Universidade Federal de Ouro Preto
Escola de Minas
Departamento de Engenharia de Minas

Mateus Góis Camilo

SELEÇÃO DE ROTAS DE TRANSPORTE VISANDO A DIMINUIÇÃO DA EMISSÃO DE POLUENTES

OURO PRETO

2021

MATEUS GÓIS CAMILO

SELEÇÃO DE ROTAS DE TRANSPORTE VISANDO A DIMINUIÇÃO DA EMISSÃO DE POLUENTES

Trabalho de conclusão de curso apresentado ao Curso de Graduação em Engenharia de Minas da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do Título de Engenheiro de Minas

Orientador: Prof. Dr. Felipe Ribeiro Souza

OURO PRETO

2021

SISBIN - SISTEMA DE BIBLIOTECAS E INFORMAÇÃO

C183s Camilo, Mateus Gois.
Seleção de rotas de transporte visando a diminuição da emissão de poluentes. [manuscrito] / Mateus Gois Camilo. - 2021.
53 f.: il.: color., gráf., tab..

Orientador: Prof. Dr. Felipe Ribeiro Souza.
Monografia (Bacharelado). Universidade Federal de Ouro Preto. Escola de Minas. Graduação em Engenharia de Minas .

1. Minas e mineração - Transporte. 2. Combustíveis - Consumo. 3. Poluentes. 4. Algoritmos. I. Souza, Felipe Ribeiro. II. Universidade Federal de Ouro Preto. III. Título.

CDU 622.68

Bibliotecário(a) Responsável: Sione Galvão Rodrigues - CRB6 / 2526

MINISTÉRIO DA EDUCAÇÃO
Universidade Federal de Ouro Preto
Escola de Minas
Departamento de Engenharia de Minas - DEMIN

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO

Aos oito dias do mês de dezembro de 2021, às 17h00min, foi instalada a sessão pública remota para a defesa de Trabalho de Conclusão de Curso do discente **Mateus Góis Camilo**, matrícula 16.1.1344, intitulado: **"SELEÇÃO DE ROTAS DE TRANSPORTE VISANDO A DIMINUIÇÃO DA EMISSÃO DE POLUENTES"**, perante comissão avaliadora constituída pelo orientador do trabalho, Prof. Dr. Felipe Ribeiro Souza, M.Sc. Tiago Mozart e M.Sc. Juliano Tessinari Zagôto. A sessão foi realizada com a participação de todos os membros por meio de videoconferência, com base no regulamento do curso e nas normas que regem as sessões de defesa de TCC. Incialmente, o presidente da comissão examinadora concedeu ao discente 20 (vinte) minutos para apresentação do seu trabalho. Terminada a exposição, o presidente concedeu, a cada membro, um tempo máximo de 20 (vinte) minutos para perguntas e respostas ao candidato sobre o conteúdo do trabalho, na seguinte ordem: primeiro o M.Sc. Tiago Mozart, segundo o M. Sc. Juliano Tessinari Zagôto em último, o Prof. Dr. Felipe Ribeiro Souza. Dando continuidade, ainda de acordo com as normas que regem a sessão, o presidente solicitou ao discente e aos espectadores que se retirassem da sessão de videoconferência para que a comissão avaliadora procedesse à análise e decisão. Após a reconexão do discente e demais espectadores, anunciou-se, publicamente, que o discente foi Aprovado por unanimidade, com a nota 8,5 (Oito e meio). O discente, por sua vez, encaminhará para o Repositório Institucional da UFOP, no prazo máximo de 15 (quinze) dias, uma versão final, contemplando todas as recomendações apresentadas pelos avaliadores. Para constar, foi lavrada a presente ata que, após aprovada, foi assinada pela presidente da comissão. Ouro Preto, 08 de dezembro de 2021.

Felipe Ribeiro

Presidente: Prof. Dr. Felipe Ribeiro Souza

Membro: M.Sc. Tiago Mozart

Membro: M.Sc. Juliano Tessinari Zagôto

Discente: Lucas Henrique Castanheira

Aos meus pais, João Carlos e Elza,

pelo apoio e amor.

AGRADECIMENTOS

Pelo apoio incondicional, amor, dedicação e esforço imensurável, agradeço a meus pais.

Aos meus irmãos, João Pedro, Maria Clara e Alice, agradeço por todos os momentos, das brigas às risadas.

Pela parceria, carinho, confiança e amor durante toda a minha caminhada, agradeço à Letícia.

Ao Divino e PCC, meus amigos, por todas as risadas, momentos e lembranças. Em especial, Samuel e Bernardo, pelos momentos juntos em Ouro Preto.

Ao meu orientador Felipe Ribeiro Souza, pelo apoio, incentivo e ensinamentos passados.

À Minera Jr. por tanto aprendizado, crescimento e experiências únicas. Avante Minera.

À Hatch e MiningMath pelos conhecimentos recebidos durante o estágio. Momentos únicos que, com certeza, contribuíram ainda mais para que eu me tornasse um profissional ainda completo.

Por fim, ao DEMIN, agradeço não somente pelos conhecimentos passados como também pelas pessoas, amigos que fiz e levarei por toda a vida.

RESUMO

A definição de rota de transporte em minerações é feita a partir dos conhecimentos práticos dos técnicos e engenheiros selecionados para o trabalho. Uma correta determinação provoca uma economia grande nos gastos, enquanto uma má seleção provoca prejuízos financeiros consideráveis. Este trabalho buscou estabelecer uma metodologia de cálculo do caminho de custo e poluição mínima do bloco lavrado até a usina de beneficiamento, por meio de quatro algoritmos de construção de caminhos, paralelamente a dois métodos de cálculo de distância diferentes. O planejamento e execução foi feito utilizando os softwares Spyder (liberado de maneira gratuita como parte da suíte Anaconda), Microsoft Excel e Datamine Studio OP. Os algoritmos de busca estão também disponibilizados na internet, de maneira gratuita, nas bibliotecas da linguagem de programação Python. O algoritmo ótimo, definido comparativamente, foi o Shortest Path, uma vez que apresentou resultados 9,8% e 32,76% menores, segundo o método de cálculo de menor consumo e de distância euclidiana, respectivamente. Quando os cálculos foram feitos utilizando a definição de rota por menor consumo os resultados foram 25,45% menores em relação à seleção por meio da distância euclidiana para todos os algoritmos, exceto o Shortest Path, que não apresentou variação de consumo para diferentes métodos de escolha de rota. Análises com outros algoritmos e topografia é sugerida para verificar a eficácia do trabalho.

Palavras-chave: consumo de combustível, emissão de poluentes, spyder, shortest path

ABSTRACT

The definition of transport route in mining is based on the practical knowledge of the technicians and engineers selected for the job. A correct determination causes great savings in expenses, while a bad selection causes considerable financial losses. This work sought to establish a methodology for calculating the path calculation path and the minimum calculation from the mined block to the processing plant, using four path construction algorithms, in parallel with two different distance calculation methods. Planning and execution were done using Spyder software (free of charge as part of the Anaconda suite), Microsoft Excel and Datamine Studio OP. Search algorithms are also available on the internet, free of charge, in Python programming language libraries. The optimal algorithm, defined comparatively, was the Shortest Path, since it presented results that were 9.8% and 32.76% lower, according to the calculation method of lower consumption and Euclidean distance, respectively. When the calculations were made using a route definition by least consumption, the results were 25.45% lower compared to selection using the Euclidean distance for all algorithms, except the Shortest path, which does not vary in consumption for different methods of choice of route. Analysis with other algorithms and topography is suggested to verify the effectiveness of the work.

Keywords: fuel consumption, GHG emissions, spyder software, shortest path

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A rota de transporte em uma operação mineira influencia fortemente o consumo de combustível e a emissão dos poluentes, frequentemente o traçado do transporte é determinado com base no conhecimento prático do engenheiro. A correta determinação da rota de transporte e os custos associados são de grande importância, pois equívocos na estimativa de custo podem resultar em perdas financeiras, segundo Barbosa (2010), na mineração, o setor de transporte pode contribuir em até 40% dos custos totais da produção, a depender do método de lavra adotado. Existe, portanto, uma correlação direta entre o consumo de combustível e a emissão de particulados e gases poluentes pelos equipamentos de transporte.

Caminhões utilizados nas operações mineiras fazem uso de muita energia (DOE, 2002) e isso incentivou produtores de caminhões e as maiores mineradoras do mundo a dedicarem vários projetos de pesquisa na eficiência energética de caminhões de transporte (EEO, 2012) e (SOOFASTAEI *et al*, 2015). Otimizar a eficiência de sistemas de transporte é um dos maiores desafios da engenharia de minas e é assunto de muitas pesquisas e projetos na indústria mineral (ERCELEBI e BASCETIN, 2009) e (LIMSIRI, 2011). Utilizando-se de modernidade e tecnologias largamente aplicadas na indústria é possível definir, ou buscar definir, rotas para tais sistemas de rodagem de maneira a gerar uma economia grande nos consumos de combustível e, consequentemente, na emissão de poluentes.

Surgidos a partir do chamado “Problema de caminho mínimo”, diversos algoritmos têm buscado solucionar e permitir a otimização nesse campo de pesquisa das mineradoras. O problema de caminho mínimo fala sobre a busca do menor caminho de distância entre dois pontos, baseado em um sistema de grafos: conjuntos de vértices e arcos utilizados para determinar e representar objetos no espaço.

Da necessidade de resolução desses grafos, bem como da determinação de um caminho mais econômico, atrelada à evolução da computação foram levantadas diversas metodologias matemáticas para a obtenção de respostas bem definidas. Surgiram-se então os primeiros algoritmos matemáticos e de programação para computadores. O mais conhecido e tido como o precursor de algoritmos mais avançados, é o chamado algoritmo de Dijkstra, inventado pelo cientista da

computação holandês Edsger Dijkstra em 1956. Este algoritmo era capaz de solucionar o famoso problema por meio de grafos com arestas de peso não negativo, identificando a partir do nó inicial qual o custo mínimo entre esse e todos os outros nós do grafo (WILHELM, 2020). A partir de Dijkstra vários outros algoritmos foram criados, propondo a solução para o que não seria capaz por este código, como por exemplo, grafos nos quais os pesos aplicados às arestas eram negativos, como os algoritmos de Bellman Ford, e o algoritmo A*, que parte da combinação entre os pesos das arestas e pesos dos nós para construir o caminho dos grafos a serem operacionalizados

Devido a toda importância vista e larga empregabilidade dos algoritmos de busca na criação e definição de rotas para os equipamentos de mineração diversas pesquisas no ramo têm sido desenvolvidas. Além de usos de sistemas de computação que podem provocar economia de combustível, outros meios também permitem essa economia como o uso de sistemas mecânicos para recuperação e reaproveitamento de energia, alterações mecânicas, uso de sistemas de despacho, etc. O presente estudo abordará somente um dos métodos, levando em consideração uma malha topográfica genérica e tendo como base para estudo as dimensões do caminhão CAT 770G.

O estudo se passa a partir da comparação entre quatro diferentes metodologias de cálculo para a definição de caminho: A*, Dijkstra, Shortest Path e Bellman Ford. O uso de mais de um algoritmo permite conhecer as singularidades de cada um e definir quais deveriam ser utilizados para determinada parte de terreno. Estudos futuros combinando mais de uma metodologia também podem surgir a partir deste estudo, permitindo, por fim, a criação de um sistema ótimo de despacho por meio da combinação de metodologias já conhecidas. Tal criação pode facilitar a revisão e desenvolvimento contínuo do sistema pelo grande conhecimento já existente acerca dos códigos.

2. OBJETIVOS

2.1. Objetivo geral

Analisar a aplicabilidade de algoritmos de caminho mínimo para a otimização do consumo de combustível em uma determinada superfície topográfica, e consequente redução da emissão de poluentes na atmosfera.

2.2. Objetivos específicos

- Determinar os algoritmos para melhor aplicação ao modelo;
- Determinar o código em linguagem Python para aplicar o algoritmo em uma topografia aleatória;
- Aplicar o programa definido tendo como base as equações de distância para seleção de rota;
- Analisar e comparar o consumo de combustível a partir das rotas definidas entre um mesmo algoritmo;
- Analisar e comparar o consumo de combustível a partir das rotas definidas entre as mesmas equações de distância.

3. REVISÃO BIBLIOGRÁFICA

3.1. GRAFO

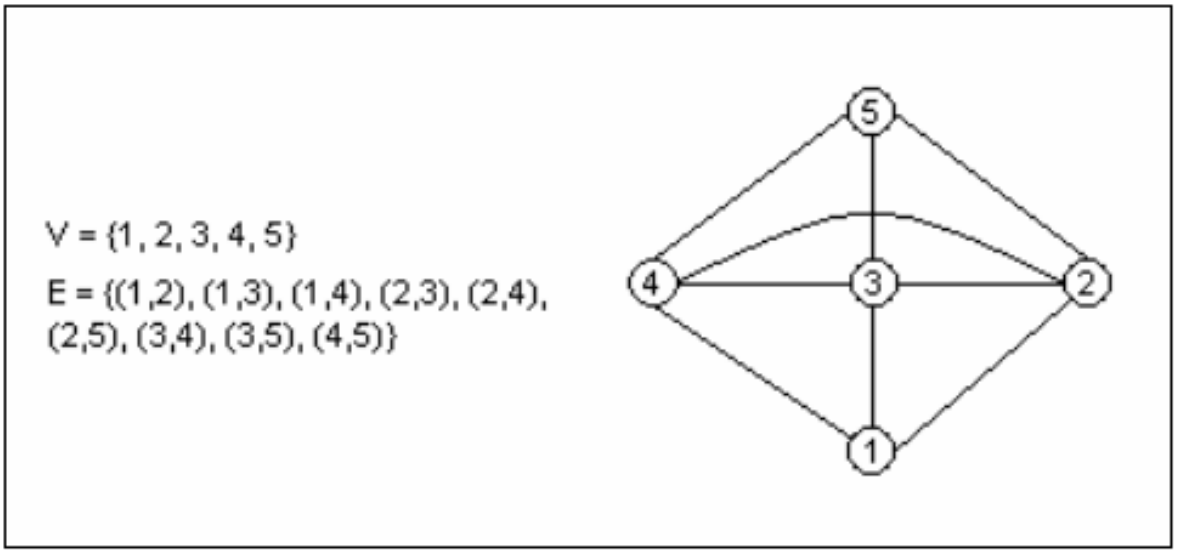
De um modo geral, a área de algoritmos em grafos pode ser caracterizada como aquela cujo interesse principal é resolver problemas de uma forma computacional. O objetivo principal é encontrar algoritmos para resolver um problema em grafos, de forma eficiente se possível.

A teoria dos grafos tem contribuído para a análise de uma ampla variedade de problemas combinatórios que pode ser utilizado para resolver problemas de análise de caminho crítico, sistema de comunicação, estudo de transmissão de informações, escolha de uma rota ótima, entre outros.

Segundo Szwarcfiter (1984), um grafo G (V, E) é definido como um conjunto finito não-vazio V e um conjunto E de pares não-ordenados de elementos distintos de V. Os elementos de V são os vértices e os de E são as arestas de G, respectivamente. Cada aresta є e E será denotada pelo par de vértices e = (v, w) que a forma. Nesse caso, os vértices v e w são os extremos (ou extremidades) da aresta e, sendo denominados adjacentes. A aresta є é incidente a ambos v e w.

Um grafo pode ser visualizado através de uma representação geométrica, na qual seus vértices correspondem a pontos distintos do plano em posições arbitrárias, enquanto cada aresta (v, w) é associada uma linha arbitrária unindo os pontos correspondentes a v e w. Um exemplo de grafo pode ser visto na Figura 1.

Figura 1 - Grafo e sua representação geométrica



Fonte: Szwarcfiter (1984)

Do ponto de vista geométrico, um grafo pode ser descrito, em um espaço euclidiano de n dimensões, como sendo um conjunto V de pontos e conjunto A de curvas contínuas que não se interceptam, satisfazendo as seguintes condições (Furtado, 1973):

a) toda curva fechada de A contém exatamente um ponto de V;

b) toda curva aberta de A contém exatamente dois pontos de V;

c) as curvas de A não têm pontos em comum, a não ser pontos de V.

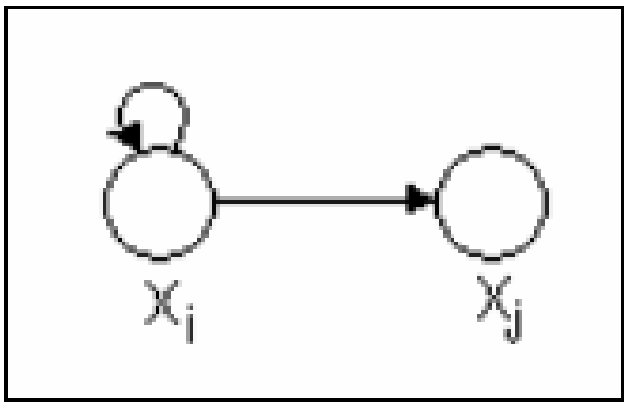
Segundo Sedgewick (1988), o número de arestas de um grafo pode variar de 0 até ½ v (v – 1), onde v é o número de vértices. Grafos com todas as arestas são chamados de grafos completos. Grafos com alguns vértices (até v log v) são denominados de esparsos. Grafos com quase todas as arestas são chamados de densos. Existem ainda os grafos direcionados e os grafos valorados, descritos nas próximas seções.

3.2. GRAFOS DIRECIONADOS

Um grafo direcionado (dígrafo) D (V, E) é um conjunto finito não vazio V (os vértices), e um conjunto E (as arestas) de pares ordenados de vértices distintos. Portanto, num dígrafo cada aresta (v, w) possui uma única direção de v para w. Dessa forma, (v, w) é divergente de v e convergente a w (Szwarcfiter, 1984).

Cada arco é representado por uma seta cujo sentido corresponde à orientação do par ordenado. Um exemplo de grafo direcionado pode ser visto na Figura 2.

Figura 2 - Grafo orientado



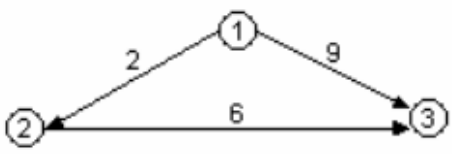
Fonte: Szwarcfiter (1984)

Em um grafo orientado, um arco de forma (xi, xi) é possível e chama-se laço. Porém, não é possível definir um sentido desse arco. Dessa forma, um vértice não pode possuir dois laços de sentidos opostos (Boaventura, 1979).

3.3. GRAFO VALORADO

Um grafo, no qual um número wij está associado a cada aresta, é denominado de grafo valorado e o número wij é chamado o custo da aresta. Em redes de comunicação ou transportes estes custos representam alguma quantidade física, tal como distância, eficiência, capacidade da aresta correspondente, etc (Rabuske, 1992). Um exemplo de grafo valorado pode ser visto na Figura 3.

Figura 3 - Exemplo de grafo valorado



Fonte: Szwarcfiter (1984)

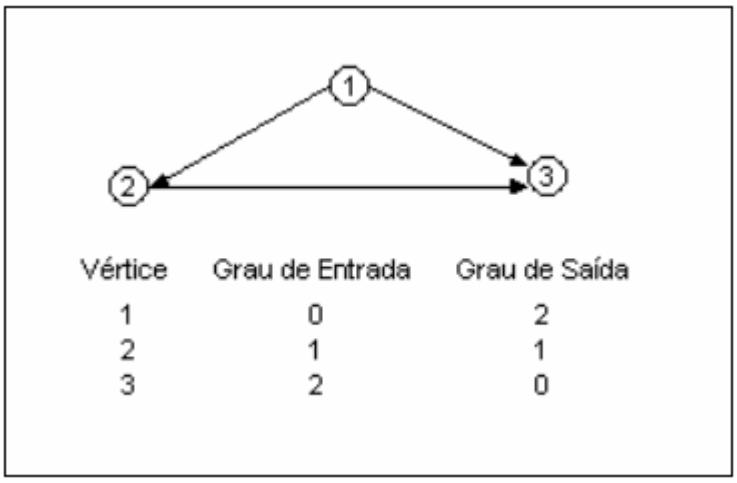
3.4. RELAÇÕES DE ADJACÊNCIA

Dois vértices v e w são ditos adjacentes em um grafo G se existe em G a aresta {v, w}. A relação de incidência é definida entre um vértice e uma linha: uma linha é incidente a um vértice se o vértice é uma de suas extremidades.

Em grafos dirigidos, um arco (v, w) é dito exteriormente incidente a v e interiormente incidente a w (Furtado, 1973)

Seja D (V, E) um dígrafo e um vértice v ϵ V. O grau de entrada de v é o número de arestas convergentes a v. O grau de saída de v é o número de arestas divergentes de v. Uma fonte é um vértice com grau de entrada nulo, enquanto um sumidouro (ou poço) é um com grau de saída nulo (SZWARCFITER, 1984). Na Figura 4, o vértice 1 é uma fonte e o vértice 3 é um sumidouro.

Figura 4 - Exemplo de grau



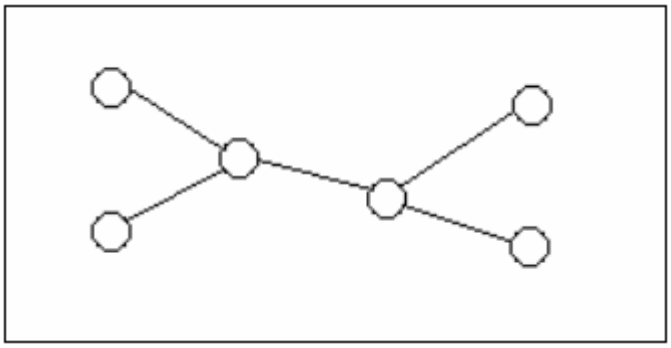
Fonte: Szwarcfiter (1984)

3.5. ÁRVORES

Um grafo que não possui ciclos é chamado acíclico. Denomina-se árvore a um grafo T (V, E) que seja acíclico e conexo. Se um vértice v da árvore T possui Grau <= 1 então v é uma folha (Szwarcfiter, 1984).

Um grafo G é uma árvore se e somente se existir um único caminho entre cada par de vértices de G. Segundo Rabuske (1992), um caminho é qualquer sequência de arestas onde o vértice final de uma aresta é o vértice inicial da próxima. Um exemplo de árvore pode ser visto na Figura 5.

Figura 5 - Árvore



Fonte: Szwarcfiter (1984)

3.6. REPRESENTAÇÃO DE GRAFOS

Segundo Szwarcfiter (1984), a representação de grafos adequados ao uso em computador tem como objetivo fornecer estruturas que correspondam univocamente a um grafo dado e possam ser armazenadas sem dificuldade, em um computador.

Dentre as várias representações para computador, as mais importantes são as representações matriciais e as por listas.

Na representação matricial é utilizada a matriz de adjacências. Em um grafo G (V, E) a matriz de adjacências R = (rij) é uma matriz n x x conforme especificado na Equação 1.

Equação 1 - Regra matricial para a definição de representação de grafos

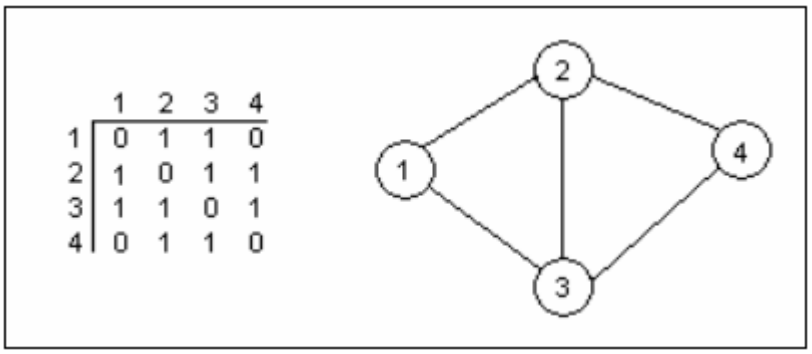
$$rij = 1 \Leftrightarrow (v_i, v_j) \in E$$

$$rij = 0, caso\ contrário$$

Ou seja, rij = 1 quando os vértices vi,vj forem adjacentes e rij = 0, caso contrário. Uma matriz de adjacências caracteriza univocamente um grafo. Um exemplo de matriz de adjacências pode ser visto na

Figura 6.

Figura 6 - Matriz de adjacências

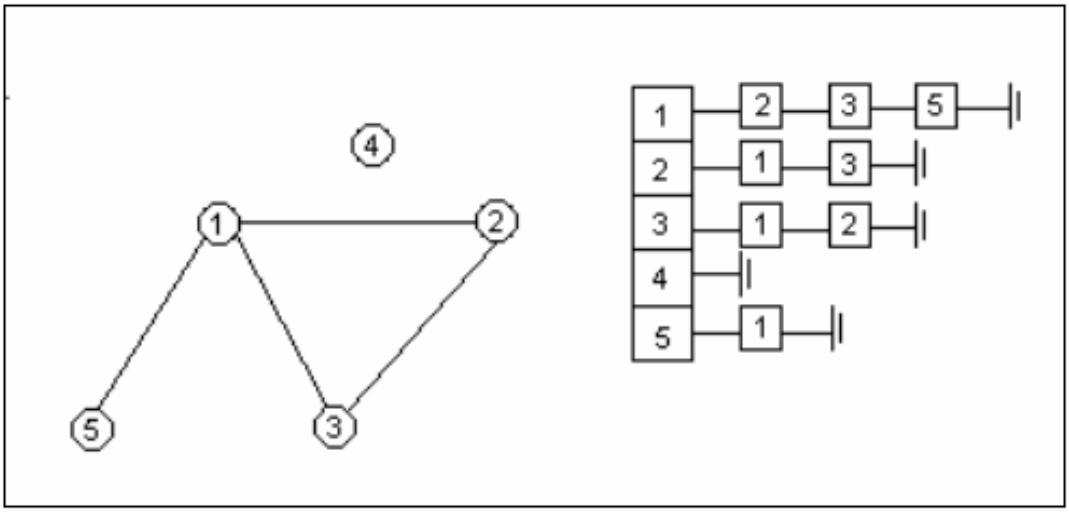


Fonte: Szwarcfiter (1984)

A principal desvantagem desta representação é o espaço necessário para o armazenamento da matriz. Essa estrutura gera uma alocação de memória desnecessária que faz com que o processo seja mais lento. Em qualquer caso existem n2 informações binárias, o que significa um espaço 0(n2). Segundo Sedgewick (1988), a matriz de adjacências é recomendada apenas para grafos densos.

Uma das principais representações de grafos por listas é a estrutura de adjacências. Seja G (V, E) um grafo. A estrutura de adjacências A de G é um conjunto de n listas A(v), uma para cada v ϵ V. Cada lista A(v) é chamada de lista de adjacências do vértice v, e possui os vértices w adjacentes a v em G. Ao adicionar uma conexão de um vértice v a um vértice w, o vértice v é adicionado na lista adjacente de w e o vértice w é adicionado na lista adjacente de v. Um exemplo de estrutura de adjacências pode ser visto na Figura 7.

Figura 7 - Estrutura de adjacências

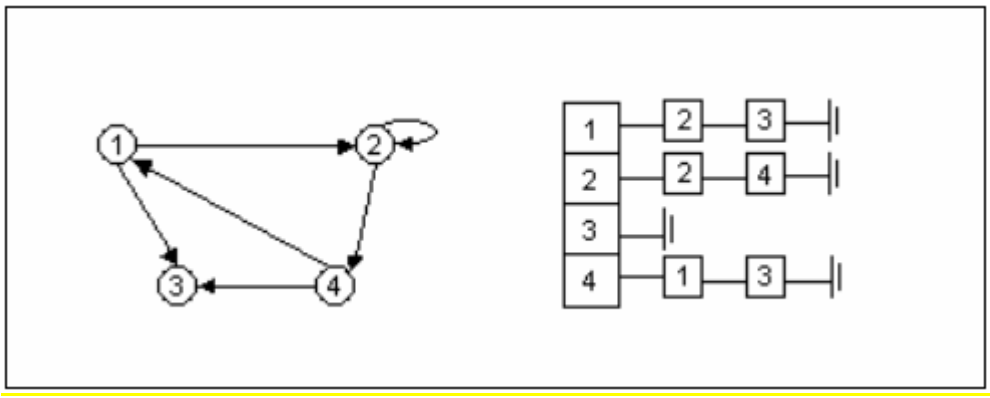


Fonte: Szwarcfiter (1984)

A estrutura de adjacências é a representação mais utilizada nas implementações em algoritmos de grafos, pois o espaço utilizado por uma estrutura de adjacências é 0 (n + m), ou seja, linear com o tamanho de G, onde n é o número de arestas e v é o número de vértices.

Segundo Sedgewick (1988), grafos dirigidos e valorados são representados com estruturas semelhantes. Para grafos dirigidos a única alteração é que cada conexão é representada apenas uma vez: uma conexão de um vértice v para um vértice w é representada por um valor *true* em [v, w] na matriz de adjacências e na representação por lista de adjacências o w está na lista de adjacências de v. A Figura 8 é um exemplo de uma estrutura de adjacências para um grafo dirigido.

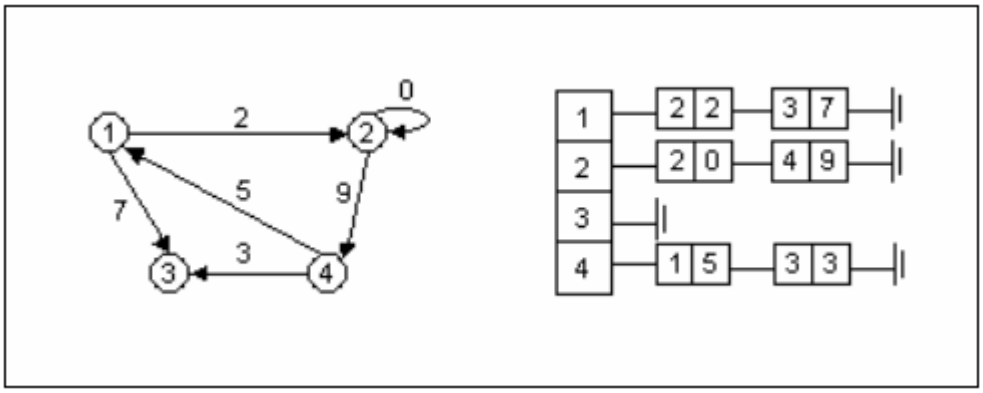
Figura 8 - Estrutura de adjacências para um grafo dirigido



Fonte: Szwarcfiter (1984)

Para grafos valorados a única alteração em uma matriz de adjacências é que se utiliza o valor ao invés de valores booleanos (utiliza-se um valor inválido para representar false); em uma estrutura de adjacências é incluído um campo para armazenar o valor em cada item da lista de adjacências. O espaço utilizado por esta estrutura de adjacência é n + 2m. A Figura 9 é um exemplo de uma estrutura de adjacência para um grafo dirigido valorado contendo o campo adicional para armazenar o valor de cada aresta.

Figura 9 - Estrutura de adjacências para um grafo dirigido e valorado



Fonte: Szwarcfiter (1984)

3.7. BUSCA EM GRAFOS

Segundo Szwarcfiter (1984), a busca é uma das técnicas mais importantes para a solução de problemas algorítmicos em grafos, devido ao grande número de problemas que a busca pode resolver. Essa importância cresce ainda mais, quando se procura algoritmos considerados eficientes.

A busca procura resolver o problema de explorar um grafo, ou seja, como caminhar pelos seus vértices e arestas.

Quando o grafo é uma árvore, a busca é mais simples. Uma das formas de fazer essa busca é denominado de preordem, seguindo os seguintes passos:

a) Se a árvore for vazia, não há nada a fazer. Caso contrário:

b) Visite a raiz da árvore,

c) Caminhe pela subárvore mais à esquerda da raiz,

d) após pela 2a mais à esquerda,

e) após pela 3a mais à esquerda,

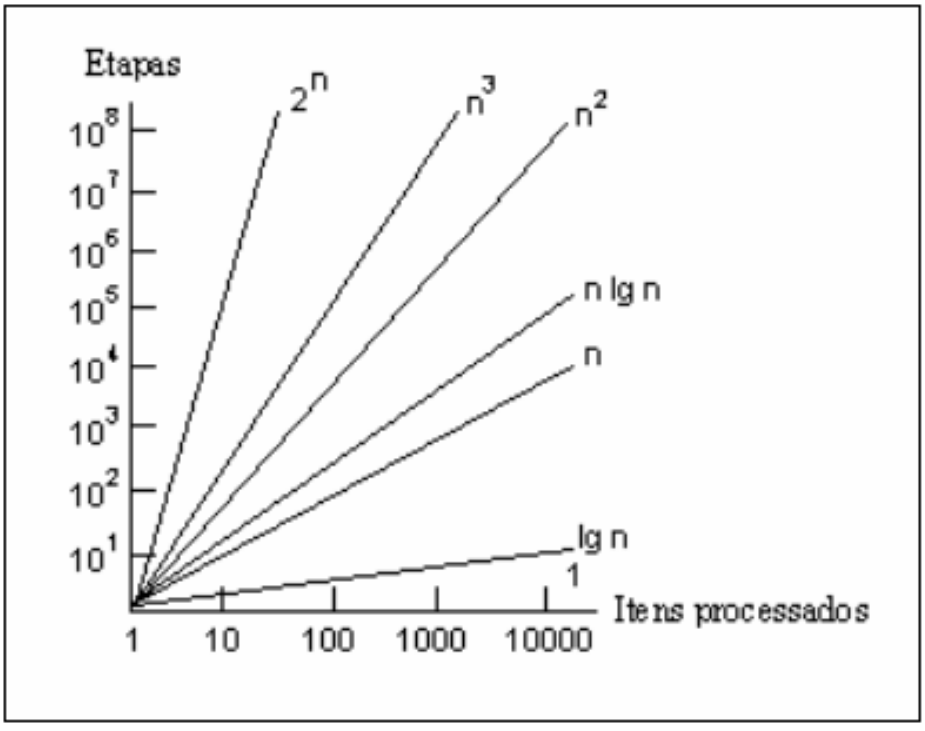
f) e assim por diante.

Outra forma de se caminhar em árvores enraizadas é conhecido como ordem de nível. Neste processo, o caminhamento é nível a nível, da esquerda para direita.

Quando o grafo não é uma árvore, a busca é mais difícil, pois não existem, de forma absoluta, os conceitos de esquerda, direita e nível. A principal dificuldade em relação aos algoritmos de busca diz respeito à complexidade do problema estudado.

Segundo Lafore (1999), a notação Big O é utilizada para dizer quão eficiente é um algoritmo. A figura 10 mostra a relação entre o crescimento do número de etapas que um algoritmo executa e o número de itens que ele processa.

Figura 10 - Big O



Fonte: Lafore (1999) baseado em Kruse (1994).

A Figura 11 mostra a quantidade de passos executados em relação ao número de itens (n) processados.

Figura 11 - Big O

N	1	lg n	N	n lg n	n^2	n^3	2^n
1	1	0,00	1	0	1	1	2
10	1	3,32	10	33	100	1.000	1024
100	1	6,64	100	66	10.000	1.000.000	1.268×10^{30}
1000	1	9,97	1000	997	1.000.000	10^9	1.072×10^{301}

Fonte: Lafore (1999) baseado em Kruse (1994).

3.8. ALGORITMO

Dentre todas as definições possíveis de um algoritmo, podemos defini-lo amplamente como "(...) uma estrutura de codificação que transforma a entrada de dados em saída ideal com base em um cálculo específico" (GILLISPIE, 2013, p.1 apud RAMOS, 2017, página 78). Ou ainda, de acordo com sua definição clássica e básica, um algoritmo nada mais é do que um conjunto limitado de regras organizadas para resolver um problema específico ou realizar uma tarefa específica (SILVEIRA, 2018). Ou, "Um algoritmo é uma receita, uma série de instruções, uma série de tarefas para atingir um resultado ou cálculo específico, como as etapas necessárias para calcular a raiz quadrada da sequência de Fibonacci."(FINN, 2017).

Dentre as técnicas existentes para a solução de problemas algorítmicos em grafos, a busca ocupa lugar de destaque, pelo grande número de problemas que podem ser resolvidos através da sua utilização (SZWARCFITER, 1984). A busca consiste em localizar um vértice dentro de um grafo, traçando o caminho percorrido para se deslocar de um vértice inicial até o vértice que se deseja procurar. Os dois métodos mais comuns que existem são a Busca em Largura e a Busca em Profundidade.

3.9. BUSCA EM LARGURA

O algoritmo de Busca em Largura é uma proposta que trabalha sob o critério FIFO (First In First Out, isto é, o primeiro que entra é o primeiro que sai). É também denominada de busca em amplitude, busca em nível e “*breadth-first*” (RABUSKE 1995).

Funciona da seguinte forma: “Aplica-se todos os operadores possíveis à raiz (isto é, ao estado inicial), em seguida, a todos os filhos da raiz (da esquerda para a direita), depois, a todos os “netos” da raiz (da esquerda para a direita) e assim por diante. A busca cessa ao se descobrir o nó objetivo.” (COLLINS 1988).

Existem duas filas de trabalho usadas pela busca em largura. A fila de abertos, que guarda todos os vértices do grafo a serem explorados; e a fila de fechados, onde se encontram todos os vértices que já foram explorados.

Neste algoritmo, todos os vértices de certo nível da árvore são examinados antes do nível abaixo. Se existe uma solução e se o grafo é finito, então por este método ela certamente será encontrada.

Este método não foi aplicado para este trabalho, especificamente. Todos os algoritmos utilizados faziam busca por meio da metodologia de “Busca em Profundidade”

3.10. BUSCA EM PROFUNDIDADE

O algoritmo de Busca em Profundidade é uma proposta que trabalha sob o critério LIFO (*Last In First Out*, isto é, o último que entra é o primeiro que sai). É também denominada de primeiro-em-profundidade ou “*depth-first*” (RABUSKE, 1995).

Quanto à estrutura do algoritmo, a única diferença em relação à busca em largura, é a utilização de uma pilha ao invés de fila. Esta técnica explora o caminho para o objetivo, dando preferência aos nós que estão mais distantes da raiz da árvore de busca. Funciona bem, quando as soluções são total e igualmente desejadas ou quando anteriormente foi feita uma varredura, detectando direções incorretas (FURTADO, 1973).

3.11. ALGORITMO SHORTEST PATH

O problema do "caminho mais curto" envolve encontrar o melhor caminho entre dois pontos chamados nós. Portanto, resolver este problema pode significar determinar o caminho entre dois nós com o menor custo, ou o menor tempo de viagem, ou a maior capacidade (Ahuja *et al.,* 1993).

O problema do caminho mais curto constitui o maior grupo no campo da pesquisa operacional e é considerado um dos problemas mais importantes da programação linear (HILLIER E LIEBERMAN, 1988).

Glover *et al.* (1985), devido ao grande número de aplicações práticas, o caminho mais curto é considerado um importante campo de pesquisa.

A aplicação do problema do caminho mais curto está relacionada à otimização de certas atividades, tais como: tráfego rodoviário, linhas de transmissão, conexões de rede, problemas de programação de rota chave PERT, planejamento de movimento de robô e outros campos mais complexos da biologia molecular (EPPSTEIN, 1994).

Ahuja *et al*. (1993), a aplicação posterior do algoritmo citado na resolução do problema do caminho mais curto envolve a otimização do plano de substituição de equipamentos, desenho do projeto, gerenciamento do fluxo de caixa, transmissão de mensagens no sistema de comunicação e fluxo de tráfego em cidades altamente congestionadas.

Glover *et al.* (1985) menciona as seguintes aplicações práticas: substituição de equipamento, planejamento de rota e tempo de viagem, rota e programação de veículos, planejamento de capacidade, design e / ou expansão de rede de transporte e comunicação e programação de caminho crítico.

Exceto para os aplicativos diretamente relacionados à busca do caminho mais curto, alguns outros problemas gerais de programação podem ser resolvidos formulando diretamente esse problema ou aplicando algumas estratégias de relaxamento.

Glover *et al.* (1985) citaram os seguintes problemas de programação inteira resolvidos pelo critério do caminho mais curto: localização, problema de atribuição generalizada, problema de correspondência máxima, problema do caixeiro viajante, problema da mochila e problema de equilíbrio de tráfego.

Hung e Divoky (1990) mencionaram que o algoritmo de busca do caminho mais curto tem sido usado para resolver o problema de correspondência, o problema de fluxo de menor custo e o problema de alocação.

As diferentes situações em que o caminho mais curto deve ser encontrado na prática, fazem com que o problema seja dividido em diferentes casos para facilitar sua pesquisa e desenvolvimento. Dreyfus (1969) propôs cinco problemas de busca do caminho mais curto.

A estratégia envolve encontrar o caminho mais curto entre dois nós para servir a demanda por uma determinada rede uma vez que uma solicitação é feita na rede, o algoritmo pode pesquisar o nó mais próximo para estabelecer um link e concluir a solicitação. (ZHAN, 1998).

3.12. ALGORITMO DIJKSTRA

O algoritmo Dijkstra foi criado para estimar o caminho mais curto entre dois vértices no gráfico. Seu criador Edsger Wybe Dijkstra o lançou em 1959 para mostrar a eficiência do processador ARMC.

"Quando todos os arcos têm comprimentos não negativos, o algoritmo de Dijkstra encontra o caminho mais curto entre quaisquer dois nós na rede. Um processo iterativo é usado. Na iteração 1, o nó mais próximo do nó 1 é determinado, e na segunda iteração determine o segundo nó mais próximo do nó 1, e assim por diante, até que o nó de destino seja alcançado em algumas iterações Arenales *et al*. (2007).

O tempo de execução do cálculo desse algoritmo em um dado grafo G com m arestas n vértices é O ([m + n] log n). O algoritmo permite o uso de execução de gráfico direcionada ou não direcionada.

Também é possível viabilizar sua execução pesando em suas bordas, a fim de determinar a "dificuldade" de transposição de uma delas. Outro problema que vale ressaltar é que o valor marginal não é negativo, ou seja, o custo só pode ser um valor nulo ou um valor positivo.

As observações finais são declaradas porque os resultados negativos fornecidos pelo algoritmo podem não ser tão precisos quanto Sedgewick (2002) afirmou.

FORSMAN *et al. (*1993) simularam uma mina a céu aberto, utilizando o modelo de simulação METAFORA, para avaliar estratégias de controle de despacho. As regras de despacho analisadas foram: despacho fixo, maximização de caminhões e maximização de carregadeiras. A simulação das estratégias mostrou que, para o modelo criado, a operação obteve melhor desempenho sob a política de maximização de caminhões. Para a seleção de uma rota ótima, durante o despacho, foi utilizado o algoritmo do menor caminho de Dijkstra.

A relação entre o comprimento da rota e o tempo de viagem necessária deve ser diretamente proporcional, no entanto, para entender como o tempo de ciclo é afetado pelo aumento da distância é de fundamental importância medir as operações de carregamento e transporte(Souza *et al,* 2009).

Para o funcionamento, o algoritmo trabalha com dois problemas:

1) Construir a arvore de menor comprimento total entre todos os nós de um grafo;

2) Encontrar o caminho de menor comprimento total entre dois determinados nós daquele grafo.

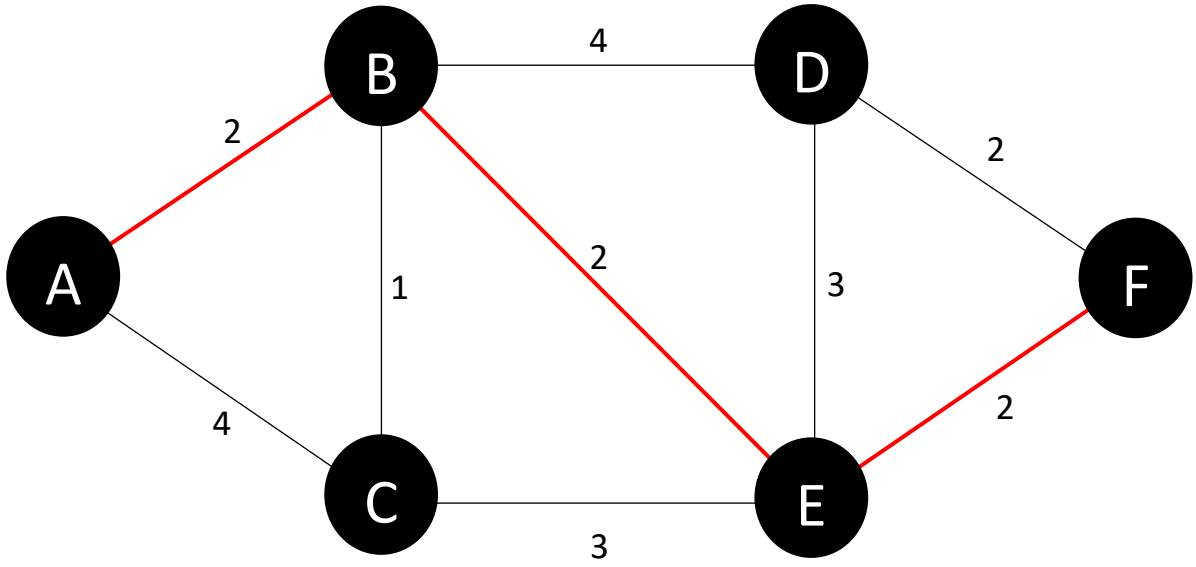
Em 1, todos os pontos (nós) do grafo são visitados, seguindo a lógica de menor distância entre um nó e o próximo. Inicialmente é atribuído um valor zero à estimativa de custo do vértice raiz de busca e infinito às demais estimativas. Atribui-se um valor aos precedentes (são estes os vértices que estão entre a raiz e o vértice destino) e é determinada a distância entre a raiz e o próximo nó. A menor distância entre o nó corrente e o próximo é armazenada em uma matriz de informações como demonstrada na Tabela 1. Após o armazenamento de todas as informações, é selecionado, de maneira reversa, o caminho mínimo, conforme indicado na figura 12.

Tabela 1 - Armazenamento de informações de distâncias de grafos

Q <= V	A	B	C	D	E	F
A	0, A	2, A	4, A	∞	∞	∞
B	0, A	2, A	3, B	6, B	4, B	∞
C	0, A	2, A	3, B	6, B	4, B	∞
D	0, A	2, A	3, B	6, B	4, B	6, E
E	0, A	2, A	3, B	6, B	4, B	6, E
F	0, A	2, A	3, B	6, B	4, B	6, E
	A-B-E-F	B => A			E => B	F => E

Fonte: O autor

Figura 12 - Estrutura de determinação de caminho de grafos em Dijkstra



Fonte: O autor

3.13. A-STAR

Em Qian *et al*. (2018) é dito que o algoritmo A-Star ("A *" ou A Star) na literatura é utilizado para processar soluções de caminhos e encontrar caminhos de baixo custo, por se tratar de um algoritmo heurístico eficiente amplamente utilizado para esse fim. Ao explorar cada nó da configuração espacial e expandir o nó mais promissor relacionado ao alvo, ele é então definido como um algoritmo BFS (Best First Search) (DUCHON *et al*., 2014). Para que isso seja possível, Hart, Nilsson e Raphael (1968)

apontam que a função de avaliação é utilizada para submetê-lo a cada nó, e então o nó de menor custo é retornado na função de expansão.

Orientar e determinar a ordem de pesquisa e exploração dos nós no gráfico indica a cena a ser avaliada, e a função de avaliação é denotada por F (n), conforme explicitado pela Equação 2.

Equação 2 - Função de avaliação de caminho

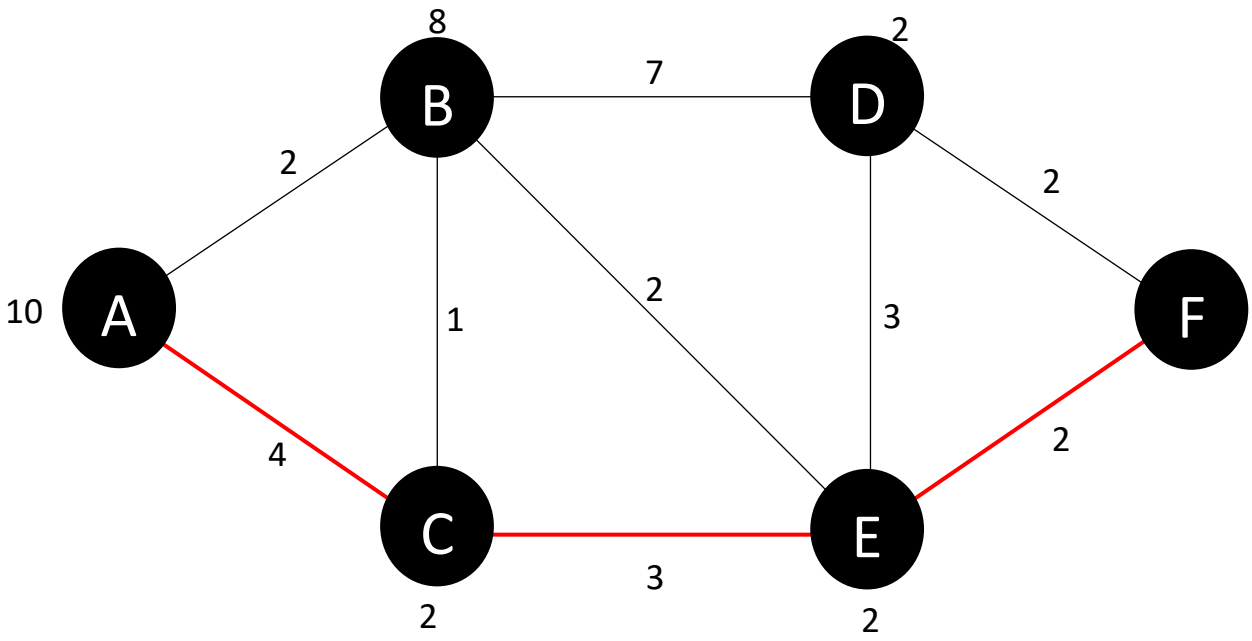
$$F(n) = G(n) + H(n)$$

Fonte: O autor

onde G (n) é o custo real do caminho ideal do ponto inicial até n. O custo estimado do caminho ótimo do ponto n ao ponto de destino é representado por H (n)

Figura 13 - Grafo determinado segundo A*

na função, e H (n) depende da informação heurística da área do problema, conforme ilustrado na Figura 13. A soma de distância de trajeto é determinada conforme a Tabela 2.



Fonte: O autor

Tabela 2 - Armazenamento de informação de caminho e determinação de rota

Etapa	A	C	E

1	F(B) = 2 + 8 = 10 F(C) = 4 + 2 = 6		
2		F(B) = 5+8 = 12 (4+1 = 5) F(E) = 7+2 = 9 (4+3 = 7)	
3			F(B) = 9+8 = 17 (4+3+2 = 9) F(D) = 10+2 = 12 (4+3+2 = 10) F(F) = 9+0 = 9 (4+3+2 = 9)
	A-C	A-C-E	A-C-E-F

Fonte: O autor

O algoritmo A-Star pode ser vinculado a todos os veículos do sistema de transporte. As cidades inteligentes geram requisitos de capacidade de computação muito elevados, porque muitos cálculos precisam ser realizados quando se busca a melhor solução. Clusters e supercomputadores podem ser utilizados neste cenário, por possuírem recursos computacionais avançados, quando se trata de poder de processamento para atender a esse requisito, porque processadores de alto desempenho podem ser usados para realizar os cálculos necessários.

Além da estrutura fornecida pelos dois, a programação paralela também pode ser usada. Além dos recursos técnicos de computação de alto desempenho, vários núcleos de processamento também podem realizar tarefas de forma independente. (HPC, High Performance Computation).

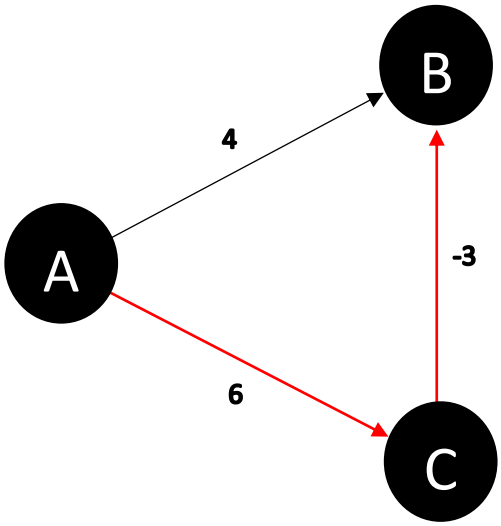
No entanto, clusters e supercomputadores são baseados em arquiteturas de hardware com altos custos de implementação (BACELLAR, 2009), pois utilizam muitos computadores interconectados para funcionar juntos em uma tarefa. Entre eles, cada computador é denominado nó (ou nó).

Os nós são conectados por meio de conexões de altíssima velocidade, de forma a evitar que o gargalo de conexão entre os nós afete a capacidade de processamento da troca de informações entre eles. Comparado com clusters e supercomputadores, o uso de GPU é uma alternativa viável e seu custo é muito menor. São hardwares específicos com placas de vídeo NVIDIA, que usam CUDA, uma extensão da linguagem de programação C para realizar operações. Portanto, quando o algoritmo está bem desenvolvido e implementado, ele pode ser processado por HPC.

3.14. ALGORITMO BELLMAN-FORD

De acordo com Cormen (1989), o algoritmo Bellman-Ford resolve o problema de encontrar o caminho mais curto de um vértice para todos os outros vértices de um grafo valorizado G = (V, E), onde o valor pode ser negativo. O algoritmo Bellman-Ford retorna um valor booleano indicando se há um loop negativo que o vértice de origem pode alcançar. Se este loop existe, o algoritmo mostra que o problema não tem solução. Se o loop não existir, o algoritmo gera o caminho mais curto e sua distância. O algoritmo Bellman-Ford usa técnicas de otimização não gananciosas para reduzir a distância estimada do vértice v ao vértice de origem s, conforme demonstrado na Figura 14.

Figura 14 - Construção de caminho por Bellman Ford



Fonte: O autor

Diferentemente de uma metodologia, não gananciosa, ao partir de A em busca de B, o algoritmo de Bellman Ford passa por A-C-B, verificando o valor deste caminho, antes de determinar e eliminar a possibilidade de utilizar C, devido seu maior valor inicial. Ao se perceber isso, o uso de arestas com pesos negativos permite a execução de iterações mais dinâmicas, abrindo a possibilidade de caminhos com menor custo.

3.15. SINGURALIDADES ENTRE ALGORITMOS

Os vértices do algoritmo de Dijkstra contêm todas as informações da rede. Não existe tal coisa que cada vértice se preocupe apenas consigo mesmo e com seus

vizinhos. Por outro lado, o algoritmo Bellman-Ford contém apenas informações relevantes. Essas informações permitem que o nó saiba apenas quais nós vizinhos podem ser conectados entre si e de qual nó vem o relacionamento. O algoritmo Dijkstra é mais rápido do que o algoritmo Bellman-Ford, mas o segundo algoritmo pode ser mais útil para resolver certos problemas (como pesos de caminho negativos).

O algoritmo Dijkstra é uma técnica gananciosa. Para implementar o algoritmo de Bellman-Ford, precisamos de um método dinâmico.

No algoritmo Dijkstra, relaxamos cada nó / vértice no loop, onde, como algo em Bellman-Ford, relaxamos apenas | v-1 | vezes.

O algoritmo Dijkstra é adequado para gráficos com pesos de aresta positivos, mas falha no caso de gráficos de arestas negativas, porque o algoritmo de Bellman-Ford tem vantagens sobre Dijkstra, mesmo se os pesos de aresta forem atribuídos negativamente.

4. METODOLOGIA

Este tópico apresenta a metodologia utilizada para realização dos procedimentos experimentais do projeto de pesquisa. A parte experimental foi desenvolvida no Laboratório de informática localizado no Departamento de Engenharia de Minas da Universidade Federal de Ouro Preto, bem como em computador pessoal de forma remota. Foram utilizados os algoritmos A* (A-Estrela), Dijkstra, Bellman-Ford e Shortest Path. Todos os quatro algoritmos, bem como os códigos para sua aplicação eram construídos sob a linguagem Python.

4.1. Seleção dos Algoritmos

Para a seleção dos algoritmos, utilizou-se a biblioteca NetworkX. O que permitiu que fosse encontrado Dijkstra, Bellman Ford, A* e Shortest Path.

Além da determinação dos algoritmos, foi indicado no código que a distância entre os pontos fosse calculada, segundo a metodologia requerida, conforme mostra a Tabela 3.

Tabela 3 - Cálculos de distância

MÉTODO DE CÁLCULO	EQUAÇÃO
Distância euclidiana	$\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$
Distância por consumo	$cateto1 = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$ $cateto2 = \lvert p_z - q_z \rvert$ $gradiente\ de\ inclinação = \frac{cateto2}{cateto1}$ $resistência\ de\ rampa = peso\ do\ equipamento * inclinação$ $resistência\ ponto\ a\ ponto = 2136.42 + resistência\ de\ rampa + 102.6362$ $r = 4.0824 * \left(0.006 + \left(0.053 * \frac{resistência\ ponto\ a\ ponto}{1000}\right)\right)$ $v = 53.8667 - 54.906 * e^{(-37.979 * r^{-1.309})}$ $p = 8.4660191 * v$ $FC = 0.0003 * p$ $peso = \lvert FC * \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$

Fonte: O autor

Após a seleção dos algoritmos e determinação das equações de distância entre pontos, foi ordenado, também, que o algoritmo determinasse as distâncias entre coordenadas para os pontos selecionados, calculando-a segundo as coordenadas X e Y, e a distância em Z, que é equivalente à deflexão. Os cálculos estão demonstrados nas Equação 3 aEquação ***5***.

Equação 3 - Determinação de Distância segundo do eixo X (DistX)

$$\Delta x = \sum_{i=2}^{n} (x_n - x_{n-1})$$

Fonte: O autor

Equação 4 - Determinação de Distância segundo o eixo Y (DistY)

$$\Delta y = \sum_{i=2}^{n} (y_n - y_{n-1})$$

Fonte: O autor

Equação 5 - Determinação de Distância segundo o eixo Z (Deflexão)

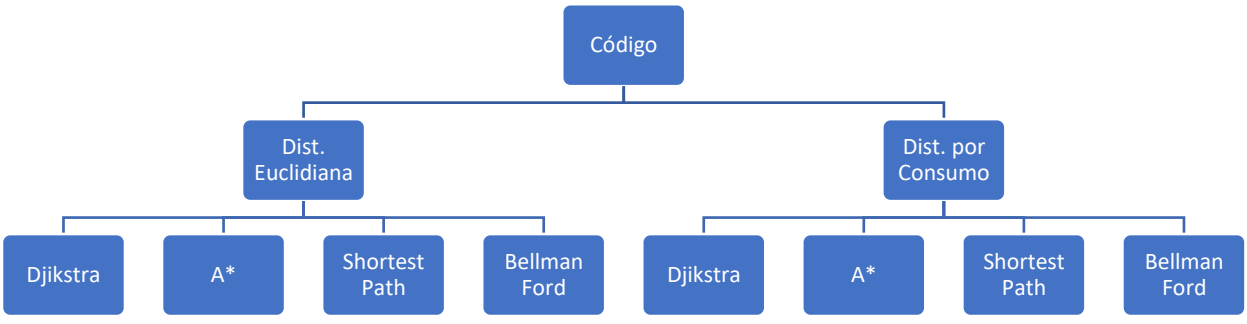
$$\Delta z = \sum_{i=2}^{n} (z_n - z_{n-1})$$

Fonte: O autor

As equações foram inseridas de forma conjunta no mesmo código para execução do trabalho, sendo necessário que fosse indicado apenas qual o módulo de distância deveria ser levado em consideração naquele momento: cálculo pelo método euclidiano ou pelo método do consumo.

Definidas as formas de cálculo, executou-se o código oito vezes, quatro para cada forma de determinar a distância, conforme descrito pela Figura 15.

Figura 15 - Execuções de código a partir das distâncias



Fonte: o Autor

Após a seleção de cada um dos algoritmos, uma matriz [adjacência], contendo e ordenando os pontos adjacentes, foi gerado, o que permitia o cálculo de adjacência e determinação de pontos, sendo assim, estes e os caminhos foram armazenados em uma segunda matriz, a matriz [Rotas], que partia de um ponto variável indicado. As equações de determinação de matrizes estão indicadas na Tabela 4.

Tabela 4 - Determinação de matrizes de pontos

Matriz	Código

Adjacência	`# MATRIZ DE ADJACENCIA - FILTRADA PELO GRADE - Filtrar pelo zero`
	`adjacencia=[]`
	`for linha in range(numero):`
	`  for coluna in range(1,Vizinho):`
	`  #if (M_Vizinhanca[linha,0] != M_Vizinhanca[linha,coluna]) and (M_Vizinhanca[linha,coluna] != 0):`
	`  ponto0=M_Vizinhanca[linha,0]`
	`  ponto1= M_Vizinhanca[linha,coluna]`
	`  xdif = abs(Pontos_Modelo[ponto0,1]- Pontos_Modelo[ponto1,1])`
	`  ydif = abs(Pontos_Modelo[ponto0,2]- Pontos_Modelo[ponto1,2])`
	`  zdif = abs(Pontos_Modelo[ponto0,3]- Pontos_Modelo[ponto1,3])`
	`  if ((Pontos_Modelo[ponto0,1] != Pontos_Modelo[ponto1,1]) and (Pontos_Modelo[ponto0,2] != Pontos_Modelo[ponto1,2]) ) :`
	`  adjacencia.append((ponto0, ponto1,Distancia(ponto0,ponto1)))`
	`print("Matriz Finalizada.")`

Rotas	`#-----------------------------------------------`
	`# PONTOS INICIAL E FINAL // DECLARAR VARIÁVEL`
	`num_cores = multiprocessing.cpu_count()`
	`DG = nx.DiGraph()`
	`DG.add_weighted_edges_from(adjacencia)`
	`inicial = 15`
	`Caminhos=[]`
	`Rotas=[]`

Fonte: O autor

4.2. Conversão e análise do consumo de combustível

Após rodar o código, um arquivo do tipo .CSV é gerado, com o qual foi necessário ainda converter os valores de coordenada em medidas, utilizando as coordenadas nas mesmas equações demonstradas para os cálculos de distância.

Esta transformação, no entanto, foi feita para todos os valores, uma vez que era necessário definir qual seria o consumo gerado por cada um dos algoritmos, bem como cada um dos modelos de cálculo de distância.

5. RESULTADOS E DISCUSSÃO

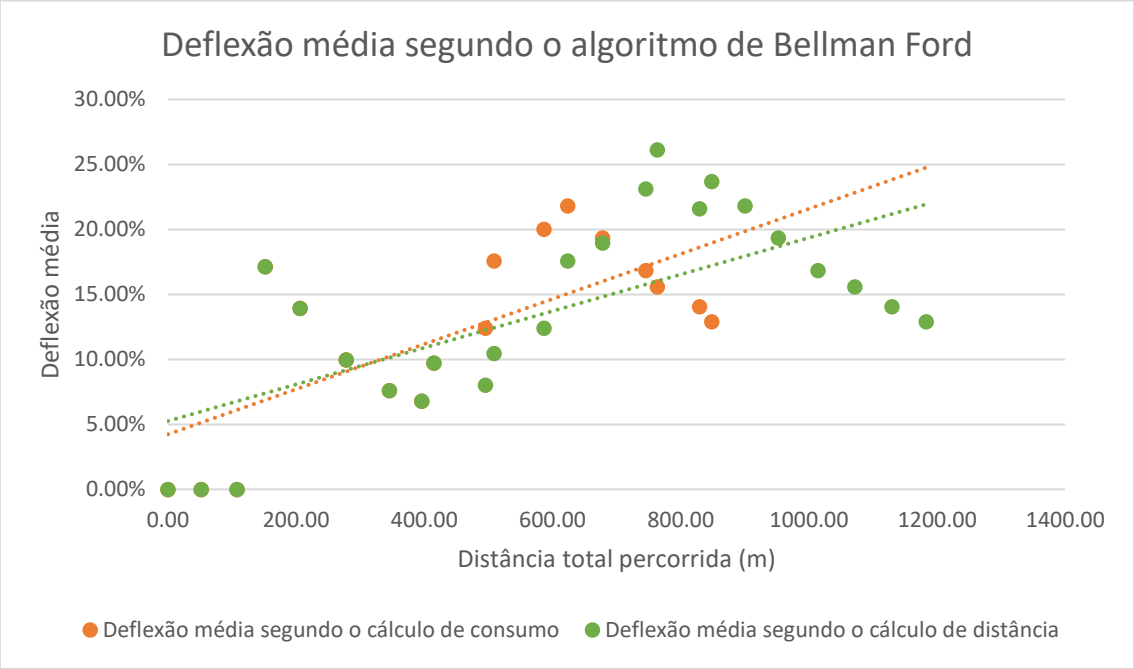
Neste tópico serão apresentados os resultados referentes aos ensaios realizados a partir da metodologia apresentada.

Abaixo estão inseridos gráficos que permitem a análise dos resultados obtidos através de cada um dos cálculos. Os resultados foram originalmente gerados para tabelas, que com o tratamento correto de informações geraram os gráficos para estudo.

Levando em consideração as informações e resultados propostos, foi necessária a utilização de dois tipos de gráficos diferentes: Gráfico de Linhas para a distância percorrida segundo o eixo X e distância percorrida segundo o eixo Y e consumo, e Gráfico de Dispersão para a deflexão de caminho.

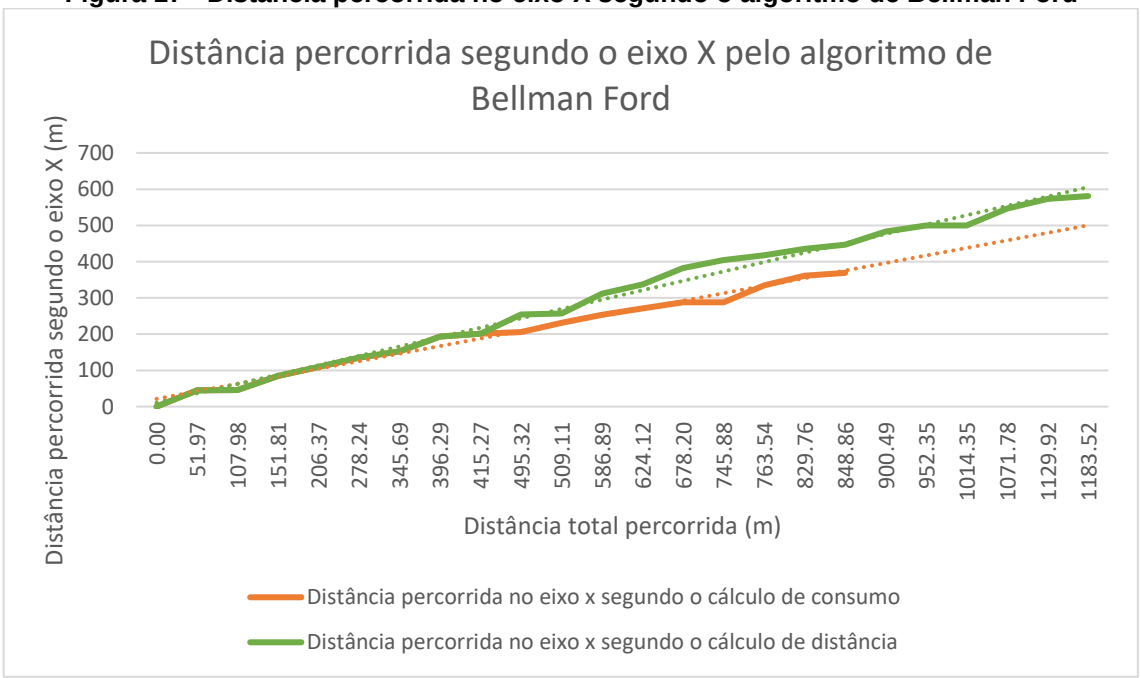
Para Bellman Ford, conforme demonstrado nas Figura 16 a Figura 19, é possível notar a insignificância da variação dos valores de deflexão média, DistX e DistY para distância total percorrida até aproximadamente 420m. Quanto ao consumo, é possível dizer que o consumo geral calculado pelo algoritmo quando a metodologia de cálculo pela distância euclidiana é utilizada para determinar a rota de operação, uma vez que esta forma de cálculo emprega em uma maior distância de operação. Ao observar a distância máxima de percurso determinada pelo algoritmo de menor consumo de combustível (aproximadamente 850m) nota-se que não há diferença considerável de consumo.

Figura 16 - Gráfico de deflexão segundo o algoritmo de Bellman Ford



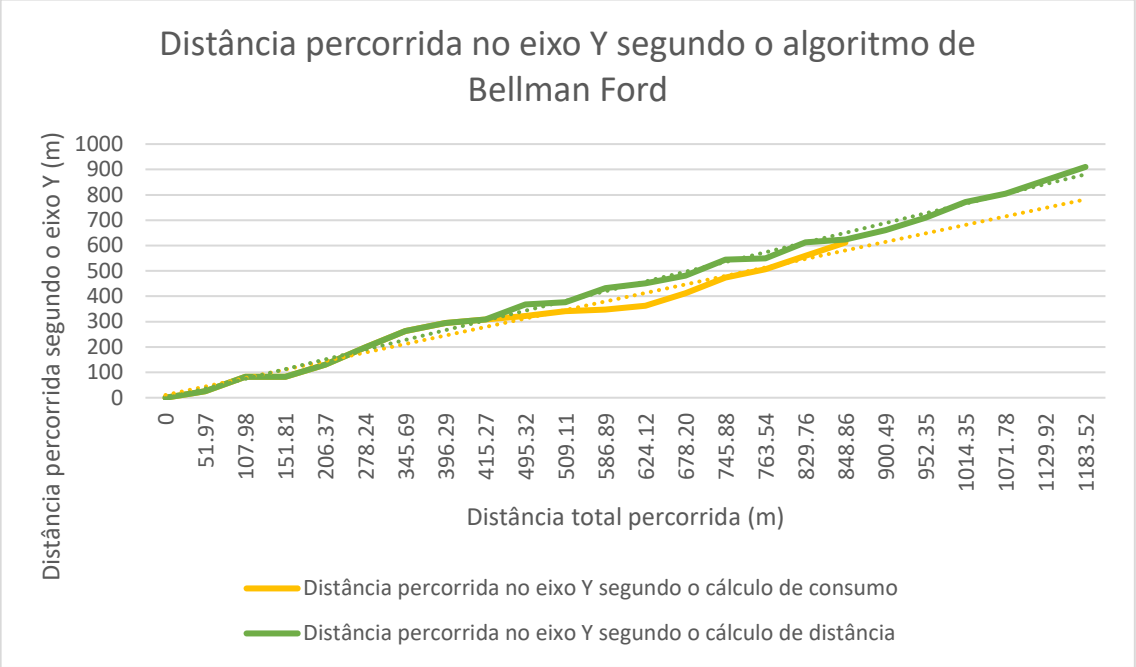
Fonte: O autor

Figura 17 - Distância percorrida no eixo X segundo o algoritmo de Bellman Ford



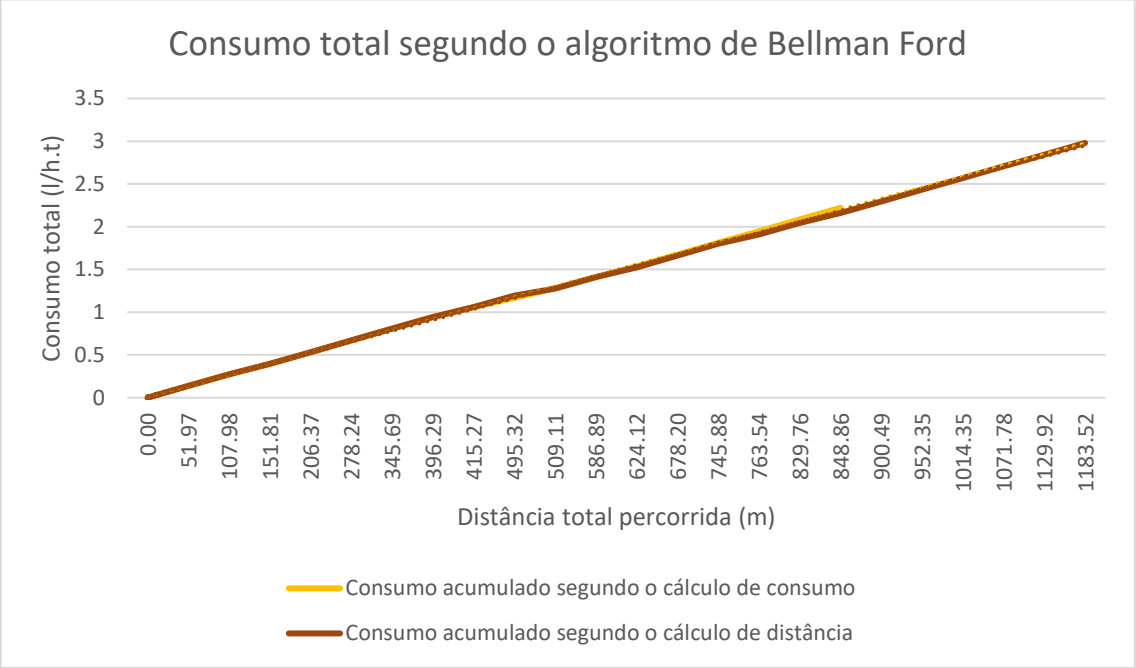
Fonte: O autor

Figura 18 - Distância percorrida no eixo Y segundo o algoritmo de Bellman Ford



Fonte: O autor

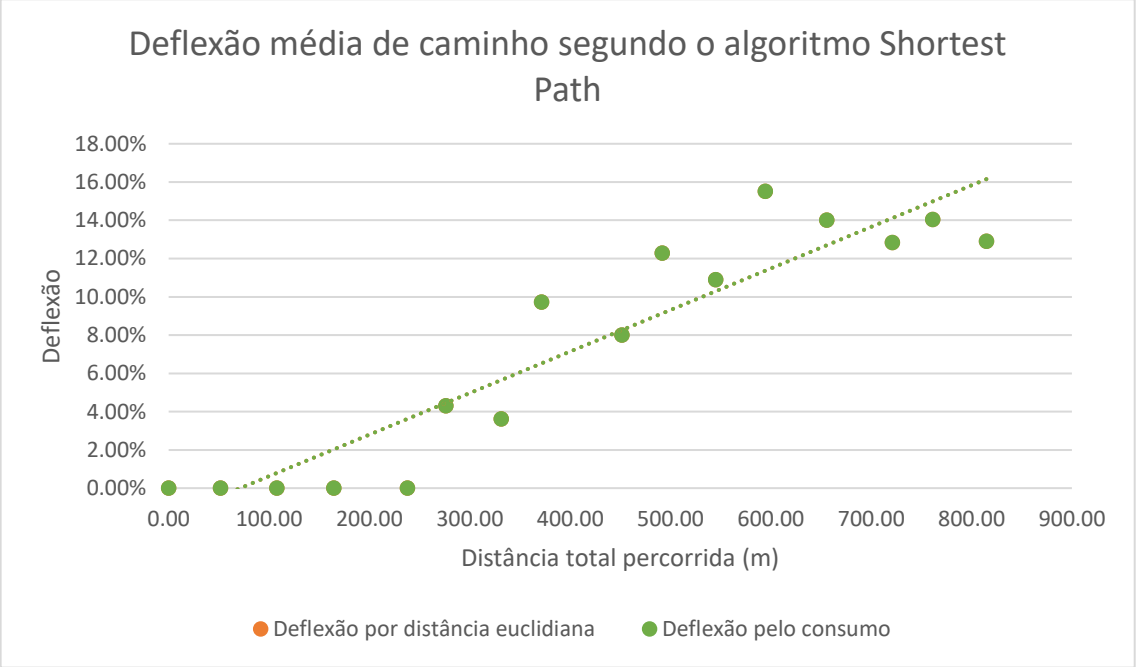
Figura 19 - Consumo total segundo o algoritmo de Bellman Ford



Fonte: O autor

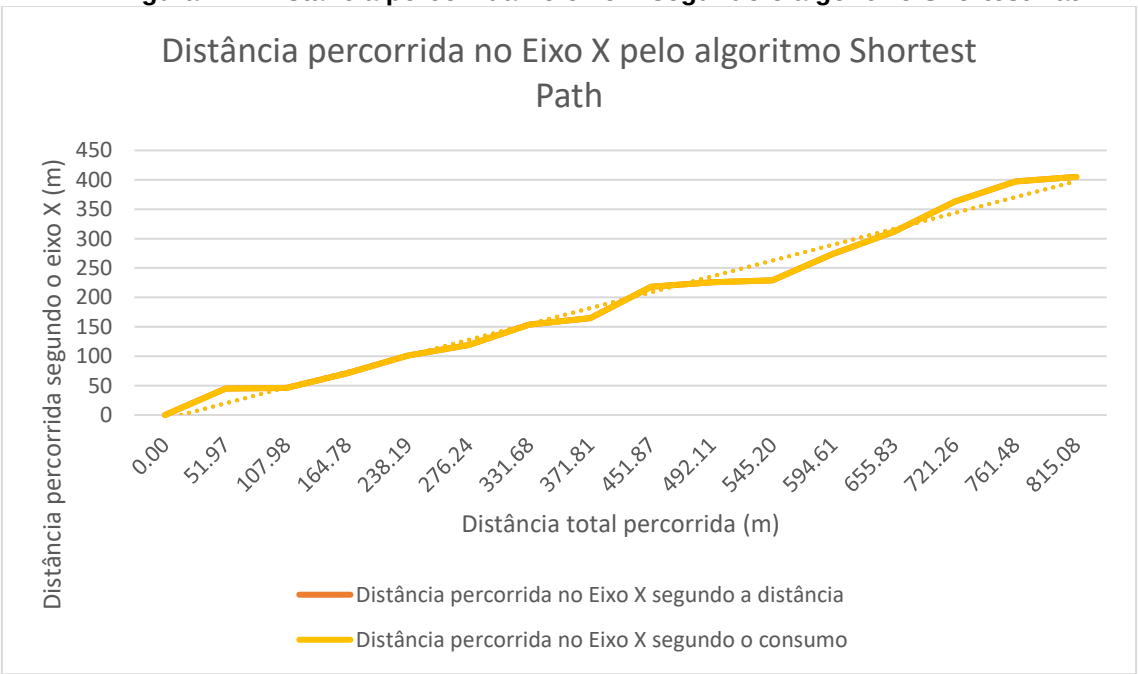
Para o cálculo por shortest path, conforme demonstrado nas Figura 20 a Figura 23, não é possível notar diferenças quanto à deflexão média, DistX e DistY, distância total percorrida e nem em relação ao consumo.

Figura 20 - Gráfico de deflexão segundo o algoritmo Shortest Path



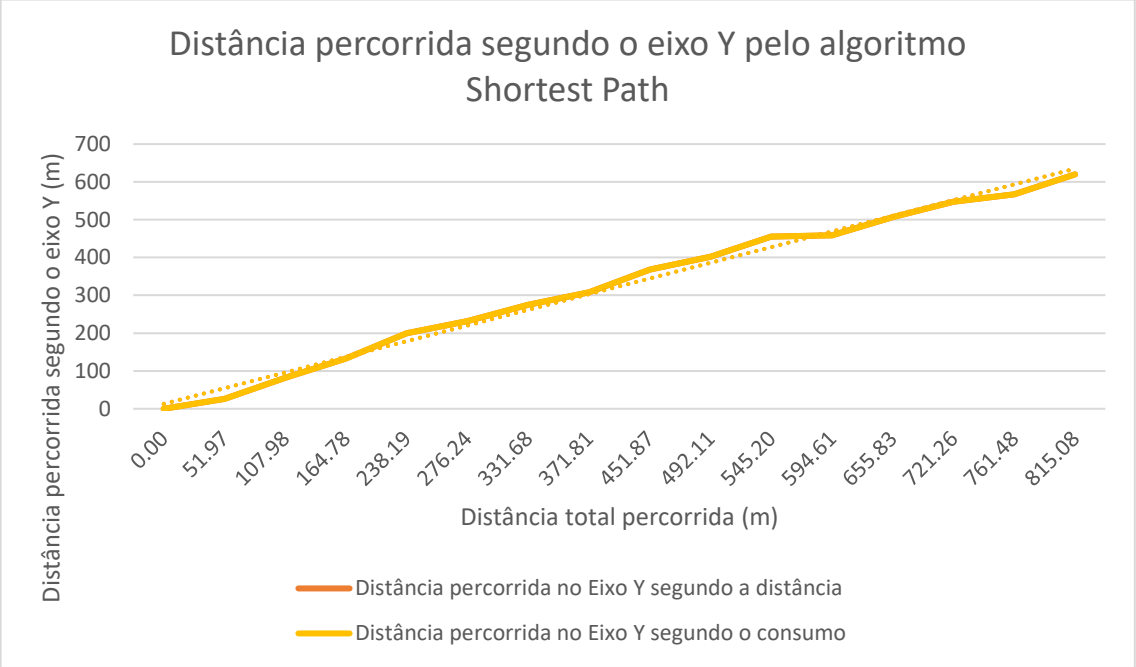
Fonte: O autor

Figura 21 - Distância percorrida no eixo X segundo o algoritmo Shortest Path



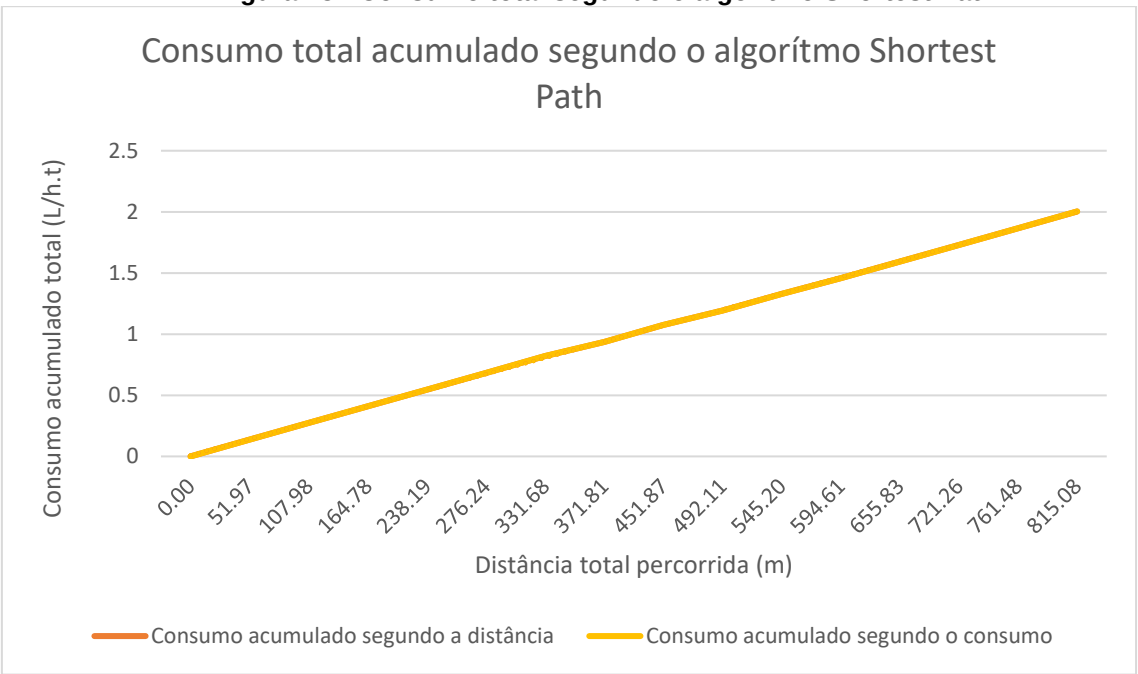
Fonte: O autor

Figura 22 - Distância percorrida no eixo Y segundo o algoritmo Shortest Path



Fonte: O autor

Figura 23 - Consumo total segundo o algoritmo Shortest Path

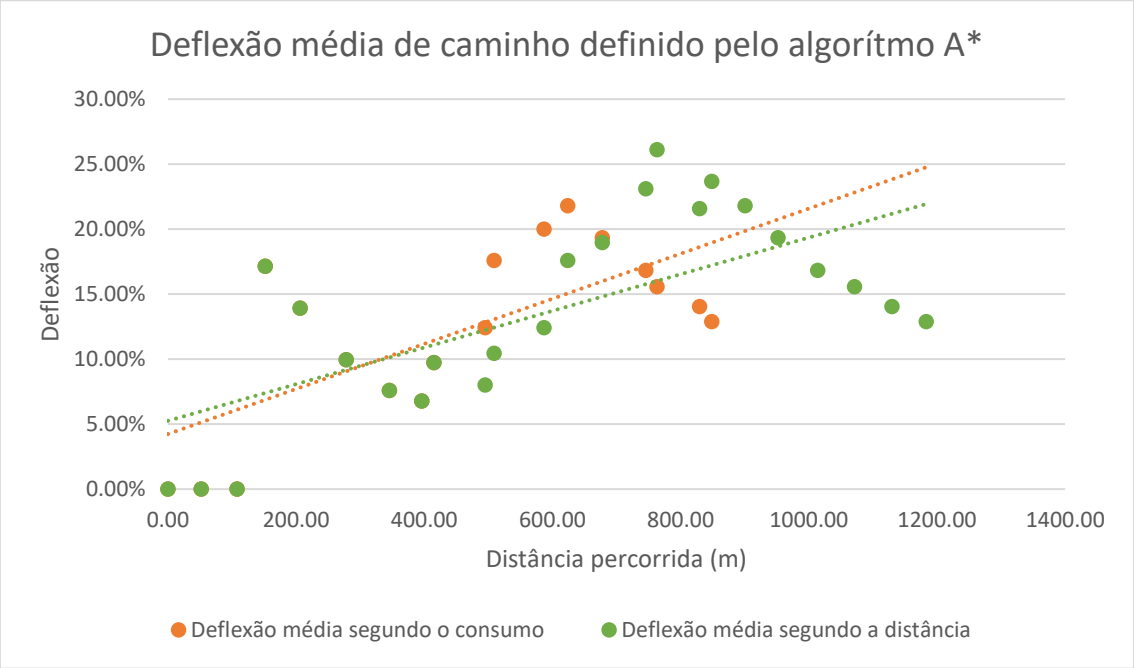


Fonte: O autor

Para A*, conforme demonstrado nas Figura 24 a Figura 27, é possível notar a insignificância da variação dos valores da deflexão média, DistX e DistY para distância total percorrida até aproximadamente 420m. Quanto ao consumo, é possível dizer que o consumo geral calculado pelo algoritmo quando a metodologia de cálculo pela distância euclidiana é utilizada para determinar a rota de operação, uma vez que esta forma de cálculo emprega em uma maior distância de operação. Ao observar a

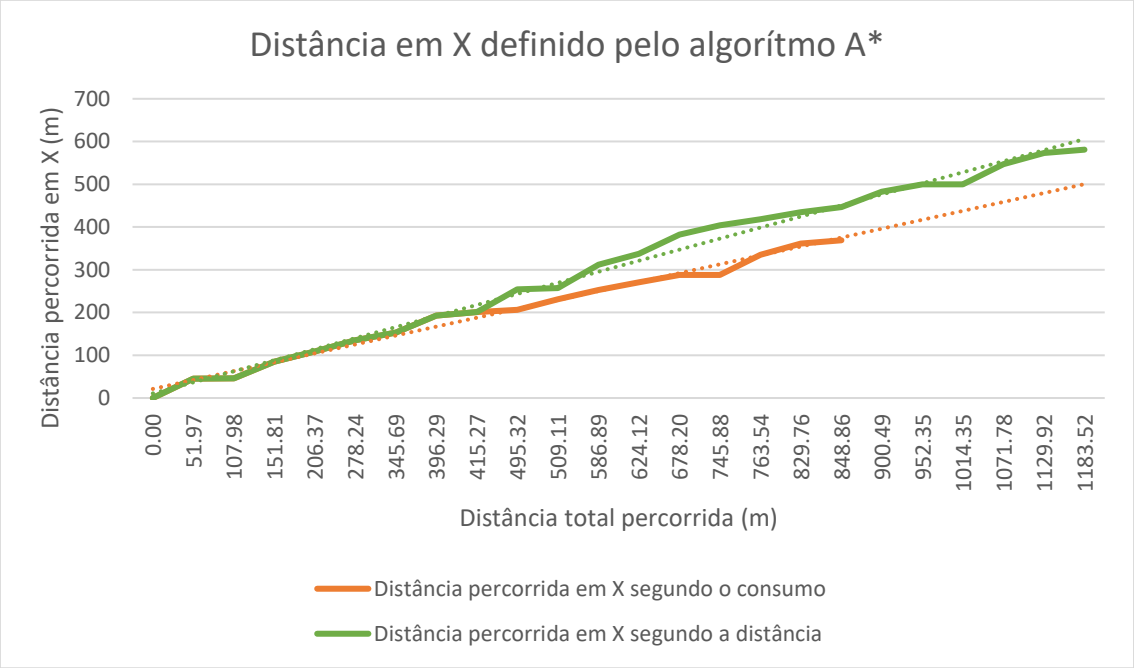
distância máxima de percurso determinada pelo algoritmo de menor consumo de combustível (aproximadamente 850m) nota-se que não há diferença considerável de consumo.

Figura 24 - Gráfico de deflexão segundo o algoritmo A*



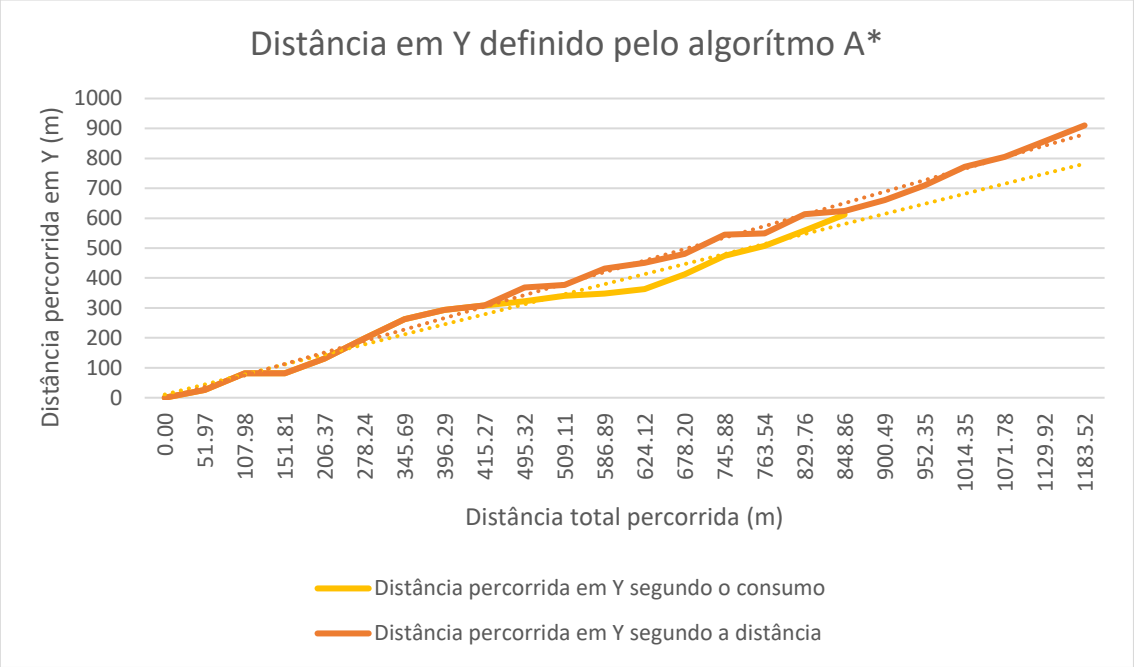
Fonte: O autor

Figura 25 - Distância percorrida no eixo X segundo o algoritmo A*



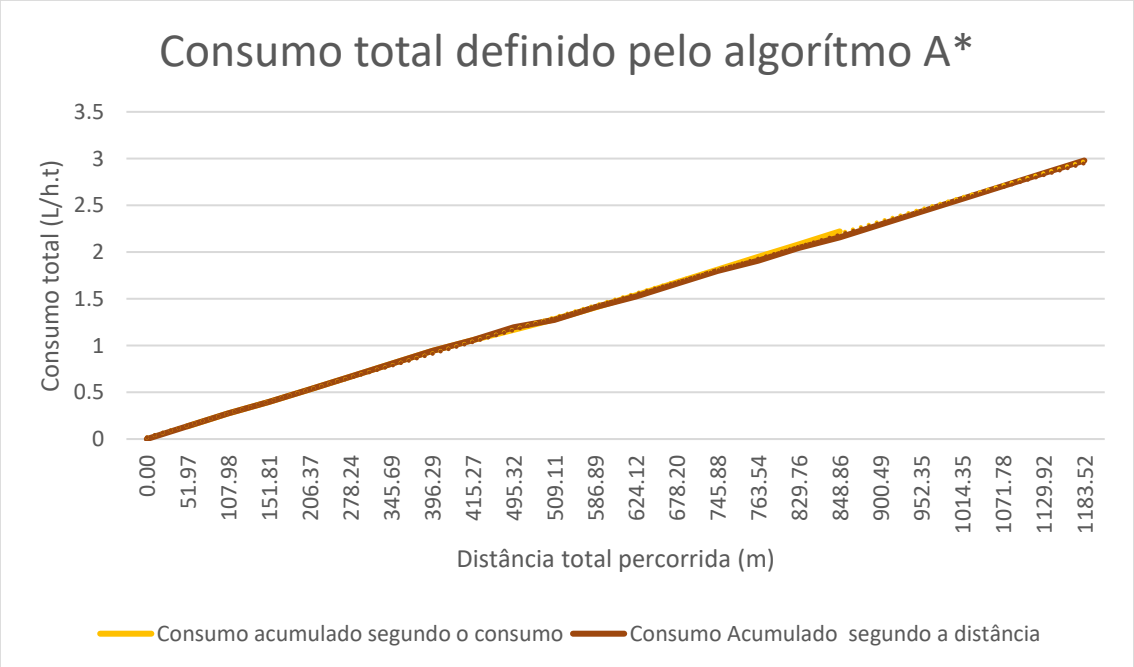
Fonte: O autor

Figura 26 - Distância percorrida no eixo Y segundo o algoritmo A*



Fonte: O autor

Figura 27 - Consumo total segundo o algoritmo A*

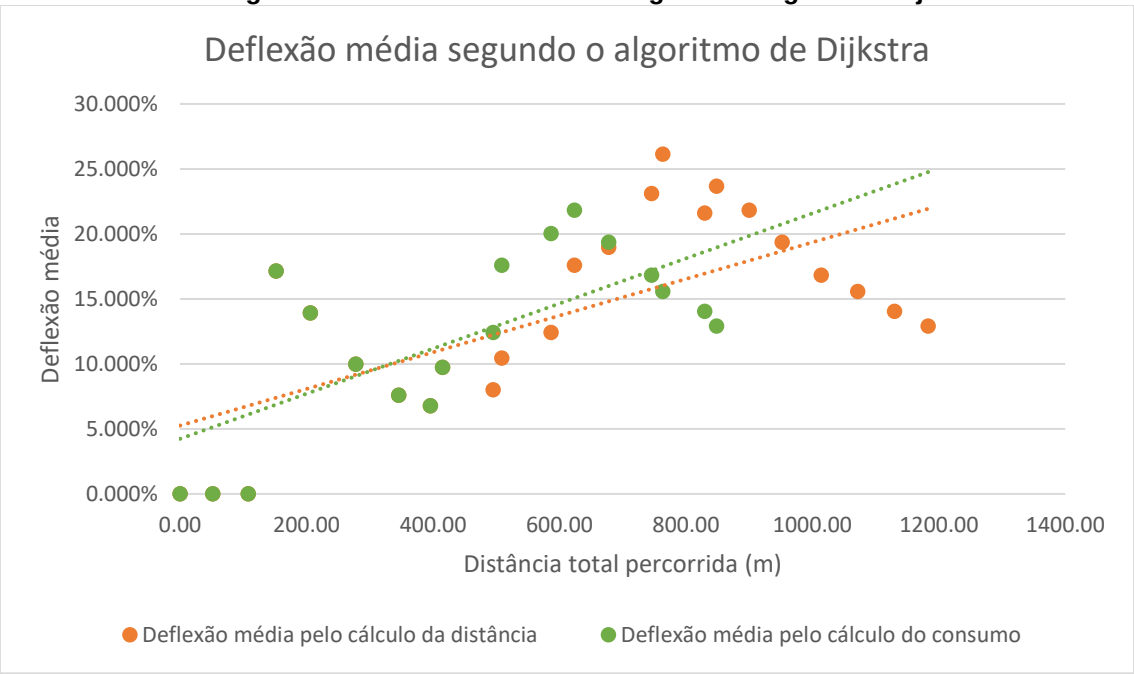


Fonte: O autor

Para Dijkstra, conforme demonstrado nas Figura 28 a Figura 31 é possível notar a insignificância da variação dos valores da deflexão média, DistX e DistY para distância total percorrida até aproximadamente 420m. Quanto ao consumo, é possível dizer que o consumo geral calculado pelo algoritmo quando a metodologia de cálculo pela

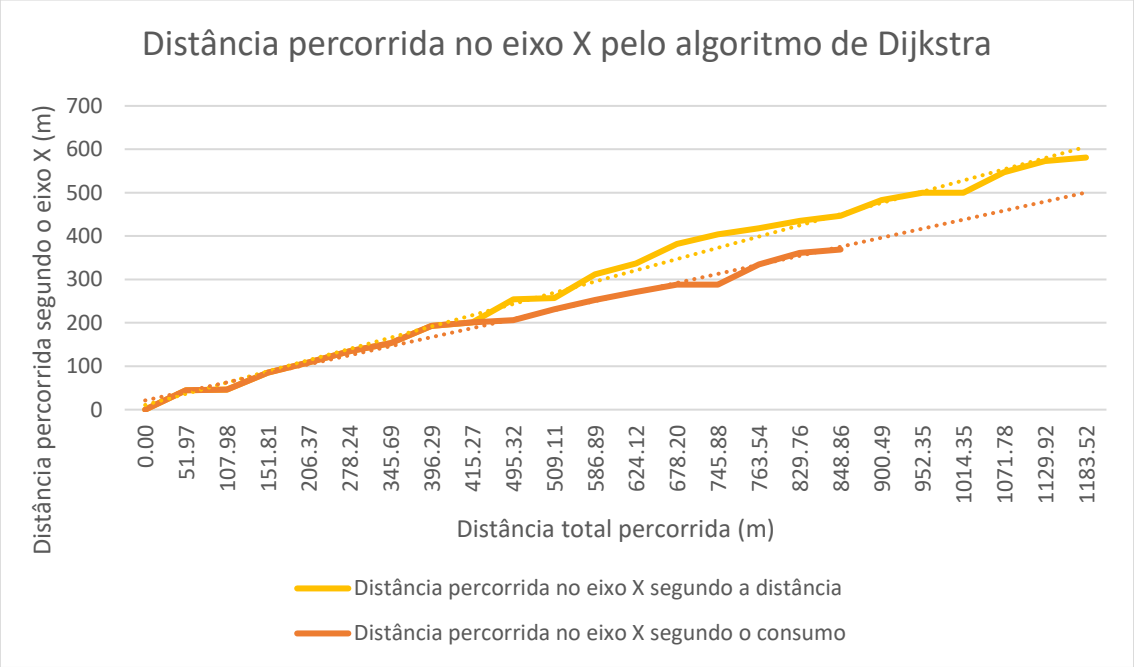
distância euclidiana é utilizado para determinar a rota de operação, uma vez que esta forma de cálculo emprega em uma maior distância de operação. Ao observar a distância máxima de percurso determinada pelo algoritmo de menor consumo de combustível (aproximadamente 850m) nota-se que não há diferença considerável de consumo.

Figura 28 - Gráfico de deflexão segundo o algoritmo Dijkstra



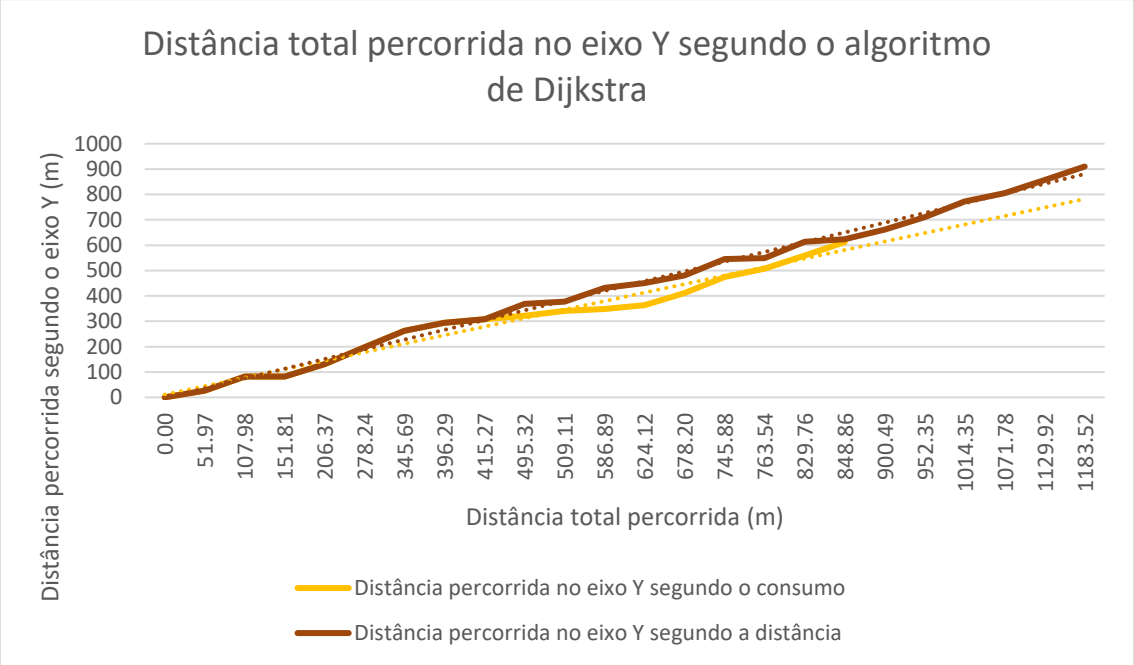
Fonte: O autor

Figura 29 - Distância percorrida no eixo X segundo o algoritmo Dijkstra



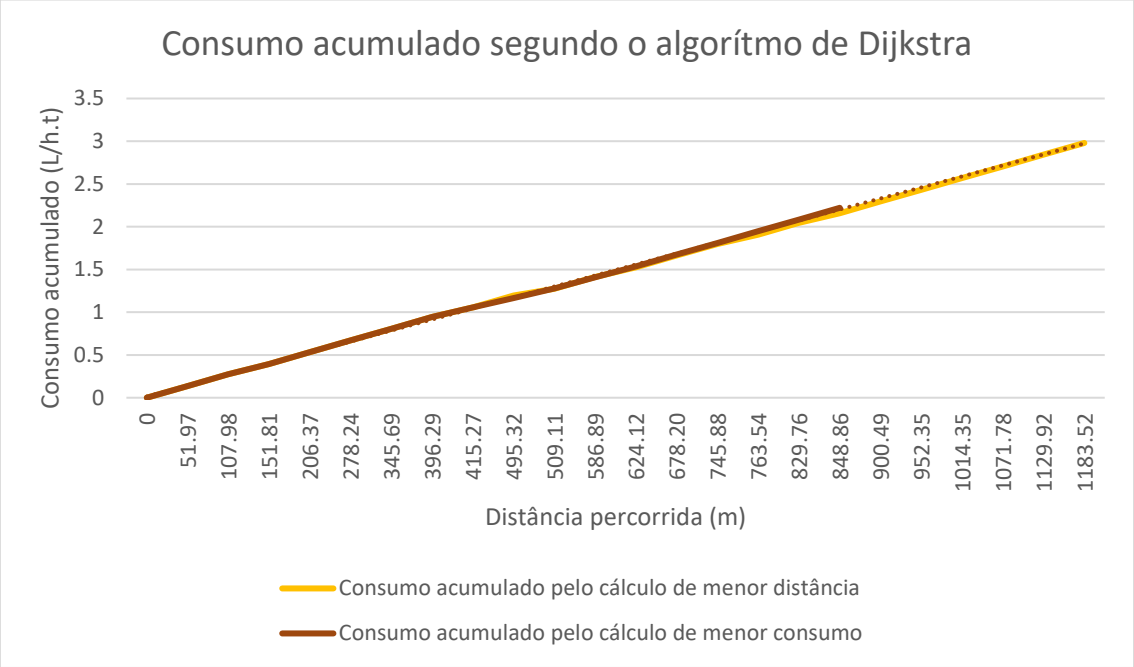
Fonte: O autor

Figura 30 - Distância percorrida no eixo Y segundo o algoritmo Dijkstra



Fonte: O autor

Figura 31 - Consumo total segundo o algoritmo Dijkstra

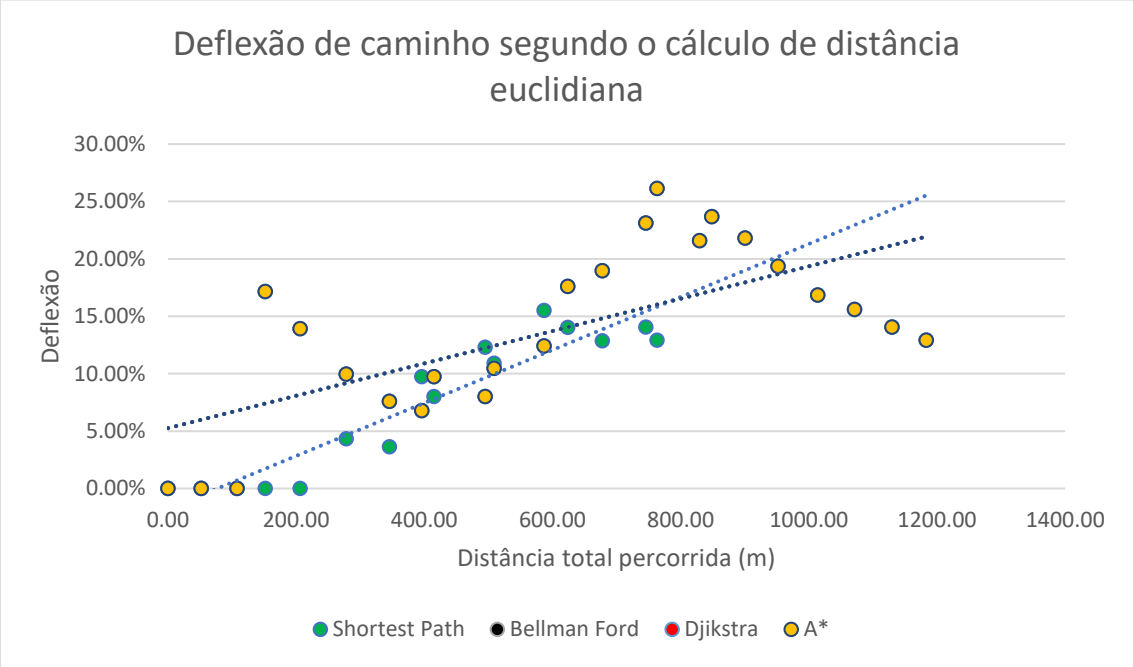


Fonte: O autor

As Figura 32 a Figura 35 demonstram, para efeitos comparativos, os resultados obtidos pelos algoritmos ao se efetuar os cálculos de rota utilizando a fórmula da distância euclidiana como referência para distância. É possível perceber que shortest path opera em uma deflexão média inferior aos outros três algoritmos até uma distância de aproximadamente 800m, ou seja, em toda a rota determinada desta maneira. É importante notar que ao longo da distância determinada por Shortest Path o consumo acumulado determinado é maior que nos outros três tipos de código. De maneira geral, entretanto, o consumo total continua sendo maior quando se utiliza o Bellman Ford, Djikstra e A*.

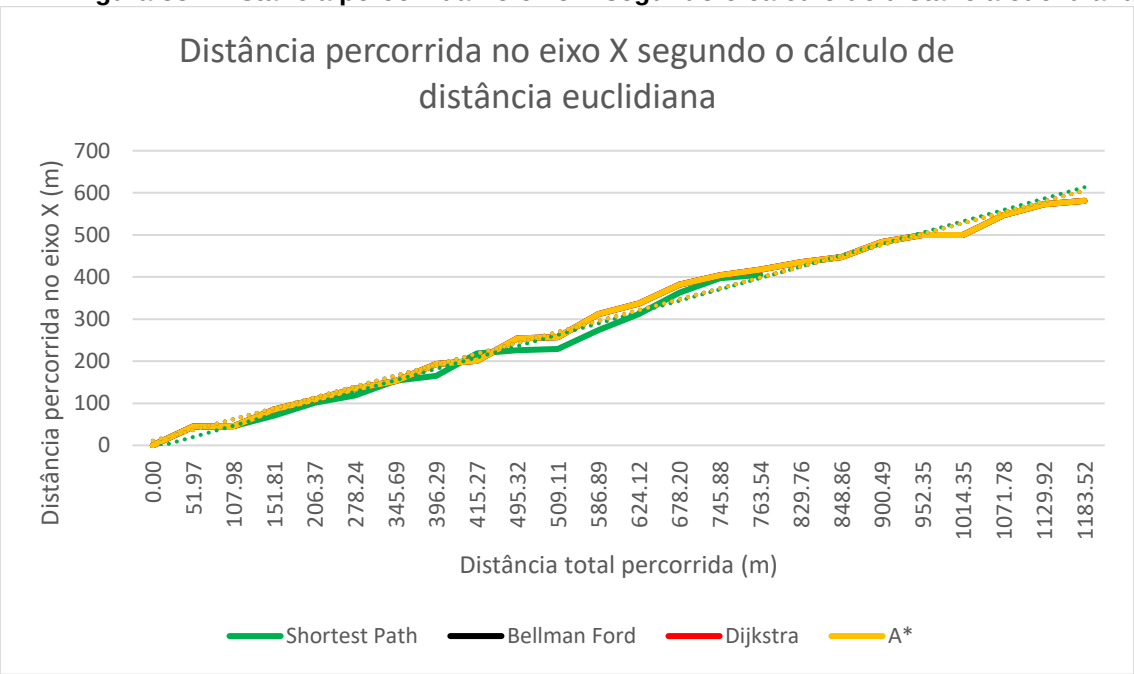
Nos gráficos apresentados abaixo, os pontos, linhas e linhas de tendência gerados pelos algoritmos de Bellman Ford, Dijkstra e A* estão sobrepostas.

Figura 32 - Gráfico de deflexão segundo o cálculo de distância euclidiana



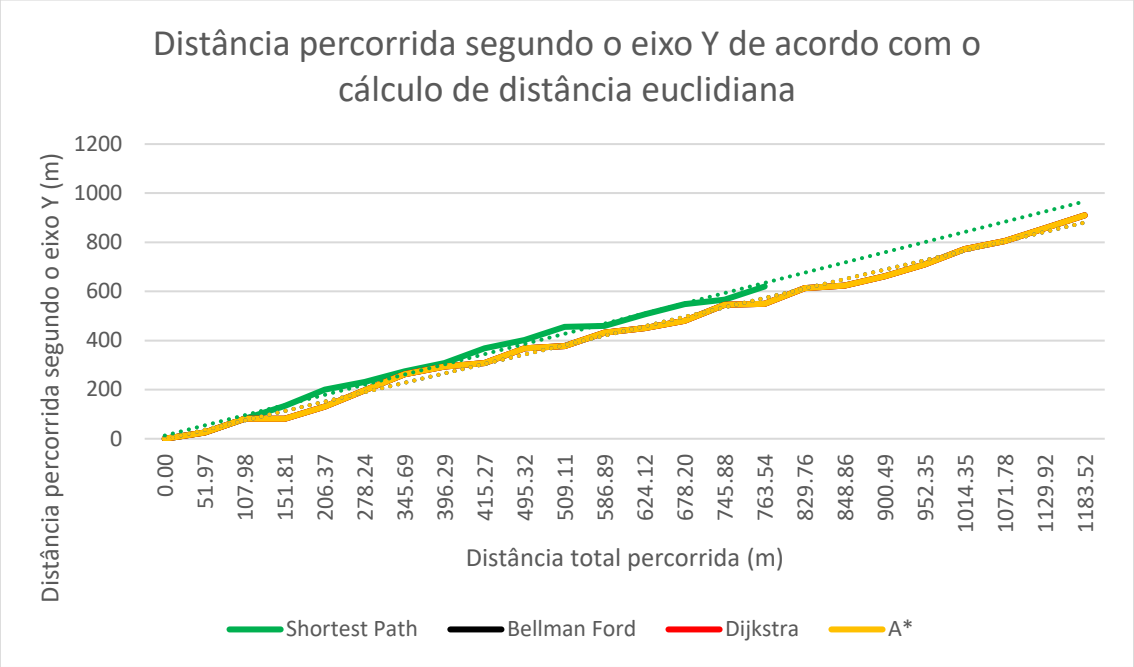
Fonte: O autor

Figura 33 - Distância percorrida no eixo X segundo o cálculo de distância euclidiana



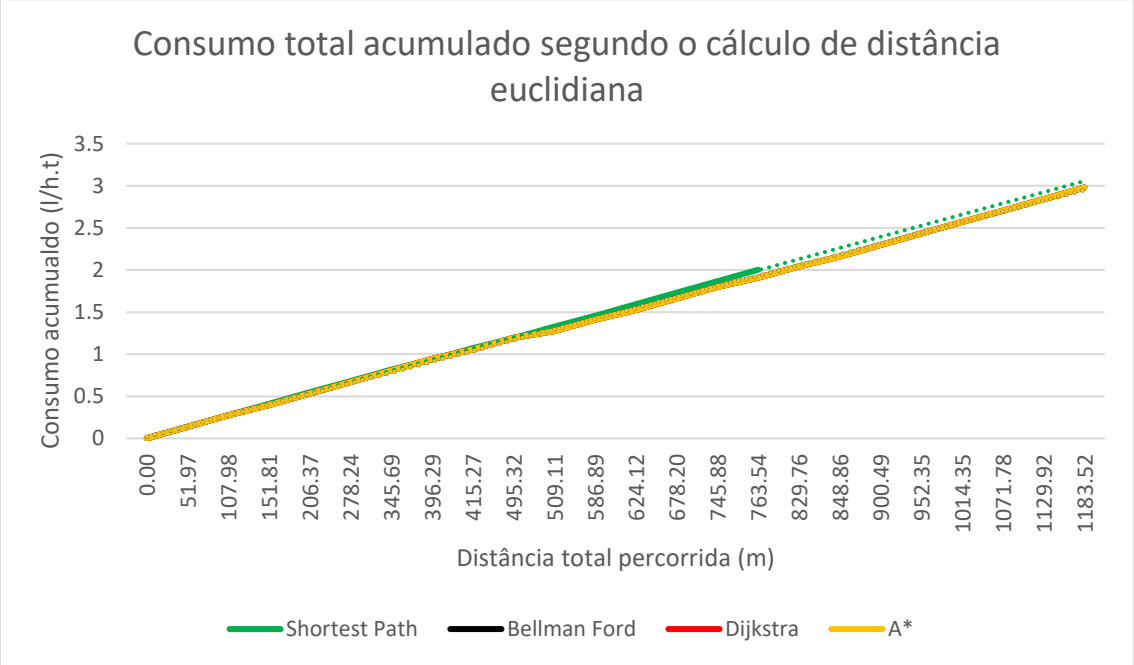
Fonte: O autor

Figura 34 - Distância percorrida no eixo Y segundo o cálculo de distância euclidiana



Fonte: O autor

Figura 35 - Consumo total segundo o cálculo de distância euclidiana



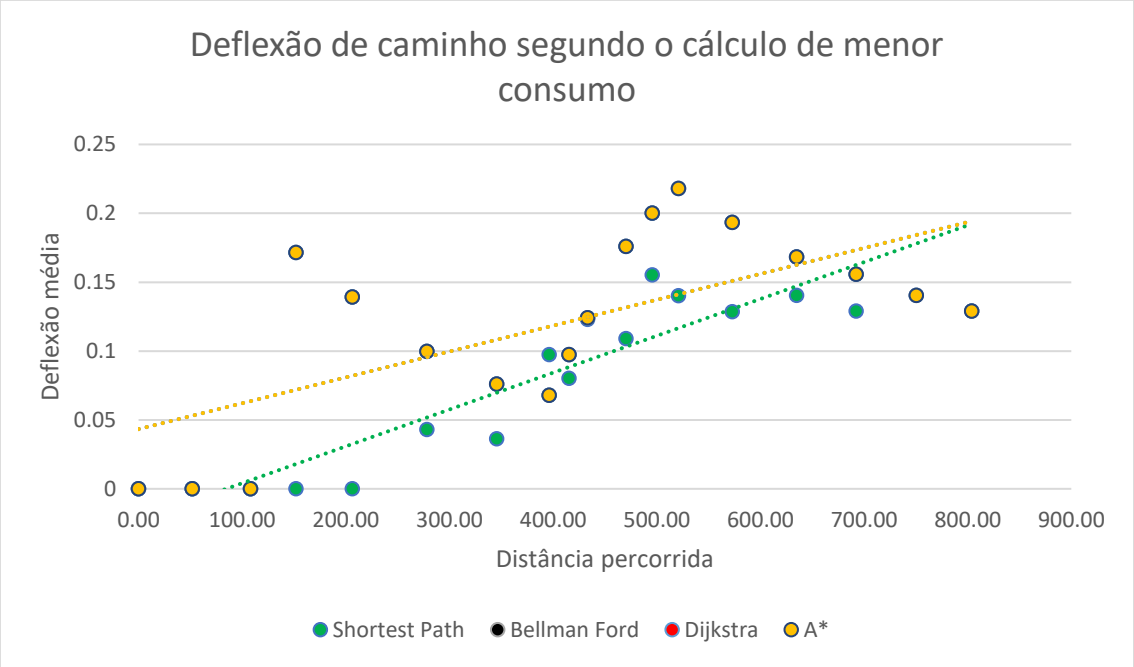
Fonte: O autor

As Figura 36 a Figura 39 demonstram, para efeitos comparativos, os resultados obtidos pelos algoritmos ao se efetuar os cálculos de rota utilizando a fórmula do menor consumo como referência para distância. É possível perceber que opera em uma deflexão média inferior aos outros três algoritmos até uma distância de aproximadamente 800m, ou seja, em toda a rota determinada desta maneira. É importante notar, também, que ao longo da distância determinada por shortest path o

consumo acumulado determinado é maior que nos outros três tipos de código, bem como há maior distância horizontal – segundo o eixo x – sedo percorrida. De maneira geral, entretanto, o consumo total continua sendo maior quando se utiliza Bellman Ford, Djikstra e A*.

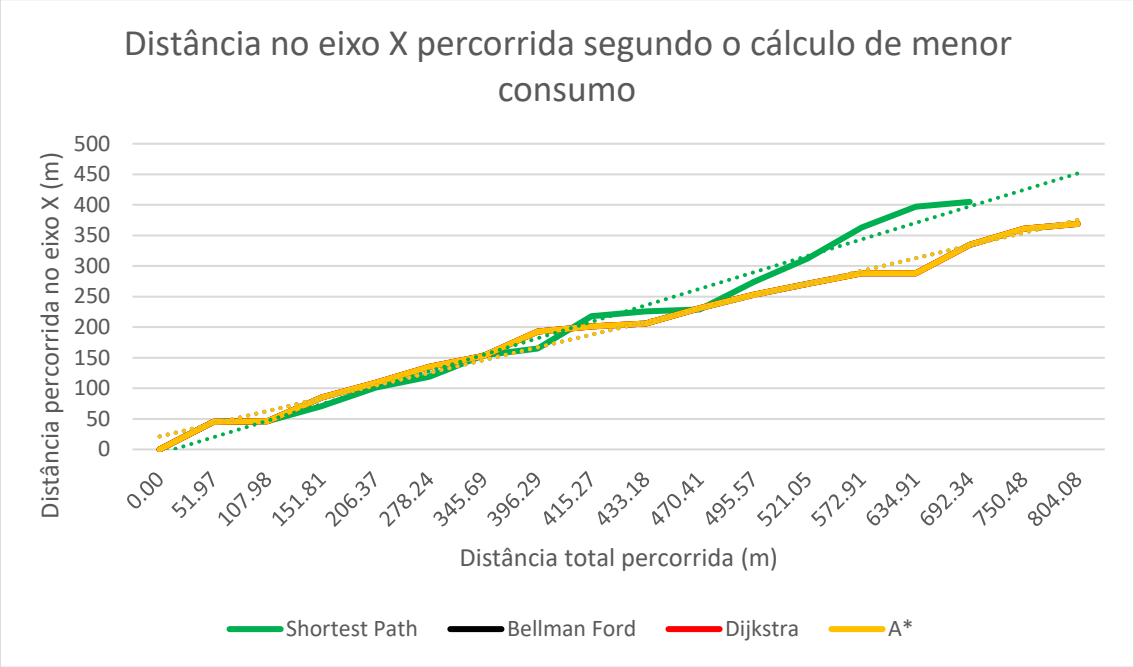
Nos gráficos apresentados abaixo, os pontos, linhas e linhas de tendência gerados pelos algoritmos de Bellman Ford, Dijkstra e A* estão sobrepostas.

Figura 36 - Gráfico de deflexão segundo o cálculo de menor consumo



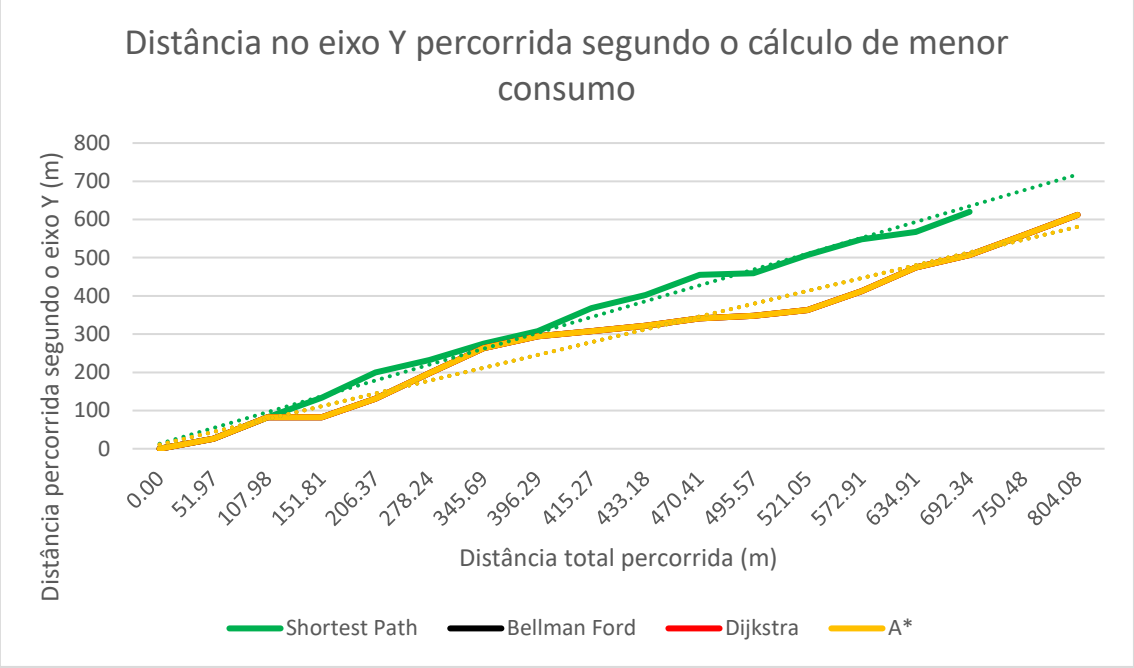
Fonte: O autor

Figura 37 - Distância percorrida no eixo X segundo o cálculo de menor consumo



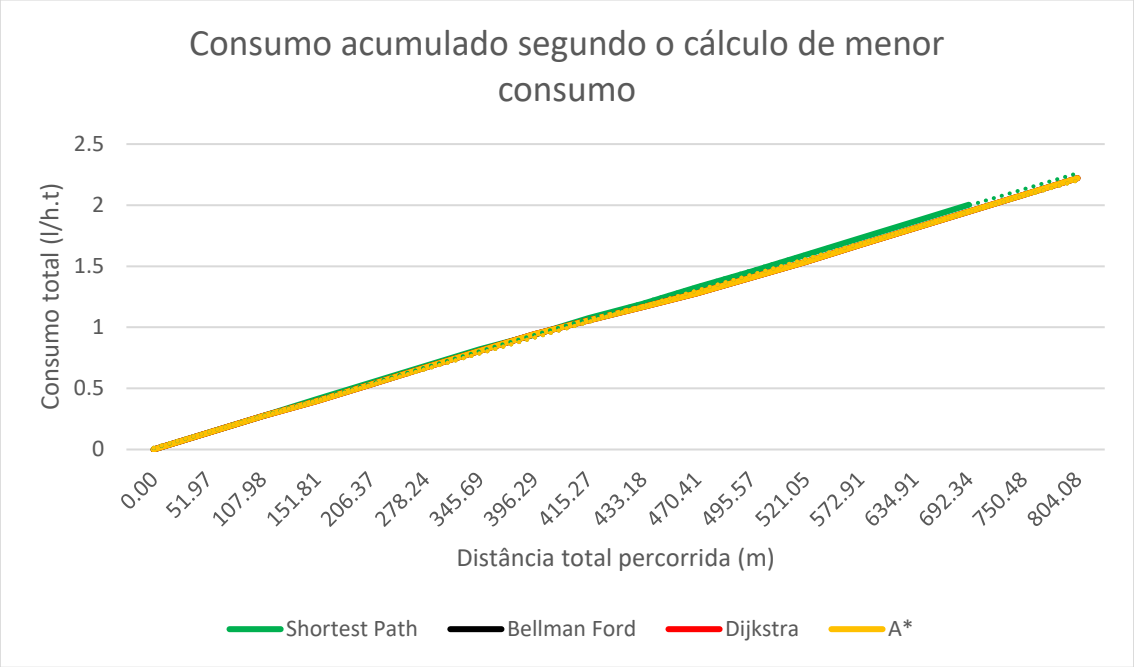
Fonte: O autor

Figura 38 - Distância percorrida no eixo Y segundo o cálculo de menor consumo



Fonte: O autor

Figura 39 - Consumo total segundo o cálculo de menor consumo



Fonte: O autor

6. CONCLUSÕES

A partir da elaboração gráfica para efeito comparativo foi possível analisar os resultados sob duas perspectivas diferentes: estudo comparativos entre os modelos de cálculo e estudos comparativos entre algoritmos.

Iniciando a análise de resultados por meio da segunda ótica, percebe-se nas Figura 35 e Figura ***39*** que o algoritmo *Shortest Path* apresentou um menor consumo quando comparado aos outros três modelos matemáticos. Essa economia foi de 9,803% quando o método de cálculo utilizado foi o de menor consumo e de 32,7658% quando o método de cálculo utilizado foi a distância euclidiana. A diferença foi obtida por meio de uma divisão simples entre o valor de consumo indicado em Shortest Path e o consumo indicado nos outros três algoritmos. Essa redução acentuada deve ser associada também à menor distância percorrida, bem como à menor deflexão média de caminho.

A segunda análise a ser feita é dentre cada um dos algoritmos, sendo comparados os resultados obtidos apenas pelos diferentes métodos de cálculo.

Nos algoritmos *Dijkstra, Bellman Ford e A** foi-se obtida uma economia de 25,4586% no consumo pelo algoritmo de menor consumo, comparando-o aos resultados obtidos por meio dos cálculos de distância euclidiana. Essa economia pode ser conferida nas Figura 19, Figura ***27*** e Figura ***31***, pela simples divisão entre os valores de consumo encontrados Já o algoritmo *Shortest Path* não apresentou nenhuma variação quando comparados os dois métodos de cálculo diferentes, conforme indica a Figura 23.

Após todas essas análises pode-se sugerir que o algoritmo de *Shortest Path,* pelo método de cálculo de menor consumo, é o melhor quando se trata de menor consumo de combustível. É importante lembrar que análises mais profundas, utilizando-se também outros diversos algoritmos de caminho, bem como outras topografias faz-se necessário para a obtenção de resultados mais precisos e conclusivos.

REFERÊNCIAS BIBLIOGRÁFICAS

AHUJA N., WENG, J. and HUANG, T. S. Optimal motion and structure estimation. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, vol. 15, no. 9, p 864-884. 1993.

BOAVENTURA NETTO, Paulo Osvaldo. **Teoria e modelos de grafos**. São Paulo. 1979

COLLINS, William J. **Programação estruturada com estudo de casos em PASCAL**. Rio de Janeiro. 1988. <trad.>

DOE. **Energy and environmental profile of the US mining indusdry.** Austrália. 2002.

DREYFUS, S, **An appraisal of some shortest path algorithms. Operations Research.** p 395 -411. 1969.

EEO. **Analyses of diesel use for mine haul and transport operations.** 2012. Austrália.

EPPSTEIN, D. **Finding the k Shortest Paths**. Report 94-26, Univ. of California, Irvine, Dept. of Information and Computer Science. p 23. 1994.

ERCELEBI, S and BASCETIN, A. **Optimization of shovel-truck sustem for surface mining.** The Journal of The Southern African Institute of Mining and Metallurgy. p 433-439. 2009.

FINN, Ed. **What Algorithms Want: Imagination in the Age of Computing**. 2017.

FORSMAN, B., RÖNNKVIST, E., VAGENAS, N. **Truck dispatch computer simulation in Aitik open pit mine**. International Journal of Surface Mining and Reclamation. p 117-120. 1993.

FREITAS, A. R. R. **Resolvendo o Problema do Caixeiro Viajante via Procedimento de Busca Adaptativa Aleatória Gulosa com Construção Baseada em Redes Neurais Auto-Organizáveis**. Monografia (Bacharel Ciência da Computação) – Universidade Federal de Ouro Preto, Curso de Ciência da Computação. 2009.

GLOVER, F., KLINGMAN, D., PHILLIPS N., SCHNEIDER, R. **New Polynomial Shortest path algorithms and their computational attributes**, Management Science (pre-1986); Vol 31, No 9, p. 1106 – 1128. 1985.

GONZAGA, C. C. **Estudo de algoritmos de buca em grafos e sua aplicalçao a problemas de planejamento.** Dissertação (Pós-Graduação em Engenharia) – Universidade Federal do Rio de Janeiro. 1973.

HILLIER F. S. and LIEBERMAN G. J. **Operations Research**. Internationale Standardlehrbucher der Wirtschafts und Sozialwissenschaften. 1988.

HUNG, M., DIVOKY, J. **Performance of shortest path algorithms in network flow problems**. 1990.

LIMSIRI, C. **Optimization of loader-hauler fleet selection**. European Journal of Scientific Research, p. 266-271. 2011

Mrówczyńska, B. **Route planning of separate waste collection on a small settlement.** 2014. <https://www.researchgate.net/publication/286822916_Route_planning_of_separate_waste_collection_on_a_small_settlement>.

NetworkX. <**https://networkx.github.io/documentation/networkx-1.8/reference/algorithms.shortest_paths.html**>. Acesso em 09 de março 2021.

RABUSKE, R. A. **Inteligência Artificial**. Florianópolis: Editora da UFSC. 1995.

RAMOS, Daniela. **A influência do algoritmo**. Revista Comunicare. 2017.

SILVEIRA, Stefanie. **Máquinas não são preconceituosas**. 2018. Blog. Disponível em: <https://tecnologia.uol.com.br/blogs-e-colunas/ coluna/ste fanie-silveira/2018/05/14/maquinas-nao-sao-preconceituosas.htm>. Acesso em 14 de outubro de 2021.

SILVA, Admilson Alcantara. **Abordagens de Otimização para apoiar a Elaboração e Análise de Roteiros Turísticos**. Tese (Doutorado em Engenharia de Produção) – Universidade Federal de São Carlos. 2017.

SOOFASTAEI, Ali *et al.* **Simulation of Payload Variance Effects on Truck Bunching to Minimise**. Energy Consumption and Greenhouse Gas emission, 15th Coal Operator' Conference, University of Wollongong, The Australian Intitute of Mining and Mettalurgy and Mine Association of Australia. p 337-346. 2015.

SOUMIS, F., ETHIER, J., ELBROND, J. **Truck dispatching in an open pit mine**. International Journal of Surface Mining, ed. 3, 115-119. 1989.

Souza, F.R *et al* **Mine fleet cost evaluation - Dijkstra's optimized path.** 2009**.** REM – International Engineering Journal. Acesso disponível em: <https://doi.org/10.1590/0370-44672018720124>. Acesso em 10 de outubro de 2021.

SZWARCFITER, J. L. **Grafos e algoritmos computacionais**. 1984.

ZHAN, F. B., NOON, C. E. **Shortest Path Algorithms: An Evaluation Using Real Road Networks.** Transportation Science. ed. 32. p 65-73. 1998.