CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS
CAMPUS TIMÓTEO

Roberto Martins Morais Júnior

ESTUDO DE CASO: ALGORITMO DE BUSCA APLICADO EM BUSCA DE MELHOR ROTA NO TRANSPORTE URBANO

Timóteo

2022

Roberto Martins Morais Júnior

ESTUDO DE CASO: ALGORITMO DE BUSCA APLICADO EM BUSCA DE MELHOR ROTA NO TRANSPORTE URBANO

Monografia apresentada à Coordenação de Engenharia de Computação do Campus Timóteo do Centro Federal de Educação Tecnológica de Minas Gerais para obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Douglas Nunes de Oliveira

Timóteo

2022

Agradecimentos

Agradeço a todos os professores por me proporcionar o conhecimento, não somente por terem me ensinado, mas por terem me feito aprender. Em especial agradeço ao professor Douglas Nunes de Oliveira pela orientação, apoio e confiança. A palavra mestre nunca fará justiça aos professores dedicados aos quais sem nominar terão os meus eternos agradecimentos.

“São as nossas escolhas, mais do que as nossas capacidades, que mostram quem realmente somos.”.

Alvo Dumbledore

Resumo

O objetivo deste trabalho é fazer uma adaptação do algoritmo de busca em profundidade A* para que este possa encontrar a melhor rota entre dois pontos utilizando o transporte público como meio de locomoção. Essa adaptação faz-se necessária tendo em vista que o algoritmo leva em consideração apenas o valor do passo para tomar sua decisão o que gera demasiadas trocas de linhas em situações que trocas não seriam necessárias, para isto foi adicionado mais um passo no algorítimo A* responsável por adicionar arestas virtuais ao grafo otimizando assim a tomada de decisão quando ao próximo passo e chegando ao destino com o caminho de menor custo e menor quantidade de trocas de linhas.

Palavras-chave: transporte público, busca em profundidade, algoritmo A*.

Abstract

The objective of this work is to adapt the A* depth-first search algorithm so that it can find the best route between two points using public transport as a means of locomotion. This adaptation is necessary considering that the algorithm takes into account only the value of the step to make its decision, which generates too many line changes in situations where changes would not be necessary, for this another step was added in the A* algorithm. responsible for adding virtual edges to the graph, thus optimizing the decision making regarding the next step and arriving at the destination with the lowest cost path and least amount of line changes.

Key-Words: public transport, depth search, A* algorithm.

Lista de ilustrações

Sumário

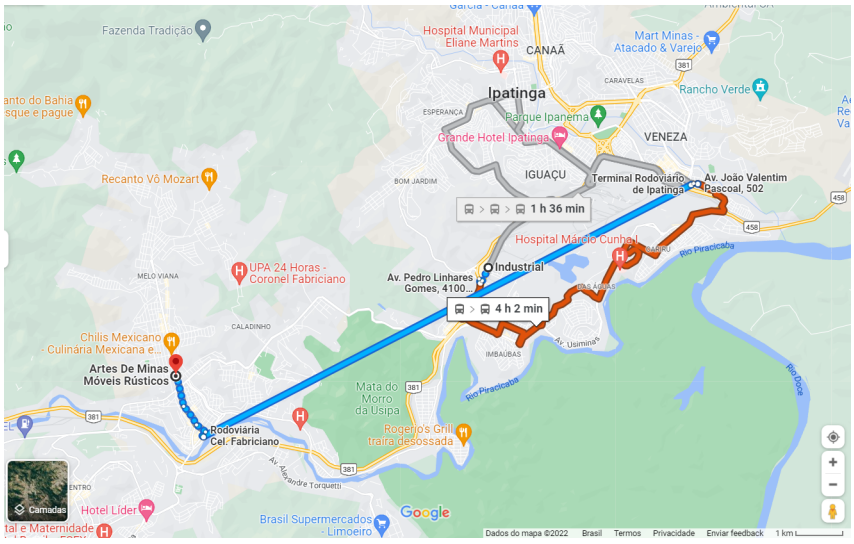
1 Introdução

Os transportes públicos são responsáveis por prover o deslocamento de pessoas de um ponto ao outro em uma cidade. Na maioria das cidades do Brasil tem-se o ônibus como meio de locomoção mais escolhido pelos seus cidadãos mesmo com os transtornos causados pela urbanização, como por exemplo a superlotação que é reflexo do mal estado de planejamento e investimento. É essencial para uma cidade possuir uma rede de transporte público urbano bem estruturada, pois este é uma excelente ferramenta para garantir o importante direito de ir e vir dos cidadãos.(VASCONCELLOS; MENDONÇA, 2010)

Encontramos hoje diversos problemas com o transporte público em questão, além do desconforto, lotação e mal cheiro temos também o problema da dificuldade de acesso, em alguns casos a inexistência, às informações sobre linhas, horários e rotas. Com o intuito de ajudar o usuário a se informar, empresas que prestam o serviço de transporte público criam sites onde publicam estas informações porém nem sempre as informações estão atualizadas e/ou são exibidas de forma intuitiva para o usuário.

Existem também a possibilidade do uso do Google Maps para traçar rotas utilizando o transporte público baseado na origem e destino, porém estes sistemas não funcionam corretamente em cidades do interior ou para viagens que envolvam mais de uma cidade. A seguir, na Figura 1, podemos ver um exemplo com a origem na cidade de Ipatinga e o destino na cidade de Coronel Fabriciano e podemos notar uma linha reta ligando a rodoviária das duas cidades e um longo trecho à pé em Coronel Fabriciano sendo que existem linhas de ônibus que podem ser utilizadas para facilitar o deslocamento do usuário além de existirem pontos de ônibus mais próximos ao destino que poderiam ter sido utilizados ao invés de ir até a rodoviária, o que causou um aumento no percurso.

Figura 1 – Pesquisa com a origem e o destino em cidades diferentes



Fonte: (GOOGLE, 2022)

De acordo com artigo publicado no site Band Uol (BandNews FM Rio, 2020) uma pesquisa feita pela Moovit, empresa israelense desenvolvedora de um aplicativo gratuito de mobilidade urbana com foco em informações de transporte público e de navegação, antes da pandemia causada pelo vírus da COVID-19 85% da população que participaram da pesquisa utilizam ônibus, trem ou metrô como meio principal de locomoção, após o início da pandemia este índice caiu para 68% e estima-se que se estabilize em 70%. Através desta pesquisa foram levantados também dados sobre incentivos ao uso do transporte público e entre os destaques estão o aumento da frota para evitar que ônibus, metrôs e trens fiquem lotados com 77%; saber a localização dos veículos em tempo real com 63%; e 45% acreditam que mais informações sobre as linhas em operação seriam um grande atrativo para o uso dos meios de transportes públicos.

Atualmente as informações, de transporte público da região do Vale do Aço, são basicamente os horários e a lista das ruas de cada linha de transporte. Esta lista de ruas muitas vezes não auxilia o usuário do transporte público a se orientar, visto que nem sempre ele conhece o nome e a posição geográfica das ruas, principalmente se o usuário vier de outra região.

O problema a ser estudado neste trabalho de conclusão de curso é, ao basear-se na origem e destino do usuário, como indicar ao usuário qual a melhor rota que lhe atende.

1.1 Objetivos

1.1.1 Objetivos Gerais

O objetivo deste trabalho é investigar o uso do algoritmo de busca em profundidade A* na busca do melhor caminho entre dois pontos.

1.1.2 Objetivos Específicos

Realizar as adaptações necessárias para que o algoritmo de busca em profundidade A* funcione da melhor forma na busca do melhor caminho entre dois pontos e aplicá-lo em rotas de transporte público urbano.

1.2 Organização do Texto

O documento esta dividido em 6 capítulos. O primeiro capítulo é responsável pela apresentação do projeto e é composto por uma breve apresentação e contextualização do problema a ser atacado, assim como os objetivos gerais e específicos.

No segundo capítulo é feita uma revisão dos fundamentos que são necessários para o entendimento do problema assim como as ferramentas e embasamentos teóricos a serem utilizados na construção da solução do problema proposto.

No terceiro capítulo é falado sobre a metodologia a ser aplicada no desenvolvimento da solução para o problema proposto, desde a coleta de dados até a interface a ser apresentada

ao usuário.

No quarto capítulo é mostrado como foi feito o desenvolvimento das camadas do projeto proposto neste trabalho desde a montagem da base de dados até a implementação do algorítimo.

No quinto e sexto capítulo constam os resultados preliminares da solução que está sendo desenvolvida, considerações finais sobre o estudo feito, as limitações, a conclusão e os possíveis trabalhos futuros.

2 Fundamentos teóricos

2.1 Algoritmos de busca

Boa parte dos problemas que podem ser representados por grafos, assim como o estudado neste trabalho, tem a busca como uma forma de chegar a solução, para alguns problemas temos que percorrer todo o grafo para encontrar a melhor solução, mas em outros casos uma busca em um subconjunto do gráfico inicial já é o suficiente para alcançarmos o resultado esperado.

Como definido no livro "*The handbook of artificial intelligence*“ (FEIGENBAUM; BARR; COHEN, 1981) Inteligência Artificial é a parte da ciência da computação envolvida no projeto de sistemas que exibem características que associamos com a inteligência do comportamento humano. Uma das utilizações da Inteligência Artificial é no desenvolvimento de algoritmos de busca, no artigo "Código de Busca em Largura e Profundidade“ (SIDNEI; LUIS, 2018) escrito por Sidnei Gomes e Luis Santos os autores separam os algoritmos de busca em duas classes: a busca em profundidade (*depth-first search*) e a busca em largura (*breadth-first search*).

Os algoritmos de busca em profundidade são mais utilizados quando precisamos reunir informações sobre os nós e arestas do grafo em que o mesmo é aplicado. O algoritmo de busca em profundidade se inicia do primeiro nó da arvore de busca e se expande para os nós filhos até que o alvo seja encontrado ou encontre um nó que não possui filhos. Com o resultado deste pré-processamento conseguimos alimentar outros tipos de algoritmos que com base nas informações coletadas conseguem encontrar uma solução para o problema.

Como os algoritmos desta classe percorrem todos os caminhos de um grafo ou árvore, podemos dizer que sua complexidade é de O (n+a), onde ’a’ e ’n’ são os números de arestas e nós respectivamente, e se aplicado a grafos ou árvores com grande número de arestas e nós, de forma que não é possível alocar todas as informações em memória, o algoritmo não finaliza.

Quando o problema envolve encontrar o melhor caminho entre dois vértices, os algoritmos de busca em largura se sobressaem sobre os de busca em profundidade. Estes algoritmos, a partir de um vértice inicial, pesquisam todos os vizinhos antes de aprofundar mais na árvore ou grafo que está sendo analisado.

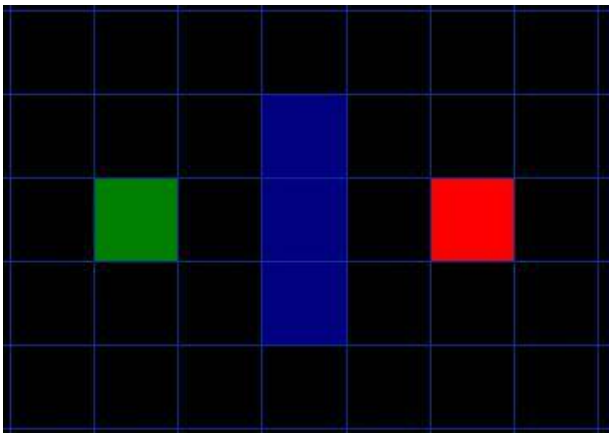
2.1.1 Algoritmo A*

Um dos algoritmos mais utilizados na busca pelo melhor caminho é o algoritmo A* (Lê-se "A estrela"). O algoritmo A* sempre encontra o caminho de menor custo, quando há algum caminho, entre dois pontos. O algoritmo atribui a cada nó um custo estimado e vai caminhando pelo nó adjacente de menor custo até chegar no destino desejado, ao chegar ele faz o caminho inverso listando todos os nós que foram percorridos para chegar no destino resultando no caminho de menor custo de deslocamento entre os dois pontos.

Abaixo segue um esquema de passo-a-passo sobre o funcionamento do algoritmo A*. As informações e imagens foram retiradas do artigo "A* Pathfinding para Iniciantes", do autor Patrick Lester (LESTER, 2005).

Supondo que a Figura 2 represente o nosso mapa de procura, neste exemplo desejamos sair do ponto verde e chegar ao ponto vermelho. Note que temos um retângulo azul no centro que não podemos passar por ele. Note também que o mapa foi dividido em quadrados para facilitar o entendimento do algoritmo.

Figura 2 – Mapa de procura.



Fonte: (LESTER, 2005)

Colocando o nosso ponto de início (quadrado verde) como o pivô do nosso algoritmo vamos:

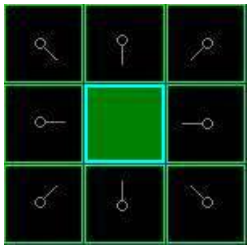
Passo 1: Acrescentá-lo a uma "lista aberta" que servirá para guardar todos os quadrados que poderão ou não compor o melhor caminho para chegarmos ao quadrado vermelho.

Passo 2: Adicionar também nesta "lista aberta" todos os quadrados que podemos passar por eles e são adjacentes ao pivô e pra cada devemos adicionar o pivô como pai.

Passo 3: Remover o pivô da "lista aberta" e adicioná-lo em uma "lista fechada"que será responsável por guardar apenas os quadrados que farão parte do melhor caminho.

A Figura 3 representa como deve estar agora o nosso mapa. O quadrado verde no centro é o quadrado pai, a bordar azul indica que ele foi adicionado na "lista fechada", os quadrados adjacentes estão com uma borda verde para indicar que já estão na "lista aberta" e os ponteiros cinza no centro deles indicam quem é o quadrado pai.

Figura 3 – Quadrados preparados para receberem seus custos



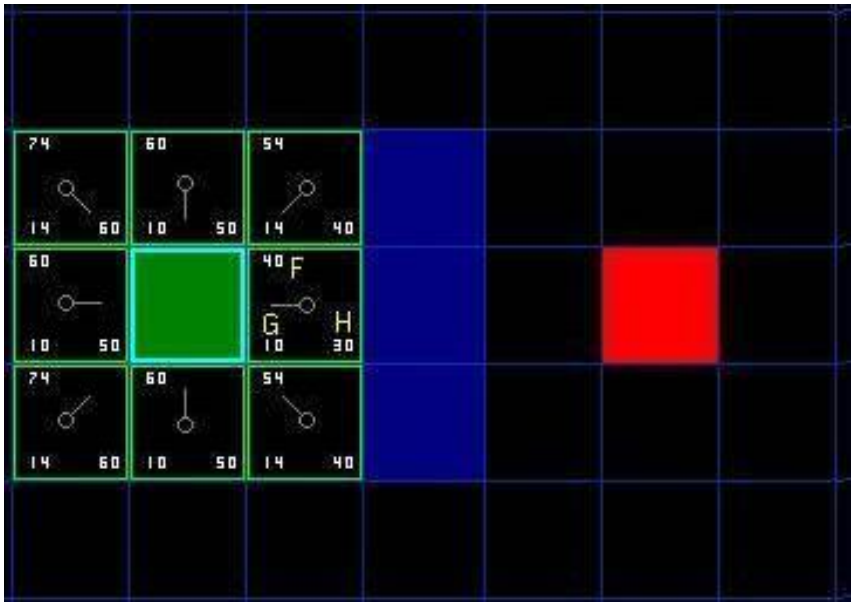
Fonte: (LESTER, 2005)

Para prosseguir com o algoritmo temos que atribuir os pesos de cada quadrado adjacente ao pivô, pra isso utilizamos a equação F = G + H, onde

- F é o custo de cada quadrado,
- G é o custo para se mover do ponto de início (neste exemplo consideramos sendo o quadrado verde) até o quadrado em questão,
- H é o custo estimado para se mover do quadrado em questão até o ponto final (neste exemplo consideramos sendo o quadrado vermelho).

Passo 4: Calcular os custos dos quadrados adjacentes ao pivô.

Figura 4 – Cada quadrado adjacente com o seu devido custo.

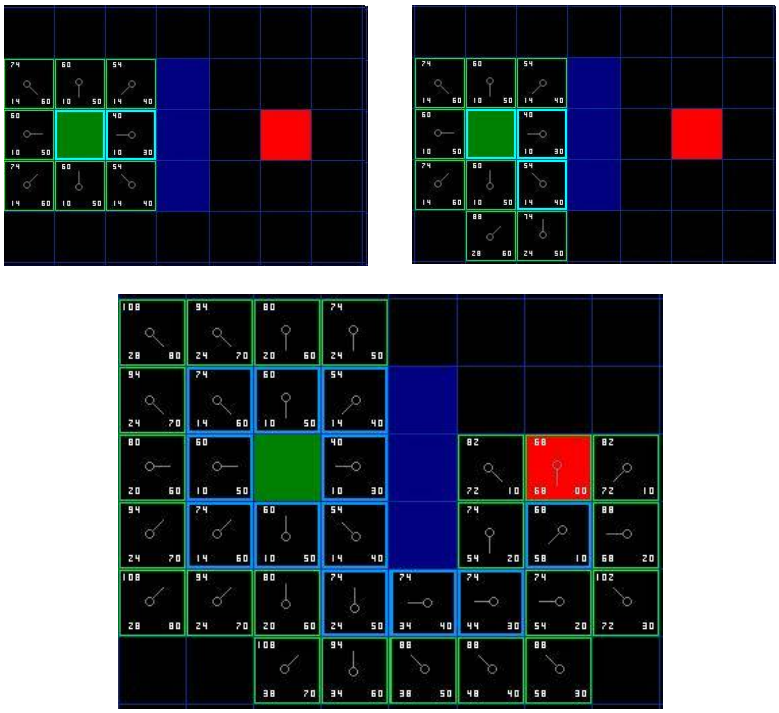


Fonte: (LESTER, 2005)

Passo 5: Escolher o Quadrado adjacente de menor custo e repetir todos os passos anteriores colocando o quadrado em questão como pivô.

Os passos anteriores irão se repetir até o algoritmo encontrar o destino (quadrado vermelho) ou até acabarem todos os quadrados da "lista aberta", se o segundo caso ocorrer não é possível encontrar um caminho entre o ponto inicial e o final.

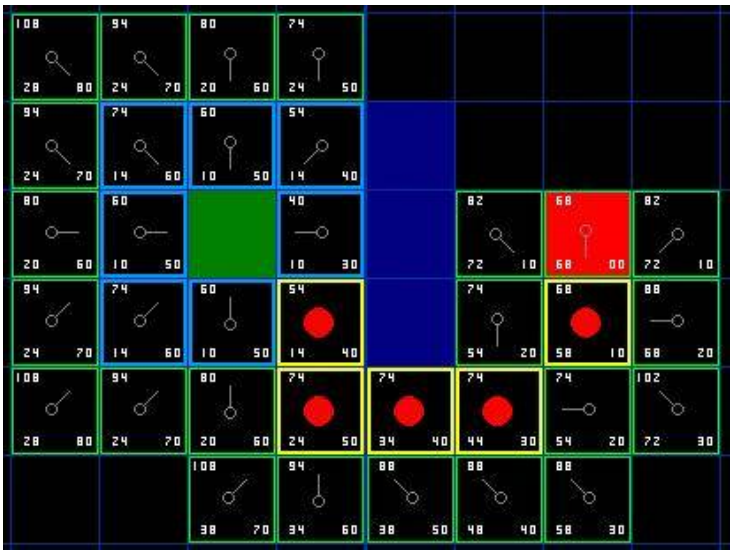
Figura 5 – Ações do algoritmo.



Fonte: (LESTER, 2005)

Ao encontrar o quadrado final o algoritmo volta de cada quadrado ao seu quadrado pai, estes quadrados formam o caminho de menor custo encontrado.

Figura 6 – Resultado do algoritmo.



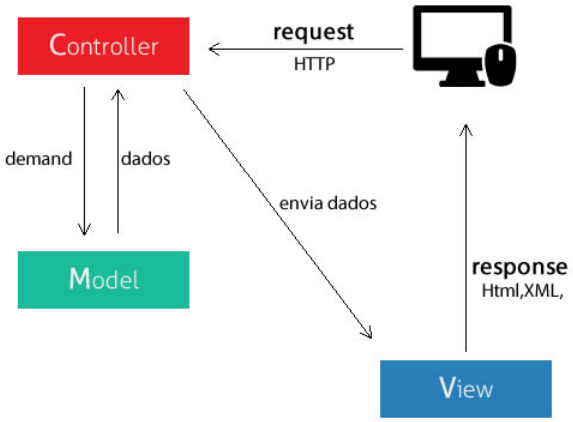
Fonte: (LESTER, 2005)

2.2 MVC

Com o aumento da complexidade dos sistemas desenvolvidos hoje em dia e o uso bem difundido da Orientação a Objetos (com características de: modularizar, generalizar, aumentar a coesão e o reaproveitamento do código, diminuir o acoplamento das partes do sistema), surgiu a ideia de criar um padrão de arquitetura de software em camadas a fim de desacoplar algumas tarefas. A arquitetura apresentada neste texto tem o nome de MVC, é uma arquitetura que possui três camadas: Modelo (Model), Visualização (View) e Controle (Controller). O desacoplamento ocorre entre as tarefas de acesso a dados e lógica da apresentação e da iteração com o usuário. Esse padrão de arquitetura foi chamado de MVC, que apesar de ter inicialmente sido desenvolvido para computação pessoal, foi posteriormente ampliado para aplicações web.

Como uma forma de facilitar o trabalho dos desenvolvedores na hora de aplicar esse novo padrão de arquitetura, frameworks pagos e gratuitos foram criados nas maiores linguagens de programação WEB. Aplicações que são desenvolvidas utilizando o padrão MVC sofrem menos impacto caso ocorra a necessidade de mudanças em uma de suas camadas, por exemplo, uma mudança no layout não afeta a camada de dados e em contrapartida, uma reorganização dos dados não irá gerar impacto no layout. Além de garantir maior aproveito de código. (SMITH, 2016)

Figura 7 – Estrutura base do padrão MVC.



Fonte: (RAMOS, 2015)

2.3 Web service

Web service é uma aplicação que é utilizada como interface entre dois ou mais programas diferentes, essencial na integração de sistemas e na comunicação entre aplicações. O objetivo dos Web Services é fornecer um conjunto de funções que, por sua vez, facilita o acesso à informação, a manipulação de dados e pode também disponibilizar ferramentas.

Web service é totalmente programado para se comunicar via rede. HTTP é o protocolo de rede mais comum entre os web services.(COSTA, 2015)

Utilizando este recurso novas aplicações conseguem interagir com outras já existentes e, também, aplicações de diferentes plataformas e linguagens conseguem se comunicar por meio de um formato intermediário como por exemplo Json, CSV ou XML que é utilizado como uma "linguagem universal" que tanto o web service quanto o sistema que está consumindo o serviço conseguem interpretar.

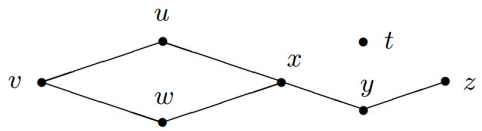
2.4 Grafos

Algumas situações podem ser representadas através de diagramas compostos por um conjunto de pontos e linhas que ligam estes pontos em pares. Por exemplo, no problema de que se trata este trabalho podemos considerar os pontos de ônibus como sendo vértices e o intervalo de linhas que passam por este ponto, ligando ele ao próximo ponto, como sendo arestas.

Como dito por Cláudio L. Lucchesi no livro “Introdução à Teoria dos Grafos” (LUCCHESI, 1979), Um grafo consiste em um conjunto de elementos chamados vértices, um conjunto finito de elementos chamados arestas e uma função de incidência que associa a cada aresta do grafo um par não ordenado de vértices (não necessariamente distintos), que são chamados de pontos extremos das arestas.

Na figura a seguir (Fig. 8) temos uma representação do grafo cujos vértices são t, u, v , w, x, y , z e cujas arestas são vw, uv , xw, xu, yz e xy.

Figura 8 – Grafo de exemplo



Fonte: (**??**)

Se o nome do grafo for G, o conjunto dos seus vértices será denotado por V(G) e o conjunto das suas arestas por A(G). O número de vértices de G é denotado por n(G) e o número de arestas por m(G); portanto,

$$n(G) = |V(G)| \text{ e } m(G) = |A(G)|.$$

Os grafos são classificados de acordo com a existência ou não de orientação, arestas múltiplas e laços de repetição. Na Tabela 1 podemos ver as classificações dos grafos de acordo com os atributos citados anteriormente, podemos também constatar que o projeto em questão pode ser representado através de um Multigrafo Orientado pois, apesar de não possuir laços, suas aresta são direcionada e podem existir mais de uma aresta ligando dois nós.

Tabela 1 – Adaptação da tabela de classificação dos grafos retirada do livro Matemática Discreta e Suas Aplicações

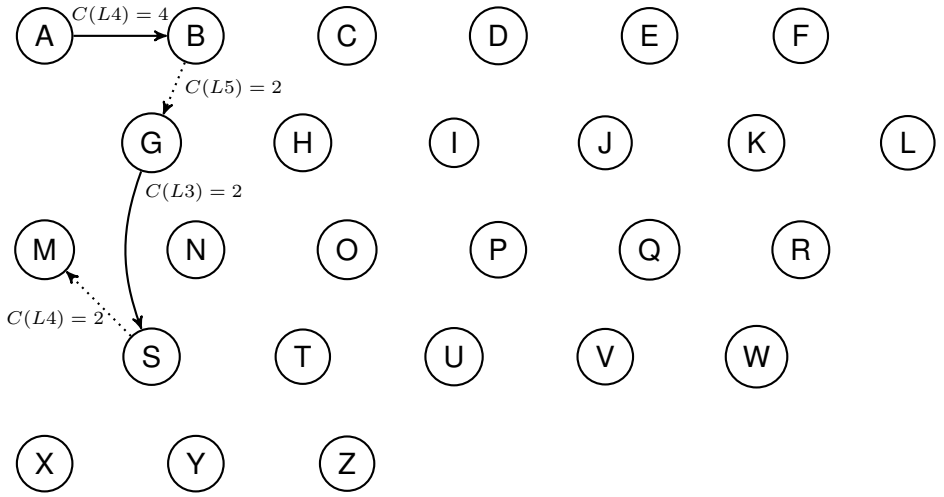
Tipo	Aresta	Arestas múltiplas?	Laços permitidos?
Grafo simples	Não dirigida	Não	Não
Multigrafo	Não dirigida	Sim	Não
Pseudografo	Não dirigida	Sim	Sim
Grafo dirigido	Dirigida	Não	Sim
Multigrafo dirigido	Dirigida	Sim	Sim

,

Fonte: (ROSEN, 2009)

A Figura 9 contém uma representação em grafo do caminho ótimo entre os pontos A e M. Observe como, além da entrada na linha inicial, três trocas de linha foram realizadas. Apesar de ser o caminho ótimo em termos de distância, não é ideal para um usuário a troca frequente de linhas de ônibus, já que além do transtorno da troca de ônibus é gerado também um custo financeiro com as passagens e um custo de tempo para realizar as trocas.

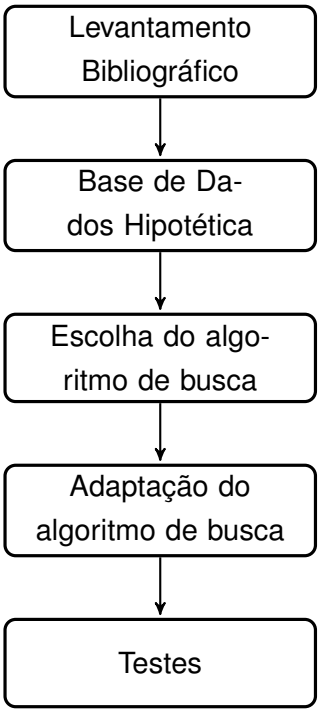
Figura 9 – Exemplo de rota traçada do ponto A ao M com $C_{troca} = 0$, com o resultado de $C_{total} = 10$ executando 3 trocas



Fonte: Elaborado pelo autor

3 Procedimentos Metodológicos

Para o desenvolvimento do projeto adotou-se alguns procedimentos metodológicos. Além do levantamento bibliográfico foram levados em consideração os procedimentos descritos nas próximas sub-sessões.



3.1 Base de dados

Neste trabalho, foi utilizado uma base de dados hipotética. Esta base contém informações dos pontos de ônibus (Código identificador, Descrição, Custo Total para chegar até ali, Latitude, Longitude, Próximos pontos alcançáveis através das linhas que passam por ele) e informações das linhas (Código identificador, Itinerário, Descrição, Tarifa, Próximos horários e os pontos de ônibus pertencentes àquela linha).

A organização dos campos, relação de dependência e demais informações à respeito da base de dados estarão melhores descritas na sessão 4.1.

3.2 Codificação do A* utilizando o grafo a ser percorrido

Depois de coletados e organizados, os dados estarão prontos para alimentar o sistema desenvolvido em C# utilizando o *framework* para construção de aplicações .NET Core 3.1. A parte mais importante de sua estrutura é uma IA que dada a origem e o destino desejado calcula as melhores rotas.

O resultado é retornado ao usuário no formato Json, a seguir podemos ver um exemplo de uma possível saída para o sistema.

```
origem:
    description: "A"
    id: 1
    latitude: -19.442932
    longitude: -42.557636
destino:
    description: "M"
    id: 13
    latitude: -19.442851
    longitude: -42.553771
results:
    0:
        route: "LD [A, B, C, I, P, V, Z, Y, X, S, M]"
        totalCost: 18
    1:
        route: "LA [B, C, D, G, E, Q, P, O, T, Z] - LD [Y, X, S, M]"
        totalCost: 45
```

Listing 3.1 – Exemplo do Json de saída

Diferentemente da implementação padrão do algoritmo A* no problema proposto não temos todos os caminhos do grafo a ser percorrido no início da execução, pois foi necessário acrescentar as arestas virtuais à medida em que o algoritmo vai avançando no caminho. Pra isso adicionamos mais um passo na lógica do processo entre os passos 3 e 4 descritos na sessão 2.1.1 deste texto. Este novo passo é responsável por adicionar à "lista aberta“ as arestas virtuais calculadas à partir do pivô e será tratado de forma mais completa na sessão 4.4.

3.3 Testes

Os testes foram feitos através da modificação dos parâmetros passados para a IA através da URL do serviço. Abaixo vemos a URL utilizada para iniciar o serviço de busca de melhor rota,

https://localhost:5001/SearchResultIA?origemId=1&destinoId=6

Os parâmetros "origemId“ e "destinoId“ definem qual o ponto de ônibus próximo à origem e qual o ponto de ônibus perto do destino.

Para garantir que a IA esteja retornando resultados satisfatórios alguns cenários específicos serão levados em consideração durante os testes, são eles:

- **Linha direta**, caso em que o melhor resultado será o usuário escolher uma única linha para ir até o ponto desejado,

- **Uma ou mais trocas**, caso em que se faz necessário uma ou mais trocas de linhas durante o trajeto da origem até o destino,
- **Sem caminho possível**, caso em que não há caminho entre a origem e o destino selecionado.

O resultado é retornado ao usuário no formato de Json e este é composto pelas informações do ponto de origem, do ponto de destino e as rotas que os ligam.

4 Desenvolvimento

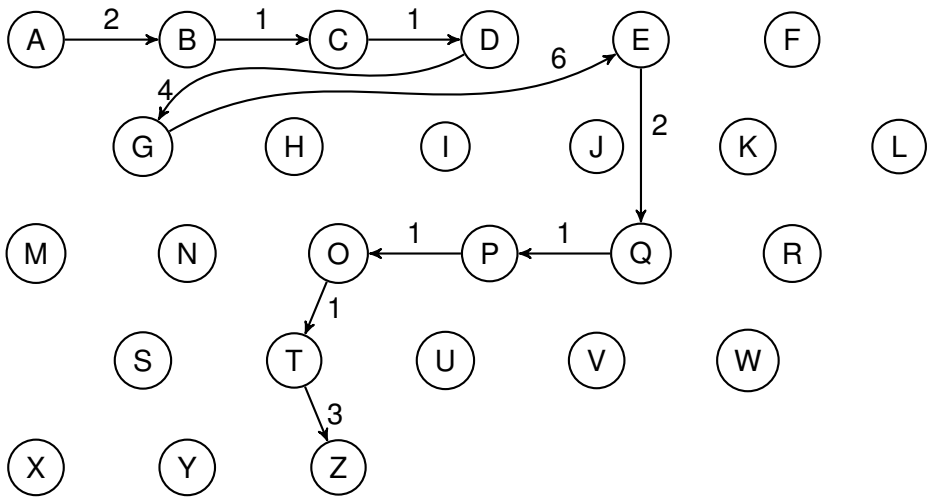
Para resolver o problema proposto neste trabalho foi criada uma IA que dado um ponto de origem e de destino mostra ao usuário sugestões de rotas para chegar ao seu destino. Por questão de maior domínio da tecnologia, foi escolhida C# como sendo a linguagem de programação utilizada para o desenvolvimento da IA.

4.1 Base de Dados

A seguir podemos ver uma representação gráfica dos pontos e das linhas que formam a base de dados utilizada do sistema.

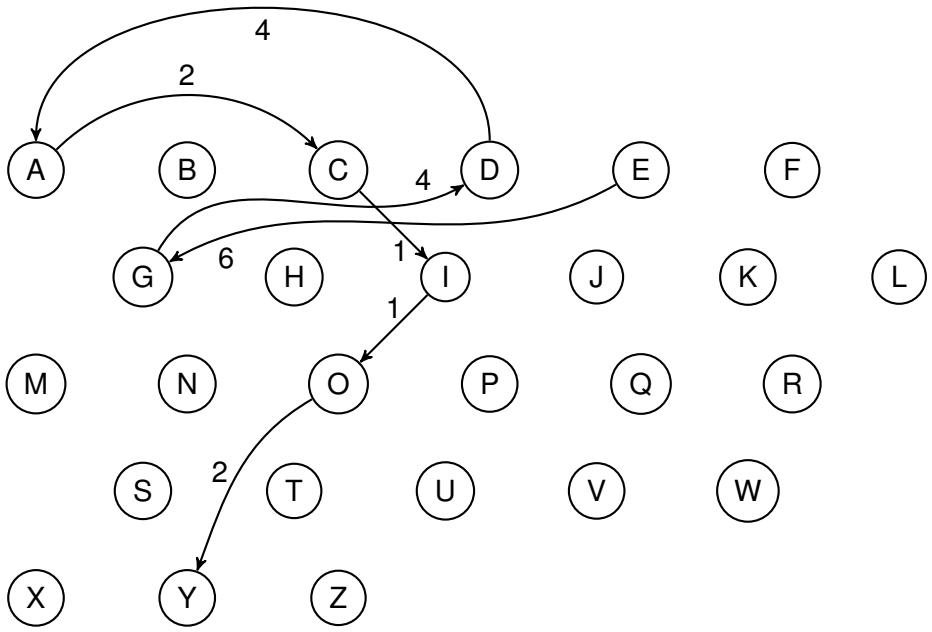
As Figuras 10, 11, 12, 13, 14 e 15 contém representações em grafo das linhas criadas como parte da base de dados e os custos associados a cada aresta em cada linha.

Figura 10 – Grafo de exemplo com as arestas da rota denominada “Linha 1”



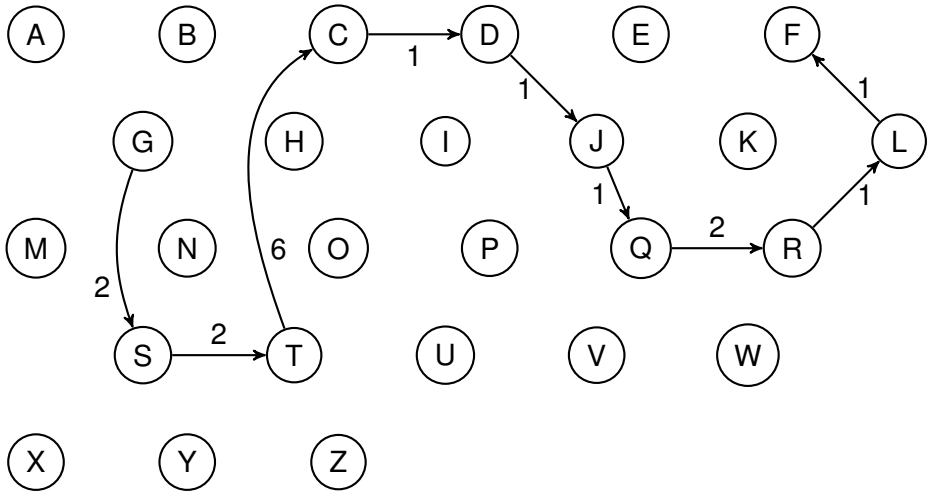
Fonte: Elaborado pelo autor

Figura 11 – Grafo de exemplo com as arestas da rota denominada “Linha 2”



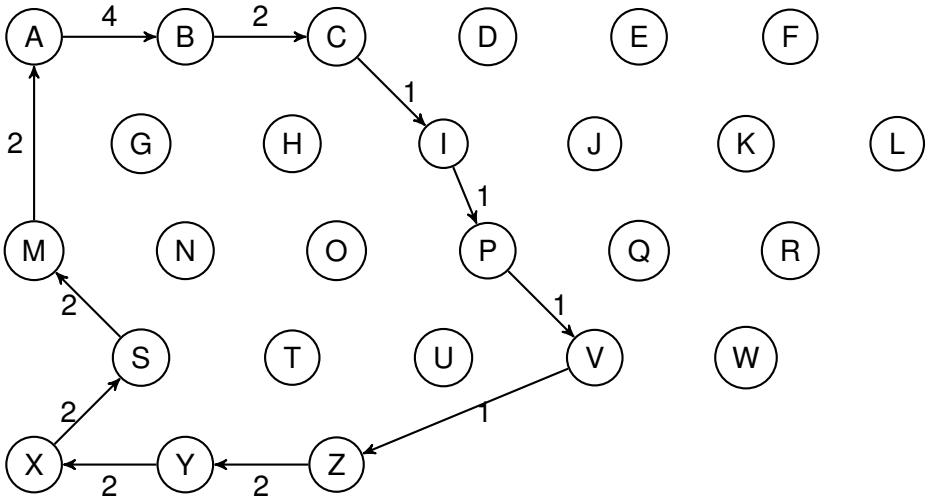
Fonte: Elaborado pelo autor

Figura 12 – Grafo de exemplo com as arestas da rota denominada “Linha 3”



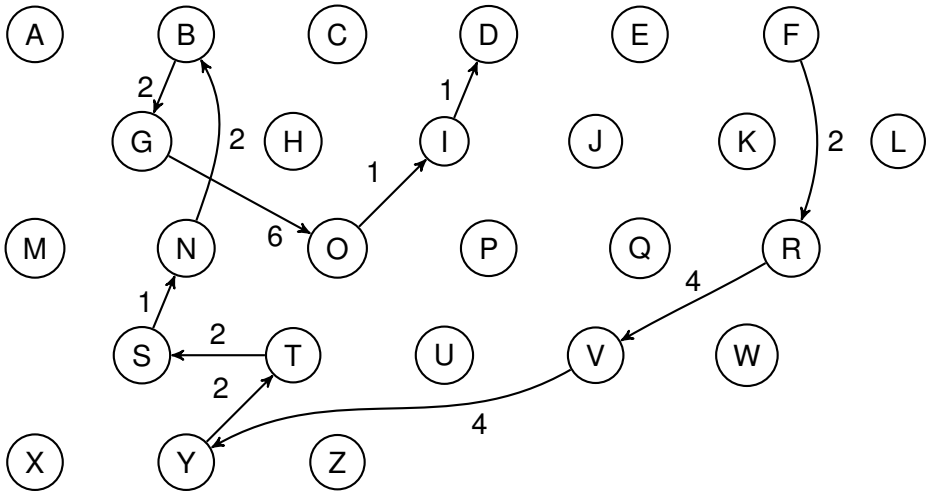
Fonte: Elaborado pelo autor

Figura 13 – Grafo de exemplo com as arestas da rota denominada “Linha 4”



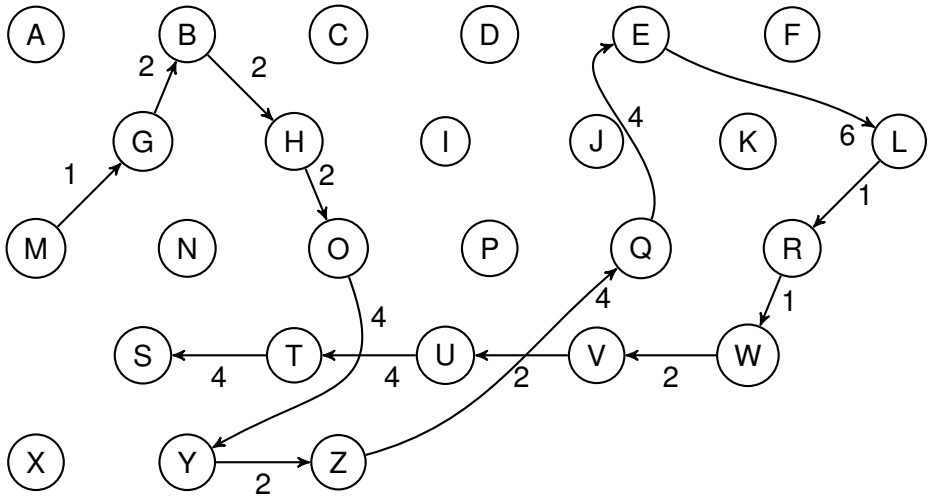
Fonte: Elaborado pelo autor

Figura 14 – Grafo de exemplo com as arestas da rota denominada “Linha 5”



Fonte: Elaborado pelo autor

Figura 15 – Grafo de exemplo com as arestas da rota denominada “Linha 6”



Fonte: Elaborado pelo autor

4.2 Métodos de escolha da melhor rota

O usuário irá informar ao sistema qual a sua origem e o seu destino, posteriormente o sistema irá montar duas listas, uma com os 2 pontos mais próximo da origem, ordenados do mais perto para o mais longe, e uma segunda lista com os 2 pontos mais próximos do destino, também ordenados do mais perto para o mais longe.

Depois de separados os pontos, o sistema aplicará os algoritmos de busca do melhor caminho para tentar resolver o problema do usuário, caso não encontre uma solução serão adicionados outros 2 pontos mais próximos às duas listas de pontos e novamente os algoritmos serão aplicados à nova lista. Essa ação se repetirá até o sistema encontrar algum caminho, ou somar 6 pontos em cada lista. Caso não encontre a alguma rota com 6 pontos em cada lista, o sistema retornará ao usuário uma mensagem dizendo que não foi possível calcular alguma rota.

Para a busca da melhor rota, por ser mais complexo, será utilizado o algoritmo de busca heurística A*. Antes de aplicar o algoritmo será necessário montar o grafo com os pontos e rotas onde este será aplicado.

O grafo será montada da seguinte forma: Os pontos de ônibus serão os vértices e segmentos das linhas serão as arestas, de cada vértice sairá um ou mais seguimentos das linhas que ligarão este ao próximo vértice (ponto de ônibus) e assim por diante. Cada segmento terá um custo e a linha que ele pertence.

O algoritmo irá percorrer os vértices do grafo, passando por suas arestas e calculando o peso da rota e quais as linhas usadas para chegar no resultado. Em cada passo a prioridade será da aresta que possui a mesma linha da anterior, a fim de escolher arestas que mudam a linha o mínimo de vezes possível, mas caso não encontre uma rota que chegue ao destino do usuário o algoritmo poderá voltar um passo e escolher uma aresta pertencente a outra linha

e se manterá nas arestas desta linha até que chegue ao fim do grafo. Este processo irá se repetir até que todo o grafo seja percorrido. Para evitar pesquisas redundantes, o algoritmo ignorará as arestas cujo a linha já foi escolhida anteriormente.

Terminada toda a pesquisa, o algoritmo retornará para o sistema um conjunto de rotas que atenderão a sua solicitação, cada rota terá os pontos, as linhas e o peso da rota. Com essas informações o sistema retornará ao usuário as rotas sugeridas.

O usuário irá informar ao sistema qual a sua origem e o seu destino, posteriormente o sistema irá montar duas listas, uma com os 2 pontos mais próximo da origem, ordenados do mais perto para o mais longe, e uma segunda lista com os 2 pontos mais próximos do destino, também ordenados do mais perto para o mais longe.

Depois de separados os pontos, o sistema aplicará os algoritmos de busca do melhor caminho para tentar resolver o problema do usuário, caso não encontre uma solução serão adicionados outros 2 pontos mais próximos às duas listas de pontos e novamente os algoritmos serão aplicados à nova lista. Essa ação se repetirá até o sistema encontrar algum caminho, ou somar 6 pontos em cada lista. Caso não encontre a alguma rota com 6 pontos em cada lista, o sistema retornará ao usuário uma mensagem dizendo que não foi possível calcular alguma rota.

Para a busca da melhor rota, por ser mais complexo, será utilizado o algoritmo de busca heurística A*. Antes de aplicar o algoritmo será necessário montar o grafo com os pontos e rotas onde este será aplicado.

O grafo será montada da seguinte forma: Os pontos de ônibus serão os vértices e segmentos das linhas serão as arestas, de cada vértice sairá um ou mais seguimentos das linhas que ligarão este ao próximo vértice (ponto de ônibus) e assim por diante. Cada segmento terá um custo e a linha que ele pertence.

O algoritmo irá percorrer os vértices do grafo, passando por suas arestas e calculando o peso da rota e quais as linhas usadas para chegar no resultado. Em cada passo a prioridade será da aresta que possui a mesma linha da anterior, a fim de escolher arestas que mudam a linha o mínimo de vezes possível, mas caso não encontre uma rota que chegue ao destino do usuário o algoritmo poderá voltar um passo e escolher uma aresta pertencente a outra linha e se manterá nas arestas desta linha até que chegue ao fim do grafo. Este processo irá se repetir até que todo o grafo seja percorrido. Para evitar pesquisas redundantes, o algoritmo ignorará as arestas cujo a linha já foi escolhida anteriormente.

Terminada toda a pesquisa, o algoritmo retornará para o sistema um conjunto de rotas que atenderão a sua solicitação, cada rota terá os pontos, as linhas e o peso da rota. Com essas informações o sistema retornará ao usuário as rotas sugeridas.

4.3 Modelagem do grafo

Dado um multigrafo valorado direcionado $G(V, E, C)$ onde V representa o conjunto de vértices, E o conjunto de arestas e C uma função que determina o custo de uma aresta,

define-se uma rota $R \rightarrow \{V_1, ..., V_N\}$ onde $N \in \mathbb{N}^+ | N \geq 2$ tal que $V_n, V_{n+1} \in V$ e $(V_n, V_{n+1}) \in E$ $\forall n \in \mathbb{N}^+, n \leq N - 1$.

Define-se também o parâmetro $C_{troca} \in \mathbb{R} | C_{troca} \geq 0$ para representar o custo associado à troca de rotas. Esse parâmetro existe com o fim de considerar os custo de trocar de linha de ônibus, como o valor da passagem e tempo de espera. Em uma aplicação real este parâmetro seria uma função que retorna um valor comparável aos valores utilizados nos pesos das arestas. Por praticidade, nesta descrição C_{troca} será tratado como uma constante.

A função $C(v_{origem}, v_{destino}, r_{utilizada})$ é definida como a função que determina o custo associado à aresta $(v_{origem}, v_{destino}) \in r_{utilizada}$.

Dadas essas definições, pode-se então definir a função de custo total de uma aresta como

$$C_{final}(v_{origem}, v_{destino}, r_{chegada}, r_{utilizada}, C_{troca}) =$$

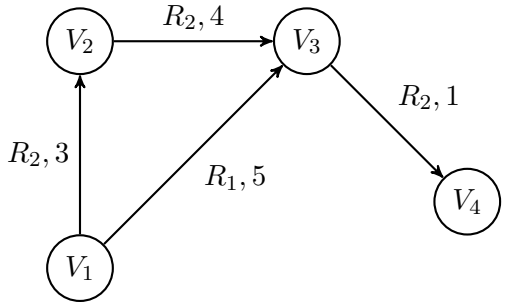
$$\begin{cases} C(v_{origem}, v_{destino}, r_{utilizada}) & r_{chegada} = \emptyset \\ C(v_{origem}, v_{destino}, r_{utilizada}) & r_{chegada} = r_{utilizada} \\ C(v_{origem}, v_{destino}, r_{utilizada}) + C_{troca} & r_{chegada} \neq r_{utilizada} \end{cases}$$

onde v_{origem} é o vértice de partida, $v_{destino}$ o vértice de destino, $r_{chegada}$ a rota utilizada para chegar ao vértice v_{origem}, $r_{utilizada}$ a rota utilizada para ir de v_{origem} a $v_{destino}$ e C_{troca} o valor do custo de uma troca de rota caso esta ocorra.

Um problema encontrado pela implementação tradicional do algoritmo A* nessa modelagem utilizando rotas é o fato do princípio da otimalidade não se aplicar em todas as situações. Dado o custo "dinâmico" das arestas, podendo ou não incluir o custo da troca de linha de ônibus, o caminho de menor custo de um dado ponto de origem V_1 a um ponto de destino V_3 passando por um ponto intermediário V_2 pode não incluir o caminho ótimo de V_1 a V_2.

Por exemplo, suponha o grafo valorado direcionado $G(V, E, C)$ com 4 vértices $V \rightarrow \{V_1,\ V_2,\ V_3,\ V_4\}$ e arestas $E \rightarrow \{(V_1, V_2), (V_1, V_3), (V_2, V_3), (V_3, V_4)\}$ com 2 rotas o percorrendo, $R_1 \rightarrow \{(V_1, V_3)\}$ onde $C(V_1, V_3, R_1) = 5$ e $R_2 \rightarrow \{(V_1, V_2), (V_2, V_3), (V_3, V_4)\}$ onde $C(V_1, V_2, R_2) = 3, C(V_2, V_3, R_2) = 4, C(V_3, V_4, R_2) = 1$. Uma representação desse grafo pode ser vista na Figura 16.

Figura 16 – Exemplo de grafo que viola o princípio da otimalidade para certos valores de C_{troca}



Fonte: Elaborado pelo autor

Neste caso, o caminho ótimo de V_1 a V_3 consiste em utilizar a aresta em R_1, com um custo total de 5. Porém, $\forall C_{troca} \in \mathbb{R} | C_{troca} > 2$ o caminho ótimo de V_1 a V_4 consistem em utilizar as arestas em R_2 por todo o percurso, totalizando um custo de 8, já que nessas condições:

$C_{final}(V_1, V_2, \emptyset, R_2, C_{troca}) + C_{final}(V_2, V_3, R_2, R_2, C_{troca}) + C_{final}(V_3, V_4, R_2, R_2, C_{troca}) < C_{final}(V_1, V_3, \emptyset, R_1, C_{troca}) + C_{final}(V_3, V_4, R_1, R_2, C_{troca})$.

$\Rightarrow C(V_1, V_2, R_2) + C(V_2, V_3, R_2) + C(V_3, V_4, R_2) < C(V_1, V_3, R_1) + C(V_3, V_4, R_2) + C_{troca}$.

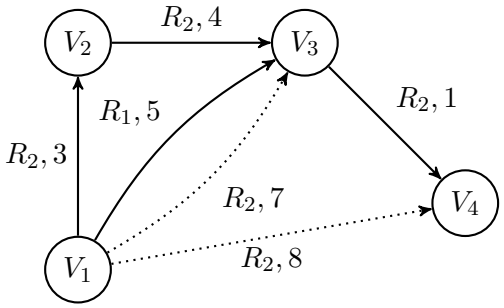
$\Rightarrow 3 + 4 + 1 < 5 + 1 + C_{troca}$

$\Rightarrow 8 < 6 + C_{troca}$

Para contornar isso, foram adicionadas às rotas o que chamou-se *arestas virtuais* para o cálculo das rotas. Define-se o conjunto das arestas virtuais de uma dada rota $R \rightarrow \{(V_1, V_2), (V_2, V_3), ..., (V_{n-1}, V_n)\}$ como $E_{virtual}(R) \rightarrow \bigcup_{i=2}^{i=|R|} \bigcup_{j=1}^{j=i-1} \{(V_i, V_j)\} - R$ e o custo de uma dada aresta virtual como $C(V_j, V_k, R) \rightarrow \sum_{i=j}^{k-1} C(V_i, V_{i+1}, R)$.

O exemplo da Figura 16 estendido para incluir as arestas virtuais pode ser visto na Figura 17.

Figura 17 – Grafo da Figura 16 estendido para incluir as arestas virtuais (em pontilhado).



Fonte: Elaborado pelo autor

4.4 Implementação do algoritmo

Nessa seção são descritas quais as técnicas empregadas na implementação do algoritmo utilizado para a determinação do melhor caminho no grafo modelado de acordo com o apresentado na seção 4.3.

O algoritmo se baseou em algoritmos amplamente difundidos na literatura, mais especificamente o algoritmo de Djisktra e o algoritmo A*. Porém, dadas certas peculiaridades do modelo abordado neste trabalho são necessárias modificações na implementação destes algoritmos. Essas modificações são descritas a seguir.

4.4.1 Cálculo do custo

A fim de ponderar o esforço de deslocamento para cada trecho entre os pontos de ônibus, adicionou-se um peso para cada intervalo da linha. Além deste peso um outro fator foi levado em consideração no calculo do custo do próximo passo do algoritmo, o custo para a troca de linha. Depois de escolhido o pivô é feito o cálculo dos pontos adjacentes que serão por onde o algoritmo vai continuar caminhando em busca do destino.

Esse custo entre dois pontos não é calculado somente baseado nos pontos sendo analisados, mas leva também em consideração qual rota está sendo utilizada, de forma a permitir a possibilidade que rotas diferente tenham custos diferentes para transitar de um ponto para outro. Enquanto a utilidade dessa variável a mais possa não se mostrar em um primeiro momento analisando um algoritmo de travessias de grafo, basta observar o fenômeno sendo modelado neste grafo para se pensar em utilidades. Entre as várias possibilidades, talvez a mais aparente seja o fato que em diferentes horários o fluxo do trânsito muda, fazendo que as linhas de ônibus possam demorar mais ou menos tempo para cobrir uma mesma distância, o que pode interferir em qual o caminho ótimo a ser tomado.

Para tomar a melhor decisão de por onde continuar, o algoritmo analisa o custo de deslocamento para os pontos adjacentes. Caso o caminho seja por uma outra linha este adiciona um custo de mudança de linha, o que faz com que trocas de linha sejam feitas somente se for o melhor caso,

Buscando trazer clareza e simplicidade ao modelo, esse custo de troca foi assumido como sendo constante. A importância do custo de troca se dá não somente pela possibilidade de modelar os custos reais associados à troca de linhas, como por exemplo o valor de custo de uma passagem de ônibus. O principal efeito desejado pela existência de um custo de troca é desestimular a constante troca entre diferentes linhas.

Em algoritmos de grafo tradicionais não há discriminação entre diferentes arestas entre um mesmo vértice, porém essa é uma distinção vital a essa implementação já que queremos permanecer em uma dada linha a maior quantidade de tempo possível. Torna-se possível então realizar o ajuste dessa constante custo de troca até que o algoritmo se comporte dentro do esperado. Durante esse estudo não foi estabelecido um procedimento teórico para a determinação do valor do custo de troca, sendo necessária a observação empírica de como sua alteração modifica o comportamento do algoritmo na base de dados que está sendo utilizada.

Em trabalhos futuros pode se modelar esse valor como sendo uma função de múltiplas variáveis, como por exemplo o ponto onde a troca ocorre e as duas rotas envolvidas nessa mudança. Essa melhoria permitiria modelar situações reais mais complexas, como por exemplo o tempo de espera envolvido entre sair do primeiro ônibus e poder entrar no segundo.

$$CustoProximoPasso = CustoDoTrecho + CustoDeTrocaDeLinha(SePertinente)$$

$$DistânciaEuclidiana = \sqrt{(ProxPonto_x + Destino_x)^2 + (ProxPonto_y + Destino_y)^2}$$

4.4.2 Arestas virtuais

Um outro fator que diferencia esta implementação de um algoritmo tradicional de travessia de grafos é o fato deste não respeitar o princípio da otimalidade. O princípio da otimalidade é a propriedade de certas soluções ótimas que afirma que para que uma solução seja ótima todas suas subestruturas também devem ser ótimas. Ou seja, em um problema de travessia de grafos de um vértice a outro, o caminho percorrido entre qualquer conjunto de dois vértices visitados também deve ser ótimo. (CORMEN et al., 2009)

Como demonstrado na seção 4.3, o modelo de arestas associadas a uma rota não respeita o princípio da otimalidade. Isso quer dizer que, sob essas circunstâncias, algoritmos como o algoritmo de Djikstra e A* não produzirão o resultado ótimo.

Para adaptar o grafo modelado de forma que este se adéque ao princípio da otimalidade foi adicionado ao algoritmo uma etapa inicial em que se realiza a criação de arestas virtuais.

Aresta virtual é o nome adotado neste trabalho a cada aresta desse conjunto que não existe diretamente no grafo mas que podem ser obtidas seguindo uma determinada rota, somando os custos das arestas percorridas entre dois vértices.

Como exemplo, assumamos a seguinte rota $R \rightarrow \{(V_1, V_2), (V_2, V_3), (V_3, V_4)\}$ onde $C(V_1, V_2, R) = 1, C(V_2, V_3, R) = 2, C(V_3, V_4, R) = 1$. Seriam criadas a partir dessa rota um conjunto de arestas virtuais $\{(V_1, V_3), (V_1, V_4), (V_2, V_4)\}$ onde $C(V_1, V_3, R) = C(V_1, V_2, R) + C(V_2, V_3, R) = 3, C(V_1, V_4, R) = C(V_1, V_2, R) + C(V_2, V_3, R) + C(V_3, V_4, R) = 4, C(V_2, V_4, R) = C(V_2, V_3, R) + C(V_3, V_4, R) = 3$.

Com a adição das arestas virtuais ao modelo inicial, as soluções produzidas para o problema de encontrar o caminho com menor custo passam a novamente observar o princípio da otimalidade. Isso ocorre devido ao fato que com a adição dessas arestas, ao procurar por uma solução o algoritmo é capaz de "pular" conjuntos de arestas que violariam o princípio da otimalidade ao tomar como caminho uma aresta virtual caso essa possua o menor custo.

No caso da aresta virtual possuir o menor custo entre dois vértices o princípio da otimalidade é respeitado por não terem sido visitados nenhum vértice intermediário, logo não há um subconjunto de vértices capaz de violar tal princípio.

Caso porém a aresta virtual não possua o menor custo entre dois vértices, isso quer dizer que existe outro caminho passando por outras arestas que possui um custo menor e portanto ótimo, o que respeita o princípio da otimalidade.

Restaurado então o princípio da otimalidade, pode ser realizado o uso de algoritmos descritos na literatura que dependem deste, como os anteriormente citados algoritmo de Djikstra e A*.

4.4.3 Utilização do A* com as arestas virtuais

A implementação proposta neste trabalho, consiste em uma adaptação do algoritmo A* para atender ao cenário do cálculo de rotas utilizando o transporte público com meio de

locomoção. Como no problema em questão existem mais de um caminho entre dois pontos foi feita uma alteração na lógica do algoritmo A* para que este possa manter o seu funcionamento e desempenho.

Além do custo de troca de linha ter sido adicionado ao custo de cada arestas uma nova etapa foi adicionada no passo 4 do algoritmo A* descrito na seção 2.1.1, na adaptação feita neste trabalho além das arestas adjacentes ao pivô foram adicionadas também na lista aberta as arestas virtuais criadas à partir do pivô e da linha atual, só depois que o algoritmo avança e escolhe o próximo pivô.

Para melhorar a performance do algoritmo, decidiu-se computar as arestas virtuais nesse momento à medida que os nós são evitados, em vez de pré-computar todas as possíveis arestas virtuais.

Antes de dar o próximo passo é aplicada uma heurística baseada na análise da distância euclidiana entre o possível próximo pivô e o destino a fim de direcionar o algoritmo a avançar no sentido do ponto final, depois de toda a análise feita o algoritmo escolhe o pivô e continua sua busca. Esses passos se repetem até que chegue ao destino ou até não haver mais vértices na lista aberta.

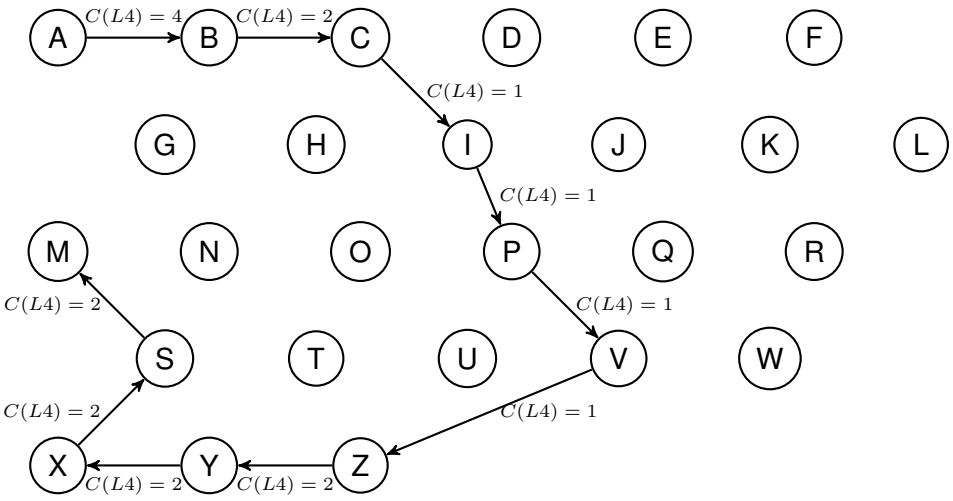
5 Resultados

O resultado obtido foi uma IA que recebe de entrada os pontos de origem e de destino do usuário e retorna o melhor caminho.

Executando as mudanças propostas na seção 4.3 ao algoritmo, podemos adicionar um custo C_{troca}, visando penalizar as trocas de linhas e incentivar o algoritmo a permanecer em uma dada linha o máximo possível, a não ser que o caminho seja impossível ou muito pior que a alternativa.

Uma rota alternativa entre os pontos A e M está representada na Figura 18, onde o custo $C_{troca} = 4$ fez com que o caminho ótimo não execute nenhuma troca de linha. A rota representada na Figura 9 possui 3 trocas, que com o novo valor de custo aumenta o custo total da rota em 12 unidades, somando 22. Em comparação, a rota da Figura 18, apesar de maior considerando somente as arestas em si, possui um custo total de 18 unidades por não efetuar nenhuma troca de linha.

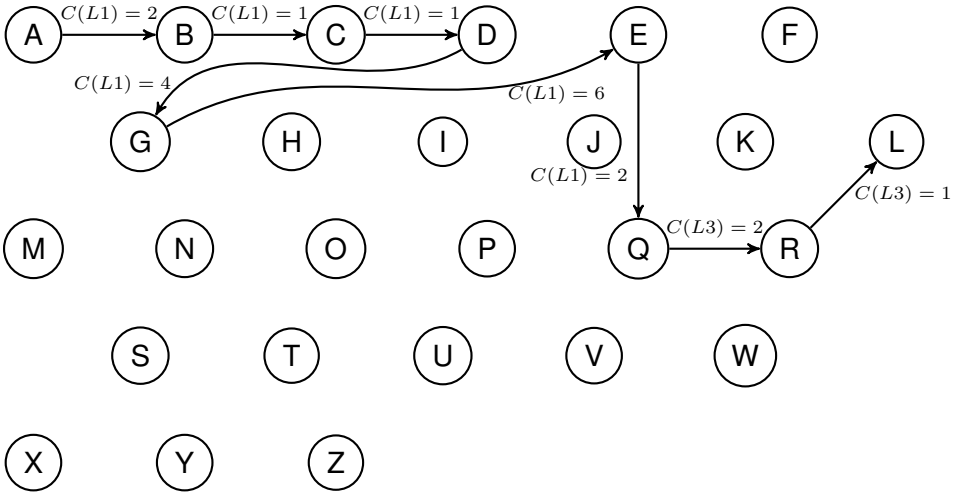
Figura 18 – Exemplo de rota traçada do ponto A ao M com $C_{troca} = 4$, com o resultado de $C_{total} = 18$ executando 0 trocas



Fonte: Elaborado pelo autor

O grafo anterior representa um dos cenários apontados na seção 3.3 em que uma linha direta seria uma melhor opção para o usuário. À seguir, na Figura 19 podemos ver o resultado da busca para os casos onde o caminho ideal esteja definido em uma ou mais trocas, é sugerido uma troca entre as linhas L1 e L3 no ponto de ônibus Q resultando no custo total do caminho de 19.

Figura 19 – Exemplo de rota traçada do ponto A ao L com $C_{troca} = 4$, com o resultado de $C_{total} = 19$ executando 1 troca



Fonte: Elaborado pelo autor

Abaixo podemos visualizar o resultado no caso de a IA não encontrar um caminho entre a origem e o destino informados.

```
origem:
    description: "A"
    id: 1
    latitude: -19.442932
    longitude: -42.557636
destino:
    description: "K"
    id: 13
    latitude: -19.442851
    longitude: -42.553771
results:
    route: "Sem resultados"
    totalCost: 0
```

Listing 5.1 – Exemplo do Json de saída caso não haja caminho possível

6 Conclusão e Considerações Finais

Tendo como objetivo o estudo do uso do algorítimo A* na busca do melhor caminho entre dois pontos e a adaptação deste algorítimo para trazer o melhor resultado quando aplicado à rotas de transporte público rodoviário, neste trabalho foi sugerida uma adaptação do algorítimo de busca A* para que seja possível resolver o problema do cálculo da melhor rota para o transporte público envolvendo múltiplas trocas de linhas. O resultado foi satisfatório tendo em vista que o algorítimo opta por manter-se na mesma linha o máximo possível ao invés de fazer muitas trocas de linha como seria no caso de implementarmos o algoritmo A* sem a modificação proposta neste trabalho.

6.1 Considerações e limitações

O projeto em questão utiliza uma base de dados estática não sendo possível a utilização de ferramentas de gerenciamento de bando de dados para editar, excluir ou adicionar novos registros e também não possui uma representação gráfica do resultado ou algum *Website* ou aplicativo para celular que facilite a iteração do usuário com a API.

6.2 Trabalhos futuros

A implementação de um banco de dados poderia ser um forte tema para trabalhos futuros assim como a criação de um *Website* ou aplicativo para celular que consumisse a API criada e retornasse ao usuário uma representação gráfica do resultado da pesquisa

Um possível tema para trabalhos futuros seria a melhoria da Inteligência Artificial atual para levar em consideração a pontuação dada às linhas pelos utilizadores do serviço, assim como a segurança e a lotação.

Referências

BandNews FM Rio. *Pesquisa aponta que brasileiros estão usando menos transporte público*. 2020. Disponível em: <https://www.band.uol.com.br/noticias/pesquisa-aponta-que-brasileiros-estao-usando-menos-transporte-publico-16310287>. Citado na página 10.

CORMEN, T. H. et al. *Introduction to algorithms*. 3. ed. Cambridge, Estados Unidos: MIT Press, 2009. Citado na página 30.

COSTA, F. *Diferença entre API e Web Service de maneira simples*. 2015. Disponível em: <https://fxcosta.wordpress.com/2015/05/31/diferenca-entre-api-e-web-service-de-maneira-simples/>. Citado na página 16.

FEIGENBAUM, E. A.; BARR, A.; COHEN, P. R. The handbook of artificial intelligence. Addison-Wesley, 1981. Citado na página 12.

GOOGLE. *Google Maps*. 2022. Disponível em: <https://www.google.com/maps>. Citado na página 9.

LESTER, P. *A* Pathfinding para Iniciantes*. 2005. Disponível em: <http://www.policyalmanac.org/games/aStarTutorial_port.htm>. Citado nas páginas 13, 14 e 15.

LUCCHESI, C. L. *Introdução à teoria dos grafos*. [S.l.]: IMPA, 1979. Citado na página 17.

RAMOS, A. *MVC – Afinal, é o quê?* 2015. Disponível em: <http://tableless.com.br/mvc-afinal-e-o-que/>. Citado na página 16.

ROSEN, K. H. *Matemática Discreta e Suas Aplicações*. 6. ed. [S.l.]: McGraw-Hill, 2009. Citado na página 18.

SIDNEI, G.; LUIS, S. Código de busca em largura e profundidade. 2018. Citado na página 12.

SMITH, S. *Overview of ASP.NET Core MVC*. 2016. Disponível em: <https://docs.microsoft.com/en-us/aspnet/core/mvc/overview>. Citado na página 16.

VASCONCELLOS, E. A.; MENDONÇA, A. Política nacional de transporte público no brasil: organização e implantação de corredores de ônibus. *Revista dos Transportes Públicos-ANTP-Ano*, p. 33–2010, 2010. Citado na página 9.