UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE FÍSICA

PROGRAMA DE PÓS-GRADUAÇÃO EM FÍSICA

MESTRADO ACADÊMICO EM FÍSICA

ANDRÉ DE ALBUQUERQUE LIMA

FRATURAS DE CAMINHOS ÓTIMOS

FORTALEZA

2023

ANDRÉ DE ALBUQUERQUE LIMA

FRATURAS DE CAMINHOS ÓTIMOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Física do Programa de Pós-Graduação em Física do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Física. Área de Concentração: Física da matéria condensada.

Orientador: Prof. Dr. Saulo Davi Soares e Reis.

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L696f Lima, André de Albuquerque.
Fraturas de Caminhos Ótimos / André de Albuquerque Lima. – 2023.
43 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Física, Fortaleza, 2023.
Orientação: Prof. Dr. Saulo Davi Soares e Reis.

1. Congestionamento. 2. Sistemas complexos. 3. Meios desordenados. 4. Caminhos ótimos. I. Título.
CDD 530

ANDRÉ DE ALBUQUERQUE LIMA

FRATURAS DE CAMINHOS ÓTIMOS

Dissertação apresentada ao Curso de Mestrado Acadêmico em Física do Programa de Pós-Graduação em Física do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Física. Área de Concentração: Física da matéria condensada.

Aprovada em: 22/06/2023.

BANCA EXAMINADORA

Prof. Dr. Saulo Davi Soares e Reis (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Humberto de Andrade Carmona
Universidade Federal do Ceará (UFC)

Prof. Dr. Francisco Heber Lacerda de Oliveira
Universidade Federal do Ceará (UFC)

A minha mãe por sempre ter me dado todo o amor e apoio que precisei.

AGRADECIMENTOS

Ao Prof. Dr. Saulo Davi Soares e Reis, por ter me orientado durante o desenvolvimento de todo o trabalho.

Ao Prof. Dr. Humberto de Andrade Carmona e ao Prof. Dr. Francisco Heber Lacerda de Oliveira por terem aceitado participar da banca de defesa da dissertação e colaborado com esclarecimentos sobre o tema.

Aos colegas do laboratório de Sistemas Complexos por todos os momentos de conversa e descontração, especialmente Germano, Lara e Laisa que estiveram comigo desde a graduação, e Rafael, Edson e Marciel, que conheci durante a pós.

A todos os professores do departamento de Física da UFC que colaboraram para minha formação.

À minha mãe Ana, por ter me dado todo o suporte necessário para seguir com meus estudos e pesquisas e por ser minha maior incentivadora.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

RESUMO

O congestionamento no trânsito de cidades desenvolvidas é um problema que acarreta em prejuízos financeiros e de saúde pública. O presente trabalho procura por meio do algoritmo de Fraturas de Caminhos Ótimos ou OPC (*Optimal Path Cracks*) entender quais os pontos mais vulneráveis à formação de congestionamentos e quais fatores influenciam essas vulnerabilidades. O OPC encontra caminhos ótimos e então bloqueia o sítio de maior peso no caminho, a fim de entender como se reorganizam os caminhos ótimos da rede à medida que os sítios de maior peso são bloqueados. Para isso, primeiramente é feita uma revisão da Teoria de Grafos, com foco no algoritmo de Dijkstra para encontrar caminhos ótimos em uma rede. Em seguida, aplicamos o modelo de OPC à cidade de Fortaleza, com resultados comparados aos encontrados na literatura quando o algoritmo é aplicado a outras cidades, em particular Boston e Manhattan, em Nova Iorque. Vemos que a porção estudada do mapa de Fortaleza tem um número médio de segmentos bloqueados semelhante à Boston, além de uma distribuição semelhante de comprimentos de vias. Além disso, os resultados encontrados indicam que os trechos mais vulneráveis da cidade estão concentrados em $3,7\%$ do total de segmentos. Desse modo, o algoritmo foi capaz de identificar em quais trechos uma intervenção seria mais efetiva para diminuir congestionamentos na cidade, comprovando sua viabilidade para o estudo do trânsito de cidades.

Palavras-chave: Congestionamento; Sistemas complexos; Meios desordenados; Caminhos ótimos.

ABSTRACT

Traffic congestion in developed cities is a problem that causes financial and public health losses. The present work seeks through the OPC (Optimal Path Cracks) algorithm to understand which points are most vulnerable to the formation of congestion and which factors influence these vulnerabilities. OPC finds optimal paths and then blocks the site of greater weight along the way, in order to understand how the optimal paths of the network are rearranged as the sites of greater weight are blocked. For this, firstly, a review of Graph Theory is made, focusing on Dijkstra's algorithm to find optimal paths in a network. Next, we apply the OPC model to the city of Fortaleza, with results compared to those found in the literature when the algorithm is applied to other cities, particularly Boston and Manhattan in New York. Furthermore, the obtained results indicate that the most vulnerable sections of the city are concentrated in $3,7\%$ of the total segments. Thus, the algorithm was capable of identifying the sections where an intervention would be most effective in reducing congestion in the city, thereby demonstrating its viability for studying urban traffic.

Keywords: Traffic congestion; Complex systems; Disordered media; Optimal paths.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Com o aumento da população e da urbanização das cidades, nasce o problema dos congestionamentos no trânsito. Além dos danos evidentes ao meio ambiente pelo aumento da emissão de CO_2 e à qualidade de vida das pessoas que passam a ter um aumento do tempo de viagem, o incremento dos congestionamentos também prejudica o crescimento econômico das cidades.

Um estudo feito com dados de 88 áreas metropolitanas dos Estados Unidos investigou os limites para os quais o tempo gasto nos congestionamentos começava a influenciar no crescimento econômico das cidades (SWEET, 2014). Como parâmetro de análise para o crescimento econômico, foram utilizados a taxa de crescimento da empregabilidade (entre 1993 e 2008) e o crescimento da produtividade por trabalhador (entre 2001 e 2007).

Os resultados do trabalho indicam que os efeitos dos congestionamentos sobre a taxa de empregabilidade não são lineares. Na verdade, existe um limite para o tempo de congestionamento que uma vez superado resulta em uma diminuição da taxa de crescimento da empregabilidade. O limite estimado foi de 35 horas anuais de atraso em trajetos de casa para o trabalho e vice-versa, ou seja, um atraso médio de apenas 4,5 minutos por viagem é o suficiente para diminuir a taxa de crescimento de empregos. Além disso, também foi concluído que um aumento de 5% no volume médio diário de tráfego por faixa autoestrada é capaz de reduzir a expectativa da taxa de crescimento anual da produtividade por trabalhador em 0,16% (SWEET, 2014).

A Figura 1 ilustra o impacto dos congestionamentos nos Estados Unidos, demonstrando que até 2018 eles custavam aproximadamente 87 bilhões de dólares anualmente, com um custo médio de 1348 dólares por motorista (REED, 2019). Esses números refletem uma média de 97 horas anuais gastas em congestionamentos. Em Boston, a cidade com os maiores índices de congestionamento, o tempo gasto em média por motorista ao longo do ano em congestionamentos chegou a 164 horas, resultando em um prejuízo de 4,1 bilhões de dólares para a cidade.

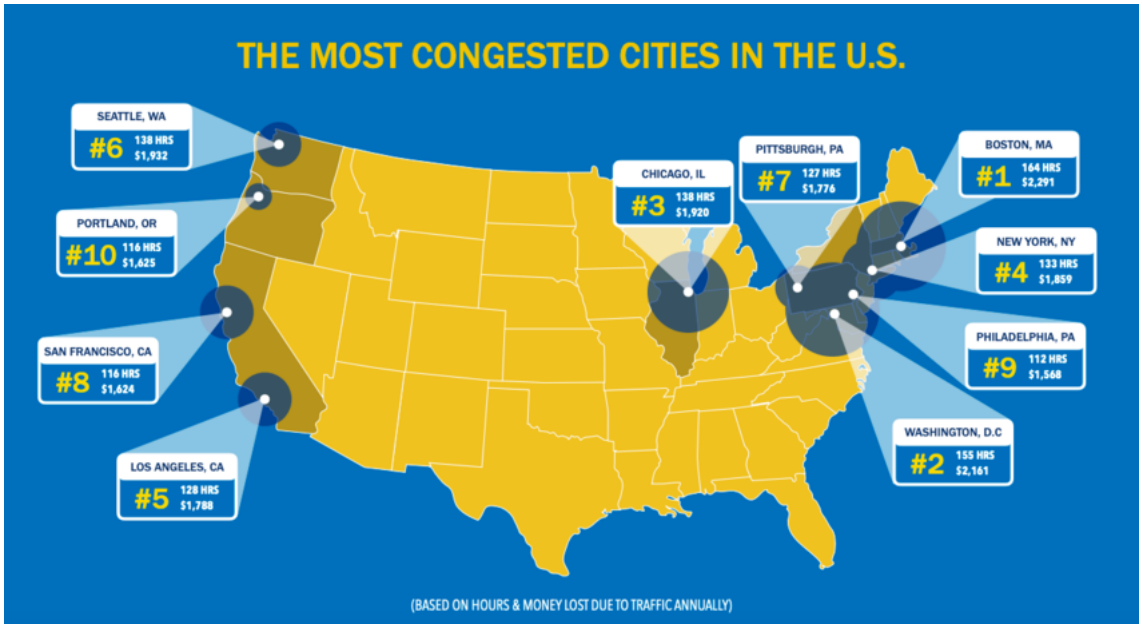


Figura 1 – Cidades Norte Americanas com mais congestionamento, com Boston em 1º lugar.
Fonte: Adaptada de (REED, 2019) pelo autor.

A Figura 2 mostra dados obtidos em 2022 para cidades do Brasil por uma empresa fabricante de sistemas de navegação para automóveis, que servem de indicativo do panorama do trânsito do país. No eixo vertical, é indicado quanto tempo as viagens demoraram acima do esperado. Os resultados indicam que Recife é a cidade brasileira com as maiores taxas, ficando em 24º lugar no ranking mundial. A cidade apresenta um índice médio de congestionamento de 40%, indicando que, na média, as viagens duraram 40% a mais de tempo do que o esperado em horários livres de trânsito. Esse índice piora quando se observa o horário de 17h nos dias úteis, nas quais a taxa de congestionamento ultrapassa os 60%, podendo chegar a 87%. Por ano, foi estimado um total de 92 horas extras despendidas em congestionamentos (TOMTOM, 2022).

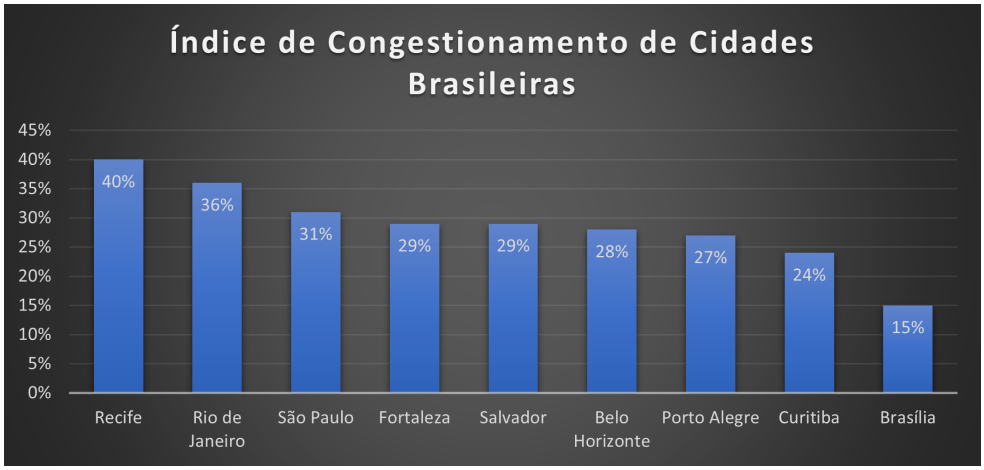


Figura 2 – Índice médio de congestionamento de cidades brasileiras.
Fonte: TomTom Traffic Index 2022, modificada pelo autor.

Em Fortaleza, o índice de congestionamento médio foi de 29%, chegando a 45% no horário de pico da manhã, por volta das 7 horas, e 57% no horário de pico da tarde, em torno de 18 horas. Em média foram gastas 66 horas extras devido, exclusivamente, a congestionamentos. Dentre as principais cidades, Brasília foi a que apresentou o menor índice médio de congestionamento, 15%, e uma média anual de 34 horas gastas em congestionamentos.

Todos esses resultados apontam para tempo e dinheiro desperdiçados no trânsito, além da redução da qualidade de vida da população e danos ambientais e sociais causados por engarrafamentos. Esses problemas são os fatores que respaldam a relevância do estudo do trânsito das cidades, em busca de entender as causas e propor soluções para os congestionamentos urbanos.

Estudos indicam que o número de quilômetros percorridos diariamente, o tamanho total da rede rodoviária, o consumo total de gasolina, a quantidade de CO_2 emitido diariamente e o tempo consumido em congestionamentos são fatores que variam como uma lei de potência com o tamanho da população de uma cidade. Resultados análogos já haviam sido encontrados anteriormente, mostrando que outras propriedades de cidades, como taxas de crimes cometidos, números de patentes, produto interno bruto e a área urbanizada na cidade também exibem uma propriedade de aumento em lei de potência com o tamanho da população (LOUF; BARTHELEMY, 2014). Isso indica que pode haver uma propriedade geral da evolução das características de cidades relacionada ao tamanho da população e a um fator que indica a sensibilidade a essa evolução. Essa hipótese é chamada de hipótese do dimensionamento urbano (BETTENCOURT *et al.*, 2013).

Para entender o conceito da hipótese do dimensionamento urbano, é preciso antes estudar as ferramentas utilizadas nesses estudos. Uma das principais abordagens utilizadas para o estudo do trânsito das cidades surge de uma área da física, mais especificamente a área de redes complexas.

Uma rede é um conjunto de pontos, chamados nós, e linhas que conectam os nós, chamadas de arestas. Sistemas presentes em física, biologia, economia e computação podem ser descritos como redes, divididas basicamente em redes tecnológicas, de informação, sociais e biológicas. A vantagem de utilizar redes para sistemas complexos é capturar apenas os padrões de conexão entre os nós para reduzir o sistema a uma representação abstrata do mesmo (NEWMAN, 2018).

Cidades seguem um dos princípios fundamentais de redes complexas: o surgimento

de propriedades coletivas não-triviais emergentes das interações entre seus componentes (NEWMAN, 2018). Cidades de forma geral podem ser divididas em um grande número de constituintes, sejam pessoas, instituições ou pontos geográficos. Fazendo então uma análise da evolução da cidade por meio das relações entre esses constituintes, é possível obter comportamentos como os da hipótese de dimensionamento urbano (BETTENCOURT *et al.*, 2013).

É utilizando a abordagem de sistemas complexos que o presente trabalho busca estudar as redes de trânsito de algumas cidades dos Estados Unidos e do Brasil. Para isso, para cada cidade teremos uma rede relacionada, nas quais os cruzamentos entre vias serão os nós da rede e as vias que ligam esses pontos são as arestas. Com a rede montada, podemos analisar caminhos entre dois pontos da rede, simulando uma viagem de um veículo ao longo da cidade. Usualmente, ao realizar um trajeto ao longo da cidade, buscamos o caminho mais curto ou o que leva menos tempo. Caminhos desse tipo são chamados de caminhos ótimos e serão definidos ao longo do capítulo 2. Partindo da hipótese que caminhos ótimos são muito utilizados e sofrer uma sobrecarga, estudaremos o que acontece quando um trecho de um caminho ótimo é bloqueado e o que isso diz sobre a estrutura da rede.

No capítulo 2, faremos uma breve introdução à Teoria de Grafos, justificando sua importância e apresentando conceitos básicos, como a definição de um grafo, seus elementos constituintes, suas representações e alguns algoritmos de mínimos caminhos, como *Breadth First Search*, *Depth First Search* e *Dijkstra*.

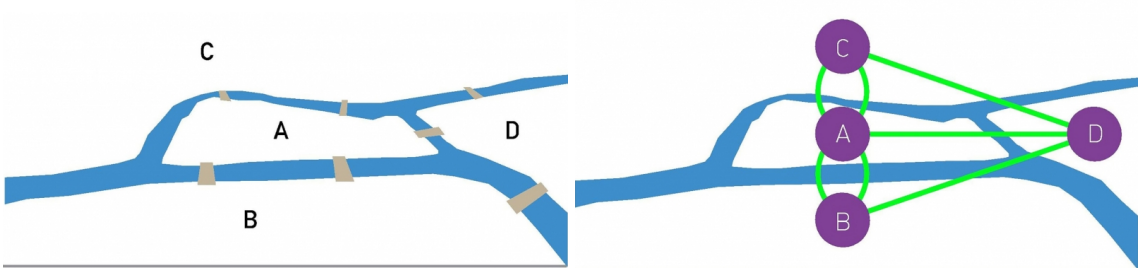
No capítulo 3, discutiremos o algoritmo *Optimal Path Cracking* (OPC), explicando sua abordagem em redes quadradas com pesos teóricos e posteriormente mostrando sua aplicação para redes reais de trânsito. Por último, no capítulo 4 mostraremos os resultados obtidos para a rede de trânsito montada para um trecho da cidade de Fortaleza e faremos a discussão dos resultados.

2 INTRODUÇÃO À TEORIA DE GRAFOS

Para estudar a fundo o comportamento das redes reais de trânsito, precisamos primeiro entender a base da teoria que descreve sistemas desse tipo, a Teoria de Grafos. Para isso, primeiro faremos uma breve introdução à história da origem desse campo de estudos.

O problema que deu origem ao que posteriormente seria a Teoria de Grafos foi o problema das 7 pontes de *Königsberg*, na Prússia (hoje Kaliningrado, Rússia) (BIGGS *et al.*, 1986). As pontes, construídas graças ao desenvolvimento do comércio local, resultaram no surgimento de um questionamento: existe algum trajeto que percorra as 7 pontes sem passar pela mesma ponte duas vezes ? Esse questionamento foi respondido em 1735 por Euler, que provou a inexistência de tal trajeto (BARABÁSI, 2013).

Para realizar esta demonstração, Euler procedeu como indica a Figura 3, identificando cada porção de terra da cidade como nós A, B, C e D e pensando nas pontes como ligações entre esses nós. Euler observou que se um nó tem um número ímpar de ligações, ele deve ser o início ou o fim de um caminho. Os nós do meio do caminho devem necessariamente apresentar um número par de ligações, pois para cada ponte usada para chegar ao nó, deve-se ter uma para sair do nó. No entanto, podemos perceber que os 4 nós do grafo têm números ímpares de ligações. Dessa forma, é impossível encontrar um caminho para percorrer todas as pontes sem repetir nenhuma.



(a) Pontes de *Königsberg* (b) Grafo das Pontes

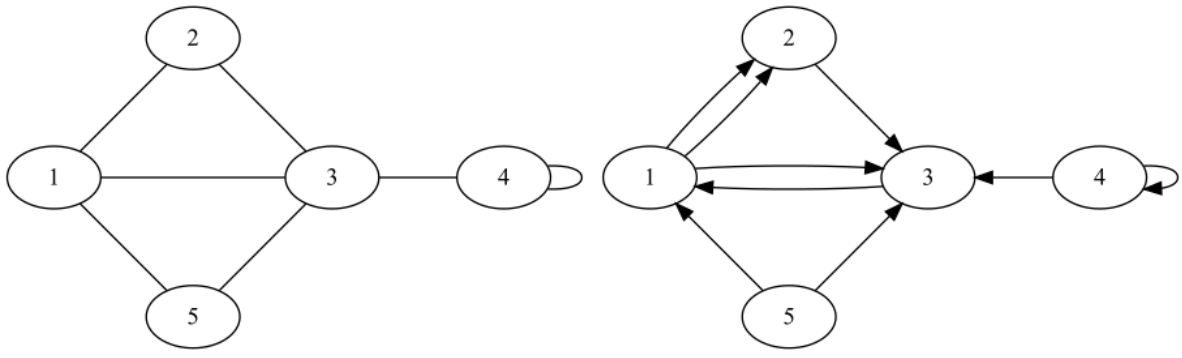
Figura 3 – Representações da cidade de *Königsberg*. (a) Representação da cidade destacando o rio que atravessa a cidade e as 7 pontes. (b) Representação em forma de grafo da cidade. Cada área é representada por um nó e cada ponte é representada por uma ligação entre os nós (BARABÁSI, 2013).

Esse problema aparentemente simples foi o primeiro a ser resolvido utilizando grafos e suas propriedades. A solução apresentada por Euler demonstra que a informação contida no número de nós e no número de ligações que sai desses nós era suficiente para acabar o problema, evidenciando a simplicidade do tratamento do problema via grafos.

2.1 Definições

Um grafo em si pode ser considerado como um conjunto $G(V,E)$ que contém um conjunto V de nós e um conjunto E de arestas. Esse conjunto é uma representação abstrata de um sistema qualquer de elementos que são representados por nós e cujas interações entre elementos do sistema são representados por arestas que ligam os nós. Por terem aplicações em muitas áreas, os grafos apresentam diferentes nomenclaturas. Os nós também podem ser chamados de vértices ou pontos e as arestas também podem ser chamadas de ligações ou linhas.

Denotaremos por $|V|$ o número de nós e $|E|$ o número de arestas de um grafo. Geralmente, os nós são representados por círculos contendo um rótulo (usualmente um número ou uma letra) e as arestas por linhas unindo os nós. Existem situações em que as interações têm uma direção preferencial, como no caso de vias públicas, onde cada via só pode ser percorrida em uma direção definida. Nesse caso, dizemos que o grafo é direcionado e a aresta (i,j) é diferente da aresta (j,i). No caso em que a ligação não tem um sentido preferencial, dizemos que o grafo é não direcionado e a aresta (i,j) é idêntica à aresta (j,i) (CORMEN *et al.*, 2022). A Figura 4 mostra dois exemplos de grafos com os mesmos nós, mas arestas diferentes. Em um dos casos, todas as arestas são não direcionadas. No outro, podemos observar que as arestas têm direção preferencial e portanto o grafo é direcionado. Além disso, é possível que um nó tenha uma ligação para ele mesmo, como acontece no nó 4 e que exista mais de uma ligação entre dois nós, como acontece entre os nós 1 e 2 e também entre 1 e 3.



(a) Grafo não direcionado (b) Grafo direcionado

Figura 4 – Grafos com 5 nós. (a) As arestas nesse caso não têm uma direção preferencial. (b) Neste caso, podemos ver que as arestas têm uma direção preferencial.
Fonte: Elaborado pelo autor.

Um conceito que aparece frequentemente em várias áreas de aplicações da Teoria de Grafos é o transporte por meio da rede. No exemplo das pontes de *Königsberg*, a ideia era de

uma pessoa caminhando pelas sete pontes. Também podemos pensar na informação viajando por meio da *World Wide Web* ou de carros se locomovendo em uma rede de trânsito. Em todos esses casos, surge o conceito de um caminho por meio da rede. Em Teoria de Grafos, um caminho é definido como uma sequência de nós da rede, em que dois nós consecutivos da sequência devem necessariamente estar ligados por uma aresta (EASLEY; KLEINBERG, 2010).

2.1.1 Representações

Além do desenho do grafo, podemos utilizar conjuntos para representar os grafos. A parte (a) da Figura 4 por exemplo, pode ser definida pelo conjunto $G(V,E)$, em que $V = \{1,\ 2,\ 3,\ 4,\ 5\}$ e $E = \{(1,2),\ (1,3),\ (1,5),\ (2,3),\ (5,3),\ (3,4)$ e $(4,4)\}$. No entanto, a fim de facilitar a computação de um grafo, existem outras representações bastante úteis.

A primeira das representações alternativas dos grafos é chamada de lista de adjacências. Para um grafo $G(V,E)$, as listas de adjacências consistem em uma coleção de $|V|$ listas em que a i-ésima lista contém todos os vértices j tais que existe uma aresta (i, j) no grafo G. A Figura 5 mostra as listas de adjacências para o grafo não direcionado da Figura 4.

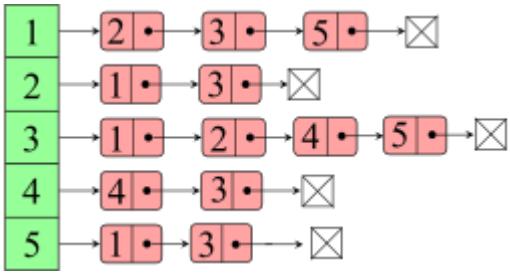


Figura 5 – Lista de adjacências do grafo (a) na Figura 4
Fonte: Elaborada pelo autor.

A outra representação consiste em uma matriz A, de dimensão $|V| \times |V|$ tal que o elemento A_{ij} é igual a 1 caso haja uma aresta que liga os vértices i e j, e 0 caso contrário. Dessa forma, para o grafo representado em (a) da Figura 4, a matriz de adjacências seria dada por:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{2.1}$$

Apesar de termos tratado apenas o caso de um grafo não direcionado, tanto a representação da lista de adjacências quanto da matriz de adjacências são facilmente estendidas para o caso de grafos direcionados. Para isso, basta levar em consideração que a aresta (i, j) é diferente da aresta (j, i) e, portanto, a matriz de adjacências não é mais necessariamente uma matriz simétrica.

Podemos perceber que ambas as representações contém as informações necessárias a respeito de um grafo. A escolha entre as duas representações depende do tipo de grafo com o qual estamos lidando. A representação via listas de adjacências é mais adequadas para grafos esparsos, que apresentam um número de arestas muito menor do que o quadrado do número de vértices. Quando o grafo apresenta um número de arestas próximo ao quadrado do número de vértices, dizemos que o grafo é denso e nesse caso a representação da matriz de adjacências é a mais adequada (CORMEN *et al.*, 2022).

2.1.2 Grafos Valorados

Até agora, temos tratado os grafos de modo que todas as arestas são idênticas entre si, mudando apenas os nós que elas estão ligando. No entanto, quando imaginamos que os grafos servem para descrever diversos sistemas de diferentes áreas, é fácil chegar a conclusão de que nem todas as ligações têm a mesma intensidade. Dessa forma, para descrever melhor como os nós de um grafo interagem entre si, introduzimos o conceito de peso de uma aresta. Mais rigorosamente, definimos um grafo valorado como um grafo cujas arestas tem uma grandeza relacionada $\omega : E \to \mathbb{R}$ que está relacionada de alguma forma à intensidade da ligação entre os dois nós da aresta (CORMEN *et al.*, 2022).

Grafos valorados podem ser representados de ambas as formas apresentadas anteriormente. Na lista de adjacências, para cada nó devemos guardar quais nós se conectam com ele e os pesos dessas ligações. Por sua vez, na matriz de adjacências, ao invés de indicar como 1 a presença de uma aresta, utilizamos o valor do peso para indicar sua existência. Caso não exista uma aresta entre dois nós, o valor é marcado como 0, como já era feito para grafos não valorados.

Ao longo do presente trabalho, trataremos de algoritmos que atuam em grafos valorados. Para isso, é preciso definir uma notação comum à todos os algoritmos pseudocódigos utilizados. Nos algoritmos que utilizaremos será preciso guardar informações sobre os objetos que estaremos tratando.

O exemplo mais básico é do próprio grafo, onde precisamos armazenar informações sobre os nós e sobre as arestas. Para um grafo $G(V,E)$, denotaremos por $G.V$ o conjunto de nós e $G.E$ o conjunto de arestas. Essa notação serve para indicar que E e V são atributos do objeto G. Se utilizarmos a representação da lista de adjacências, podemos escrever $G.adj[u]$ para representar a lista de adjacências do nó u, pertencente ao grafo G. Da mesma forma, caso seja necessário guardar alguma informação x do nó u, escrevemos $u.x$ e para guardar uma informação y da aresta (u,v) escrevemos $(u,v).y$.

2.2 Mínimos Caminhos

Em todos tipos de redes, quando estudamos o transporte de algum tipo de objeto ou informação ao longo da rede é importante conhecer informações sobre o caminho escolhido a fim de entender comportamentos da rede, planejar ações ou identificar problemas. Para fazer o estudo dos caminhos ao longo da rede, definimos o comprimento de um caminho em um grafo não valorado como o número passos entre nós durante o caminho. Dentre os caminhos possíveis, o comprimento do caminho com o menor número de passos determina a distância entre dois nós do grafo.

2.2.1 Algoritmos de Busca

Um algoritmo que serve de introdução para o estudo de caminhos ao longo de redes é o algoritmo de busca em largura ou *BFS* (*Breadth-First-Search*). Esse algoritmo serve para encontrar a distância entre um nó, denotado por s, escolhido como ponto de partida e todos os outros nós da rede.

A inicialização do algoritmo é feita definindo uma distância 0 para o nó escolhido como ponto de partida. Na primeira iteração, atribuímos a distância 1 para todos os nós vizinhos de s, isto é, que possuem uma aresta com s. Em seguida, identificamos os vizinhos dos vizinhos de s, que por consequência têm distância 2 do nó de origem. Dessa forma, repetindo esse processo e atribuindo distâncias em cada iteração a nós ainda não visitados, conseguimos encontrar todas as distâncias dos nós da rede ao nó de partida s (NEWMAN, 2018).

Esse algoritmo pode ser utilizado em sites de redes sociais, em que as pessoas que utilizam a rede são representadas por nós e as interações são representadas por arestas. Assim, é possível encontrar pessoas dentro de uma certa distância de interação entre si. Além disso,

a busca em largura também é utilizada para a automação do gerenciamento de memória via algoritmo de Cheney (CHENEY, 1970) e para encontrar o fluxo máximo em uma rede de fluxo, via algoritmo de Ford-Fulkerson (FORD; FULKERSON, 1956).

Outro importante algoritmo de busca é o algoritmo de busca em profundidade ou *DFS* (*Depth-First-Search*). Esse algoritmo percorre o grafo por meio das arestas do último nó descoberto. Se o último nó descoberto tem arestas saindo dele, o algoritmo continua percorrendo as arestas para chegar em nós ainda não visitados. Quando encontra um nó que não tem mais arestas a serem percorridas, o algoritmo faz o caminho inverso até encontrar um nó com arestas ligando a nós ainda não visitados. Esse processo continua até que tenhamos descoberto todos os nós para os quais existe um caminho a partir de um nó escolhido como origem.

O algoritmo de busca em profundidade também desempenha um papel fundamental no algoritmo de Kosaraju para a identificação de componentes fortemente conectados em um grafo direcionado. Especificamente, o algoritmo de Kosaraju utiliza o DFS para localizar subgrafos nos quais todos os vértices são mutuamente alcançáveis (ALFRED *et al.*, 1983). Nesse contexto, os subgrafos encontrados são caracterizados pela propriedade de que qualquer vértice dentro do subgrafo pode ser alcançado a partir de qualquer outro vértice, enfatizando a forte conexão entre seus componentes.

Outra aplicação da busca em profundidade é no algoritmo de ordenação topológica de um grafo direcionado acíclico. Um grafo é acíclico quando não existe um ciclo, isto é, um caminho de pelo menos 3 nós em que o primeiro e o último nó do caminho são iguais e o restante diferentes entre si. A ordenação topológica funciona de forma que se existe uma aresta (u, v) em um grafo G, então o nó u deve necessariamente aparecer antes de v na ordenação (CORMEN *et al.*, 2022).

2.2.2 Algoritmo de Dijkstra

Diariamente quando precisamos nos locomover de um ponto a outro de uma cidade fazemos, mesmo que automaticamente ou por meio de um GPS, a escolha do melhor caminho possível entre esses dois pontos. O caminho que minimiza algum fator determinado, como o tempo de viagem entre dois pontos da cidade, é chamado de caminho ótimo. A determinação do caminho ótimo entre dois pontos encontra aplicação em várias áreas da ciência, como transporte em meios porosos e polímeros aleatórios (MÉZARD *et al.*, 1984; ANSARI *et al.*, 1985; KIRKPATRICK; TOULOUSE, 1985).

Para resolver problemas desse tipo, foram inventados diversos algoritmos de caminho mínimo. O algoritmo de Bellman-Ford, por exemplo, serve para encontrar o caminho mínimo para grafos valorados que contém pesos positivos e negativos (BELLMAN, 1958; JR, 1956). Para um propósito semelhante existe o algoritmo de Floyd-Warshal, que cálcula o peso do caminho de uma fonte até todos os outros nós em um grafo valorado, com pesos positivos ou negativos, mas sem ciclos de pesos negativos. Por sua vez, o algoritmo A^* encontra o menor caminho entre 2 pontos e pode ser utilizado em sistemas especializados em encontrar rotas (CORMEN *et al.*, 2022).

Neste trabalho, utilizaremos o algoritmo de Dijkstra para encontrar o caminho ótimo entre dois pontos de uma rede (DIJKSTRA *et al.*, 1959). Esse algoritmo é uma espécie de generalização do algoritmo de busca em profundidade, sendo utilizado em redes valoradas cujos pesos das arestas são todos positivos. O algoritmo de Dijkstra calcula o melhor caminho de uma fonte até todos os outros nós da rede minimizando a soma dos pesos das arestas ao longo dos caminhos. Além de ser diretamente utilizado para encontrar o melhor caminho a ser percorrido em uma cidade, o algoritmo de Dijkstra também é utilizado em aplicações recentes no planejamento e suavização da rota de robôs móveis (LI *et al.*, 2022), na análise da interação entre alimento e medicamentos (RAHMAN *et al.*, 2022) e em algoritmos de aprendizado de máquina para encontrar representações significativas de sequências de proteínas (DETLEFSEN *et al.*, 2022).

Antes de definir os passos do algoritmo, iremos abordar os conceitos necessários para o mesmo. Desse modo, consideremos um grafo valorado $G(V,E)$ cujas arestas estão associadas com um peso por meio de uma função $\omega : E \to \mathbb{R}$. Queremos então encontrar o caminho entre dois nós do grafo que tem o menor peso $\omega(p)$, definido como a soma dos pesos das arestas que constituem o caminho. Isto é, dado um caminho $p = \langle v_0, v_1, ..., v_k \rangle$, o peso desse caminho é dado pela Equação 2.2:

$$\omega(p) = \sum_{i=1}^{k} \omega(v_{k-1}, v_k). \tag{2.2}$$

Considerando esse conceito, podemos partir para a definição do caminho ótimo entre dois nós u e v como mostra a Equação 2.3 (CORMEN *et al.*, 2022):

$$\delta(u,v) = \begin{cases} min\{\omega(p) : u \xrightarrow{\text{p}} v\} \text{ se há caminho de u para v,} \\ \infty, \text{ caso contrário.} \end{cases} \tag{2.3}$$

Existem algoritmos de caminhos mínimos que calculam os caminhos entre todos os pares possíveis de nós e algoritmos de fonte única, que calculam os caminhos ótimos de todos os nós do grafo partindo de uma fonte escolhida. Neste trabalho, focaremos nos algoritmos de fonte única, como o algoritmo de Bellman-Ford e especialmente o algoritmo de Dijkstra.

Vale ressaltar que os algoritmos de caminhos ótimos com fonte única podem ser modificados para resolver diferentes problemas. Se invertermos a direção dos caminhos, podemos calcular os caminhos mínimos de todos os nós do sistema até um nó de destino. Quando resolvemos o problema de mínimos caminhos com fonte única, também resolvemos o problema de encontrar o mínimo caminho entre dois nós específicos do grafo, bastando desconsiderar os caminhos da fonte até os outros nós. Além disso, apesar de existirem formas mais eficientes de fazer isso, podemos repetir o algoritmo de fonte única para todos os nós do grafo, determinando assim todos os mínimos caminhos entre todos os pares possíveis de nós do grafo. Em todos os casos, os algoritmos se beneficiam do fato de que ao determinar o caminho mínimo entre dois nós, fica determinado também o caminho mínimo entre os nós intermediários.

Na maioria dos casos em que desejamos encontrar o mínimo caminho entre dois nós de um grafo, é interessante conhecer todos os nós do caminho, ao invés de apenas determinar o peso do menor caminho. Para fazer isso, os algoritmos de mínimo caminho armazenam atributos específicos para cada vértice. Assim, para todo nó $v.\pi \in V$, com V sendo o conjunto de nós de um grafo $G(V,E)$, o algoritmo armazena um atributo $v.\pi$ chamado de predecessor de v. O predecessor de um nó, é outro nó ou um valor nulo (caso o nó em questão seja a fonte do caminho, em que obviamente não existe predecessor ou no caso em que o algoritmo ainda não determinou nenhum caminho até o nó). Dessa forma, quando o algoritmo tiver terminado todas as iterações, todos os nós terão um outro nó como predecessor, exceto a fonte.

Para determinar o caminho até qualquer um dos nós basta fazer uma lista de antecessores partindo do nó de destino até que se chegue ao nó que serviu de fonte para o algoritmo. No final do algoritmo, obtemos um subgrafo de predecessores $G_\pi = (V_\pi, E_\pi)$ definidos pelos valores π. Os conjuntos V_π e E_π são respectivamente o conjunto de nós de G com predecessores não nulos mais a fonte s e o conjunto de arestas dirigidas induzidas pelos valores de π para os nós pertencentes a V_π. Desse modo, podemos escrever os conjuntos V_π e E_π como mostram as Equações 2.4 e 2.5:

$$V_\pi = \{v \in V : v.\pi \neq \text{NULO}\} \cup \{s\} \tag{2.4}$$

e

$$E_\pi = \{(v.\pi, v) \in E : v \in E_\pi - \{s\}\}. \tag{2.5}$$

O algoritmo de Dijkstra utiliza a cada iteração a técnica de relaxamento. Essa técnica necessita da definição de um novo atributo para cada nó do grafo. Assim, dado $G(V,E)$, todo nó $v \in$ V agora tem associado a si um atributo $v.d$ que serve de estimativa para o peso do menor caminho partindo da fonte s e indo até o nó v. Em cada iteração, o algoritmo verifica se a estimativa pode ser melhorada e em caso positivo atualiza a estimativa e altera o predecessor do nó. Ao final da execução, o atributo $v.d$ guardará a informação do peso efetivo do menor caminho de s até v.

Agora que já fizemos a introdução das características necessárias para que possamos aplicar o algoritmo de Dijkstra a um grafo e já definimos os atributos que precisaremos armazenar, podemos dar início ao passo a passo do algoritmo. O primeiro passo consiste em inicializar os atributos dos nós do grafo.

Para inicializar os atributos dos nós, daremos valor infinito à estimativa de caminho para todos os nós do grafo exceto a fonte. Isso serve para indicar que inicialmente os nós ainda não têm nenhum caminho descoberto desde a fonte. A estimativa de caminho para o nó que serve de fonte é inicializada como 0 e deve permanecer assim por todo o caminho, pois obviamente a distância de um nó para ele mesmo é 0. O outro atributo, por sua vez, que indica qual o nó predecessor, é inicializado com valor nulo. Isso é feito pois ainda não fizemos nenhuma estimativa de caminho e portanto não conhecemos o predecessor de nenhum nó.

Na notação computacional, para um gravo $G(V,E)$ e para um nó $s \in$ V que serve de fonte, escrevemos que o atributo predecessor é inicializado como $v.\pi =$ NULO para todo nó $v \in$ V. Por sua vez, o atributo de estimativa de distância é inicializado como $v.d = \infty$ para todo nó $v \in V - \{s\}$ e $s.d = 0$.

O segundo passo consiste em criar dois objetos que servirão para acompanhar o andamento do algoritmo e para indicar quando o algoritmo está finalizado. O primeiro objeto a ser criado é um conjunto S, que armazenará todos os nós para os quais o caminho mínimo partindo da fonte s já foi determinado pelo algoritmo. Naturalmente, esse conjunto inicialmente é vazio.

O outro objeto é uma fila de prioridade mínima. Essa estrutura de dados tem como princípio um conjunto Q, em que cada elemento tem uma propriedade associada chamada de chave. Sobre a fila de prioridade mínima, podem atuar 4 operações. A primeira consiste em

inserir um elemento no conjunto Q. A segunda propriedade retorna o elemento com o menor valor de chave dentre todos os elementos de Q. A terceira operação é semelhante à anterior, porém além de retornar o elemento com a menor chave, esse elemento é retirado do conjunto Q. A última propriedade baseia-se em incrementar o valor chave de um dos elementos de Q.

Uma fila de prioridades mínimas pode ser implementada de diferentes formas. A forma mais simples consiste em armazenar os nós em um vetor com número de posições igual ao número de nós do grafo. No entanto, apesar de simples essa forma usualmente é mais lenta, de modo que se faz necessário apresentar uma estrutura de dados chamada de *(*heap). *Heaps* binárias são estruturas baseadas em árvores binárias, em que cada nó da árvore está relacionado a um elemento entre os que devem ser ordenados. Outra possibilidade de implementação de filas de prioridades é utilizando as *F-heaps* ou *Heaps* de Fibonnacci. Entretanto, essas estruturas de dados além de serem mais complexas têm a particularidade de que algumas iterações demoram muito para serem executadas.

No algoritmo de Dijkstra, utilizamos uma fila de prioridade mínima Q para armazenar nós do grafo G, com a chave de cada nós sendo o valor do seu atributo de estimativa de caminho d.

Uma vez criados o conjunto S e a fila de prioridades Q, começamos as iterações do algoritmo que atualizarão os atributos. Pelas definições de S e Q, podemos perceber que o conjunto total de vértices do grafo pode ser escrito como $V = Q + S$. Lembrando das operações que atuam sobre uma fila de prioridade mínima, utilizaremos a operação que extrai da fila o elemento com o menor valor de chave. Assim, cada iteração do algoritmo irá identificar na fila Q qual elemento u tem o menor valor do atributo $u.d$ e retirar da fila.

A Figura 6 contém imagens dos passos de implementação do algoritmo para uma rede dirigida com 6 nós. Os números sobre as arestas indicam os pesos da aresta. Os valores dentro dos círculos indicam as estimativas de caminhos mínimos. O nó s é escolhido como a fonte. Como esperado, o nó s tem peso final 0.

Na Figura 6, o quadro (a) mostra a inicialização do algoritmo, atribuindo os primeiros valores dos atributos da rede. Na primeira iteração, necessariamente o nó retirado de Q será a fonte s, visto que todos os nós são inicializados com o atributo $d = \infty$, exceto s que tem $s.d = 0$. O nó retirado de Q nessa iteração é então adicionado ao conjunto S e dizemos que esse nó foi visitado e então ele não será mais movido de S nem de Q. Assim, dizemos que Q é um invariante, pois antes e depois de cada iteração a propriedade $Q = V - S$ se mantém verdadeira. Além disso,

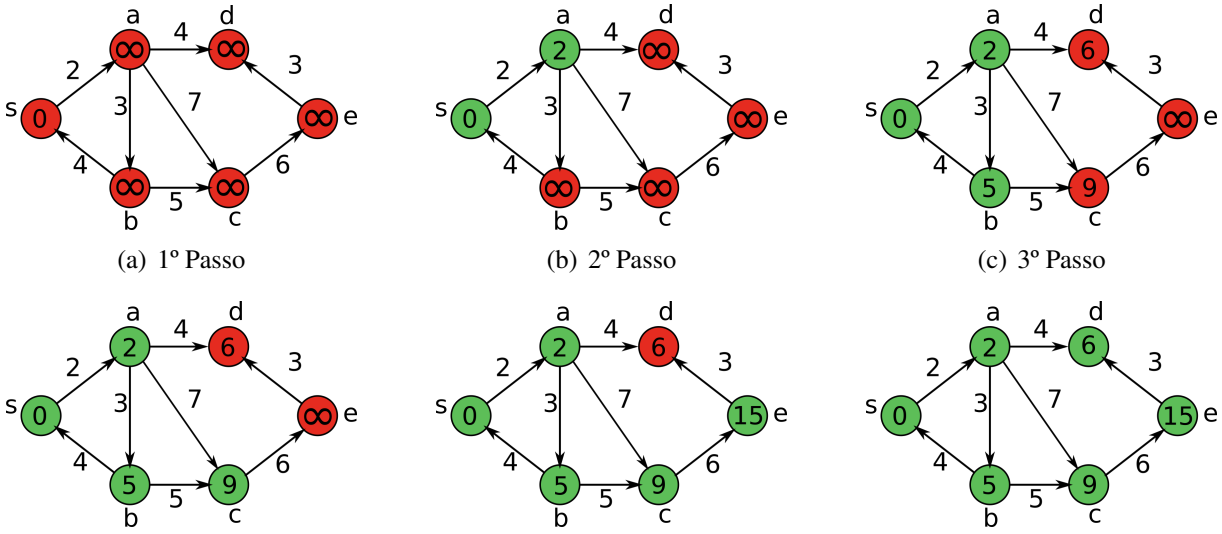


Figura 6 – Iterações do algoritmo de Dijkstra. (a) Passo de inicialização do algoritmo. (b)O peso do caminho até o nó s é 0 e até o nó a é 2. (c) São feitas estimativas de caminhos para todos os nós vizinhos ao nó a e então partimos para o nó b, que tem o menor peso. (d) Do nó b vamos para o nó c, mantendo a estimativa anterior. (e)Partimos do nó d para o nó e .(f) Por último, verificamos o peso para o nó d, mantendo a primeira estimativa.

é possível demonstrar que, uma vez que o nó u é adicionado ao conjunto S, a distância mínima da fonte até u está determinada, isto é, $u.d = d(s,u)$ (CORMEN *et al.*, 2022). O quadro (b), mostra a rede após a primeira iteração, quando a fonte já foi visitada e o começo da segunda iteração, quando partimos para o nó mais próximo da fonte. Os quadros (c), (d) e (e) mostram as seguintes iterações, à medida em que a rede vai sendo percorrida e os nós vão sendo retirados da fila de prioridade Q e adicionados ao conjunto S. Por último, o quadro (f) mostra a rede completa e os valores dos pesos de cada caminho desde a fonte até cada nó da rede.

O algoritmo de Dijkstra pertence à classe de algoritmos gulosos. Isso significa que à cada etapa o algoritmo escolhe tomar o caminho que otimiza o resultado localmente, sem garantir que isso resultará em uma solução geral ótima.

3 FRATURAS DE CAMINHOS ÓTIMOS

No capítulo passado apresentamos as definições e conceitos necessários para o estudo de algoritmos de busca de caminhos ótimos entre dois pontos de um grafo. Em sistemas complexos, tais problemas chamados de problemas de caminhos ótimos. Como citado anteriormente, podemos representar uma cidade como uma rede e buscar percorrer caminhos ótimos, seja com base na experiência diária ou utilizando um aparelho como um GPS para fazer o cálculo do caminho ótimo.

No entanto, em uma cidade com um grande número de habitantes, nos horários de pico diversas pessoas buscam percorrer um caminho ótimo em um curto intervalo de tempo. Com isso, os caminhos ótimos que ligam dois pontos muito visados na rede acabam sendo muito utilizados e portanto são mais suscetíveis à formação de engarrafamentos. Descobrir quais são esses pontos é essencial para que sejam feitas intervenções capazes de melhorar o trânsito de forma geral. Para analisar os pontos mais vulneráveis de uma rede de trânsito, utilizaremos o algoritmo de Fraturas de Caminhos Ótimos que será explicado a seguir.

3.1 Algoritmo

Como visto anteriormente, os pesos de um grafo não necessariamente representam uma distância geométrica. Caso o peso do grafo não seja a distância entre dois pontos, o caminho ótimo difere do caminho mínimo geométrico.

O peso de uma ligação entre dois elementos de uma rede serve para representar uma interação entre esses dois elementos. Também é possível atribuir pesos aos sítios de uma rede, de forma que cada peso representa uma interação entre o elemento relacionado ao sítio e o meio em que o elemento está inserido. Dessa forma, os pesos podem ser dados por medidas de tempo de percurso ou de energia de interação. Em uma rede criada para representar um meio físico, as ligações são associadas a pesos que representam a energia de interação de um determinado sítio com o meio que o cerca.

Partindo disso, podemos definir um meio como homogêneo se todas as ligações têm os mesmos valores de energia. Nesse caso, todos caminhos de mesmo comprimento têm o mesmo valor de energia total. No caso em que as energias das ligações são diferentes, dizemos que o meio é desordenado. Se as energias atribuídas às ligações não são correlacionadas, isto é, se a energia de uma ligação não afeta a energia de ligações próximas, então a desordem é

aleatória.

Para iniciar o estudo do algoritmo de fraturas de caminhos ótimos, consideremos uma rede quadrada, com condições de contorno periódicas na direção horizontal e condições de contorno fixas nas extremidades de cima e de baixo. Queremos estudar falhas nos caminhos ótimos em meios desordenados. Para isso, atribuímos à cada sítio um peso representado por uma energia ε_i dada por:

$$\varepsilon_i = \exp\left[\beta \cdot (p_i - 1)\right]. \tag{3.1}$$

Essa distribuição é equivalente a atribuir valores a partir de uma distribuição em lei de potência da forma $P(\varepsilon_i) \sim \frac{1}{\varepsilon_i}$. Para mostrar isso, consideremos a condição de normalização:

$$\int_{\varepsilon_{min}}^{\varepsilon_{max}} p(\varepsilon) d\varepsilon = 1. \tag{3.2}$$

A fim de tornar possível a normalização, definimos os limites de integração $\varepsilon_{max} = 1$ e $\varepsilon_{min} = \exp(-\beta)$. Desse modo, realizando a integral temos:

$$\int_{e^{-\beta}}^{1} \frac{C}{\varepsilon} d\varepsilon = 1 \Rightarrow C = \frac{1}{\beta}. \tag{3.3}$$

Uma vez que encontramos a constante de normalização $C = 1/\beta$, podemos fazer uma transformação de variáveis nas probabilidades:

$$|p(y')dy'| = |p(x')dx'|. \tag{3.4}$$

Substituindo a variável y por ε e x por uma variável aleatória p distribuída uniformemente, temos:

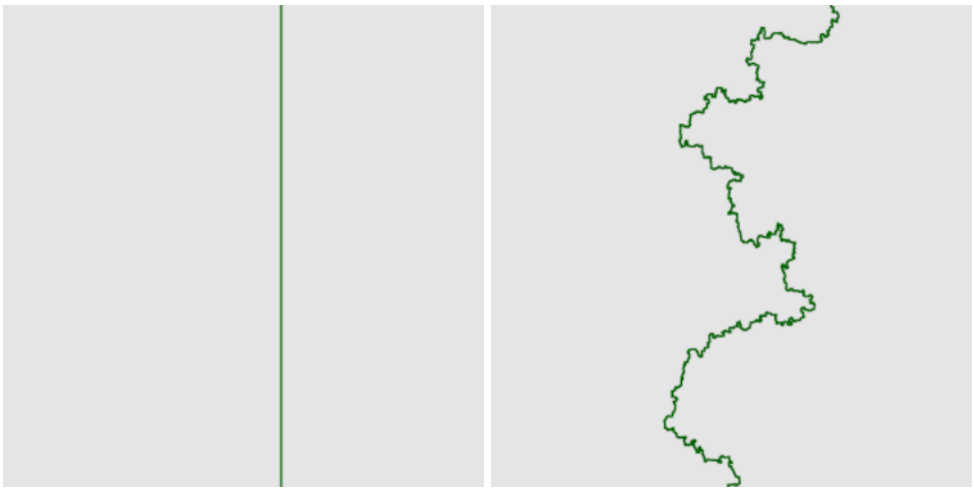
$$\int_{e^{-\beta}}^{\varepsilon} \frac{1}{\beta \varepsilon'} d\varepsilon' = \int_0^p dp' \Rightarrow \varepsilon = \exp\left[\beta \cdot (p - 1)\right]. \tag{3.5}$$

Na equação 3.1, p_i é uma variável aleatória uniforme pertencente ao intervalo [0,1] e β é um parâmetro que regula a intensidade da desordem do sistema. Quanto maior o valor de β, mais significativa é a discrepância nos valores de energia obtidos para os sítios e por

consequência, maior a desordem do sistema. Os sistemas que trataremos são classificados qualitativamente como tendo desordem fraca ou forte.

Uma vez criada a rede, estudaremos inicialmente o problema dos caminhos ótimos. Nesse caso, queremos encontrar o caminho conectando as extremidades inferior e superior da rede cuja soma de energias dos sítios é a menor dentre todos os caminhos possíveis. Como os valores de energia são todos positivos, podemos utilizar o algoritmo de Dijkstra para encontrar a trajetória de menor energia.

Naturalmente, se o meio é homogêneo, a soma das energias será a menor possível no caminho que passa pelo menor número de sítios e, portanto, o caminho percorrido é uma linha reta conectando a parte superior à parte inferior da rede. No entanto, quando existe a presença de desordem os caminhos deixam de ser retas e passam a depender da organização das energias dos sítios do sistema. A Figura 7 mostra os caminhos ótimos encontrados para uma rede quadrada com as condições de contorno explicitadas anteriormente e com $L = 1024$, mas com desordem diferente. Para valores de beta muito pequenos a desordem é muito fraca, e o meio é aproximadamente homogêneo. Quando a desordem é forte, vemos que o caminho difere bastante de uma reta, como esperado (OLIVEIRA, 2012).



(a) Desordem fraca (b) Desordem forte

Figura 7 – Caminhos ótimos para redes quadradas de tamanho L = 1024. (a) Para uma desordem fraca ($\beta = 0.0625$, o meio é, aproximadamente, homogêneo e, portanto, o caminho ótimo é uma reta. (b) Para desordem forte ($\beta = 1024$, o caminho deixa de ser uma reta, tomando uma forma que depende das energias atribuídas a cada sítio.
Fonte: (OLIVEIRA, 2012)

O caminho ótimo encontrado na Figura 7 para o caso de desordem forte pertence a uma classe de objetos de interesse físico chamados de fractais (OLIVEIRA, 2012). Para

entender o conceito de fractais, primeiro precisamos entender dois tipos de dimensões que podem ser atribuídas a objetos geométricos. Sistemas descritos usuais são descritos pela Geometria Euclidiana apresentam uma dimensão topológica D_T que pode ser entendida como o número de coordenadas necessárias para representar o objeto no espeço. No entanto, para descrever objetos de maior complexidade, surgiu a definição de dimensão de Haussdorf-Besicovich.

Um objeto arbitrário imerso em um espeço de dimensão topológica d_E pode ser coberto por $N(\varepsilon)$ bolas de raio ε e volume proporcional a ε^{d_E}, de modo que seu volume é dado por:

$$V(\varepsilon) = N(\varepsilon) \cdot \varepsilon^{d_E}, \tag{3.6}$$

de modo que em geral, objetos obedecem a uma equação do tipo $N(\varepsilon) \propto \varepsilon^{-d_E}$. Fractais são objetos para os quais $N(\varepsilon) \propto \varepsilon^{-D}$, onde D é dado por (FEDER, 2013):

$$D = \lim_{\varepsilon \to 0} \left[\frac{ln(N(\varepsilon))}{ln(\frac{1}{\varepsilon})} \right]. \tag{3.7}$$

Caminhos ótimos são os mais acessados em uma rede e, portanto, têm uma tendência a serem sobrecarregados. O modelo de fraturas de caminhos ótimos, tenta entender quais os pontos mais suscetíveis a sobrecarga na rede. Para isso, uma vez identificado o caminho ótimo que conecta a extremidade inferior à extremidade superior da rede, escolhemos o sítio com o maior valor de energia e o bloqueamos. Chamaremos esse processo de microfratura. Computacionalmente falando, a microfratura pode ser feita retirando o sítio da rede ou atribuindo ao sítio uma energia de valor infinito. Assim, garantimos que o sítio não pode ser acessado e então estudamos como se modificam os caminhos ótimos na rede sem a presença do sítio bloqueado.

Depois de realizar o processo de microfratura, realizamos mais uma iteração aplicando o algoritmo de Dijkstra para encontrar um novo caminho ótimo, mas dessa vez sem a presença do sítio bloqueado. Quando encontrarmos o novo caminho, novamente bloqueamos o sítio com maior energia do caminho. Esse processo se repete até que tenham sido bloqueados sítios suficientes para que não haja mais caminhos conectando a parte inferior à parte superior da rede. A Figura 8 mostra um exemplo de uma realização do algoritmo de fraturas de caminhos ótimos para uma rede quadrada de tamanho $L = 512$, com energias atribuídas de acordo com a distribuição 3.1 e parâmetro de desordem $\beta = 0.002$.

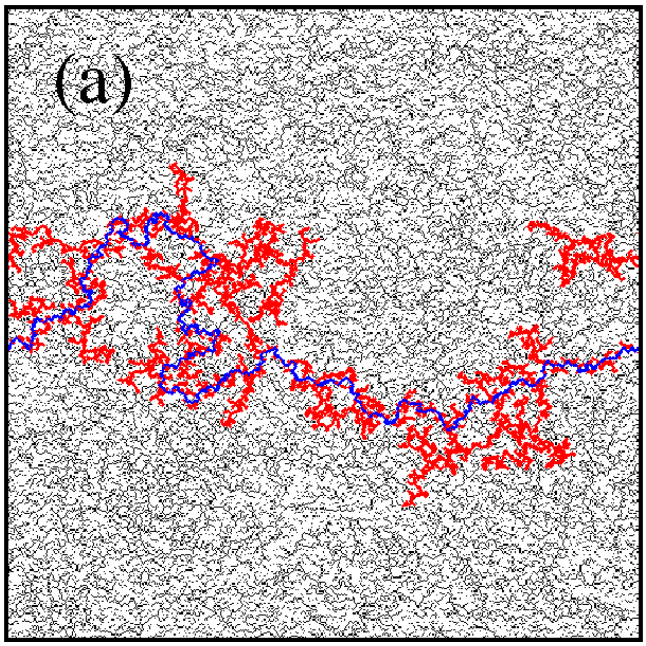


Figura 8 – Sítios bloqueados pelo algoritmo de fraturas de caminhos ótimos em meio com desordem fraca ($\beta = 0.002$).
Fonte: (JR *et al.*, 2009)

A Figura 9 demonstra o comportamento do algoritmo para a mesma rede com pesos retirados de uma distribuição com $\beta = 6.0$, isto é, em um meio desordenado. Em ambas as Figuras vemos que os sítios bloqueados formam uma fratura percolante, ou seja, um agregado de sítios que quebra a conectividade do sistema. Os sítios bloqueados formam um conjunto que pode ser dividido em três partes: a fratura percolante, o esqueleto da fratura percolante, que é o menor caminho da fratura que desconecta a rede (indicada em azul), e os agregados isolados (indicados em preto), que são os pequenos grupos de sítios bloqueados que não estão ligados ao maior agregado (JR *et al.*, 2009).

O comportamento observado nas Figuras 8 e 9 indica uma tendência com o aumento da desordem. A medida que aumentamos o valor de β o agregado percolante tende a diminuir e para valores muito altos, apenas o esqueleto indicado em azul permanece. Além disso, esse esqueleto é o mesmo para todos os valores de β (JR *et al.*, 2009).

A Figura 10 mostra a dependência da massa do esqueleto do OPC no caso de desordem fraca. Podemos perceber que a massa do esqueleto obedece a uma lei de potência da forma $M_b \cong L^{D_b}$, com $D_b \cong 1.22 \pm 0.02$. Isso significa que para o caso de desordem fraca, o esqueleto do OPC tem uma dimensão fractal idêntica a do caminho ótimo em um meio fortemente desordenado. Esse valor é o mesmo para a dimensão de caminhos em árvores geradores mínimas, de *strands* em percolação invasiva (DOBRIN; DUXBURY, 2001; CIEPLAK *et al.*, 1996).

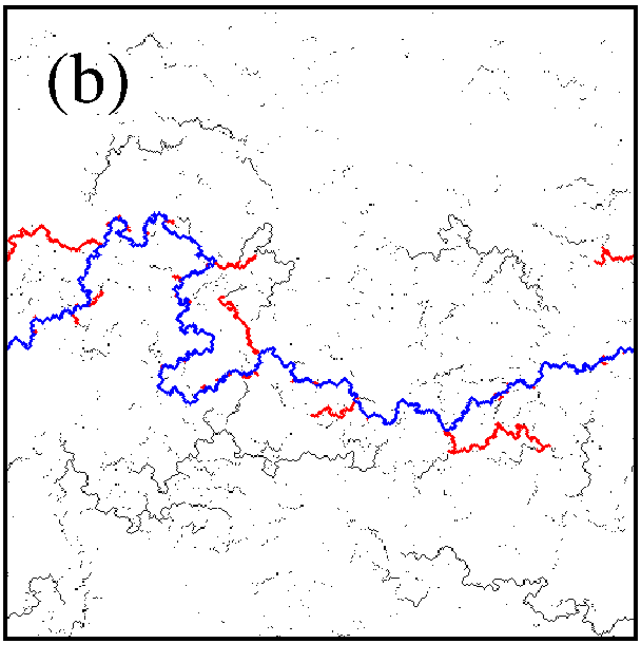


Figura 9 – Sítios bloqueados pelo algoritmo de fraturas de caminhos ótimos em meio com desordem forte ($\beta = 6.0$) .
Fonte: (JR *et al.*, 2009)

A Figura 10 também indica também um comportamento exponencial para a massa da fratura e para a massa total dos sítios bloqueados. A fratura obedece a uma lei do tipo $M_f \cong L^{D_f}$, com $D_f \cong 1.59 \pm 0.02$, enquanto a massa total é constante e tem a a forma $M_t \cong L^{D_t}$, com $D_f \cong 2.00 \pm 0.02$.

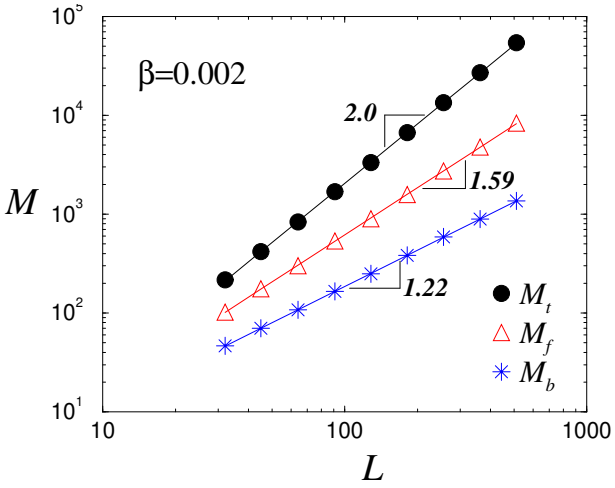


Figura 10 – Dependência logarítmica da massa dos sítios bloqueados com o tamanho da rede para o caso de desordem fraca. M_t é a massa total dos sítios bloqueados, M_f é a massa da fratura e M_b é a massa do esqueleto da fratura .
Fonte: (JR *et al.*, 2009)

Já a Figura 11 mostra a dependência da massa com o tamanho da rede para um meio fortemente desordenado. Pela análise do gráfico, é possível identificar que à medida que aumentamos o tamanho da rede o sistema sofre uma transição de desordem forte para desordem fraca, indicando que o parâmetro β não é o único regulador da desordem do sistema e que o comprimento L da rede também tem influência.

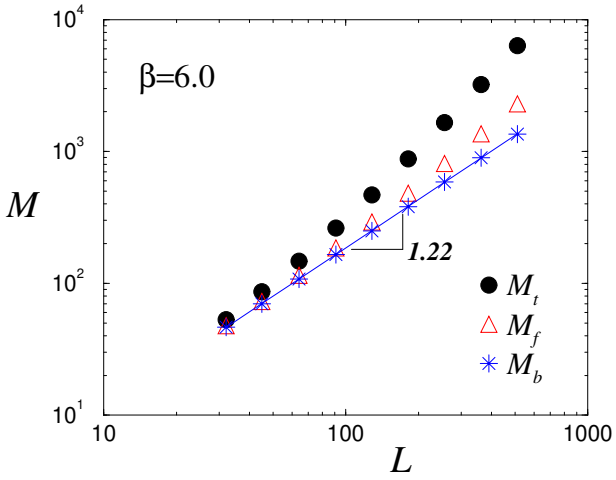


Figura 11 – Dependência logarítmica da massa dos sítios bloqueados com o tamanho da rede para o caso de desordem forte. M_t é a massa total dos sítios bloqueados, M_f é a massa da fratura e M_b é a massa do esqueleto da fratura
Fonte: (JR *et al.*, 2009).

3.2 Aplicações em Redes de Trânsito

Assim como para o caso da seção anterior, de meios desordenados, o algoritmo de fraturas de caminhos ótimos pode ser utilizado para compreender alguns aspectos de redes de trânsito. Os caminhos ótimos que ligam dois pontos de uma cidade são amplamente utilizados, principalmente em horários de pico, e, portanto, imagina-se que pontos pertencentes a caminhos ótimos sejam os mais vulneráveis à formação de congestionamentos.

Para entender o funcionamento do algoritmo aplicado à cidades, imaginemos o deslocamento de veículos entre bairros de uma região. Se um grande número de veículos tentar se deslocar utilizando o caminho ótimo, em determinado momento um engarrafamento será formado. Nesse momento, imediatamente alguns motoristas tentarão buscar outro caminho que os leve ao seu destino. Podemos entender isso como um processo de microfraturas apresentado na seção anterior e dizer que o sítio equivalente ao ponto em que o engarrafamento começou foi bloqueado na rede.

O algoritmo de fraturas de caminhos ótimos foi aplicado para o centro de Boston e para Manhattan (CARMONA *et al.*, 2020). Para aplicar o algoritmo a essas cidades, é necessário criar uma rede a partir dos mapas, onde cada cruzamento entre vias é um sítio da rede e as vias em si são as ligações. Os pesos atribuídos às ligações são dados por t/l, em que t é o tempo médio de viagem por via e l é o comprimento do trecho. É importante ressaltar que esse tempo deve ser relativo a um horário fora do horário de pico, pois o algoritmo busca refletir o estado das ruas livres de trânsito.

Uma vez construída a rede e atribuídos os pesos às ligações, partimos então para a aplicação do algoritmo. Inicialmente, um ponto geográfico é escolhido aleatoriamente dentro da região e o sítio mais próximo desse ponto é tomado como origem dos caminhos. Então, dado um raio L, construímos um círculo centrado no sítio de origem e então selecionamos um ponto sob o círculo. O sítio mais próximo ao segundo ponto selecionado é tomado como destino do caminho, desde que a distância entre origem e destino esteja dentro de um intervalo de tolerância de 5% de L. Esse valor de tolerância é encontrado experimentalmente e tem como função garantir que se encontre um sítio de destino, uma vez que não necessariamente existirá um sítio na rede que esteja a exatamente L metros de distância da origem.

Uma vez escolhido o caminho ótimo, que minimiza a soma de t/l ao longo do caminho, o sítio com o maior valor de t/l é bloqueado. O algoritmo para quando não for possível encontrar um caminho ligando origem e destino. No total, foram realizadas 2000 iterações para diferentes pares origem-destino com o intuito de obter uma média dos resultados que seja significativa e não devida apenas a fatores aleatórios do algoritmo.

A Figura 12 mostra como o valor médio de sítios removidos N_r varia com o tamanho do raio L escolhido. A Figura também mostra os resultados obtidos da aplicação do OPC para redes construídas em Boston e Manhattan preservando a geometria da rede, mas permutando os valores de t/l. Quanto menor o valor de N_r, menos sítios são necessários remover para quebrar a conectividade da rede e portanto menos resistente a falhas ela é. Podemos ver então, que Boston é significativamente mais suscetível a falhas do que Manhattan, independentemente do tamanho do raio L escolhido, como era esperado.

Além disso, sendo λ_i o comprimento l do segmento de rua removido na iteração i do OPC, podemos definir uma quantidade Λ, dada por:

$$\Lambda = \sum_{i=1}^{N_r} \lambda_i, \tag{3.8}$$

onde N_r é o número médio de vias removidas em uma realização do OPC. Como mostra o

gráfico inserido na Figura 12, o parâmetro Λ varia linearmente com N_r, tanto para Boston ($\langle\Lambda\rangle = 98.9 \cdot N_r$), como para Manhattan($\langle\Lambda\rangle = 129.8 \cdot N_r$).

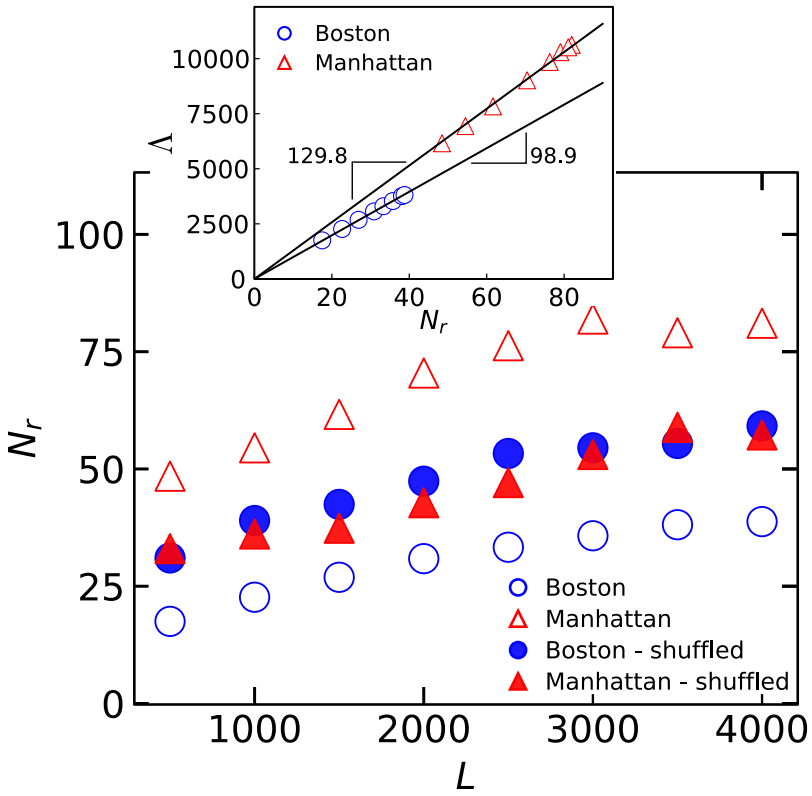
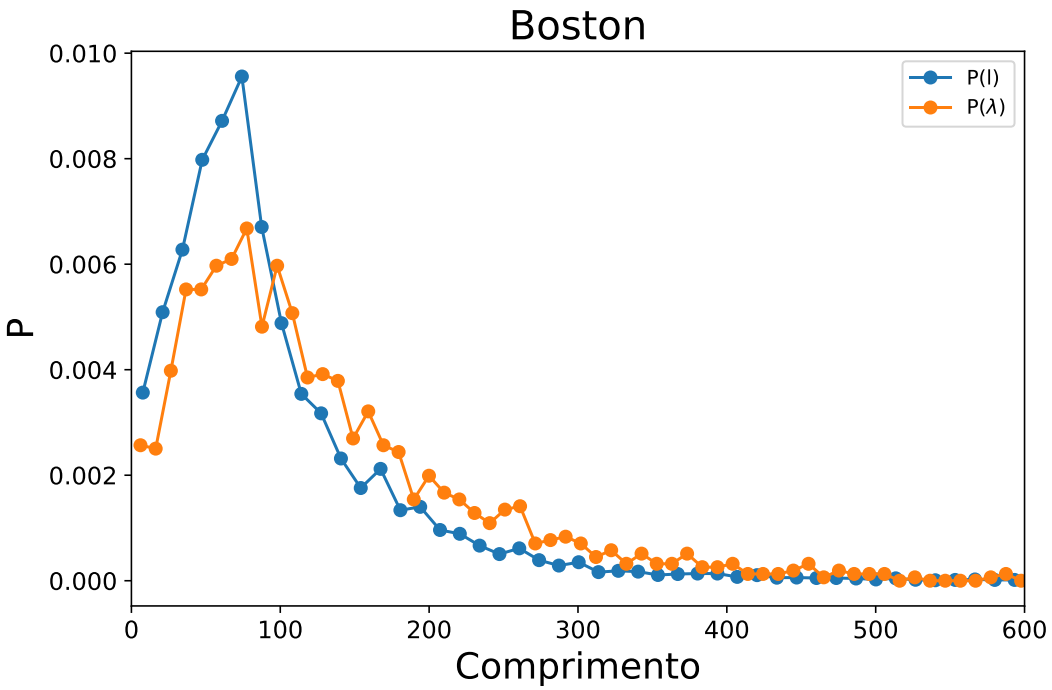
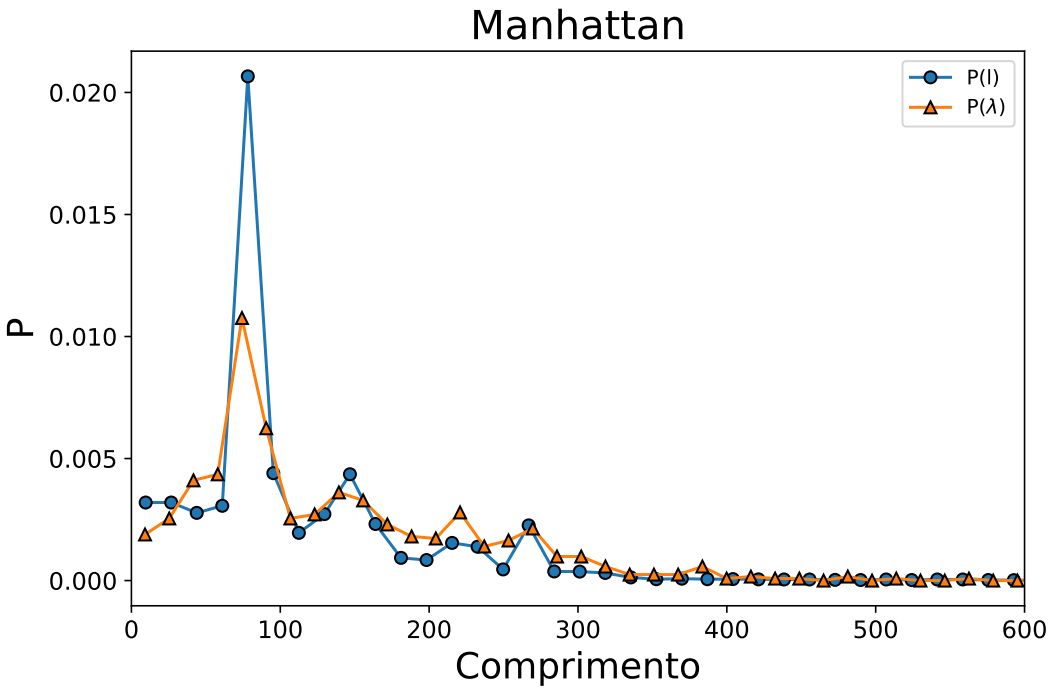


Figura 12 – Dependência do número total de segmentos removidos N_r pela distância entre origem e destino L. O outro gráfico inserido representa a dependência da soma total dos comprimentos removidos no OPC com o número médio de segmentos removidos.
Fonte: (CARMONA *et al.*, 2020)

O comportamento similar de Λ para as duas cidades indica que o OPC não é afetado por correlações espaciais na seleção dos comprimentos a serem removidos. A Figura 13 mostra a distribuição $P(l)$ dos comprimentos das vias das cidades e a distribuição $P(\lambda)$ dos comprimentos dos segmentos removidos durante a execução do algoritmo OPC. Podemos ver que para ambas as cidades as distribuições apresentam um comportamento semelhante. Isso mostra que durante o OPC, o valor λ é retirado de um intervalo que cobre todos os valores possíveis de l. Apesar da semelhança entre as distribuições, é possível notar que valores maiores de l são mais frequentemente escolhidos pelo algoritmo. No entanto, isso reflete apenas características locais da rede e aparece em ambas as cidades, não sendo portanto a causa da diferença entre os resultados do número médio de sítios removidos.



(a)

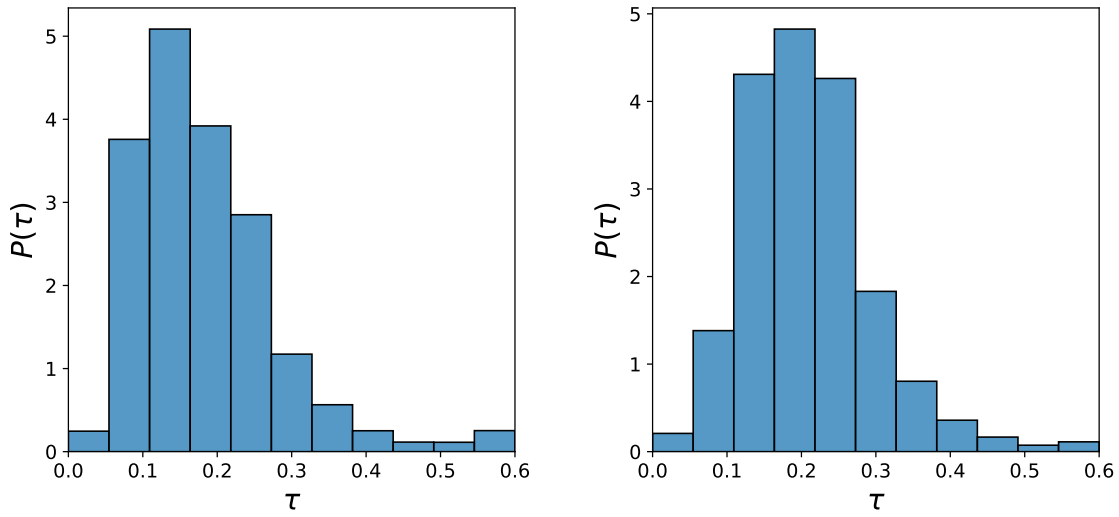


(b)

Figura 13 – Distribuições dos comprimentos das vias na cidade P(l) e dos comprimentos removidos durante o OPC P(λ). (a) Distribuição para Boston. (b) Distribuição para Manhattan

Em busca da origem das diferenças entre as susceptibilidades à falhas das redes, podemos analisar a distribuição do tempo médio de viagem por comprimento dos segmentos das cidades. Como podemos ver na Figura 14, ambas as distribuições seguem um comportamento

muito similar, de modo que essa também não aparenta ser a causa das diferenças obtidas.



(a) (b)

Figura 14 – Distribuição do parâmetro $\tau = t/l$ dos segmentos de vias para as cidades de (a) Boston e (b) Manhattan.

Por último, observando novamente a Figura 12, podemos analisar os resultados obtidos para o número médio de nós removidos quando os valores de t/l são trocados aleatoriamente entre as vias. O valor de N_r para Boston cresce e o de Manhattan decresce, ficando praticamente idênticos. Isso mostra que removendo as correlações espaciais, a diferença identificada pelo OPC para as cidades praticamente desaparece.

4 RESULTADOS

A princípio, tentamos reproduzir os resultados da literatura para as cidades de Boston e Manhattan. Para isso, obtivemos os mapas das duas cidades através do *site OpenStreetMap* (OpenStreetMap, 2017). Utilizando a linguagem de programação Python para manipular os dados, montamos uma rede sobre o mapa, em que cada encontro entre vias corresponde a um sítio.

Para atribuir pesos às ligações, utilizamos dados do site *Google Directions* (GOOGLE, 2022). Primeiramente, obtivemos o valor dos tempos que se leva em média para percorrer cada trecho de via presente na rede. O mesmo procedimento foi feito para uma porção da cidade de Fortaleza, contendo bairros como Aldeota, Meireles, Montese, entre outros. A Figura 15 mostra um mapa da região de Fortaleza cujas vias serão analisadas.

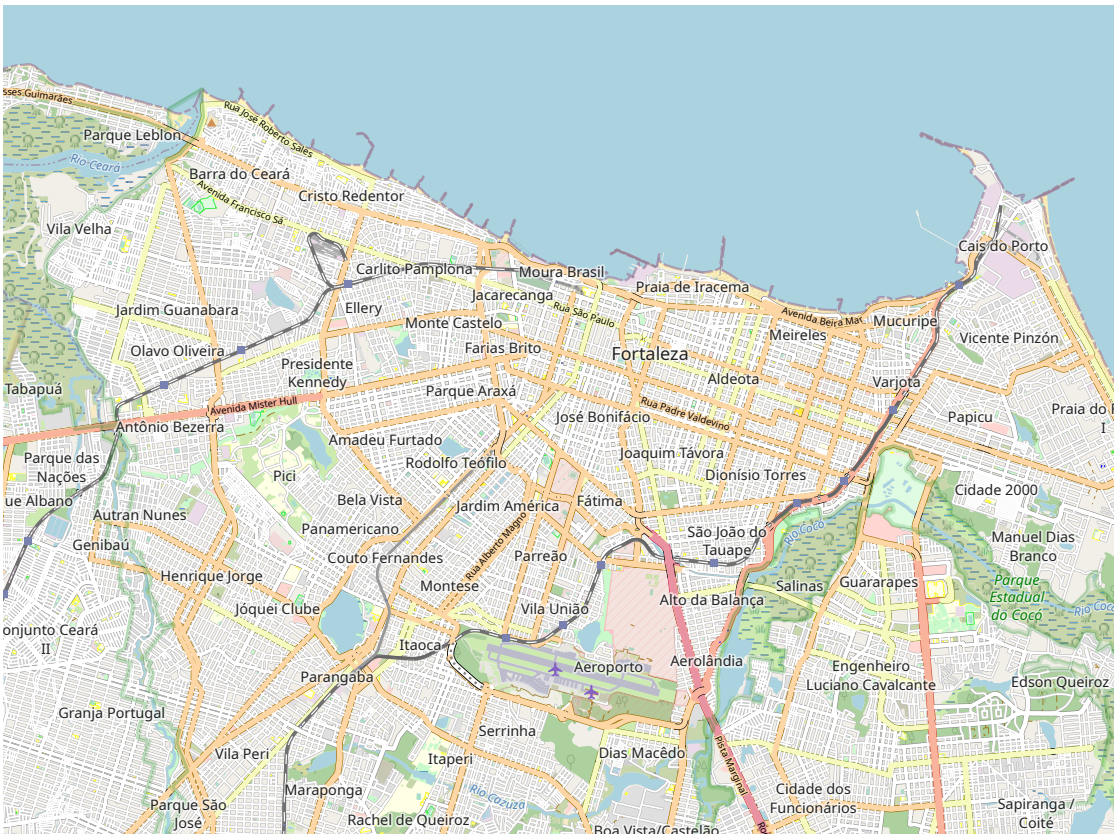


Figura 15 – Mapa do trecho de Fortaleza utilizado para montar a rede, englobando bairros da cidade como Aldeota, Meireles, Montese, Farias Brito entre outros.

A partir das redes montadas para esses 3 locais, atribuímos a cada trecho um peso igual a t/l, onde t é o tempo médio levado para percorrer o trecho e l é o comprimento do trecho. Uma vez criadas as redes para as cidades, foram implementados os algoritmos como explicado

nos capítulos anteriores. O primeiro resultado a ser reproduzido foi o do número médio de segmentos removidos N_r em função do raio L de distância entre origem e destino.

A Figura 16 mostra os resultados obtidos para as três cidades. O comportamento segue a mesma tendência da Figura 12, mas com valores numéricos diferentes. Essa diferença pode ser atribuída ao fator de aleatoriedade presente na escolha dos pares origem-destino durante a execução do OPC, uma vez que o comportamento observado segue coerente com a literatura.

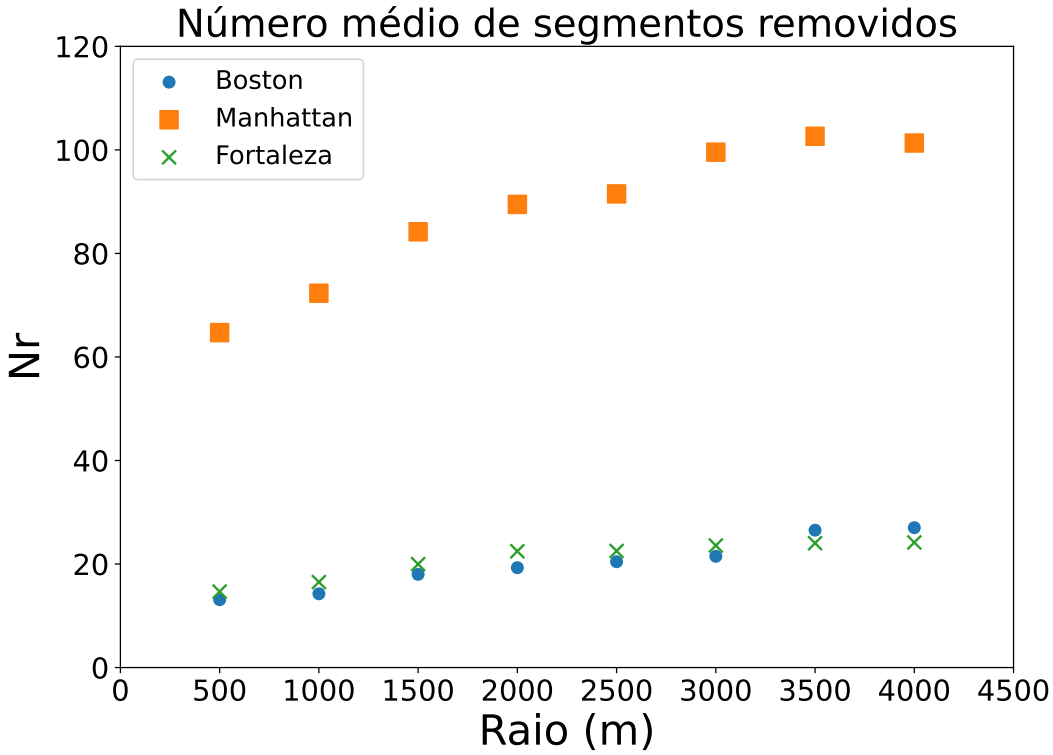


Figura 16 – Gráfico do número médio de segmentos removidos durante a execução do OPC em função do raio utilizado como distância entre origem e destino.

Além disso, a Figura 12 também mostra um comportamento muito semelhante entre Boston e Fortaleza. Para investigar essa relação, é preciso analisar outras propriedades das redes e dos resultados obtidos pelo OPC. Um fator a se considerar é o tamanho das redes montadas: enquanto a rede de Manhattan apresenta 10466 segmentos e 4867 nós, Boston tem 24840 segmentos e 11134 nós e Fortaleza tem 28221 segmentos e 11690 nós. Desse modo, em termos de tamanho da rede, as duas últimas apresentam números bastante próximos.

Em seguida, podemos observar o comportamento do parâmetro Λ, definido como a soma dos comprimentos de todos os segmentos bloqueados durante as execuções do OPC ($\Lambda = \sum_i \lambda_i$). A Figura 17 mostra a relação entre Λ e o número médio de segmentos bloqueados, N_r.

Como podemos observar, nas três redes temos uma relação da forma $\langle\Lambda\rangle = a \cdot N_r$. Comparando os resultados com os da Figura 12, vemos que Boston apresentou um resultado idêntico, enquanto Manhattan teve uma diferença de 1,8%, que pode novamente ser considerada devida à escolha aleatória de pares origem e destino durante o algoritmo.

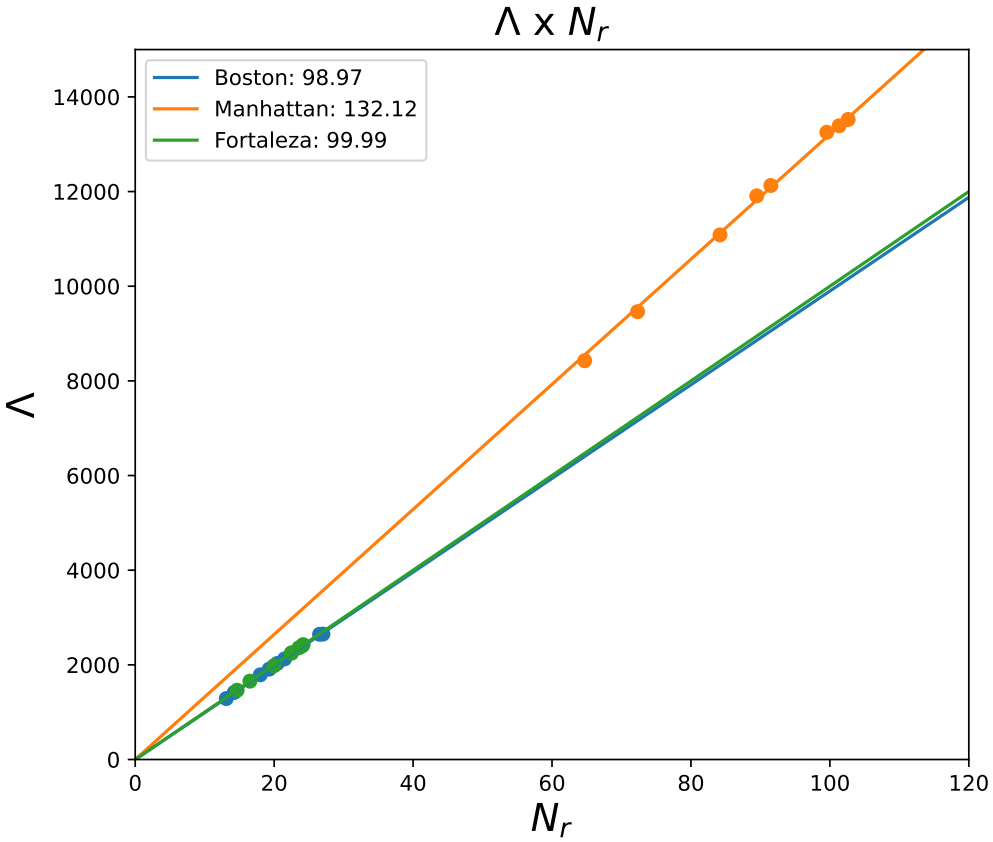


Figura 17 – Gráfico do parâmetro *Lambda* em função do número médio de segmentos removidos N_r. Manhattan, identificada em laranja, apresentou um comportamento da forma $\langle\Lambda\rangle = 132,12 \cdot N_r$, enquanto Boston teve como resultado a reta$\langle\Lambda\rangle = 98.97 \cdot N_r$. Já Fortaleza, no trecho estudado teve uma reta com coeficiente angular 99.99, apenas 1% diferente de Boston.

Concluímos, portanto, que o algoritmo implementado é capaz de reproduzir os resultados presentes na literatura. Além disso, assim como no resultado do gráfico do número de segmentos bloqueados em função do raio mostrado na figura 16, vemos que a rede do trecho da cidade de Fortaleza apresenta um comportamento bastante semelhante ao de Boston. Isso era esperado, visto que o parâmetro Λ tem um comportamento linear com N_r.

Outra característica da rede que pode ser observada é as distribuições dos comprimentos de segmentos ao longo da rede. Na Figura 18 em azul temos a distribuição $P(l)$ de todos os comprimentos da rede do trecho em questão de Fortaleza, enquanto em laranja temos a distribuição $P(\lambda)$ dos comprimentos bloqueados durante o OPC.

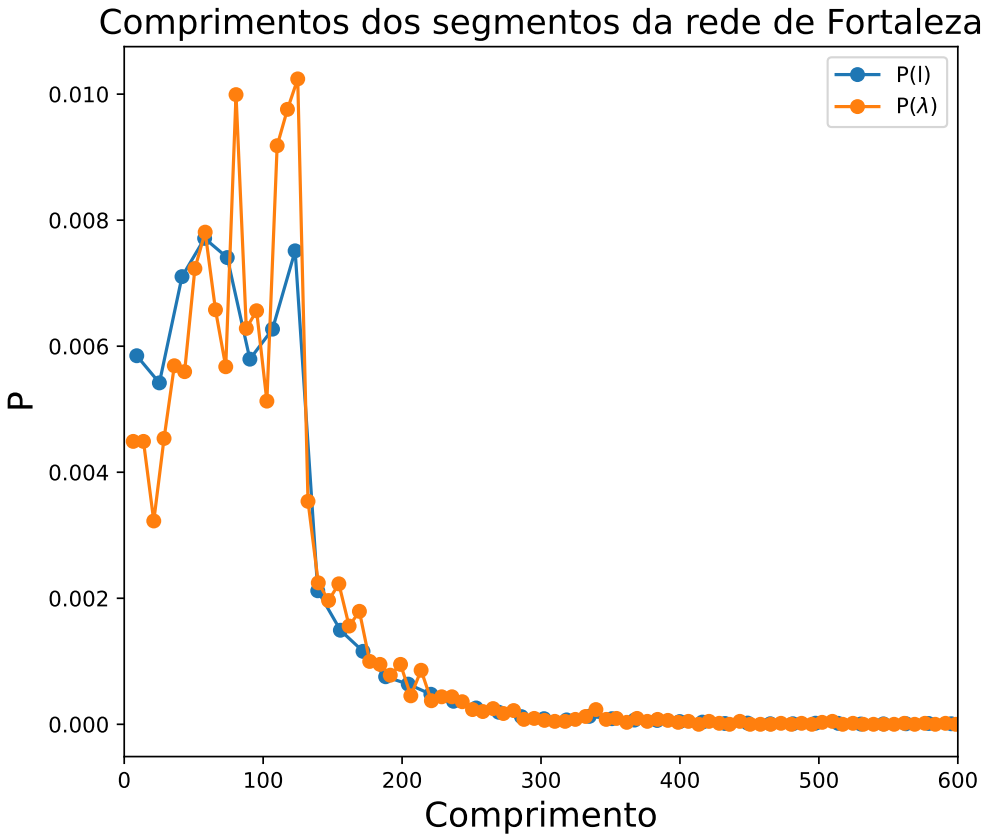


Figura 18 – Em azul, temos o gráfico da distribuição $P(l)$ de todos os comprimentos para a rede do trecho em questão de Fortaleza. Em laranja temos a distribuição $P(\lambda)$ dos trechos que foram bloqueados em algum momento durante a execução do OPC.

Quando comparamos a Figura 18 com a Figura 13, vemos o comportamento das distribuições dos comprimentos de Fortaleza, não segue o mesmo padrão de Boston nem de Manhattan. Enquanto Boston tem picos para comprimentos menos e depois decai quase continuamente, Manhattan apresenta vários picos ao longo dos comprimentos. Para Fortaleza, vemos que a distribuição tem um comportamento intermediário, apresentando alguns picos para comprimentos menores e decaindo continuamente na sequência. Isso tem origem, provavelmente, na geometria das cidades. Enquanto Manhattan apresenta uma estrutura bem definida, com quarteirões quadrados e comprimentos característicos, Boston tem uma geometria menos organizada, com formatos retangulares ou irregulares e comprimentos variados, e a porção escolhida de

Fortaleza apresenta áreas mais parecidas com Boston em relação à estrutura.

Por último, podemos analisar quais pontos são os mais vulneráveis à formação de congestionamentos. A Figura 19 mostra a rede montada para o trecho selecionado de Fortaleza. Os segmentos destacados em vermelho são aqueles que foram os primeiros bloqueados em cada iteração do OPC executada com um raio de 2.000 metros. Esses segmentos representam 3,7% do total de segmentos da rede montada. Na literatura, os valores encontrados para Boston e Manhattan são 5,4% e 10,0%, respectivamente, do total de segmentos da rede (CARMONA *et al.*, 2020). Esse resultado indica que, para o trecho estudado de Fortaleza, os pontos mais frequentemente associados a falhas na rede estão concentrados em um número baixo de vias. Desse modo, intervenções feitas nesses trechos podem ser mais eficazes para aumentar a resiliência da rede, isto é, aumentar a capacidade da rede de manter sua conectividade quando segmentos são bloqueados ao longo de caminhos ótimos.

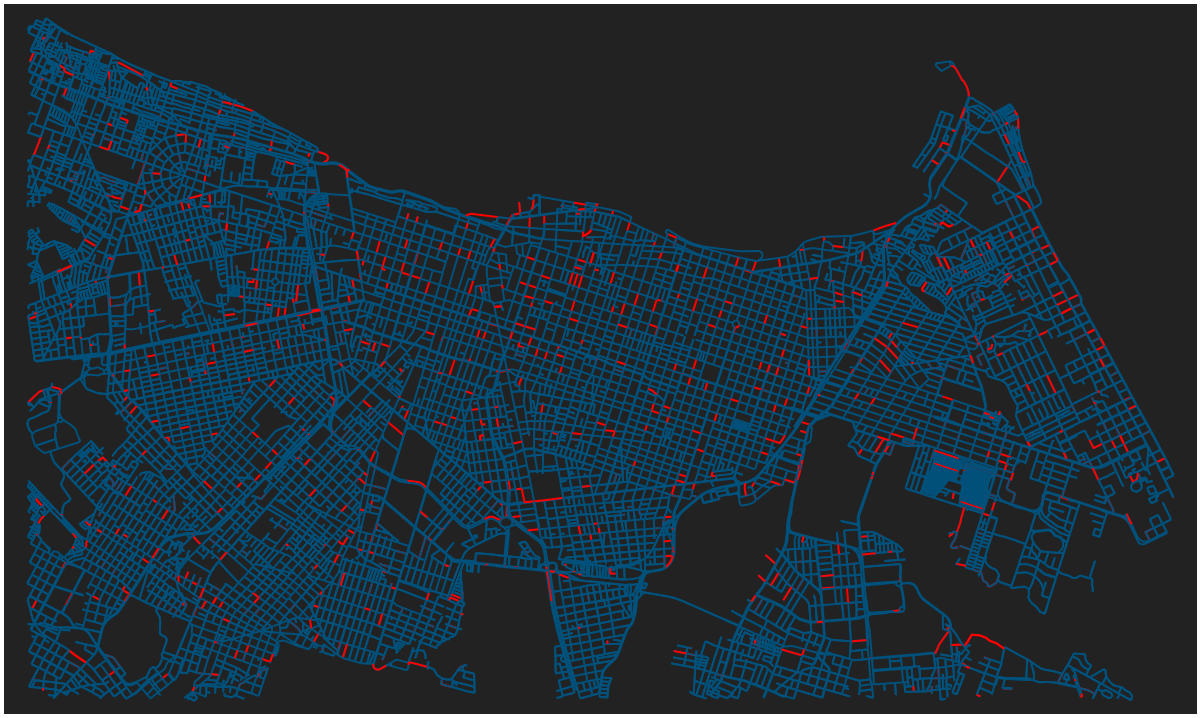


Figura 19 – Rede de Fortaleza com todos os segmentos que foram os primeiros removidos ao longo das 2000 execuções considerando um raio L = 2000m como distância entre os pares origem e destino. Nessa rede, os trechos removidos primeiro representam 3,7% do total de vias.

5 CONCLUSÕES E TRABALHOS FUTUROS

Foi feita uma análise de uma porção do mapa de Fortaleza, contendo bairros como Aldeota, Meireles, Parangaba Farias Brito, Montese, entre outros. Os resultados obtidos apresentaram grande semelhança à Boston quanto ao número médio de segmentos bloqueados que são necessários para quebrar a conectividade da rede. Foi mostrado que essa porção de Fortaleza tem uma distribuição de segmentos semelhante à rede de Boston, o que é uma das possíveis causas para a semelhança nos resultados.

A partir do número de segmentos que foram os primeiros removidos no OPC, podemos concluir que os trechos vulneráveis à falhas na rede estão concentrados em uma pequena parte do total e que, portanto, uma intervenção nesses trechos pode ser altamente efetiva. O trecho escolhido de Fortaleza tinha uma rede de tamanho semelhante à rede de Boston, fato que pode ter acarretado na semelhança encontrada. No entanto, para analisar completamente a influência desse fator é necessário que tenhamos dados da rede completa de Fortaleza e de mais cidades. A partir daí poderemos organizar as cidades em um ranking de resiliência das redes de trânsito, além de identificar quais são os pontos mais vulneráveis em cada uma delas.

Portanto, para futuros desenvolvimentos precisamos analisar outras redes, com características diversas, a fim de analisar com mais profundidade quais fatores influenciam no algoritmo. A obtenção de dados de outras cidades pode ser feita a utilizando a técnica de *Web Scraping*, extraindo as informações necessárias de sites especializados em dados de trânsito como o *Google Maps* ou o *Bing Maps*. Outra alternativa, é utilizar soluções fornecidas pelos próprios sites que detém os dados. Através do site *Google Directions*, podemos fazer requisições do tempo médio de viagem em um determinado trajeto. No entanto, para que isso seja viável é preciso desenvolver um algoritmo que retorne uma lista de caminhos ao longo da rede de forma que todo trecho da cidade esteja em um dos caminhos da lista, para garantir que não ficaremos com nenhum trecho sem informação do tempo de viagem. Para isso, devemos testar adaptações para algoritmos como *Breadth-first search (BFS)* e *Depth-first search (DFS)*.

Por último, também podemos testar modificações no modelo, além de verificar se o resultado obtido para Boston e Manhattan de que ao se retirar as correlações espaciais das redes o número médio de segmentos removidos tende a ser o mesmo se mantém para as demais cidades, indicando que as correlações espaciais são de fato a causa principal do aumento da vulnerabilidade de uma rede de trânsito.

REFERÊNCIAS

ALFRED, V. A.; JOHN, E. H.; JEFFREY, D. U.; ALFRED, V. A.; GLENN, H. B.; KENNETH, D. H.; JULIAN, C. S.; JEAN-PIERRE, B.; SAMLER, B. A.; PETER, B. A. *et al.* **Data structures and algorithms**. [*S. l.*]: USA: Addison-Wesley, 1983.

ANSARI, A.; BERENDZEN, J.; BOWNE, S. F.; FRAUENFELDER, H.; IBEN, I.; SAUKE, T. B.; SHYAMSUNDER, E.; YOUNG, R. D. Protein states and proteinquakes. **Proceedings of the National Academy of Sciences**, National Acad Sciences, v. 82, n. 15, p. 5000–5004, 1985.

BARABÁSI, A.-L. Network science. **Philosophical Transactions of the Royal Society A**, The Royal Society Publishing, v. 371, n. 1987, p. 20120375, 2013.

BELLMAN, R. On a routing problem. **Quarterly of applied mathematics**, v. 16, n. 1, p. 87–90, 1958.

BETTENCOURT, L.; LOBO, J.; YOUN, H. The hypothesis of urban scaling: formalization, implications and challenges. **arXiv preprint arXiv:1301.5919**, 2013.

BIGGS, N.; LLOYD, E. K.; WILSON, R. J. **Graph Theory, 1736-1936**. [*S. l.*]: Oxford University Press, 1986.

CARMONA, H.; NORONHA, A. de; MOREIRA, A.; ARAÚJO, N.; JR, J. A. Cracking urban mobility. **Physical Review Research**, APS, v. 2, n. 4, p. 043132, 2020.

CHENEY, C. J. A nonrecursive list compacting algorithm. **Communications of the ACM**, ACM New York, NY, USA, v. 13, n. 11, p. 677–678, 1970.

CIEPLAK, M.; MARITAN, A.; BANAVAR, J. R. Invasion percolation and eden growth: Geometry and universality. **Phys. Rev. Lett.**, American Physical Society, v. 76, p. 3754–3757, May 1996.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. **Introduction to algorithms**. [*S. l.*]: MIT press, 2022.

DETLEFSEN, N. S.; HAUBERG, S.; BOOMSMA, W. Learning meaningful representations of protein sequences. **Nature communications**, Nature Publishing Group, v. 13, n. 1, p. 1–12, 2022.

DIJKSTRA, E. W. *et al.* A note on two problems in connexion with graphs. **Numerische mathematik**, v. 1, n. 1, p. 269–271, 1959.

DOBRIN, R.; DUXBURY, P. M. Minimum spanning trees on random networks. **Phys. Rev. Lett.**, American Physical Society, v. 86, p. 5076–5079, May 2001.

EASLEY, D.; KLEINBERG, J. **Networks, crowds, and markets**: reasoning about a highly connected world. [*S. l.*]: Cambridge university press, 2010.

FEDER, J. **Fractals**. [*S. l.*]: Springer Science & Business Media, 2013.

FORD, L. R.; FULKERSON, D. R. Maximal flow through a network. **Canadian journal of Mathematics**, Cambridge University Press, v. 8, p. 399–404, 1956.

GOOGLE. **Google Maps**. Google, 2022. Disponível em: https://developers.google.com/maps/documentation/directions. Acesso em: 14 jul. 2022.

JR, J. S. A.; OLIVEIRA, E.; MOREIRA, A.; HERRMANN, H. J. Fracturing the optimal paths. **Physical review letters**, APS, v. 103, n. 22, p. 225503, 2009.

JR, L. R. F. **Network flow theory**. [*S. l.*], 1956.

KIRKPATRICK, S.; TOULOUSE, G. Configuration space analysis of travelling salesman problems. **Journal de Physique**, Société française de physique, v. 46, n. 8, p. 1277–1292, 1985.

LI, F.-F.; DU, Y.; JIA, K.-J. Path planning and smoothing of mobile robot based on improved artificial fish swarm algorithm. **Scientific Reports**, Nature Publishing Group, v. 12, n. 1, p. 1–16, 2022.

LOUF, R.; BARTHELEMY, M. How congestion shapes cities: from mobility patterns to scaling. **Scientific reports**, Nature Publishing Group, v. 4, n. 1, p. 1–9, 2014.

MÉZARD, M.; PARISI, G.; SOURLAS, N.; TOULOUSE, G.; VIRASORO, M. Nature of the spin-glass phase. **Physical review letters**, APS, v. 52, n. 13, p. 1156, 1984.

NEWMAN, M. **Networks**. [*S. l.*]: Oxford university press, 2018.

OLIVEIRA, E. A. d. Linha divisórias de águas e fraturas de caminhos ótimos em meios desordenados. 2012.

OpenStreetMap. **Planet dump retrieved from https://planet.osm.org** . 2017. Disponível em: https://www.openstreetmap.org. Acesso em: 14 jul. 2022.

RAHMAN, M.; VADREV, S. M.; MAGANA-MORA, A.; LEVMAN, J.; SOUFAN, O. *et al.* A novel graph mining approach to predict and evaluate food-drug interactions. **Scientific Reports**, Nature Publishing Group, v. 12, n. 1, p. 1–16, 2022.

REED, T. Inrix global traffic scorecard. IRIX Research, 2019.

SWEET, M. Traffic congestion's economic impacts: evidence from us metropolitan regions. **Urban Studies**, Sage Publications Sage UK: London, England, v. 51, n. 10, p. 2088–2110, 2014.

TOMTOM. **Traffic congestion ranking**. TomTom, 2022. Disponível em: https://www.tomtom.com/en_gb/traffic-index/ranking/. Acesso em: 14 jul. 2022.