UNIVERSIDADE ESTADUAL PAULISTA - UNESP

Faculdade de Engenharia e Ciências de Guaratinguetá - Campus de Guaratinguetá

FERNANDO JUN OTA

Algoritmo bioinspirado para roteamento de distribuição de alimentos do produtor aos consumidores finais

Guaratinguetá

2024

Fernando Jun Ota

Algoritmo bioinspirado para roteamento de distribuição de alimentos do produtor aos consumidores finais

Trabalho de Conclusão de Curso, apresentada à Universidade Estadual Paulista (UNESP), Faculdade de Engenharia e Ciências de Guaratinguetá, Guaratinguetá, para obtenção do título de Engenheiro em Engenharia Elétrica.

Orientador: Profª Dra. Paloma Maria Silva Rocha Rizol
Coorientador: Dr. Vitor Pinto Ribeiro

Guaratinguetá
2024

Ota, Fernando Jun
O87a Algoritmo bioinspirado para roteamento de distribuição de alimentos do produtor aos consumidores finais / Fernando Jun Ota - Guaratinguetá, 2024.
47 f : il.
Bibliografia: f. 46-47

Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia e Ciências de Guaratinguetá, 2024.
Orientadora: Profa. Dra. Paloma Maria Silva Rocha Rizol
Coorientador: Dr. Vitor Pinto Ribeiro

1. Algoritmos. 2. Logística. 3. Grafos - Teoria dos. I. Título.

CDU 519.712

Luciana Máximo
Bibliotecária/CRB-8 3595

FERNANDO JUN OTA

ALGORITMO BIOINSPIRADO PARA ROTEAMENTO DE DISTRIBUIÇÃO DE ALIMENTOS DO PRODUTOR AOS CONSUMIDORES FINAIS

Trabalho de Conclusão de Curso apresentado à Universidade Estadual Paulista (UNESP), Faculdade de Engenharia e Ciências de Guaratinguetá, Guaratinguetá, para obtenção do título de Engenheiro em Engenharia Elétrica.

Data de defesa: 11 de novembro de 2024

BANCA EXAMINADORA

Profª Dra. Paloma Maria Silva Rocha Rizol
UNESP – Faculdade de Engenharia e Ciências de Guaratinguetá – Campus de Guaratinguetá

Prof Me. Thiago José Michelin
UNESP - Faculdade de Engenharia e Ciências de Guaratinguetá – Campus de Guaratinguetá

Dr. Vitor Pinto Ribeiro
USP - Escola de Engenharia de São Carlos

Dedico todo o esforço empenhado no desenvolvimento deste trabalho de graduação à minha família e, em especial, aos meus queridos avós maternos, Armando Massanori Yokoyama e Olga Takahara Yokoyama, e paternos, Shiro Ota e Kiuyoko Ota, em memória.

AGRADECIMENTOS

Gostaria de expressar minha mais profunda e eterna gratidão à minha família — Jorge Akira Ota, Márcia Yoshie Ota, Karen Tiemi Ota e meu cachorro Yann — por todo o apoio constante ao longo de minha trajetória. Agradeço também à minha namorada, Giulia Mei Kanazawa, por estar sempre ao meu lado e celebrar comigo todas as conquistas. Manifesto ainda minha gratidão à Profª Dra. Paloma Maria Silva Rocha Rizol pela oportunidade e ao Dr. Vitor Pinto Ribeiro pelo suporte, pela confiança e pelos inúmeros ensinamentos transmitidos ao longo do desenvolvimento do projeto. Por fim, mas não menos importante, agradeço aos meus colegas de graduação e de república que compartilharam essa jornada acadêmica ao meu lado.

“Litterarum radices amaræ, fructus dulces”
Aristóteles

RESUMO

As *commodities* são produtos de suma importância para a economia brasileira e então é necessário que as logísticas desses produtos primários utilizem os trajetos mais eficientes desde o produtor até os consumidores finais. Além disso, é importante que haja uma redução na emissão de gases poluentes, já que, no Brasil, os caminhões são responsáveis por grande parte do transporte das *commodities* e emitem uma grande quantidade de gases poluentes. Uma rota ótima seria aquela que há um menor tempo para chegar de um lugar a outro e, consequentemente, possui a menor emissão de gases poluentes. Neste trabalho de graduação será desenvolvido um algoritmo bioinspirado que encontre as melhores rotas partindo de um único ponto representando o produtor até os diferentes locais de chegada que seriam os mercados e, para isso, será utilizado um grafo não direcionado e georreferenciado para representar a malha viária a ser seguida. Um grafo é representado por vértices que são conectados por arestas que podem ter direção ou não, sendo uma possível aplicação do grafo a representação de um mapa geográfico. O problema de roteamento consiste em encontrar a rota ótima entre dois nós diferentes, sendo ela definida como a menor soma dos comprimentos de cada uma das arestas que estão entre o ponto de partida e o de chegada. O algoritmo de Dijkstra é usado para calcular os menores caminhos de um ponto inicial e todos os outros restantes a partir de um grafo não direcionado, sendo este algoritmo o mais utilizado em problemas de roteamento. O algoritmo bioinspirado denominado *Physarum solver* foi desenvolvido com o objetivo de identificar a rota ótima entre dois pontos distintos em um grafo não direcionado. Neste trabalho de graduação utilizou-se o grafo não direcionado para simplificar a representação de uma localização geográfica e diminuir o tempo de execução do algoritmo. Além disso, o algoritmo bioinspirado desenvolvido neste trabalho oferece uma abordagem diferente do algoritmo de Dijkstra, pois permite encontrar rotas ótimas de um ponto de partida para múltiplas saídas, enquanto o Dijkstra se concentra em rotas de entrada e saída única. Para aplicar o *Physarum solver* partindo de um ponto para outros destinos finais, foi necessário modificar o algoritmo original. Para realizar a aplicação final na cidade de Guaratinguetá - SP, utilizou-se outros dois exemplos de localizações com áreas menores, sendo eles um bairro na cidade de São Paulo chamado Vila Esperança e a cidade de Iracemápolis - SP. Uma vez que tentativas e correções foram feitas, comparou-se o desempenho do *Physarum solver* com o algoritmo de Dijkstra e, mesmo com o tempo do algoritmo bioinspirado sendo muito maior, as rotas obtidas para ambos os algoritmos foram idênticas. A partir desta comparação, foram realizadas diferentes aplicações do *Physarum solver* no bairro Vila Esperança localizado na cidade de São Paulo e em Iracemápolis - SP utilizando rotas já validadas pelo algoritmo de Dijkstra para encontrar a rota ótima entre um ponto para diferentes locais. Após essas execuções, aplicou-se o *Physarum solver* para a cidade de Guaratinguetá-SP resultando as rotas ótimas.

PALAVRAS-CHAVE: Algoritmo bioinspirado; *Physarum polycephalum*; *Physarum solver*.

ABSTRACT

Commodities are products of utmost importance for brazilian economy, and therefore it is essential that the logistics of these primary products use the most efficient routes from the producer to the final consumers. Additionally, it is important to reduce pollutant gas emissions, as in Brazil, trucks are responsible for a significant part of *commodities* transport and emit a large amount of pollutant gases. An optimal route would be one with the shortest time to travel from one place to another and, consequently, the lowest emission of pollutant gases. In this undergraduate project, a bio-inspired algorithm will be developed to find the best routes from a single point representing the producer to various destination points representing the markets. For this, an undirected and georeferenced graph will be used to represent the road network to be followed. A graph is represented by vertices connected by edges, which may or may not have direction, and one possible application of the graph is the representation of a geographic map. The routing problem consists of finding the optimal route between two different nodes, defined as the shortest sum of the lengths of each of the edges between the starting and ending points. Dijkstra's algorithm is used to compute the shortest paths from an initial point to all remaining points in an undirected graph, being the most widely used algorithm in routing problems. The bio-inspired algorithm known as the *Physarum solver* was developed to identify the optimal route between two distinct points in an undirected graph. In this undergraduate project, the undirected graph was used to simplify the representation of a geographical location and reduce the algorithm's runtime. Additionally, the bio-inspired algorithm developed in this work offers a different approach from Dijkstra's algorithm, as it allows for finding optimal routes from a starting point to multiple exits, while Dijkstra focuses on single-source to single-destination routes. To apply the *Physarum solver* from one point to multiple final destinations, it was necessary to modify the original algorithm. For the final application in the city of Guaratinguetá - SP, two other examples of locations with smaller areas were used, namely a neighborhood in the city of São Paulo called Vila Esperança and the city of Iracemápolis located in the state of São Paulo. After some experiments and some fine-tuning, the performance of the *Physarum solver* was compared with Dijkstra's Algorithm. Despite the much longer runtime of the bio-inspired algorithm, the routes obtained for both algorithms were identical. Based on this comparison, various applications of the *Physarum solver* were carried out in the neighborhood of Vila Esperança and in the city of Iracemápolis, using routes already validated by Dijkstra's Algorithm to find the optimal route from one point to different locations. After executing the algorithm for these use cases, the *Physarum solver* was applied to the city of Guaratinguetá-SP, resulting in the optimal routes.

KEYWORDS: Bioinspired algorithm; *Physarum polycephalum*; *Physarum solver*.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

As *commodities* são produtos primários feitos em larga escala com características uniformes e são essenciais à economia brasileira. Para que isso seja possível, um dos componentes principais ao sucesso desse setor é a logística que é responsável por fazer o trajeto mais eficiente dos produtos primários até ao consumidor ou à exportação. Os alimentos são um dos integrantes desse grupo de matérias-prima e constituem cerca de 26,6% do Produto Interno Bruto do país, evidenciando o destaque do setor na economia (CONFEDERAÇÃO DA AGRICULTURA E PECUÁRIA DO BRASIL., 2020).

A rede de distribuição de alimentos é um dos fatores inerentes ao desenvolvimento de um país, já que promove um crescimento econômico para diferentes setores da sociedade (Soliani; Argoud, 2018). No Brasil, o setor de transporte em caminhões é responsável por 60% de toda a distribuição de produtos pelo país (Feltrin, **Estradão**, São Paulo, 2020). Como o transporte em veículos de grande porte que fazem uso de combustíveis fósseis acarreta na emissão de grande volume de gases poluentes, existem algumas abordagens que promovem a redução das emissões de gases nocivos ao meio ambiente. Para uma menor emissão de poluentes, um dos fatores relevantes é a utilização das rotas com o menor caminho entre ponto de carga dos alimentos até o destino, usualmente um intermediário, como um mercado ou depósito, pois esses caminhos além de acarretarem em um menor consumo de combustível, os mesmos também apresentam o menor tempo de entrega se comparado com as outras possíveis rotas, supondo condições como o tráfego, o clima, o pavimento, entre outros fatores como constantes ao longo das rotas.

Para tratar de problemas que têm como objetivo encontrar a menor rota, são utilizadas diferentes heurísticas que simulam a situação e, com os dados obtidos por ela baseado na abordagem escolhida, são criados parâmetros e com os cálculos realizados para atingir o objetivo final que é identificar a distância mais curta entre dois pontos. Existem diferentes algoritmos que buscam o caminho mais curto entre diferentes locais, como o algoritmo da colônia de formiga (do inglês *Ant Colony Optimization* - ACO) (Luo *et al.*, 2019), o do enxame (do inglês *Particle Swarm Optimization* - PSO) (Adamu *et al.*, 2018), o algoritmo desenvolvido por Dijkstra (Noto; Sato, 2000), entre outros.

Os algoritmos bioinspirados são desenvolvidos com base nos comportamentos observados na natureza para resolver diferentes tipos de problemas (Passarini,). Uma das inspirações naturais é o comportamento do *Physarum polycephalum* que pertence ao reino protista e possui apenas uma célula com diversos núcleos (Howard, 1931). Quando esse organismo busca por alimentos e que estão distribuídos em diferentes regiões, ele se espalha pela superfície e se conecta aos alimentos para consumí-los e essas ligações realizadas apresentarão sempre o menor caminho conectando todas as fontes de alimentos (Dussutour *et al.*, 2010). Com essa característica, foram desenvolvidos algoritmos de acordo com o comportamento do *Physarum polycephalum* para encontrar a menor rota entre dois locais e um exemplo dessa aplicação pode ser a otimização

logística dos alimentos entre os múltiplos fornecedores e os pontos de destino (Chua *et al.*, 2021).

Na literatura científica, é comum que os problemas que buscam os caminhos mais curtos entre os pontos sejam resolvidos com o algoritmo desenvolvido por Dijkstra (1959), porém, segundo Tero, Kobayashi & Nakagaki (2006), o algoritmo de Dijkstra exige um alto tempo de execução quando é submetido a um grande número de locais. No mesmo artigo, outro impasse com o algoritmo de Dijkstra é que ele apresenta dificuldades em problemas de navegação que exigem uma adaptabilidade para encontrar novas rotas ótimas. Essas rotas ideais são aquelas que minimizam o tempo de viagem do ponto de partida até o destino final. No entanto, devido ao dinamismo das condições de transporte, o algoritmo de Dijkstra não apresenta essa flexibilidade.

As dificuldades citadas anteriormente serviram como base para o algoritmo bioinspirado no *Physarum polycephalum* denominado *Physarum solver* (Tero; Kobayashi; Nakagaki, 2006). Esse algoritmo consegue encontrar a menor rota entre dois locais independentemente da quantidade de pontos envolvidos e tem adaptabilidade em criar e recriar outros caminhos otimizados mesmo com alterações durante o trajeto. Exemplos foram derivados do *Physarum solver* original como apresentado por Watanabe *et al.* (2011), que foi aplicado à otimização do sistema ferroviário.

Outro artigo relevante é Chua *et al.* (2021), que apresenta um algoritmo inspirado no *Physarum polycephalum* para um produtor que tem como objetivo reduzir os gastos financeiros da logística de seus produtos. Entretanto, o trabalho de Chua *et al.* (2021) não realizaou uma abordagem com um detalhamento tão profundo quanto Zhang *et al.* (2017) que desenvolveu um algoritmo bioinspirado no *Physarum polycephalum* para um produtor que procura minimizar os custos financeiros do transporte de seus alimentos e também a redução de poluentes emitidos tanto pela produção quanto pela distribuição. Além disso, o artigo de Zhang *et al.* (2017) utilizou uma abordagem que envolve a produção agrícola, os centros de armazenamento, a distribuição, e os pontos de entrega e de demanda. O grupo de pesquisadores citados incluiu os cálculos da emissão de gases poluentes desde a plantação até a distribuição desses produtos para encontrarem a melhor rota a ser seguida.

1.1 OBJETIVO

A expectativa deste trabalho de graduação é o desenvolvimento de um *Physarum solver* que possa aplicar o algoritmo em diferentes grafos não direcionados georreferenciados de forma que o resultado obtido seja a melhor rota de um ponto de partida para outros pontos de chegada e o tempo de execução do algoritmo seja razoável. A rota ótima é aquela que possui o melhor tempo de entrega se comparado com diferentes rotas e também uma economia na utilização de combustíveis e, consequentemente, uma menor emissão de gases poluentes. Essas rotas obtidas pelo o *Physarum solver* possuem como referência o mesmo ponto de partida. Além disso, espera-se deste trabalho de graduação que as rotas ótimas obtidas pelo algoritmo bioinspirado sejam as mesma que foram calculadas a partir do algoritmo de Dijkstra. É de suma importância ressaltar que enquanto o algoritmo de Dijkstra gera rotas de um ponto a outro, o *Physarum solver*

é capaz de obter múltiplas rotas de um local de partida para diferentes destinos finais.

Este trabalho de graduação é dividido em cinco capítulos, sendo eles, introdução, referencial teórico, materiais e métodos, resultados e discussão, conclusão e trabalhos futuros. O primeiro capítulo é utilizado para evidenciar a importância da otimização de rotas e a utilização de algoritmos bioinspirados nesse tema. Outro ponto importante deste capítulo é demonstrar a literatura utilizada neste trabalho. O segundo capítulo descreve as bases teóricas utilizadas para o desenvolvimento do trabalho de graduação. O terceiro capítulo é responsável por apresentar os materiais e os métodos. O quarto capítulo descreve tanto os resultados obtidos quanto as análises sobre eles. O último capítulo é dedicado para a conclusão e comentários sobre os possíveis trabalhos futuros relacionados a este trabalho de graduação.

1.2 JUSTIFICATIVA

Uma aplicação do *Physarum solver* é a utilização para a redução de custos e, consequentemente, aumento do lucro a partir da logística pela melhor rota do local desde o inicio, passando pelo locais intermediários até o destino final dos produtos como foi realizado pelos cientistas Chua *et al.* (2021). Eles utilizaram diferentes tipos de informações para o desenvolvimento do cálculo que definirá a melhor rota e isso supera outras abordagens que consideram apenas a distância entre os locais. O algoritmo desenvolvido teve o tempo de execução comparado tanto com o PSO quanto com um tipo de algoritmo genético e os resutados obtidos após as comparações foi que o tempo de execução do *Physarum solver* foi muito menor do que o tempo tanto do *Particle Swarm Optimization* quanto do algoritmo genético.

Outra aplicação do algoritmo bioinspirado é para a otimização de rotas ferroviárias. Para isso, Watanabe *et al.* (2011) utilizou como base uma das versões do *Physarum solver* de forma que incluisse outros pontos como o destino final. Além disso, considerou-se a quantidade total de passageiros a bordo, e cada estação do sistema ferroviário recebe tanto pessoas que estão à espera de um trem quanto aquelas que desembarcarão nesse mesmo local. Portanto, cada uma dessas estações deve ser priorizada de acordo com o fluxo de indivíduos. Como comparação em relação à performance do algoritmo desenvolvido no trabalho, utilizou-se o algoritmo de Dijkstra e foi observado que, em alguns locais, a quantidade calculada do fluxo de pessoas pelo algoritmo de Dijkstra é maior do que no algoritmo modificado do *Physarum solver*, e os valores obtidos mais próximos da realidade foi do algoritmo bioinspirado.

Zhang *et al.* (2017) realiza uma aplicação modificada do *Physarum solver* com objetivo de minimizar tanto os custos financeiros de um produtor para distribuir seus produtos para diferentes locais como fábricas, centros de distribuição e pontos de venda ou demanda, quanto a emissão dióxido de carbono. O algoritmo desenvolvido considera um ponto de partida, outros lugares como intermédio e diversos destinos finais e, para encontrar o melhor caminho a ser percorrido, são utilizados os parâmetros de custo do transporte até os locais intermediários e finais e de emissão total de gases poluentes desde a produção até o quanto será emitido por

conta do transporte dos produtos. O algoritmo desenvolvido foi comparado com um método de Naugurney que é utilizado em problemas que envolvem rede de distribuição de alimentos de forma sustentável e obteve que, apesar dos parâmetros terem valores numéricos com uma diferença desprezível, as iterações e o tempo de execução no algoritmo desenvolvido são expressivamente menores do que no método de Naugurney.

É importante ressaltar que nenhum dos artigos citados anteriormente trata diretamente sobre o desenvolvimento de um algoritmo para encontrar a rota ótima com o ponto de partida sendo o local do produtor até os destinos que receberão os alimentos. A rota ótima é aquela que possui tanto o menor tempo de entrega se comparado com outras possíveis rotas, quanto a menor emissão de gases poluentes. Isso ocorre já que o caminho escolhido terá uma economia de combustível em relação a outros trajetos por ser a melhor escolha. Outra observação relevante é sobre a revisão bibliométrica que é mostrada pela Tabela 1 é notório que existe uma grande quantidade de artigos relacionados aos algoritmos bioinspirados, porém não há artigos sobre aplicações para o *Physarum solver* em otimização de rotas de distribuição de alimentos.

Tabela 1 – Revisão bibliométrica utilizando Scopus.

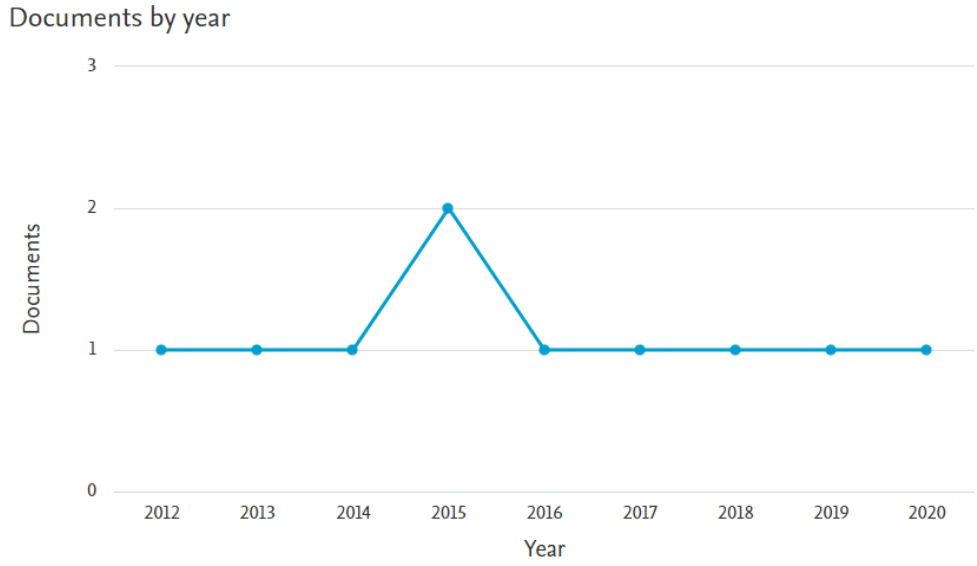
Termo utilizado na pesquisa	Número de artigos encontrados
"Bio-inspired algorithm"	2.332
"Bio-inspired algorithm" AND "Physarum polycephalum"	24
"Bio-inspired algorithm" AND "Physarum polycephalum" AND "shortest path"	10
"Bio-inspired algorithm" AND "Physarum polycephalum" AND "shortest path" AND "supply chain"	0

Fonte: Extraido de *Scopus* (2024).

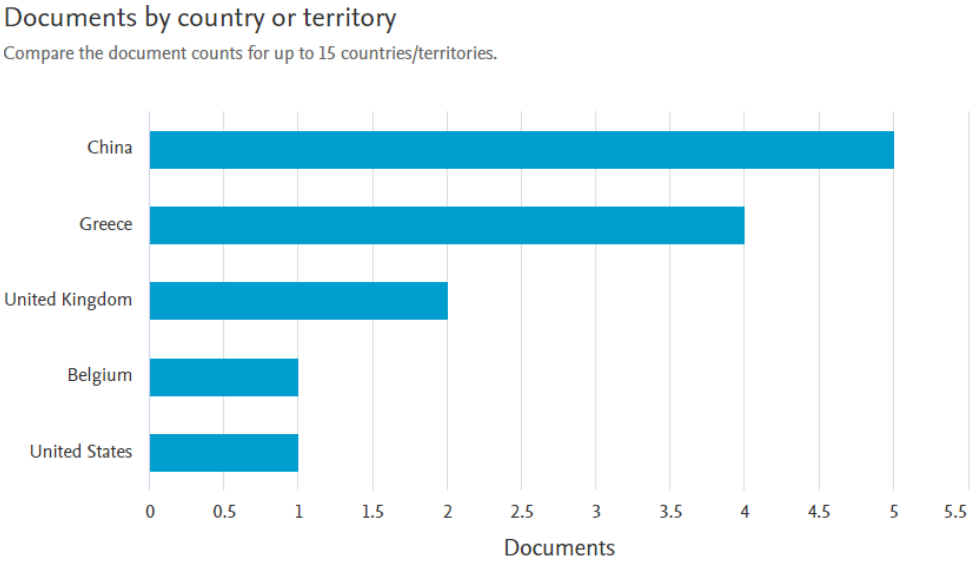
A Figura 1 mostra a análise feita com os documentos publicados com os termos "*Bio-inspired algorithm*" AND "*Physarum polycephalum*" AND "*shortest path*", já que não houve resultados para o último grupo de palavras chaves. A Figura 1a mostra os documentos publicados por ano e nota-se que é um assunto a ser explorado, já que não há muitos documentos publicados com essa temática. A Figura 1b também mostra os artigos publicados utilizando os termos citados, mas utilizando os países em que foram submetidos.

Figura 1 – Análise da revisão bibliométrica.

(a) Documentos publicados por ano.



(b) Documentos publicados por pais.



Fonte: Ambas as Figuras foram extraidas do *Scopus* (2024).

1.3 DELIMITAÇÃO DA PESQUISA

Foram realizado diferentes testes para definir o tipo de grafo a ser aplicado para a representação dos locais georreferenciados como multigrafo ou grafo direcionado ou não direcionado. Para a representação de uma cidade, é necessário uma quantidade expressiva de nós de um grafo, o que ocasiona um tempo de execução consideravel. Para um multigrafo direcionado, o número de arestas pode ser muito maior se comparado com um grafo não direcionado, o que aumenta mais ainda o tempo de execução do programa. Com essas análises, escolheu-se o grafo não direcionado como representação das regiões para serem aplicadas no *Physarum solver*.

2 REFERENCIAL TEÓRICO

Este capítulo tem como objetivo estabelecer as bases teóricas utilizadas para o desenvolvimento deste trabalho de graduação.

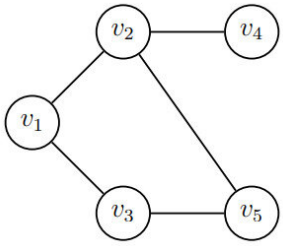
2.1 GRAFO

Um grafo G é definido por $G = \{E, V\}$, sendo V o conjunto de vértices e E o de arestas. Um par de nós $\{u, v\}$ pode ser arestas que têm um valor numérico associado a cada uma delas, como o comprimento (l). Uma aresta que conecta os vértices u e v que estão em contidos em G tem o comprimento do arco representado por $l_{u,v}$. Além disso, as arestas podem ser direcionadas ou não. Isso significa que, se as arestas $u, v \neq v, u$, o grafo é considerado direcionado. Por outro lado, se as arestas são equivalentes, ou seja, $u, v = v, u$, o grafo é classificado como não direcionado. (Bender; Williamson, 2010).

A característica de um grafo ser conexo é se o seu conjunto de vértices puder ser dividido em dois conjuntos disjuntos e não-vazios, V_1 e V_2, de tal forma que exista pelo menos uma aresta com uma extremidade em V_1 e outra extremidade em V_2. Isso garante que haja pelo menos uma aresta entre dois nós distintos (Chartrand, 1985).

O grafo G pode ser representado a partir de um diagrama no qual os vértices são representados por pontos e as arestas por linhas ou curvas (Deo, 2016). A Figura 2 representa um grafo não direcionado no qual $V = \{v_1, v_2, v_3, v_4, v_5\}$ e $E = \{v_1v_2, v_1v_3, v_2v_4, v_2v_5, v_3v_5\}$.

Figura 2 – Grafo não direcionado.

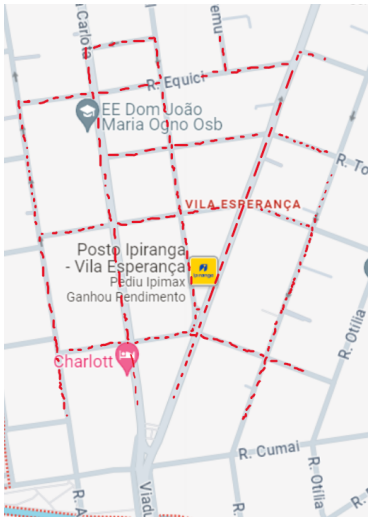


Fonte: Autoria própria.

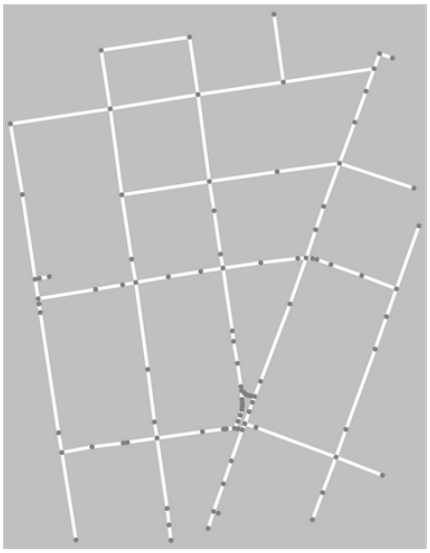
É possível associar um grafo a diferentes aplicações, como redes de computadores, cadeias alimentares, entre outros exemplos (Adal; Ortega, 2018). A Figura 3 mostra a comparação entre uma localização geográfica gerada pelo *Google Maps*, exibida na Figura 3a, e um grafo georreferenciado gerado a partir dela, mostrado na Figura 3b por meio da plataforma *OpenStreetMaps*. A localização utilizada corresponde a uma área do bairro Vila Esperança, na cidade de São Paulo, e o traçado vermelho, feito manualmente sobre as vias do bairro, representa o grafo exibido na Figura 3a.

Figura 3 – Comparação entre uma localização geográfica e um grafo georreferenciado.

(a) Localização geográfica.



(b) Grafo georreferenciado.



Fonte: (a) Extraído de *Google Maps* (2024); (b) Autoria própria, via biblioteca *OpenStreetMaps*.

2.2 PROBLEMAS DE ROTEAMENTO

O problema de roteamento (problema do caminho mais curto) consiste em utilizar a rota entre dois nós distintos de um grafo com o menor custo de travessia possível. Uma rota (P) é determinada pela soma de comprimentos entre os vértices u e v, u, v pertencentes a V e $u \neq v$, sendo $u, v \in V$, de tal forma que $\sum_{p \in P} f(p)$ seja mínimo entre todos os caminhos possíveis entre u e v, tendo $f(p)$ como o comprimento da aresta p (Gallo; Pallottino, 1988).

Para um vértice v e cada inteiro $i \leq k$, sendo k o número total de arestas E em G, determina-se uma função $dist(v, i)$ que corresponde à menor distância de um nó origem s até o vértice v utilizando i arestas. Inicialmente, os valores iniciais para $dist(v, 0)$ são ∞ para todos os vértices, exceto s que possui um valor numérico nulo, pois é o ponto de partida. Com isso, obteve-se uma equação geral para $dist(v, i)$, como mostra a equaçao 1 (Henig, 1986).

$$dist(v, i) = min_{(u,v) \in E} \{dist(u, i - 1) + l_{u,v}\} \tag{1}$$

2.3 ALGORITMO DE DIJKSTRA

Dijkstra (1959) apresenta um algoritmo para identificar o menor caminho entre um vértice inicial e todos os demais. Para isso, utilizou-se um grafo não direcionado descrito por $G = \{E, V\}$, com arestas que possuem comprimentos com valores numéricos não negativos. Como trata-se de um *shortest path problem*, utiliza-se a função da distância $dist(v, i)$, sendo $dist(s, i) = 0$ e $dist(v, 0) = \infty$. É escolhido iterativamente uma aresta $\{v, u\}$ sendo u um dos vértices de tal forma que $dist(v, u)$ seja o menor valor possível.

Além disso, são determinados os vértices ainda não visitados (Q) e, a cada iteração, seleciona-se o vértice mais próximo do nó atual ainda não visitado (anterior). Recalcula-se as distâncias totais entre o novo nó de partida e os vértices vizinhos (alternativa). O vértice atual passa a ser parte do grupo de nós já visitados e todo esse processo é repetido até que não haja vértices não visitados. Com isso, o resultado obtido é um conjunto de distâncias mínimas de um ponto inicial a todos os outros vértices. O Quadro 1 apresenta o pseudocódigo para a aplicação do algoritmo de Dijkstra.

Quadro 1 – Algoritmo de Dijkstra.

```
1:  função DIJKSTRA(G, s)
2:      para todos v in G.V faça
3:          dist[v] ← ∞
4:          anterior[v] ← INDEFINIDO
5:          adicionar v a Q
6:      fim para
7:      dist[s] ← 0
8:      enquanto Q não está vazio faça
9:          u ← vértice em Q com min dist[u]
10:         remover u de Q
11:         para todos v em vizinhos de u ainda em Q faça
12:             alternativa ← dist[u] + G.E(u, v)
13:             se alternativa < dist[v] então
14:                 dist[v] ← alternativa
15:                 anterior[v] ← u
16:             fim se
17:         fim para
18:     fim enquanto
19:     retorne dist[], anterior[]
20: fim função
```

Fonte: Adaptado de Dijkstra (1959).

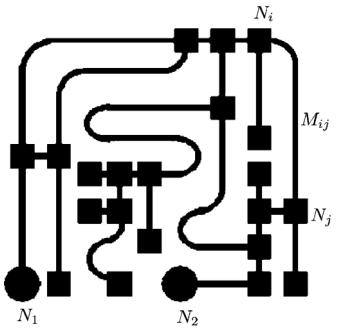
O algoritmo de Dijkstra tornou-se o mais utilizado para problemas de roteamento devido a sua simples aplicação em grafos que não possuem um número elevado de arestas e vértices (Risald; Suyoto, 2017). O algoritmo desenvolvido por Dijkstra (1959) serviu como base para outros importantes algoritmos como algoritmo de Floyd-Warshall (Cormen *et al.*, 2001), algoritmo de Bellman-Ford (Bannister; Eppstein, 2012) e algoritmo A* (Hart; Nilsson; Raphael, 1968).

2.4 *PHYSARUM SOLVER*

Tero, Kobayashi & Nakagaki (2006) desenvolve um algoritmo bioinspirado no *Physarum polycephalum* para resolver o problema do caminho mais curto denominado *Physarum solver* e o seu objetivo principal é encontrar a menor rota entre dois pontos diferentes. Foi determinado um grafo não direcionado como ilustrado pela Figura 4, na qual os nós são representados por N_i,

sendo *i* os diferentes nós, N_1 o ponto de partida e o N_2 destino. Além disso, M_{ij} representa a seção das conexões realizadas pelo *Physarum polycephalum* entre os nós N_i e N_j, sendo *j* outros nós presentes no grafo.

Figura 4 – Labirinto representado por um grafo.



Fonte: Extraído de Tero, Kobayashi & Nakagaki (2006).

A equação de Poiseuille descreve o fluxo de um fluido em um tubo cilíndrico devido a uma diferença de pressão. Para o *Physarum solver*, utilizou-se uma aproximação para a equação de Poiseuille para calcular o fluxo Q_{ij} que passa por M_{ij} de N_i a N_j. A equação 2 representa essa adaptação para o cálculo do fluxo.

$$Q_{ij} = \frac{D_{ij}}{L_{ij}}(p_i - p_j) \tag{2}$$

sendo D_{ij} e L_{ij} a condutivade e comprimento do segmento M_{ij} respectivamente. A variável p_i representa a pressão presente no nó N_i.

A Lei de Kirchhoff é utilizada para a análise de circuitos elétricos, mas também pode ser aplicada a redes de fluxo de fluido. A lei dos nós é um dos princípios da Lei de Kirchhoff que consiste na soma dos fluxos que entram em um nó é igual à soma dos fluxos que saem, garantindo a conservação dos fluxos. Aplicando-se a Lei de Kirchhoff em cada um dos nós, temos as equações 3, 4 e 5

$$\sum_i Q_{ij} = 0 \quad (j \neq 1, 2) \tag{3}$$

$$\sum_i Q_{i1} + I_0 = 0 \tag{4}$$

$$\sum_i Q_{i2} - I_0 = 0 \tag{5}$$

sendo I_0 o fluxo constante do nó de partida.

Para a adaptação de espessura tubular do *Physarum polycephalum*, assume-se que a condutividade varia em função do tempo. Esta variação depende do fluxo, como mostrado na equação 6

$$\frac{d}{dt}D_{ij} = f(|Q_{ij}|) - rD_{ij} \tag{6}$$

como *f(Q)* é uma função crescente com *f(0) = 0*, isso implica que a condutividade tende a diminuir exponecialmente, porém ela é aumentada pelo fluxo que passa pelo tubo. A equação 7 mostra a adaptação realizada.

$$\frac{d}{dt}D_{ij} = \alpha|Q_{ij}| - rD_{ij} \tag{7}$$

sendo α um coeficiente positivo que ajusta a taxa de crescimento da espessura do tubo (D_{ij}) em resposta ao fluxo $|Q_{ij}|$. A variável α representa a eficiência com que o fluxo $|Q_{ij}|$ promove o crescimento do tubo, onde um valor maior de α significa que a espessura do tubo aumenta mais rapidamente com o fluxo. Já r é um coeficiente positivo que representa a taxa de decaimento ou desgaste da espessura do tubo (D_{ij}). O termo r representa a tendência natural do tubo de diminuir a espessura, onde um valor maior de r indica que a espessura do tubo decresce mais rapidamente ao longo do tempo.

Como o comprimento dos segmentos mantém-se constante, a equação desenvolvida por Tero, Kobayashi & Nakagaki (2006) mostrada pela equação 8 para a pressão é gerada a partir das equações 2, 4 e 5.

$$\sum_i \frac{D_{ij}}{L_{ij}}(p_i - p_j) = \begin{cases} -I_0 \text{ para j = 1,} \\ \quad I_0 \text{ para j = 2,} \\ 0 \text{ outros casos.} \end{cases} \tag{8}$$

Definindo p_2 = *0*, as outras pressões são calculadas a partir da equação 8, obtendo-se também todos os fluxos da equação 2. Com isso, obtém-se o caminho mais curto entre os pontos de partida e de destino.

2.5 DIFERENÇA ENTRE O ALGORITMO DE DIJKSTRA E O *PHYSARUM SOLVER*

Embora ambos os algoritmos sejam utilizados para resolver o problema da rota mais curta, eles operam de maneiras distintas. O algoritmo de Dijkstra calcula diretamente o menor caminho em um grafo, enquanto o algoritmo inspirado no *Physarum polycephalum* ajusta iterativamente as condutâncias dos caminhos, simulando o comportamento adaptativo do organismo para encontrar a rota ideal entre dois pontos.

Além disso, o algoritmo de Dijkstra gera a soluções ótimas em todas as execuções, enquanto o algoritmo bioinspirado resulta possíveis rotas e as melhores rotas podem estar contida nesses resultados. Além disso o *Physarum solver* acaba sendo mais adequado para resolver problemas em grafos nos quais a estrutura e o peso das arestas podem mudar ao longo do tempo.

3 MATERIAIS E MÉTODOS

Este capítulo tem como objetivo descrever os materiais e os métodos utilizados durante o desenvolvimento deste trabalho de graduação, sendo eles: etapas para o desenvolvimento deste trabalho de graduação, configurações de *hardware* e *software*, tipo de grafo utilizado, modificações no algoritmo, critério de comparação do *Physarum solver* em relação ao algoritmo de Dijkstra e localizações georreferenciadas.

3.1 ETAPAS PARA O DESENVOLVIMENTO DESTE TRABALHO

O desenvolvimento deste trabalho de graduação seguiu diferentes etapas para ser realizado, sendo elas: planejamento inicial e definição do problema, revisão da literatura e pesquisa teórica, desenvolvimento do algoritmo, testes preliminares e ajustes, aplicação do algoritmo em cenários reais, análise e discussão dos resultados e conclusões e perspectivas futuras.

- **Definição do problema**: O problema envolve a otimização do roteamento para distribuição de alimentos, visando minimizar custos logísticos e emissões de gases poluentes. Para isso, foram levantados dados sobre transporte de alimentos e algoritmos de roteamento com o objetivo de obter as melhores rotas. O objetivo principal é desenvolver um algoritmo bioinspirado para gerar rotas otimizadas e reduzir o impacto ambiental.

- **Revisão e pesquisa da literatura**: A pesquisa revisou algoritmos bioinspirados, como o *Physarum solver*, comparando-os com métodos tradicionais, como o algoritmo de Dijkstra. Também foram revisados aspectos logísticos da distribuição de alimentos no Brasil e conceitos de grafos direcionados e não direcionados para representação de malhas viárias.

- **Desenvolvimento do algoritmo bioinspirado**: Optou-se por utilizar o *Physarum solver*, implementado em Python com as bibliotecas *NumPy* e *NetworkX*. O algoritmo foi adaptado para lidar com múltiplos destinos em grafos não direcionados, e foram integradas bibliotecas para para obter malhas viárias georreferenciadas.

- **Testes e ajustes do *Physarum solver***: O algoritmo foi testado em cenários simplificados, onde foram feitos ajustes para garantir rotas otimizadas.

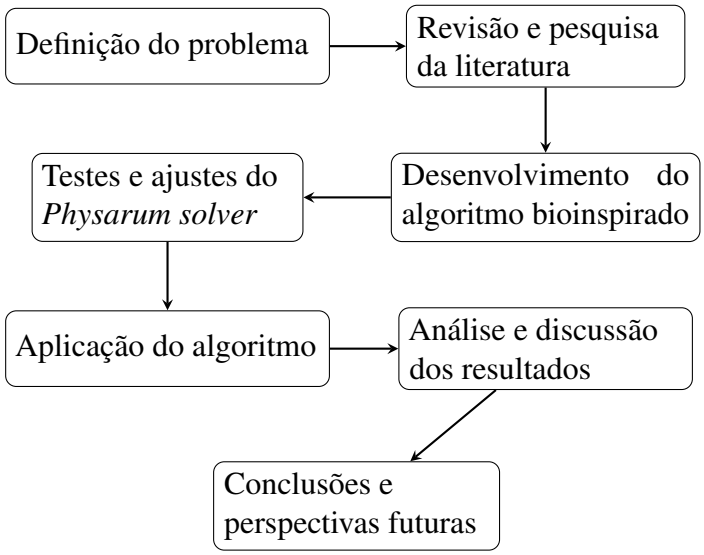
- **Aplicação do algoritmo**: O algoritmo foi testado em áreas geográficas reais como um bairro da cidade de São Paulo-SP Vila Esperança, uma cidade de pequeno porte como Iracemápolis e uma cidade de médio porte como Guaratinguetá. Seus resultados foram comparados com os do algoritmo de Dijkstra, considerando rotas geradas e tempo de execução.

- **Análise e discussão dos resultados**: As rotas geradas foram avaliadas quanto à eficiência. Uma análise comparativa do tempo de execução entre o *Physarum solver* e o algoritmo de Dijkstra foi realizada, validando os resultados obtidos.

- **Conclusões e perspectivas futuras**: Os resultados confirmaram a aplicabilidade do *Physarum solver* na otimização de rotas de distribuição de alimentos. Sugeriram-se melhorias no algoritmo, como otimizações no tempo de execução e adaptação para grafos direcionados, além de novas explorações em cenários maiores e mais complexos.

Para o desenvolvimento deste trabalho de graduação, foi seguido o fluxograma de organização do trabalho mostrado pela Figura 5.

Figura 5 – Fluxograma de organização do trabalho.



Fonte: Autoria própria.

3.2 CONFIGURAÇÕES DE *HARDWARE* E *SOFTWARE*

Para implementar este trabalho utilizou-se um computador e código desenvolvido na linguagem de programação *Python*. A configuração do computador utilizado é um processador *Intel(R) Core(TM) i5-8300H* CPU de 2,30 GHz, memória RAM de 16,0 GB e sistema operacional *Windows 11 Home*.

O desenvolvimento do algoritmo bioinspirado neste trabalho de graduação foi realizado utilizando a linguagem de programação *Python 2.7.18* e os seus módulos, sendo eles: *OSMNX*, *NetworkX* e *NumPy*.

- ***OSMNX 1.1.2***: O módulo é responsável por construir um grafo baseado em características georreferenciadas utilizando a tecnologia *OpenStreetMaps*. Ele permite baixar dados geoespaciais e criar grafos de redes de ruas, ciclovias, trilhas, entre outras funções.

- ***NetworkX 2.6.3***: Este módulo realiza manipulações em diferentes tipos de grafos. Ele fornece ferramentas para a criação, manipulação e estudo da estrutura, dinâmica e funções de redes complexas.

- ***NumPy 1.21.4***: Este módulo realiza diferentes operações matemáticas em vetores e matrizes. Ele é a base para a diversas bibliotecas científicas em *Python*, fornecendo suporte para grandes vetores e matrizes multidimensionais, juntamente com uma coleção de funções matemáticas de alto nível para operar com esses diferentes vetores.

3.3 TIPO DE GRAFO UTILIZADO

Para este trabalho de graduação será utilizado um grafo não direcionado para a simulação de uma cidade ou bairro no qual as arestas representam as vias e os locais são representados pelos vértices. Um desses pontos será utilizado como ponto de partida e alguns outros serão usados como os destinos finais.

3.4 MODIFICAÇÕES NO ALGORITMO

O *Physarum solver* tem como objetivo principal encontrar a menor rota entre dois pontos distintos, porém foi utilizado neste trabalho de graduação um ponto de partida e múltiplos pontos, sendo necessárias alterações no algoritmo bioinspirado orginal. Os pseudo-códigos de algumas destas alterações foram disponibilizados no Apêndice A.

Os valores da condutividade presente na equação 2 foram definidos a partir da função *default_rng()* e *uniform()* do submódulo *random* do *NumPy 1.21.4* gerandos números aleatórios a partir de uma distribuição de probabilidade uniforme. O Quadro 2 mostra essa modificações citadas.

Outra mudança foi a determinação dos valores para α e *r* presentes na equação 7, sendo ambos unitários.

Alterou-se o valor do fluxo I_0 de forma que único ponto de partida descrito pelo conjunto $O = \{v_1\}$ tenha um valor de I_0 e os destinos representados pelo outro conjunto $S = \{v_1, v_2, \ldots, v_n\}$ possuam um fluxo descrito pela modificação das equações 3 e 5 resultando nas equações 9 e 10. As modificações realizadas no *Physarum solver* são mostradas no Quadro 3.

$$\sum_i Q_{ij} = 0, \quad j \notin S,\ j \notin O \tag{9}$$

$$\sum_i Q_{is_k} = 0, \forall s_k \in S \tag{10}$$

Para que as rotas sejam calculadas e traçadas, é necessário que os valores dos fluxos descritos pela equação 2 sejam todos positivos, assim esta mesma equação foi alterada como mostra a equação 11. O Quadro 4 mostra as modificações feitas no *Physarum solver*.

$$Q_{ij} = \frac{D_{ij}}{L_{ij}} |p_i - p_j| \tag{11}$$

Uma vez que os valores do fluxo foram definidos, utilizou-se *plot_graph()* do módulo da *NetworkX 2.6.3* para traçar o grafo com mudança na coloração das rotas ótimas, do nó de partida e dos destinos finais. O Quadro 5 mostra as alterações feitas para mostrar o grafo, as rotas, o nó de partida e os nós de chegada de forma colorida.

Além disso, um ponto importante é a redução da emissão de gases poluentes a partir da escolha da rota ótima, já que, se comparado com uma outra rota qualquer, levaria um maior tempo até chegar no destino final, o que está associado a uma maior utilização de combustível e, consequentemente, uma maior emissão de gases poluentes.

A partir da determinação da condutividade utilizando os métodos presentes na biblioteca *Numpy* e as modificações realizadas para o desenvolvimento das equações 9, 10 e 11, foi possível aplicar todos esses conceitos no algoritmo bioinspirado para gerar as rotas partindo de um único ponto para diferentes outros pontos.

3.5 CRITÉRIOS DE COMPARAÇÃO ENTRE *PHYSARUM SOLVER* E O ALGORITMO DE DIJKSTRA

A comparação entre o *Physarum solver* e o algoritmo de Dijkstra torna-se necessária, já que é de suma importância tanto a comparação do tempo de execução quanto da rota obtida de um ponto de partida a outro de saída para validar a aplicação do algoritmo bioinspirado.

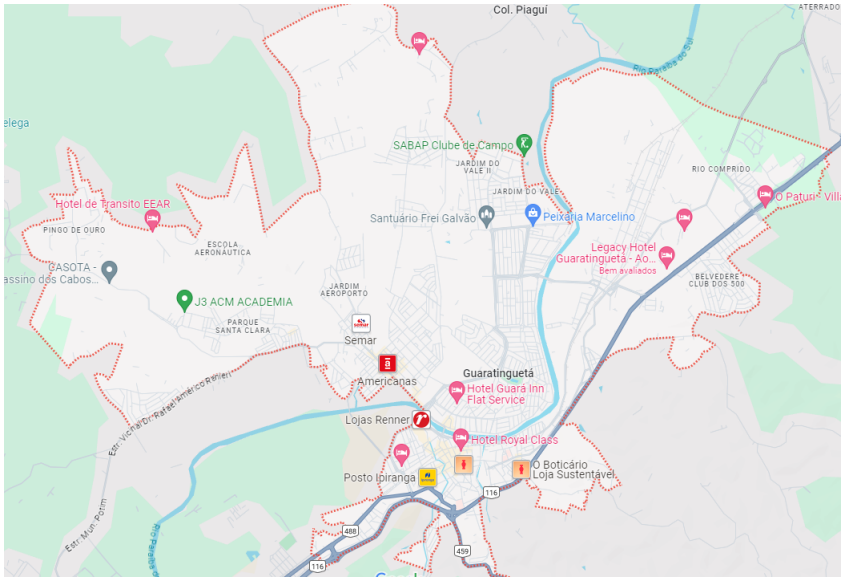
3.6 LOCALIZAÇÕES GEORREFERENCIADAS

Este trabalho de graduação tem como objetivo aplicar o algoritmo a ser desenvolvido utilizando a cidade de Guaratinguetá. Essa cidade possui uma população de aproximadamente 118 mil habitantes e possui 752,636 km² localizada no Vale do Paraíba, região sudeste do Brasil entre as cidades de São Paulo e Rio de Janeiro (22° 48’ 58” S, 45° 11’ 37” O).

Além disso, a cidade conta com desenvolvimento industrial, abrigando o maior complexo da indústria química da América do Sul. A Figura 6 representa a cidade de Guaratinguetá[1] e a Figura 7 é a representação gráfica da cidade em forma de grafo retirada diretamente utilizando *OpenStreetMaps*.

[1] Cidade de Guaratinguetá acessada a partir de ***Google Maps***. 2023. Disponível em: https://www.google.com/maps/place/GuaratinguetÃą,+SP/@-22.7921564,-45.244938,13z

Figura 6 – Cidade de Guaratinguetá.



Fonte: Extraído de *Google Maps* (2024).

Figura 7 – Grafo relacionado à cidade de Guaratinguetá.

Fonte: Autoria própria, via biblioteca *OpenStreetMaps*.

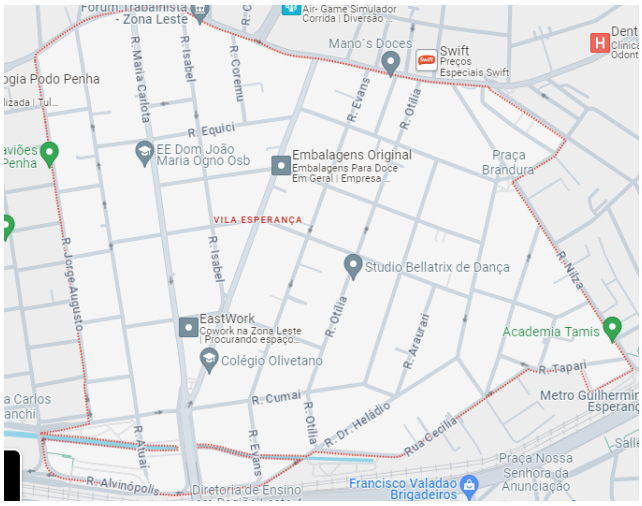
Para uma aplicação inicial do algoritmo, utilizou-se parte de um bairro na cidade de São Paulo chamado Vila Esperança[2], já que sua dimensão geográfica de, aproximadamente, 1,308 km² é menor do que a cidade de Guaratinguetá para serem realizadas avaliações e validações iniciais do algoritmo implementado e também há uma familiaridade com a região. A Figura 8

[2] Bairro Vila Esperança acessada a partir de ***Google Maps***. 2023. Disponível em: https://www.google.com/maps/place/Vila+EsperanÃğa,+SÃčo+Paulo+-+SP/@-23.5266277,-46.5295448,16z

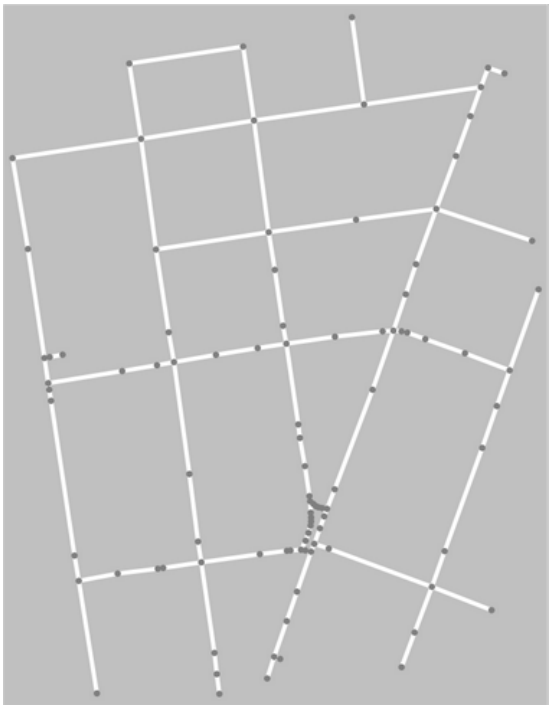
ilustra tanto o bairro em questão mostrado pela Figura 8a extraído do *Google Maps*, como o bairro em forma de grafo como mostrado na Figura 8b extraido utilizando *OpenStreetMaps*.

Figura 8 – Comparação entre o bairro Vila Esperança e o seu grafo georreferenciado.

(a) Bairro Vila Esperança.



(b) Grafo do bairro Vila Esperança.



Fonte: (a) Extraído de *Google Maps* (2024); (b) Autoria própria, via biblioteca *OpenStreetMaps*.

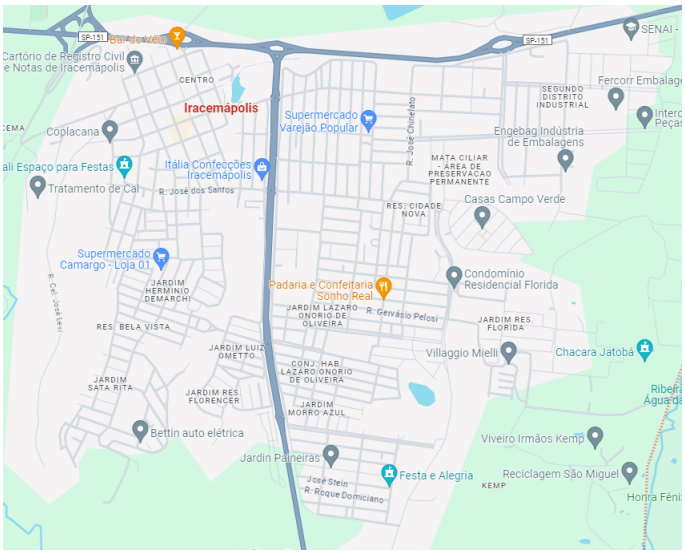
Na próxima execução optou-se por uma cidade com menor área do que Guaratinguetá e maior do que o bairro utilizado anteriormente, sendo a cidade de Iracemápolis[3] escolhida. Essa região possui 115,118 km² de área geográfica. Uma vez que os resultados foram obtidos nesse município, aplicou-se o algoritmo em Guaratinguetá. A Figura 9 demonstra tanto a cidade de

[3] Cidade de Iracemápolis acessada a partir de ***Google Maps***. 2023. Disponível em: https://www.google.com/maps/place/Iracemapoli+-+SP

Iracemápolis mostrado pela Figura 9a retirada do *Google Maps*, como o grafo representando ela na Figura 9b gerada a partir de *OpenStreetMaps*.

Figura 9 – Comparação entre a cidade de Iracemápolis e o seu grafo georreferenciado.

(a) Cidade de Iracemápolis.



(b) Grafo da cidade de Iracemápolis.

Fonte: (a) Extraído de *Google Maps* (2024); (b) Autoria própria, via biblioteca *OpenStreetMaps*.

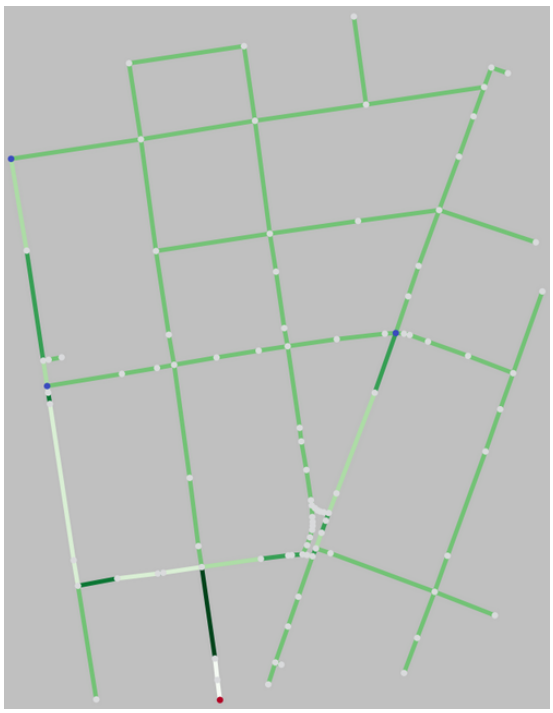
4 RESULTADOS E DISCUSSÃO

O quarto capítulo será utilizado para descrever o histórico de tentativas, os resultados obtidos e discussões em relação a eles.

4.1 HISTÓRICO DE TENTATIVAS

Adaptou-se o *Physarum solver* para ser aplicado em grafos com diferentes características para analisar a diferença do tempo de execução. Uma vez definido o tipo de grafo, foram realizados testes para traçar a melhor rota, entretanto os resultados não estavam consistentes com a realidade das rotas aplicadas ao bairro utilizado para os primeiros teste. A Figura 10 mostra um dos resultados obtidos, sendo as rotas não utilizadas na coloração branca e as ótimas em tons da cor verde. Além disso, o ponto de partida é representado pela cor vermelha, os de destino pela cor azul e os outros locais estão na coloração cinza claro.

Figura 10 – Resultado da aplicação inicial a um grafo não direcionado.



Fonte: Autoria própria, via biblioteca *OpenStreetMaps*.

O problema da inconsistências das rotas foi solucionado utilizando apenas valores positivos para o fluxo como mostra a equação 11, pois o seu valor é utilizado para traçar as rotas ótimas a serem utilizadas.

4.2 RESULTADOS DA COMPARAÇÃO DO *PHYSARUM SOLVER* EM RELAÇÃO AO ALGORITMO DE DIJKSTRA

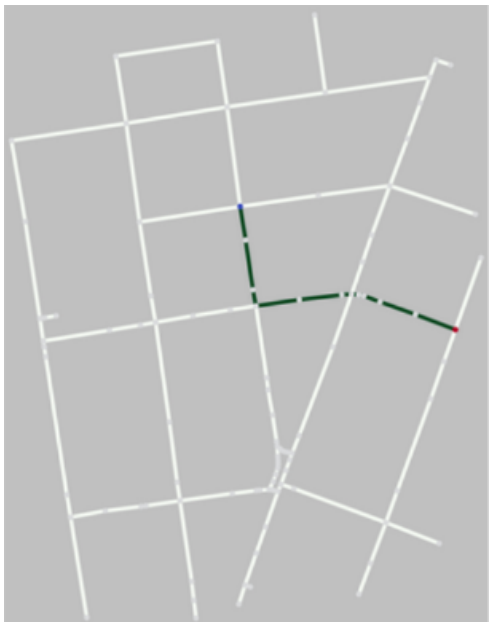
4.2.1 Primeira aplicação: bairro Vila Esperança

A primeira comparação foi utilizando o bairro Vila Esperança e para isso foram aplicados cinco diferentes pares de pontos de partida e chegada para comparar o tempo de execução e as rotas obtidas tanto pelo *Physarum solver* quanto pelo algoritmo de Dijkstra. As rotas ótimas são apresentadas em tonalidades de verde e as não utilizadas em branco. Além disso, os locais são representados pela cor cinza claro, os de destino pela cor azul e o ponto de partida é usado a cor vermelha.

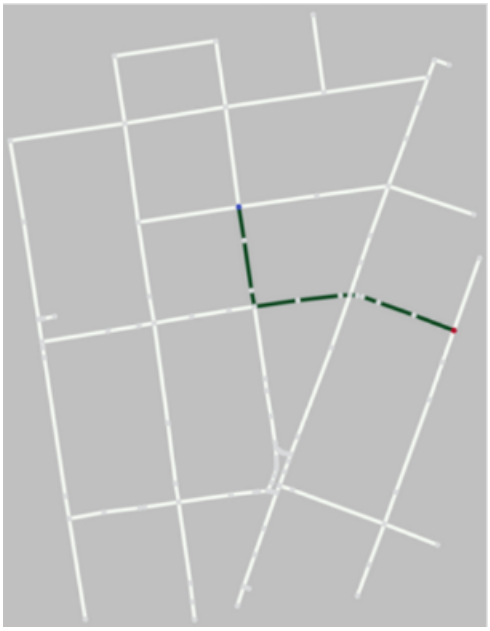
A Figura 11 mostra o grafo utilizado para representar o bairro com 96 nós e 106 arestas e as rotas obtidas pelos algoritmos, sendo a Figura 11a gerada pelo algortimo de Dijkstra e a Figura 11b pelo algoritmo bioinspirado.

Figura 11 – Aplicação (A): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra.



(b) *Physarum solver*.



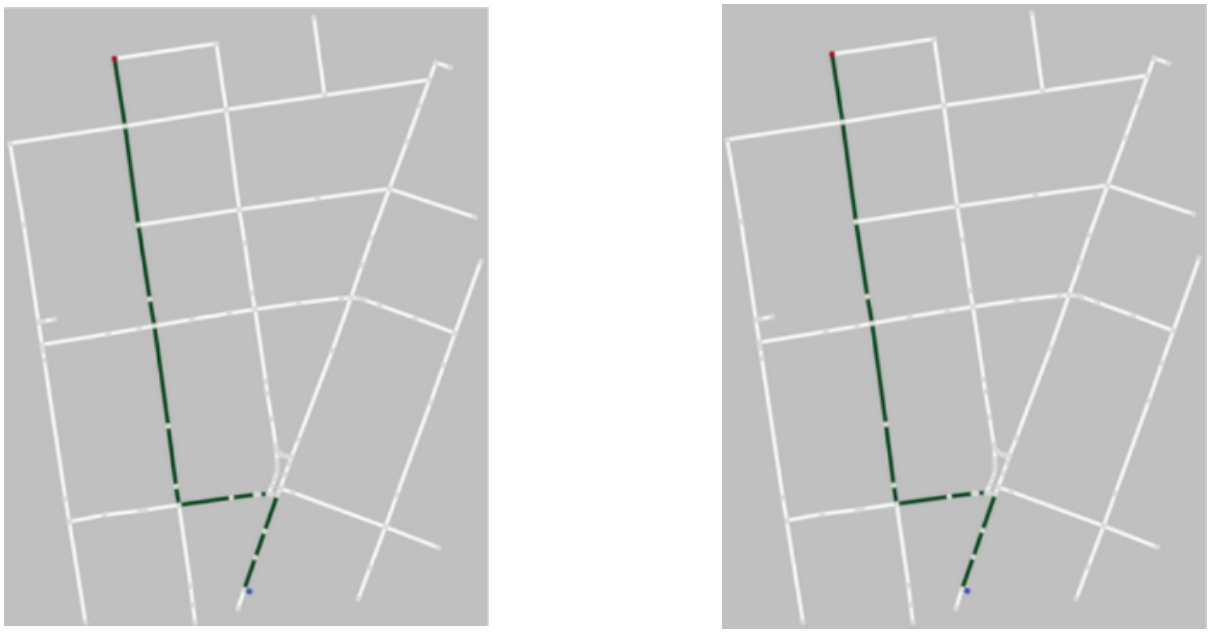
Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

As Figuras 11a e 11b apresentam a mesma rota para os dois pontos com o tempo de execução total de 0,78 s e 74,84 s respectivamente.

A Figura 12 representa as rotas calculadas pelos dois algoritmos para o mesmo par de pontos distintos, sendo a Figura 12a gerada pelo Algortimo de Dijkstra e a Figura 12b pelo *Physarum solver*.

Figura 12 – Aplicação (B): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra. (b) *Physarum solver*.



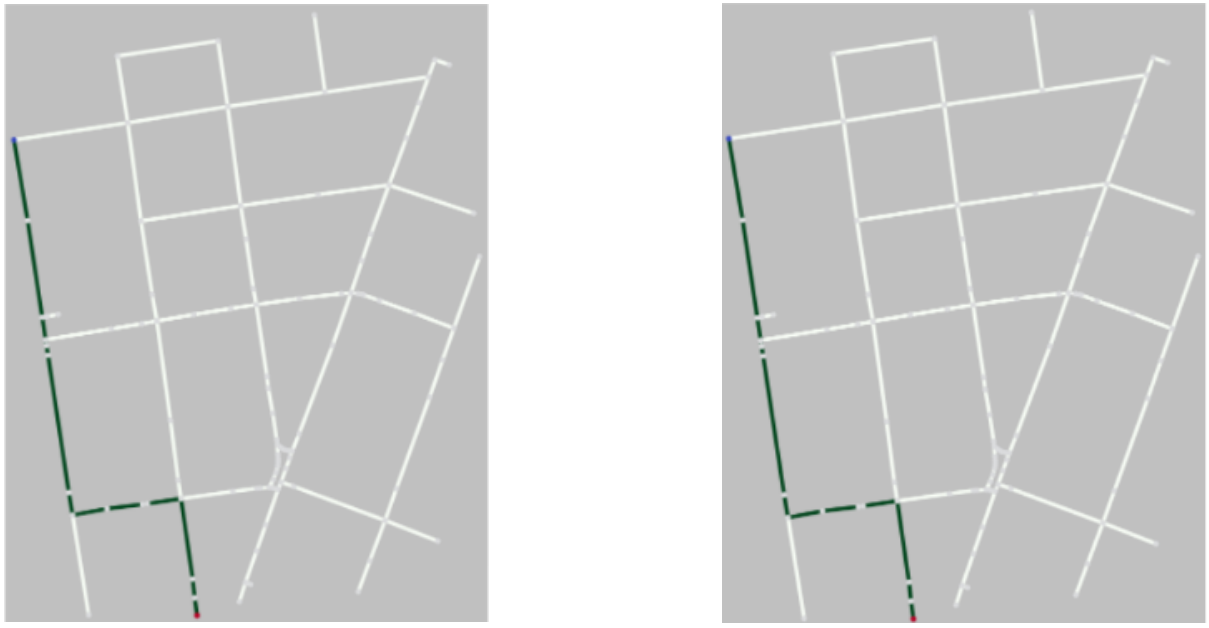
Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

As Figuras 12a e 12b apresentam a mesma rota para o mesmo par de pontos com o tempo de execução de 0,87 s e 77,04 s respectivamente.

A Figura 13 mostra as duas rotas calculadas pelos algoritmos, sendo a Figura 13a gerada pelo algortimo de Dijkstra e a Figura 13b pelo *Physarum solver*.

Figura 13 – Aplicação (C): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra. (b) *Physarum solver*.



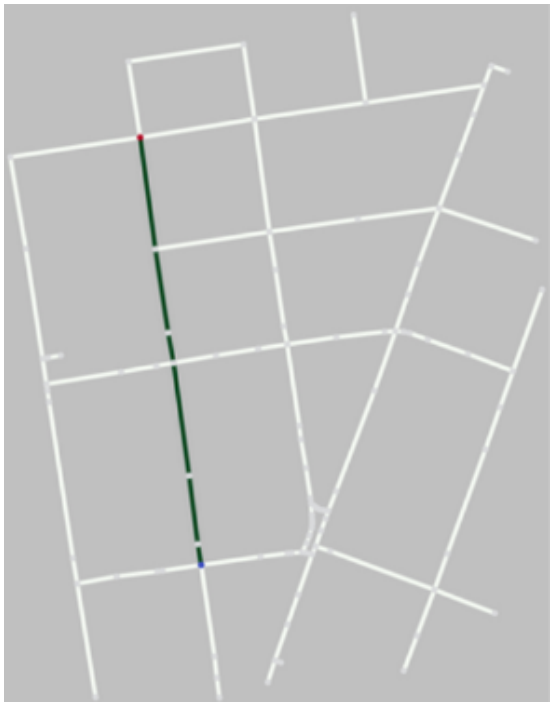
Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

As Figuras 13a e 13b apresentam a mesma rota e obteve-se um tempo de execução para cada um dos algoritmos de 1,27 s e 76,04 s respectivamente.

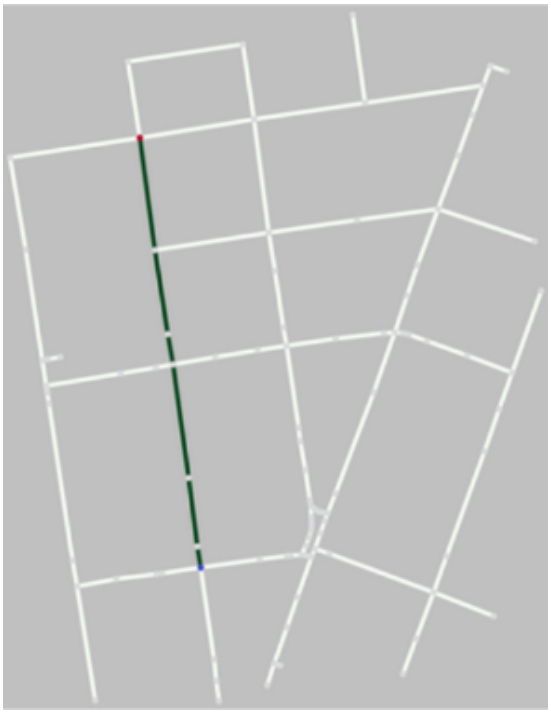
A Figura 14 representa as duas rotas geradas pelos dois algoritmos, sendo a Figura 14a gerada pelo algortimo de Dijkstra e a Figura 14b pelo *Physarum solver*.

Figura 14 – Aplicação (D): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra.



(b) *Physarum solver*.



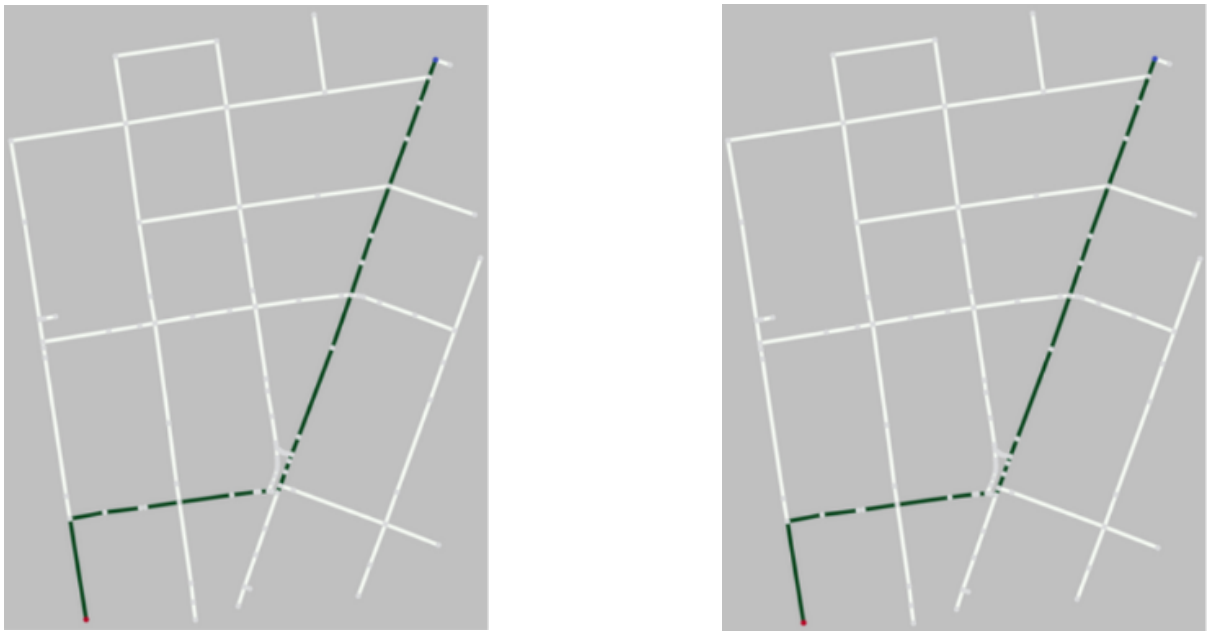
Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

Ambas as Figuras 14a e 14b apresentam a mesma rota e tiveram um tempo de execução total de 0,97 s e 79,78 s respectivamente.

A Figura 15 representa as duas rotas obtidas pelos dois algoritmos, sendo a Figura 15a gerada pelo algortimo de Dijkstra e a Figura 15b pelo *Physarum solver*.

Figura 15 – Aplicação (E): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra. (b) *Physarum solver*.



Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

As Figuras 15a e 15b apresentam a mesma rota e tiveram um tempo de execução de 1,30 s e 87,26 s respectivamente.

Os tempos obtidos para cada uma das aplicações e as distâncias aproximadas entre os pontos são mostrados na Tabela 2.

Tabela 2 – Tempo de execução do algoritmo de Dijkstra e do *Physarum solver* e a distância entre os pontos de partida e de saída.

Aplicações	**Algoritmo de Dijkstra**	***Physarum Solver***	**Distância**
Aplicação (A)	0,78 s	74,84 s	274,84 m
Aplicação (B)	0,87 s	77,04 s	586,87 m
Aplicação (C)	1,27 s	76,04 s	565,96 m
Aplicação (D)	0,97 s	79,78 s	422,01 m
Aplicação (E)	1,30 s	87,26 s	736,87 m

Fonte: Autoria própria.

É possível notar que, na primeira aplicação, as rotas obtidas pelo *Physarum solver* são idênticas às do algoritmo de Dijkstra de forma independente das distâncias entre os dois pontos. Além disso, o tempo de execução do algoritmo bioinspirado é significantemente maior se comparado ao de Dijkstra, já que o segundo é a referência para a solução de problemas de roteamento para um ponto de entrada e um de saída. Outro ponto importante para a grande diferença de tempo de execução é que o *Physarum solver* depende de um processo iterativo de adaptação e convergência, enquanto o algoritmo de Dijkstra utiliza um cálculo direto e otimizado.

É importante ressaltar que o algoritmo de Dijkstra não é utilizado para um local de partida para diversos de chegada e, para isso, optou-se pela aplicação do *Physarum solver*.

4.2.2 Segunda aplicação: cidade de Iracemápolis

A segunda comparação também teve o objetivo de validar as rotas obtidas pelo *Physarum solver* em relação ao algoritmo de Dijkstra, porém foi realizada aplicando ambos os algoritmos na cidade de Iracemápolis que é representada por um grafo com 1135 nós e 1466 arestas.

A Figura 16 mostra as rotas obtidas pelos algoritmos, sendo a Figura 16a gerada pelo algortimo de Dijkstra e a Figura 16b pelo algoritmo bioinspirado.

Figura 16 – Aplicação (F): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra. (b) *Physarum solver*.

Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

As Figuras 16a e 16b apresentam a mesma rota para o mesmo par de pontos com o tempo de execução de 2,4 s e 02h:32min:38s respectivamente.

A Figura 17 demonstra as rotas obtidas pelos algoritmos para o mesmo par de pontos, sendo a Figura 17a gerada pelo algortimo de Dijkstra e a Figura 17b pelo algoritmo bioinspirado.

Figura 17 – Aplicação (G): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra. (b) *Physarum solver*.

Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

Nota-se que, na aplicação representada pela Figura 17, as rotas obtidas pelo *Physarum solver* são semelhantes às do algoritmo de Dijkstra, porém não idênticas, já que, na Figura 17b, o algoritmo bioinspirado sugere também outras rotas entre dois pontos. Isso ocorre por conta dos comprimentos muito próximos entre as duas rotas, neste caso, a rota sugerida pelo algoritmo de Dijkstra possui um comprimento de, aproximadamente, 167,02 m, mostrado na Figura 17a. Já a rota alternativa sugerida pelo *Physarum solver* possui um comprimento total aproximado de 168,21 m. Além disso, as Figuras 17a e 17b obtiveram o tempo de execução de 2,4 s e 02h:41min:35s respectivamente.

A Figura 18 mostra as rotas obtidas pelos algoritmos para o mesmo par de nós, sendo a Figura 18a gerada pelo algortimo de Dijkstra e a Figura 18b pelo *Physarum solver*.

Figura 18 – Aplicação (H): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra. (b) *Physarum solver*.

Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

As Figuras 18a e 18b apresentam a mesma rota e com o tempo total de 2,6 s e 02h:45min:18s respectivamente.

A Figura 19 mostra as rotas obtidas pelos algoritmos, sendo a Figura 19a gerada pelo algortimo de Dijkstra e a Figura 19b pelo algoritmo bioinspirado.

Figura 19 – Aplicação (I): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra. (b) *Physarum solver*.

Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

As Figuras 19a e 19b demonstram a mesma rota para o mesmo par de pontos e obteve-se o tempo total de execução de 2,2 s e 02h:42min:58s respectivamente.

A Figura 20 mostra as rotas geradas pelos dois algoritmos, sendo a Figura 20a gerada pelo algortimo de Dijkstra e a Figura 20b pelo *Physarum solver*.

Figura 20 – Aplicação (J): Comparação entre as rotas dos dois algoritmos.

(a) Algoritmo de Dijkstra. (b) *Physarum solver*.

Fonte: Ambas as Figuras são de autoria própria, via biblioteca *OpenStreetMaps*.

As Figuras 20a e 20b apresentam a mesma rota e o tempo de execução foi 1,30 s e 02h:45min:58s respectivamente

Assim como na primeira comparação, os tempos obtidos para cada uma das aplicações da segunda comparação e as distâncias aproximadas entre os pontos são mostrados pela Tabela 3.

Tabela 3 – Tempo de execução do algoritmo de Dijkstra e do *Physarum solver* e a distância entre os pontos de partida e de saída.

Aplicações	**Algoritmo de Dijkstra**	***Physarum Solver***	**Distância**
Aplicação (F)	2,4 s	02h:32min:38s	949,64 m
Aplicação (G)	2,4 s	02h:41min:35s	805,13 m
Aplicação (H)	2,6 s	02h:45min:18s	644,80 m
Aplicação (I)	2,2 s	02h:32min:15s	1,31 km
Aplicação (J)	1,30 s	02h:45min:58s	987,10 m

Fonte: Autoria própria.

Para esta segunda comparação, é importante ressaltar que as diferentes opções de caminhos a serem seguidos calculados pelo *Physarum solver* incluem os caminhos sugeridos pelo algoritmo de Dijkstra, sendo possível a validação do *Physarum solver*. Além disso, as rotas obtidas por ambos os algoritmos não dependem da distância física entre o ponto de partida e o de saída.

Outro ponto relevante é o tempo de execução do algoritmo de Dijkstra, que é significativamente menor em comparação ao algoritmo bioinspirado. O modo de funcionamento entre os

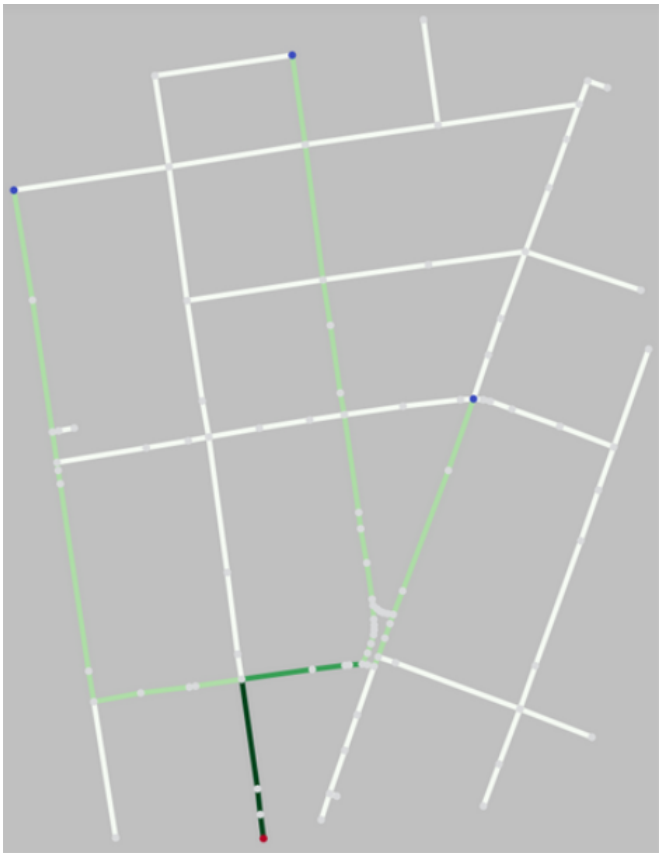
dois algoritmos é distinto, já que, enquanto o algoritmo de Dijkstra encontra o menor caminho de forma direta para uma distância fixa, o *Physarum solver* realiza um processo iterativo de adaptação das condutâncias para determinar o caminho ideal. Esse processo iterativo torna o *Physarum solver* mais lento, pois simula o comportamento adaptativo do organismo, exigindo múltiplas iterações até a convergência da solução.

Com isso, há a validação das rotas obtidas pelo algoritmo bioinspirado, mesmo que a diferença de tempo de execução seja grande se comparado com o algoritmo de Dijkstra. O *Physarum solver* será utilizado neste trabalho de graduação pois é possível utilizá-lo para um local de entrada para diversos de saída, ao contrario do algoritmo de Dijkstra que não é possível de ser aplicado nessa situação.

4.3 APLICAÇÃO DO *PHYSARUM SOLVER* PARA UMA ENTRADA E MÚLTIPLAS SAÍDAS

Uma vez que as rotas obtidas pelo *Physarum solver* foram validadas, inicialmente, aplicou-se o algortimo bioinspirado no bairro Vila Esperança utilizando um ponto de partida para outros três pontos de chegada em três situações diferentes. A Figura 21 mostra as rotas ótimas obtidas para a primeira aplicação e o tempo de execução total foi de 71,86 segundos.

Figura 21 – Primeira aplicação do *Physarum solver* utilizando o bairro da Vila Esperança em São Paulo.

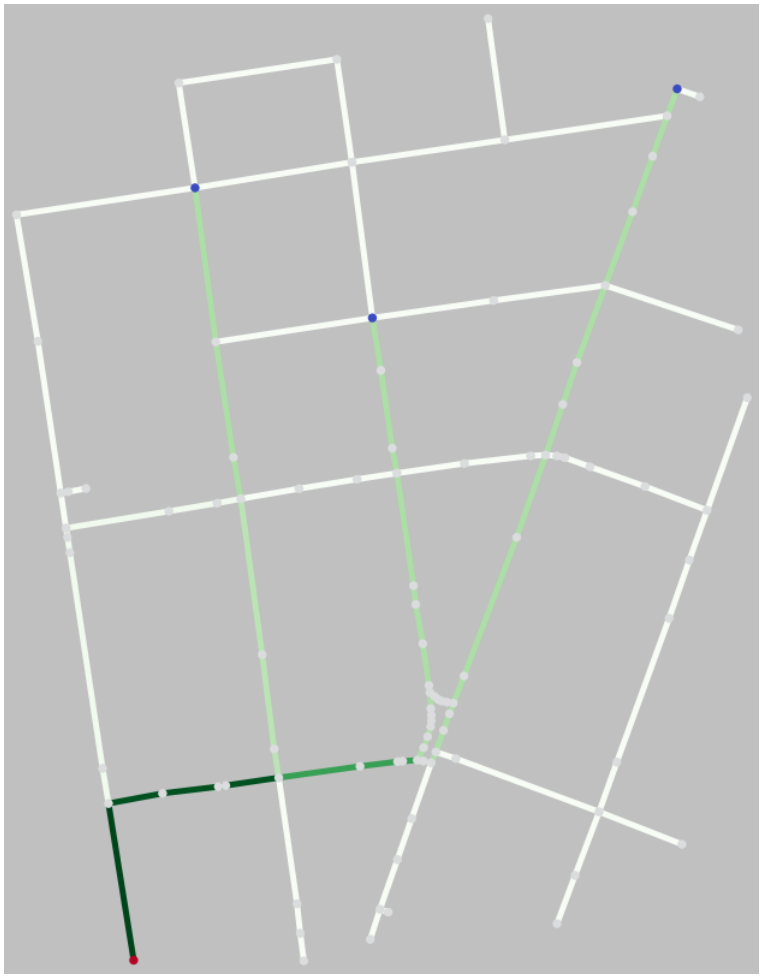


Fonte: Autoria própria, via biblioteca *OpenStreetMaps*.

Os pontos utilizados para a primeira aplicação também foram utilizados para comparar as soluções do *Physarum solver* em relação ao algoritmo de Dijkstra. Desse forma, é possível verificar a qualidade das rotas obtidas pelo algoritmo bioinspirado. Para esta primeira utilização, as rotas obtidas foram semelhantes às geradas pelo algoritmo de Dijkstra o que valida a funcionalidade do algoritmo desenvolvido neste trabalho de graduação.

A segunda aplicação foi utilizado outro conjuntos de pontos tanto de saída quanto de entrada. A Figura 22 mostra os locais de chegada e de partida e as rotas obtidas pelo *Physarum solver* e o tempo total de execução foi de 99,81 segundos.

Figura 22 – Segunda aplicação do *Physarum solver* utilizando o bairro da Vila Esperança em São Paulo.



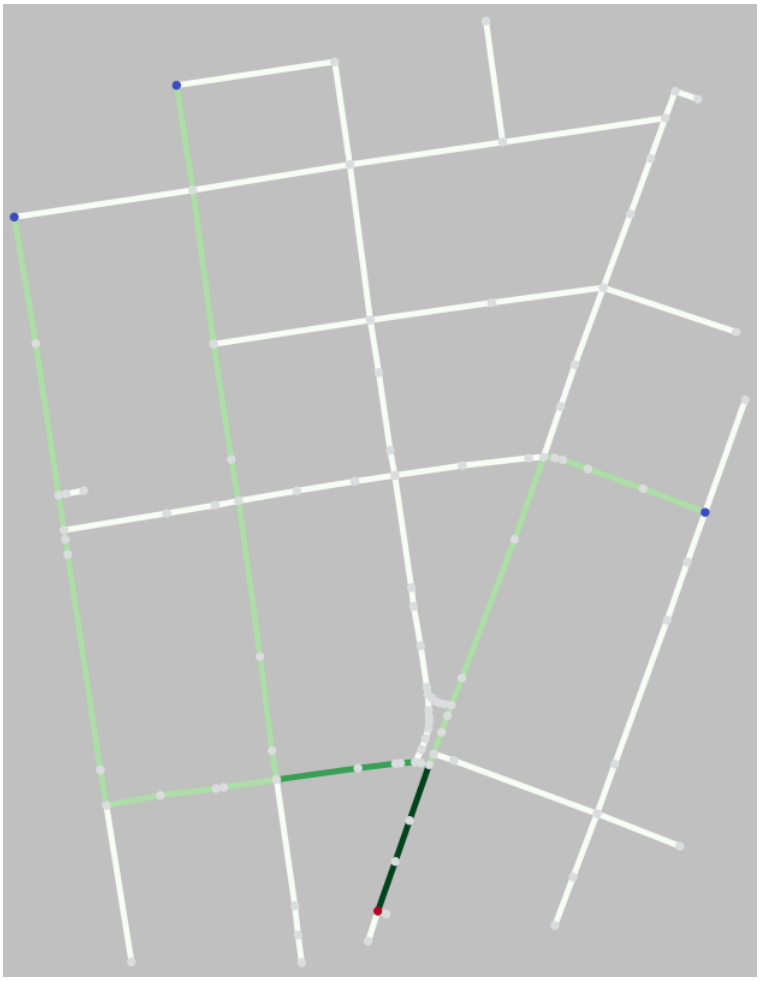
Fonte: Autoria própria, via biblioteca *OpenStreetMaps*.

Assim como na primeira aplicação, para a segunda, também foram utilizados alguns mesmos pontos usados na comparação dos dois algoritmos. As rotas calculadas pelo algoritmo bioinspirado foram bem próximas das geradas pelo algoritmo de Dijkstra, o que mostra o funcionamento adequado do *Physarum solver*.

A última aplicação utilizou-se o bairro Vila Esperança como localização geográfica. Para isso, utilizou-se um ponto de partida e outros três de chegada usados também na comparação

entre o *Physarum solver* e o algoritmo de Dijkstra. A Figura 23 mostra tanto os locais utilizados como as rotas obtidas pelo algoritmo desenvolvido neste trabalho de graduação. Além disso, o tempo de execução do algoritmo para o cálculo das rotas foi de 86,97 segundos.

Figura 23 – Terceira aplicação do *Physarum solver* utilizando o bairro da Vila Esperança em São Paulo.



Fonte: Autoria própria, via biblioteca *OpenStreetMaps*.

As rotas obtidas para esta última aplicação foram semelhantes as geradas pelo algoritmo de Dijkstra, o que valida as rotas obtidas pelo algoritmo bioinspirado.

Para uma próxima aplicação, utilizou-se a cidade de Iracemápolis para aplicar o algoritmo bioinspirado em duas distintas execuções. Para a primeira, foram colocados três locais de destino para um único de partida que também foram utilizados na comparação entre o algoritmo de Dijkstra *Physarum solver* para ter parâmetros de comparação e validação das rotas obtidas. A Figura 24 mostra as rotas ótimas calculadas pelos *Physarum solver* entre os pontos utilizados para esta primeira aplicação. Neste caso, o tempo total que o *Physarum solver* levou para obter as rotas ótimas foi de 02h:35min:22s.

Figura 24 – Primeira aplicação do *Physarum solver* na cidade de Iracemápolis.

Fonte: Autoria própria, via biblioteca *OpenStreetMaps*.

Um ponto importante são as sugestões de rotas calculadas pelo *Physarum solver* como é mostrado pela Figura 24. Isso ocorre por conta dos comprimentos muito próximos às possíveis rotas, já que, no caso do destino que está localizado na parte nordeste, as rotas sugeridas possuem uma diferença de comprimento aproximado de 11 m. Para o ponto localizado à sudeste da Figura 24, há uma semelhança, já que as duas rotas indicadas possuem uma diferença de 7,75 m aproximadamente. Mesmo com as sugestões de rotas obtidas pelo *Physarum solver*, elas incluem as rotas calculadas pelo algoritmo de Dijkstra, o que demonstra o funcionamento adequado do algoritmo desenvolvido.

Para a segunda execução para a cidade de Iracemápolis utilizou-se um conjunto distinto de pontos e que também foram usados na comparação realizada entre os dois algoritmos. A Figura 25 mostra os locais usados e as rotas obtidas pelo algoritmo bioinspirado. Para esta aplicação, o tempo de execução foi de 2h:38min:27s.

Figura 25 – Segunda aplicação do *Physarum solver* na cidade de Iracemápolis..

Fonte: Autoria própria, via biblioteca *OpenStreetMaps*.

Assim como na primeira aplicação utilizando a cidade de Iracemápolis, a segunda também possui mais de um caminho como rota calculada por conta do comprimento muito próximo entre elas em um dos pontos escolhidos como destino em ambas as execuções. Além disso, como os pontos utilizados também foram aplicados na comparação entre o algoritmo bioinspirado e o de Dijkstra, as rotas obtidas foram semelhantes às obtidas no segundo, o que valida o funcionamento adequado do *Physarum solver*.

A última utilização do *Physarum solver* foi na cidade de Guaratinguetá e, assim como nos outros dois exemplos, utilizou-se um ponto inicial para três diferentes pontos finais. A Figura 26 é o grafo representando Guaratinguetá com 3736 nós e 5352 arestas, os pontos utilizados para representar o local de partida e os destinos e as rotas ótimas a serem utilizadas.

Figura 26 – *Physarum solver* aplicado à cidade de Guaratinguetá.

Fonte: Autoria própria, via biblioteca *OpenStreetMaps*.

A aplicação do *Physarum solver* na cidade de Guaratinguetá levou, aproximadamente, 81 horas que é um tempo excessivamente maior se comparado com qualquer outra aplicação desenvolvida neste trabalho de graduação. Outro ponto adicional sobre as rotas obtidas é que todas elas foram geradas em um caminho único, ou seja, não houve caminhos alternativos até o destino final. Isso ocorre por não houver caminhos que possuam o comprimento tão semelhante ao ponto do *Physarum solver* retornar um caminho alternativo do local de partida até o ponto de chegada.

5 CONCLUSÃO E TRABALHOS FUTUROS

Os resultados obtidos mostram que o *Physarum solver* é capaz de calcular as rotas ótimas entre um ponto de partida para diferentes pontos de destinos utilizando um grafo não direcionado georreferenciado.

Para a primeira aplicação no bairro de Vila Esperança, o algortimo bioinspirado obteve os melhores caminhos para os três diferentes pontos de destino e nas três situações distintas, não havendo uma rota diferente como sugestão e tendo um tempo total de 71,86s, 99,81s e 86,97s para a primeira, segunda e terceira aplicação respectivamente.

A segunda aplicação na cidade Iracemápolis contou com duas diferentes execuções e obteve as rotas ótimas para os três locais de chegada, e em ambas aplicações, para um mesmo local de destino, houve a sugestão de rotas alternativas por terem o comprimento próximos. A primeira e a segunda execução levaram um período de 02h:35min:22s e 2h:38min:27s nessa mesma ordem.

Para a aplicação final na cidade de Guaratinguetá, as rotas ótimas para os três pontos diferentes partindo de um mesmo lugar foram obtidas uma única rota sem outros caminhos alternativos, assim como nas aplicações para o bairro e a segunda execução utilizando a cidade de Iracemápolis. O tempo total para o *Physarum solver* aplicado a Guaratinguetá foi de 81h.

Mesmo com áreas geográficas com diferentes dimensões e, consequentemente, diferente número de nós, o algoritmo bioinspirado em *Physarum polycephalum* foi capaz de definir as melhores rotas a serem seguidas e elas foram validadas a partir da comparação com as rotas obtidas de um ponto a outro em relação ao algoritmo de Dijkstra.

Além disso, outro ponto a ser considerado é o tempo de execução do *Physarum solver* comparado ao do algoritmo de Dijkstra. Enquanto o primeiro algoritmo citado tem como objetivo calcular as melhores rotas de um ponto para outros pontos de destino e o segundo tem como função encontrar a melhor rota de um local para outro, o tempo de execução do *Physarum solver* é muito superior ao do algoritmo de Dijkstra.

Para a realização de trabalhos futuros, é possível a realizar tanto a otimização do *Physarum solver* como a simplificação do grafo utilizado de forma que o grafo possua um número menor de nós e mantenha o formato da localização geográfica na qual ele foi baseado. Com essas melhorias poderá facilitar a utilização um grafo direcionado para representar a direção das vias georreferenciadas.

Um outro possível trabalho futuro seria fazer que o *Physarum solver* ao gerar uma rota ótima, o ponto de partida do próximo trajeto fosse alterado para o destino final do caminho gerado anteriormente. Com isso, o resultado seria uma rota ótima contínua assemelhando-se aos aplicativos de roteamento utilizados como o *Google Maps*.

REFERÊNCIAS

ADAL, T.; ORTEGA, A. Applications of graph theory. **Proceedings of the IEEE**, Istanbul, v. 106, n. 5, p. 784–786, 2018.

ADAMU, P. I. *et al.* Shortest path planning algorithm: a particle swarm optimization (PSO) approach. *In*: PROCEEDINGS OF THE WORLD CONGRESS ON ENGINEERING 2018, 1., jul 4-6, 2018, London, U.K. **Proceedings [...]**. London: Springer, 2018.

BANNISTER, M. J.; EPPSTEIN, D. Randomized speedup of the Bellman–Ford algorithm. *In*: ANALYTIC ALGORITHMICS AND COMBINATORICS (ANALCO12), 1., jan 16, 2012, Kyoto. **Proceedings [...]**. Kyoto: Society for Industrial and Applied Mathematics, 2012.

BENDER, E. A.; WILLIAMSON, S. G. **Lists, decisions and graphs**: with an introduction to probability. San Diego: University of California, 2010.

CHARTRAND, G. **Introductory graph theory**. New York: Dover Publications, 1985.

CHUA, D. *et al.* A Physarum-inspired algorithm for logistics optimization: from the perspective of effective distance. **Swarm and evolutionary computation**, Singapore, v. 64, p. 100–890, 2021.

CONFEDERAÇÃO DA AGRICULTURA E PECUÁRIA DO BRASIL. **PIB do agronegócio alcança participação de 26,6% no PIB brasileiro em 2020**: relatório publicado pela confederação da agricultura e pecuária do Brasil. Brasília: Confederação da Agricultura e Pecuária do Brasil, 2020.

CORMEN, T. H. *et al.* The floyd-warshall algorithm. *In*: CORMEN, T. H. *et al.* (Ed.). **Introduction to Algorithms**. 2. ed. Cambridge: MIT Press, 2001. cap. 25.2, p. 535–541.

DEO, N. **Graph theory with applications to engineering and computer science**. New York: Dover Publications, 2016.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. **Numerische mathematik**, Amsterdam, p. 269–271, 1959.

DUSSUTOUR, A. *et al.* Amoeboid organism solves complex nutritional challenges. **Proceedings of the national academy of sciences of the United States of America**, Toulusse, v. 107, n. 10, p. 4607–4611, 2010. ISSN 0027-8424.

FELTRIN, A. Retrospectiva 2020 do setor de transporte de cargas no Brasil. **Estradão**, São Paulo, 2020. Disponível em: https://estradao.estadao.com.br/caminhoes/retrospectiva-2020-do-setor-de-transporte-de-cargas-no-brasil/. Acesso em: 21 mar. 2024.

GALLO, G.; PALLOTTINO, S. Shortest path algorithms. **Annals of operations research**, Pisa, v. 13, p. 1–79, 1988.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE transactions on systems science and cybernetics**, Stanford, v. SSC4, n. 2, p. 100–107, 1968.

HENIG, M. I. The shortest path problem with two objective functions. **European journal of operational research**, Tel Aviv, v. 25, n. 2, p. 281–291, 1986.

HOWARD, F. L. The life history of *Physarum polycephalum*. **American journal of botany**, Washington D.C., v. 18, n. 2, p. 116–133, 1931.

LUO, Q. *et al.* Research on path planning of mobile robot based on improved ant colony algorithm. **Neural computing and applications**, Beijing, v. 32, p. 1555–1566, 2019.

NOTO, M.; SATO, H. A method for the shortest path search by extended Dijkstra algorithm. *In*: 2000 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 1., oct 8-11, 2000, Nashville. **Proceedings [...]**. Tucson: IEEE, 2000. v. 2, p. 2316–2320.

PASSARINI, G. M. **Algoritmos bioinspirados**. Porto União: IA Expert Academy, 2020.

RISALD, A. E.; SUYOTO. Best routes selection using Dijkstra and Floyd-Warshall algorithm. *In*: INTERNATIONAL CONFERENCE ON INFORMATION & COMMUNICATION TECHNOLOGY AND SYSTEM (ICTS), 11., oct 31, 2017, Surabaya. **Proceedings [...]**. Surabaya: IEEE, 2017. p. 155–158.

SOLIANI, R. D.; ARGOUD, A. R. T. T. A emissão de gases poluentes no transporte rodoviário de cargas brasileiro. **Espacios**, São Carlos, v. 39, n. 48, p. 14, 2018.

TERO, A.; KOBAYASHI, R.; NAKAGAKI, T. Physarum solver: a biologically inspired method of road-network navigation. **Physica A – Statistical Mechanics and its Applications**, Sapporo, v. 363, n. 1, p. 115–119, 2006.

WATANABE, S. *et al.* Traffic optimization in railroad networks using an algorithm mimicking an amoeba-like organism, Physarum plasmodium. **Biosystems**, Tokyo, v. 105, n. 3, p. 225–232, 2011.

ZHANG, X. *et al.* Physarum solver: a bio-inspired method for sustainable supply chain network design problem. **Annals of operations research**, Changsha, v. 254, p. 533–552, 2017.

APÊNDICE A – MODIFICAÇÕES REALIZADAS NO *PHYSARUM SOLVER* ORIGINAL

Quadro 2 – Modificação no *Physarum solver* para determinar a condutividade.

```
importar networkx como nx
importar numpy como np
importar osmnx como ox
função DETERMINARCONDUTIVIDADE
    rng ← np.random.default_rng()
    low ← 0.78
    high ← 0.79
    para todos aresta (i, j) em maze.edges() faça
        maze.edges[i, j]["conduct"] ← rng.uniform(low, high)
    fim para
fim função
```

Fonte: Autoria própria.

Quadro 3 – Modificação no *Physarum solver* para adaptar o fluxo constante do nó de partida.

```
importar networkx como nx
importar numpy como np
importar osmnx como ox
sink ← [7175707874, 1706413629, 1706348217]
I0 ← 9
função ADAPTARFLUXOCONSTANTE
    res[node_list.index(source)] ← (-1) * I0
    para todos n em sink faça
        res[node_list.index(n)] ← I0 / len(sink)
    fim para
fim função
```

Fonte: Autoria própria.

Quadro 4 – Modificação no *Physarum solver* para calcular o fluxo.

```
importar networkx como nx
importar numpy como np
importar osmnx como ox
função CALCULARFLUXO
    para todos aresta (i, j) em maze.edges() faça
        maze.edges[i, j]["flux"] ← np.abs((maze.edges[i, j]["conduct"] / maze.edges[i, j]["length"]) * (pres[node_list.index(i)] - pres[node_list.index(j)]))
    fim para
fim função
```

Fonte: Autoria própria.

Quadro 5 – Modificação no *Physarum solver* para gerar o grafo com as rotas, o ponto de partida e as múltiplas saídas.

```
importar networkx como nx
importar osmnx como ox
função TRAÇARGRAFO
    G ← nx.MultiGraph(maze)
    nc ← ox.plot.get_node_colors_by_attr(G, attr='type', cmap='coolwarm')
    ec ← ox.plot.get_edge_colors_by_attr(G, attr='flux', cmap='Greens')
    fig, ax ← ox.plot.plot_graph(G, ax=None, node_color=nc, edge_color=ec, show=True)
fim função
```

Fonte: Autoria própria.