q(V3). Como antes, arestas de peso negativo podem estar presentes, mas supomos que não existe nenhum ciclo de peso negativo." **(CORMEN, Leiserson, Rivest e Stein, 2012, p. 566).**

- **Implementação de Floyd-Warshall (Pseudocódigo):**

```
1 para i de 1 até N
2       para j de 1 até N
3               se existe Aresta ( i , j )
4                       dist ( i , j ) = pesoAresta ( i , j )
5                       p ( i , j ) = i
6               se não
7                       dist ( i , j ) = infinito
8  para k de 1 até N
11      para i de 1 até N
12              para j de 1 até N
13                      se dist ( i , j ) > dist ( i , k ) + dist ( k , j )
14                              dist ( i , j ) = dist ( i , k ) + dist ( k , j )
15                              p ( i , j ) = p ( k , j )
```

3. Metodologia

Sobre a metodologia que será empregada neste Projeto de TCC:

1. O trabalho terá início com uma pesquisa literária sobre assuntos específicos ao tema. A pesquisa terá o foco de conceitos básicos de teoria de grafo; como funcionam rotas aeronáuticas e quais são os fatores que pesam no preço dessas rotas e por último uma revisão sobre o algoritmo de Dijkstra e como implementá-lo.
2. Este alicerce teórico será obtido a partir de autores: Ashworth, Cook, Billig e Sedgewick. Será também analisado material de pesquisas em periódicos científicos, sites, publicações de autoridades governamentais e publicações de associações técnicas.
3. A sistematização de dados será o próximo passo a ser realizado, onde a partir dos dados coletados será feito uma análise dos fatores que contribuem com o custo das rotas aeronáuticas.
4. A próxima etapa será modelar o grafo com os dados obtidos na etapa anterior e por último teremos a simulação do algoritmo de Dijkstra e Floyd-Warshall sobre este grafo para obter um resultado e conclusão final do projeto.

3.1 Pesquisa do referencial

Nesse primeiro passo foi realizada a pesquisa de referências para algoritmos que calculam o caminho de menor custo, conceitos básicos de grafos e introdução a como

rotas aeronáuticas são calculadas. Foi feita uma pesquisa geral sobre como abordar o problema e os próximos passos a seguir: quais métricas será possível utilizar para custos de rotas e como modelar o grafo para ser utilizado como entrada nos algoritmos em questão.

3.2 Diferença entre Dijkstra e Floyd-Warshall

- **Busca do caminho com menor custo**

  No algoritmo de Dijkstra o resultado final será o caminho mais curto a partir de um vértice inicial para todos os outros vértices do grafo, já no Floyd-Warshall o resultado obtido é o caminho mais curto entre todos os pares de vértices do grafo.

- **Guloso vs Dinâmico**

  Dijkstra é um algoritmo guloso, ou seja, a cada iteração o algoritmo busca a solução local ótima, sem preocupação com o escopo global do problema, já Floyd-Warshall é um algoritmo que utiliza programação dinâmica, cada iteração do problema a partir de iterações menores e então é criado uma tabela que armazena essas soluções menores, para no final retornar o resultado completo.

- **Complexidade**

  Em termos de complexidade de tempo, Dijkstra é executado em O(E log V), já Floyd-Warshall em $O(V^3)$. Podemos também utilizar o algoritmo de Dijkstra para achar o caminho mais para todos os pares de vértices, porém podemos ter um tempo de execução, no pior caso, de até $O(V^3 \log V)$.

- **Pesos na aresta**

  Outro benefício de Floyd-Warshall é que podemos ter arestas de peso negativo, porém não podemos ter um ciclo negativo, já Dijkstra não aceita nenhuma aresta com pesos negativos.

3.3 Cálculo das rotas aéreas

Ao invés de uma única aresta representando o caminho de uma local para outro, como foi mostrado na figura 1, será feita uma análise mais cuidadosa destrinchando melhor essa rota, logo teremos múltiplos vértices para representar um caminho, podendo então chegar a uma análise mais exata, tomando em contas os fatores explicados abaixo:

USA Today (2018, Ask the Captain: How are airline routes determined?, tradução nossa) diz que para escolher a rota de um ponto A para B diversos fatores são levados em consideração: “Na maioria das vezes é a rota é provida pelo controle de tráfego aéreo para melhor fluxo de tráfego. Se não é escolhida a rota mais eficaz, levando em conta ventos turbulência ”. Outros fatores a serem levados em consideração são: “O escritório de despacho de voos da companhia aérea analisará a rota mais

eficiente. A seleção da rota pode incluir a quilometragem, o vento e *overflight-fees*. Com base nesses critérios, a resposta à sua pergunta seria porque é o mais econômico.”.

- **Tempo de Voo**

  O tempo de voo será normalmente o fator mais importante, pois quanto menor ele é, menor o custo de todos os fatores de um trajeto, porém outros fatores podem ter uma influência maior sobre o custo total como é mostrado abaixo.

- **Ventos**

  Os fatores relacionados a ventos e turbulências são: vento a favor ou contra a aeronave, ou seja, com o vento a favor há uma maior velocidade máxima logo menos tempo de voo e no final menor consumo de combustível, já com o vento contra a aeronave os fatores invertem, logo maior tempo de voo e maior consumo de combustível.

- **Turbulência**

  O fator turbulência não necessariamente causará um custo exponencialmente maior, porém há rotas que serão evitadas caso o fator turbulência seja muito grande, causando perigo de segurança. Podemos ter uma turbulência, que não traga risco à segurança, e ainda proporcionando um tempo de voo mais curto, tornando a rota viável.

- ***Overflight-fees***

  As *overflight-fees* são calculadas pelas horas de voo no território controlado, ou seja se sobrevoamos por uma área que custa ‘x’ por horas, por 6 horas, teremos um custo de 6x. E podemos ter uma segunda opção tendo como destino o mesmo local onde a *overflight-fee* é de ‘2x’ por hora, porém com um tempo menor de voo.

- **Distância de voo**

  A aeronave, tal como um carro, se desgasta conforme a quantidade de quilômetros percorridos e apesar de não ser um grande fator na escolha de uma rota, as empresas tomam em conta a distância, pois com o tempo o custo de manutenção de uma aeronave é alto.

Para modelar o grafo foi montado uma fórmula para ser seguida, logo os pesos dos vértices foram baseados nessa fórmula levando em consideração o preço de cada um dos fatores desta rota escolhida. Os valores não são 100% fiéis à realidade, porém com a pesquisa o objetivo foi se aproximar da realidade o máximo possível.

Fórmula final para modelar o grafo que utilizaremos:

Custo = (Tempo de voo $\times$ 0.5) + (Turbulência $\times$ 0.1) + ((*Overflight-fee* $\times$ horas de voo) $\times$ 0.3) + (Distância $\times$ 0,1)

3.4 Grafo modelado

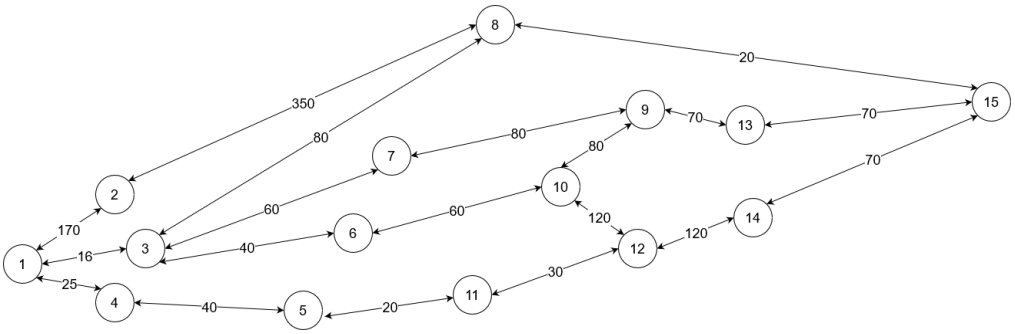

Figura 3: Grafo com vértices de 1 a 15 e já com pesos nas arestas

O vértice 1 representa o aeroporto de Chicago (Chicago O'Hare International Airport - ORD), com um plano de voo para Londres (Heathrow Airport - LHR) representado pelo vértice 15.

Neste plano de voo há múltiplas rotas possíveis, duas que passam pelo território Canadense: **[1 - 2 - 8 - 15] e [1 - 3 - 8 - 15]**, o preço inicial destas rotas, do vértice 1 até 8, são mais custosas, pois as overflight-fees passando pelo Canadá são mais caras comparadas a dos EUA, porém no final o resultado é um custo baixo, com curto tempo de voo, vento a favor e pouca turbulência, logo o custo mais baixo.

Outras quatro rotas possíveis não passam pelo território Canadense: **[1, 3, 7, 9, 13, 15], [1, 3, 6, 10, 9, 13, 15], [1, 3, 6, 10, 12, 14, 15] e [1, 4, 5, 11, 12, 14, 15],** logo as overflight-fees são pagas aos EUA, então é iniciado com um custo l menor. Porém não é uma rota tão eficaz quando sobrevoamos o oceano Atlântico por mais tempo, tendo vento contra a aeronave e mais turbulência, ou seja, um custo final de rota maior.

3.5 Grafo modelado com o mapa

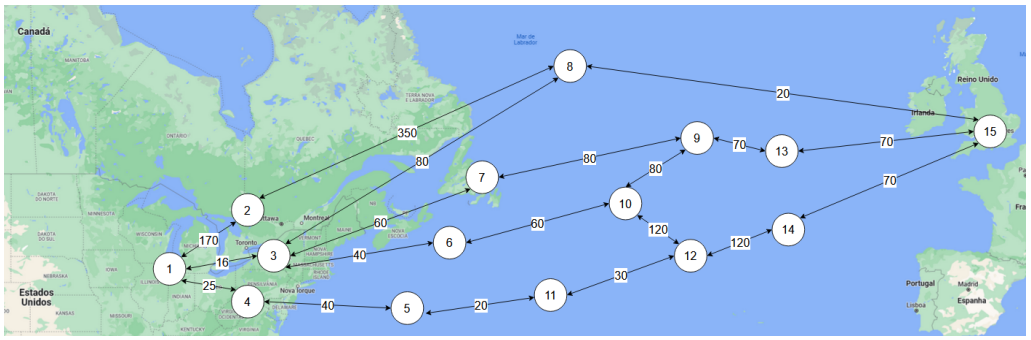

Figura 4: Grafo modelado com o mapa (Chicago - Londres)

4. Aplicações

Essa seção tem como finalidade apresentar a aplicação dos algoritmos utilizando o grafo modelado acima, tanto em código como descrevendo o algoritmo passo-a-passo.

4.1 Aplicação em C

Foram feitas duas aplicações em C para executar os dois algoritmos. O grafo pode ser alterado no código, assim como os vértices e pesos. Para inserir dois vértices conectados e com peso, basta adicionar na função main( ) a seguinte linha: **acrescentaAresta(G,ordemG,1,2,170);** onde G é o grafo, ordemG é o tamanho do grafo + 1, logo como nosso grafo tem 15 vértices, ordemG = 16, 1 e 2 são os vértices e 170 é o peso entre estes dois vértices.

Como resultado teremos a lista de adjacência que é uma forma de representar o grafo como texto. Teremos no caso do Dijkstra a tabela com o vértice, menor distância e menor caminho e o usuário pode escolher para qual vértice ele quer descobrir o menor caminho, tendo o vértice 1 como início (vértice inicial pode ser alterado no código):

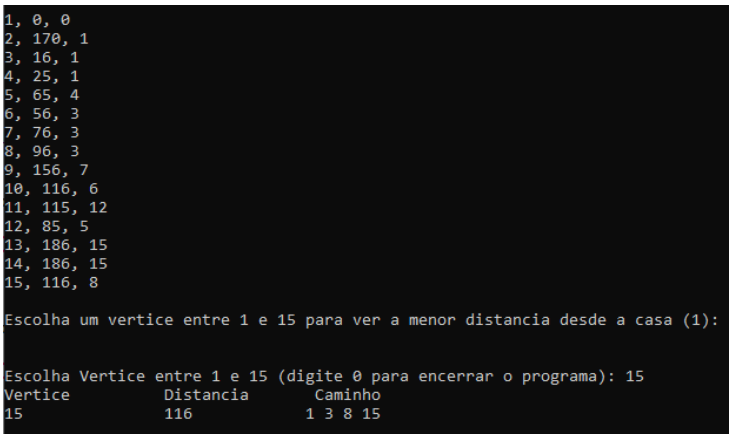

Figura 4: Resultado da execução do algoritmo Dijkstra

Como resultado para o algoritmo de Floyd-Warshall teremos a lista de adjacência, a matriz de menor distância e a matriz de menor caminho, é a descrição do caminho a ser percorrido para obter esse resultado.

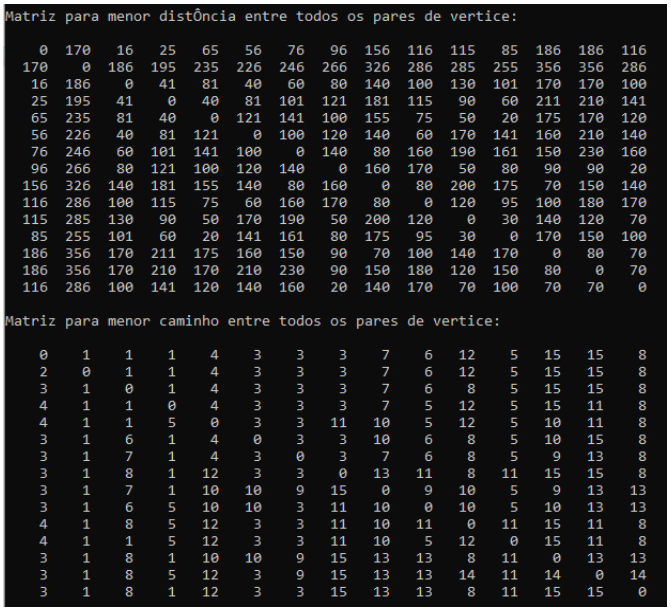

Figura 5: Resultado da execução do algoritmo Floyd-Warshall

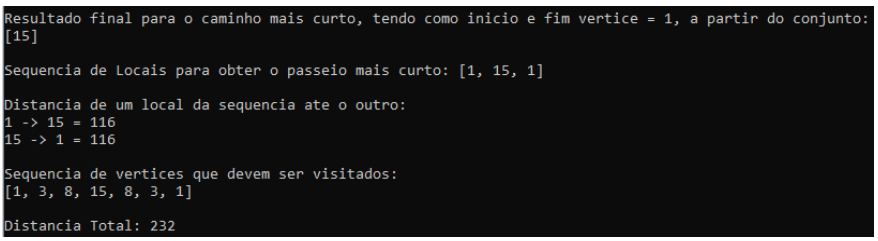

Figura 6: Resultado da execução do algoritmo Floyd-Warshall (continuação)

Link para o repositório do código de ambos os algoritmos: Dijkstra - github.com/Savoy91/TCC-II/blob/main/Dijkstra.c | Floyd-Warshall - github.com/Savoy91/TCC-II/blob/main/Floyd-Warshall.c

4.2 Aplicação do algoritmo de Dijkstra

- **Preparação para executar o algoritmo**

Para a demonstração da execução do algoritmo de Dijkstra sob o grafo que já foi modelado será iniciado uma tabela onde a primeira coluna terá todos os vértices do grafo, a segunda coluna com o menor custo de uma rota a partir do vértice 1 (vértice inicial) e a última coluna sendo o vértice que foi visitado anteriormente para obter o menor custo.

Para a segunda coluna, todos os valores serão iniciados como infinito para comparar com os novos custos que serão descobertos ao longo da execução do algoritmo, ou seja, será atualizada a coluna se o custo novo for menor que o custo atual, que no caso inicial será infinito. E para o vértice 1 o menor custo será 0 e não terá vértice anterior, pois este mesmo é o vértice inicial, como mostrado na figura 5 abaixo.

Antes de iniciar o algoritmo também será declarado dois *arrays*, o primeiro será o de vértices visitados que será vazio (visitados = [ ]) e o segundo será o de vértices não-visitados que conterá todo o conjunto de vértices do grafo (não-visitados = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15].

Vértice	Menor custo de 1	Vértice anterior
1	0	
2	∞	
3	∞	
4	∞	
5	∞	
6	∞	
7	∞	
8	∞	
9	∞	
10	∞	
11	∞	
12	∞	
13	∞	
14	∞	
15	∞	

Figura 5 - Inicialização da tabela para execução de Dijkstra

- **Execução do algoritmo**

Será mostrado a execução detalhada do primeiro passo do algoritmo, onde será necessário analisar os vértices de 1 a 4 como mostrados na figura abaixo:

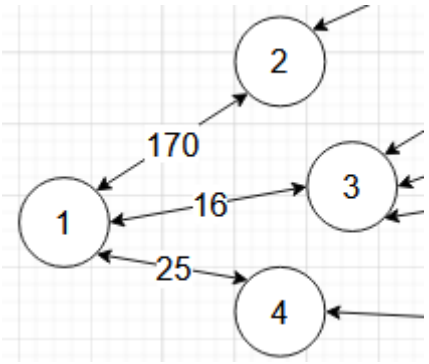

Figura 6 - Parte do grafo que será analisada nesta sessão [1,2,3,4]

Primeiro olhando o *array* de vértices visitados, como este está vazio, o algoritmo será iniciado a partir do primeiro vértice do grafo (1). E então será analisado seus vizinhos que não foram visitados, ou seja, os vizinhos de 1 são os vértices 2, 3 e 4 que estão presentes no *array* não-visitados.

Analisando o vértice 2:

1. É descoberto um caminho novo de custo = 170.
2. Comparando então se este novo custo é menor que o custo atual do vértice 1 para o vértice 2 que é agora igual a infinito.
3. Logo como 170 é menor que infinito, 170 será adicionado ao menor custo do vértice 1 para 2, substituindo assim o valor de infinito.
4. E por último será adicionado o vértice visitado para obter este novo custo, que no caso será 1.

Para os vértices 3 e 4 faremos os mesmos passos, onde descobrimos um custo menor que o atual e então substituímos e adicionamos os valores a tabela. Para o vértice 3 teremos um menor custo de 16 e o vértice visitado anteriormente foi o 1 e para o vértice 4 teremos um menor custo de 25 e também o vértice visitado anteriormente é igual a 1.

Após a análise destes três vértices os *arrays* são atualizados. O *array* de visitados não será mais vazio, pois como analisamos todos os vizinhos de 1, podemos dizer que ele já foi completamente visitado (analisado), ou seja, visitados = [1]. E para o vértice de não-visitados, podemos agora tirar o 1, ou seja quando um vértice é adicionado ao *array* de visitados este mesmo vértice é retirado do *array* de não-visitados.

E então com todas as atualizações feitas será obtida a tabela abaixo:

Vértice	Menor custo de 1	Vértice anterior
1	0	
2	170	1
3	16	1
4	25	1
5	∞	
6	∞	
7	∞	
8	∞	
9	∞	
10	∞	
11	∞	
12	∞	
13	∞	
14	∞	
15	∞	

Figura 7 - Tabela após visitar o vértice 1

Estes passos serão executados até que todos os vértices do grafo sejam adicionados ao *array* de visitados, logo acabamos a execução quando o *array* de vértices não-visitados for vazio.

- **Analisando resultados da execução total de Dijkstra**

Após a execução completa do algoritmo, o resultado é a tabela mostrada abaixo, onde está preenchido o menor custo para todos os vértices a partir do vértice 1 e também os vértices visitados anteriormente para obter este custo.

Vértice	Menor custo de 1	Vértice anterior
1	0	
2	170	1
3	16	1
4	25	1
5	65	4
6	56	3
7	76	3
8	96	3
9	156	7
10	116	6
11	115	12
12	85	5
13	186	15
14	186	15
15	116	8

Figura 8: Execução do algoritmo de Dijkstra finalizada

Agora analisando o resultado para obter o menor custo e o caminho do vértice 1 até o vértice 15, ou seja, como exemplificado no problema inicial o menor custo de Chicago para Londres:

Basta olhar para a linha do vértice 15 onde o resultado final é de **116.** Já para obter o caminho total para chegar neste custo o processo será:

1. Buscar o vértice anterior para o vértice 15 que neste caso será igual ao vértice 8.
2. Partir para a linha do vértice 8, onde é obtido que o vértice anterior é igual a 3.
3. E por último para o vértice 3 onde o vértice anterior foi o vértice 1, que é o igual ao vértice inicial.
4. Fim da execução.

Como resultado final:

- Menor custo do vértice 1 até 15 é = 116
- Caminho a ser percorrido é igual a 1 → 3 → 8 → 15

4.3 Aplicação do algoritmo de Floyd-Warshall

- **Preparação para executar o algoritmo**

Será iniciado o algoritmo declarando duas matrizes, a primeira matriz será distância[][] e a segunda será caminho[][], de tamanho V × V, onde V é igual o número de vértices do grafo, neste caso V = 15. E também será declarado que distância[i][j], será igual a 0 quando i == j .

Observação - será mostrado exemplos até o vértice 6 só para poder visualizar melhor a tabela, no final mostraremos a tabela com todos os 15 vértices completos.

A partir disso as duas matrizes, distância e caminho, ficam como a tabela mostrada abaixo:

	1	2	3	4	5	6
1	0					
2		0				
3			0			
4				0		
5					0	
6						0

Figura 9: Tabela inicializada para o algoritmo de Floyd-Warshall

O segundo passo é analisar se aresta existe, adicionamos distância[i][j] = peso-da-aresta(i, j) e caminho[i][j] = i, se aresta não existe, distancia[i][j] = infinito.

Logo teremos a matriz distância como a tabela abaixo:

	1	2	3	4	5	6
1	0	170	16	25	∞	∞
2	170	0	∞	∞	∞	∞
3	16	∞	0	∞	∞	40
4	25	∞	∞	0	40	∞
5	∞	∞	∞	∞	0	∞
6	∞	∞	40	∞	∞	0

Figura 10: Tabela distância com os primeiros valores adicionados

E teremos a matriz caminho como a tabela abaixo:

	1	2	3	4	5	6
1	0	1	1	1	∞	∞
2	2	0	∞	∞	∞	∞
3	3	∞	0	∞	∞	3
4	4	∞	∞	0	4	∞
5	∞	∞	∞	∞	0	∞
6	∞	∞	6	∞	∞	0

Figura 11: Tabela caminho com os primeiros valores adicionados

- **Execução do algoritmo**

O algoritmo Floyd-Warshall tem uma ordem de complexidade $O(V^3)$ que são os três *nested for loops* (for loops aninhados), onde teremos três variáveis *k*, *i* e *j* para iterar pelo grafo e atualizar as matrizes de distâncias e caminhos, onde se **distância[i][k] + distância[k][j] < distância[i][j]** atualizamos as duas matrizes com os novos valores:

```
for (k = 1; k < V; k++) {
        for (i = 1; i < V; i++) {
                for (j = 1; j < V; j++) {
                        if (distância[i][k] + distância[k][j] < distância[i][j]) {
                                distância[i][j] = distância[i][k] + distância[k][j];
                                caminho[i][j] = caminho[k][j];
                        }
                }
        }
}
```

Exemplo da execução onde k = 4; i = 2 e j = 5 :

k → 1 2 3 4 5 6

i → 1 2 3 4 5 6

j → 1 2 3 4 5 6

Logo:

Se distância[2][4] + distância[4][5] < distância[2][5]:

distância[2][5] = distância[2][4] + distância[4][5];

caminho[2][5] = caminho[4][5];

Ou utilizando os dados durante a execução:

Se 195 + 40 < 235:

distância[2][5] = 195 + 40

caminho[2][5] = 4

Esta condição é falsa e então os valores de distância e caminho não serão atualizados.

O algoritmo será executado enquanto k < V e como resultado final teremos a tabela de distância e caminho completas como mostrado no item abaixo (clique no link para visualizar melhor as tabelas):

- **Analisando resultados da execução total de Floyd-Warshall**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	170	16	25	65	56	76	96	156	116	115	85	186	186	116
2	170	0	186	195	235	226	246	266	326	286	285	255	356	356	286
3	16	186	0	41	81	40	60	80	140	100	130	101	170	170	100
4	25	195	41	0	40	81	101	121	181	115	90	60	211	210	141
5	65	235	81	40	0	121	141	100	155	75	50	20	175	170	120
6	56	226	40	81	121	0	100	120	140	60	170	141	160	210	140
7	76	246	60	101	141	100	0	140	80	160	190	161	150	230	160
8	96	266	80	121	100	120	140	0	160	170	50	80	90	90	20
9	156	326	140	181	155	140	80	160	0	80	200	175	70	150	140
10	116	286	100	115	75	60	160	170	80	0	120	95	100	180	170
11	115	285	130	90	50	170	190	50	200	120	0	30	140	120	70
12	85	255	101	60	20	141	161	80	175	95	30	0	170	150	100
13	186	356	170	211	175	160	150	90	70	100	140	170	0	80	70
14	186	356	170	210	170	210	230	90	150	180	120	150	80	0	70
15	116	286	100	141	120	140	160	20	140	170	70	100	70	70	0

Figura 7: Tabela menor custo entre todos os pares de vértices - https://github.com/Savoy91/TCC-II/blob/main/floyd-warshall-tabela.jpg

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	1	1	4	3	3	3	7	6	12	5	15	15	8
2	2	0	1	1	4	3	3	3	7	6	12	5	15	15	8
3	3	1	0	1	4	3	3	3	7	6	8	5	15	15	8
4	4	1	1	0	4	3	3	3	7	5	12	5	15	11	8
5	4	1	1	5	0	3	3	11	10	5	12	5	10	11	8
6	3	1	6	1	4	0	3	3	10	6	8	5	10	15	8
7	3	1	7	1	4	3	0	3	7	6	8	5	9	13	8
8	3	1	8	1	12	3	3	0	13	11	8	11	15	15	8
9	3	1	7	1	10	10	9	15	0	9	10	5	9	13	13
10	3	1	6	5	10	10	3	11	10	0	10	5	10	13	13
11	4	1	8	5	12	3	3	11	10	11	0	11	15	11	8
12	4	1	1	5	12	3	3	11	10	5	12	0	15	11	8
13	3	1	8	1	10	10	9	15	13	13	8	11	0	13	13
14	3	1	8	5	12	3	9	15	13	13	14	11	14	0	14
15	3	1	8	1	12	3	3	15	13	13	8	11	15	15	0

Figura 8 - Tabela com caminho para menor custo entre todos os pares de vértices - https://github.com/Savoy91/TCC-II/blob/main/floyd-warshall-tabela-2.jpg

Extrair o resultado que estamos buscando quando exemplificamos o problema é similar ao algoritmo de Dijkstra. Primeiro olhando a tabela de menor custo (Figura 7) para a linha 1 e coluna 15 é obtido um menor custo de **116.**

E para extrair o caminho necessário para o custo de 116, olhamos para a tabela de caminhos:

1. Linha 1, Coluna 15 o resultado é 8, logo caminho = [1, 8, 15]
2. Linha 1, Coluna 8 o resultado é 3, logo caminho = [1, 3, 8, 15]
3. Linha 1, Coluna 3 o resultado é 1, logo o caminho está finalizado.

5. Resultados Obtidos

Como resultado houve uma dificuldade em achar dados concretos sobre o valor exato do custo das rotas, e o peso que cada fator tem na escolha da rota. Porém foi obtido uma boa representação de como estas rotas funcionam e como podemos representar estes custos e principalmente como colocá-los em um grafo.

Já para os algoritmos, o projeto obteve êxito em demonstrar os algoritmos passo-a-passo e criar uma aplicação em C para cada um dos algoritmos. Com a aplicação podemos observar que alterando os pesos do grafo obtemos resultados diferentes ao executar a aplicação. E é possível modelar a aplicação para outras rotas, alterando o grafo, e também para outros problemas que envolvem a procura de um menor caminho, modelando um grafo deste problema e inseri-lo na aplicação.

6. Considerações Finais

O projeto trouxe exemplificar o sistema de rotas aeronáuticas e mostrar um pouco de sua complexidade, desde o início onde foi mostrado que temos mais de uma rota possível para ir de um ponto A até um outro ponto B até a execução dos algoritmos.

Foi analisado os fatores que contribuem com a escolha de uma rota, ou seja, o custo de uma rota e de onde vem esses custos. Levando em conta *overflight-fees,* tempo de voo, ventos, turbulência e distância de voo. Com apoio da pesquisa realizada foi feita então uma fórmula para calcular o custo de uma rota específica.

A partir dos fatores analisados, o grafo final foi modelado, que então foi utilizado para a execução dos algoritmos. Com o grafo foi possível mostrar as várias escolhas de rotas possíveis e os vários cálculos que são feitos para que um voo tenha sucesso. Além de modelar o grafo foi possível mostrá-lo em cima do mapa, como na figura 4, trazendo o problema para uma situação real.

Executando o algoritmo de Dijkstra e Floyd-Warshall e mostrando como cada passo é executado ficou evidente o menor custo entre os vértices escolhidos e foi possível extrair o caminho de menor custo, mostrando também o passo a passo para encontrar estas informações que o algoritmo nos retorna.

Cada um dos algoritmos tem suas características específicas, enquanto Dijkstra só nos retorna o menor caminho de um vértice para todos os outros, Floyd-Warshall nos

retorna o menor caminho entre todos os pares de vértice do grafo. Logo, ambos os algoritmos resolvem o problema proposto, tendo cada um suas peculiaridades.

São dois algoritmos que exercitam diversos conceitos da computação e os transformando em resoluções para um problema real, podemos transmitir as ferramentas e conhecimentos necessários para estes serem reproduzidos por estudantes da no futuro.

7. Trabalhos Futuros

Como o projeto teve o objetivo de demonstrar, principalmente para estudantes de execução, como executar estes algoritmos e como modelar problemas do mundo real em um grafo, a sugestão para projetos futuros são: criação de uma GUI *(graphics user interface)* utilizando o código dos dois algoritmos para melhor demonstrar como estes algoritmos funcionam e deixá-los interativos; criação de um projeto que possa destrinchar o código das aplicações, levando em conta como transformamos um algoritmo (pseudocódigo) em uma aplicação e técnicas utilizadas otimização dos códigos; e por último um projeto que melhore o estudo sobre rotas e seus custos para aproximar mais ainda este projeto a realidade.

8. Referências

DE SOUZA, AUDEMIR LIMA. TEORIA DOS GRAFOS E APLICAÇÕES. 2013. TCC (Mestrado Profissional em Matemática) - UNIVERSIDADE FEDERAL DO AMAZONAS.

ASHWORTH, Mark; FORSYTH, Peter. Civil aviation policy and the privatization of British Airways. [*S. l.*: *s. n.*], 1984.

COOK N., Gerald; G. BILLIG, Bruce. Airline Operations and Management. [*S. l.*: *s. n.*], 2017.

SEDGEWICK, Robert; WAYNE, Kevin. Algorithms. Quarta Edição. ed. [*S. l.*: *s. n.*], 2011.

EN Route Operations - Airways. *In*: INSTRUMENT Procedures Handbook. [*S. l.*: *s. n.*], 2017.

FEOFILOFF, Paulo. ALGORITMOS para Grafos via Sedgewick. [*S. l.*], 24 abr. 2020. Disponível em: https://www.ime.usp.br/~pf/algoritmos_para_grafos/index.html#contents. Acesso em: 26 out. 2021.

ISIKLI, Erkan; AYDIN, Nezir; CELIK, Erkan; GUMUS, Alev. Identifying Key Factors of Rail Transit Service Quality: An Empirical Analysis for Istanbul. Journal of Public Transportation, [*S. l.*], 9 mar. 2017

NUMBER of flights performed by the global airline industry from 2004 to 2022. [*S. l.*], 12 out. 2021. Disponível em:

https://www.statista.com/statistics/564769/airline-industry-number-of-flights/. Acesso em: 26 out. 2021.

ALGORITMOS Teoria e Prática: Tradução da Terceira edição Americana. 3°. ed. [*S. l.*: *s. n.*], 2012.

COX, John. Ask the Captain: How are airline routes determined?. **USA TODAY**, [*S. l.*], p. 1, 8 fev. 2018. Disponível em: https://www.usatoday.com/story/travel/columnist/cox/2018/02/04/ask-captain-how-airline-routes-determined/1088837001/. Acesso em: 8 nov. 2022.