PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Lucas Angel Larios Prado

Navegação inteligente com ênfase na segurança do usuário

Projeto Final

Relatório de Projeto Final, apresentado ao programa de Ciência da Computação da PUC-Rio como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Augusto Cesar Espíndola Baffa

Rio de Janeiro, dezembro de 2024

PONTIFÍCIA UNIVERSIDADE CATÓLICA
DO RIO DE JANEIRO

Lucas Angel Larios Prado

Navegação inteligente com ênfase na segurança do usuário

Relatório de Projeto Final, apresentado ao programa de Ciência da Computação da PUC-RIO como requisito parcial para obtenção do título de Bacharel em Ciência da Computação. Aprovada pela Comissão Examinadora abaixo:

Prof. Augusto Cesar Espíndola Baffa
Orientador
Departamento de Informática – PUC-Rio

Rio de Janeiro, dezembro de 2024

Lucas Angel Larios Prado

Ficha Catalográfica

Larios, Lucas

Navegação inteligente com ênfase na segurança do usuário / Lucas Angel Larios Prado; orientador: Augusto Baffa. – 2024.

50 f: il. color. ; 30 cm

Relatório (Projeto Final de Graduação) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Informática – Teses. 2. GPS. 3. Algoritmos heurísticos. 4. VNS. 5. Segurança Urbana. 6. Otimização de Rotas. 7. Algoritmo A*. 8. Geoestatística. 9. Krigagem. 10. Índice de Insegurança. 11. Teoria dos Grafos. 12. Sistemas de Navegação.
I. Baffa, Augusto. II. Pontifícia Universidade Católica do Rio

CDD: 004

Agradecimentos

A conclusão deste trabalho não seria possível sem o apoio de diversas pessoas, às quais expresso minha mais profunda gratidão.

Agradeço, em primeiro lugar, à minha família, pelo suporte incondicional, pela paciência e pelo incentivo constante durante todas as etapas deste projeto. Vocês foram minha fonte de motivação e equilíbrio nos momentos mais desafiadores.

Ao meu orientador, Augusto Baffa, pela orientação técnica, pelas valiosas contribuições e pela confiança depositada em mim durante o desenvolvimento deste trabalho. Sua experiência e dedicação foram fundamentais para o alcance dos objetivos deste projeto.

Aos professores da Pontifícia Universidade Católica do Rio de Janeiro, pelos conhecimentos compartilhados ao longo da minha trajetória acadêmica, que serviram de base para este estudo.

Por fim, aos meus amigos de infância e aos meus amigos de curso, pelo apoio, pelas discussões enriquecedoras e por compartilharem comigo os desafios e conquistas dessa jornada. A cada um de vocês, meu sincero agradecimento.

Resumo

Larios, Lucas; Baffa, Augusto. **Navegação inteligente com ênfase na segurança do usuário**. Rio de Janeiro, 2023. 50p. Relatório de Projeto Final – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Este projeto de ciências da computação aborda o desenvolvimento de um sistema de navegação veicular (VNS) com ênfase em segurança. Um sistema de navegação veicular é uma aplicação que visa resolver o problema de encontrar o caminho mais curto entre um ponto A e um ponto B em uma área geográfica com mobilidade automotiva. Os VNS atuais tendem a otimizar apenas a menor distância ou tempo de viagem, desconsiderando a segurança do trajeto escolhido. Em ambientes urbanos, essas soluções podem expor os usuários a situações perigosas. A falta de consideração pela segurança pode levar os usuários a seguir rotas que aumentam a vulnerabilidade a assaltos, furtos, violência e homicídios. Este projeto propõe e prototipa uma solução para esse problema utilizando o algoritmo A* e a base de dados da polícia do estado do Rio de Janeiro, buscando garantir que o caminho traçado seja não apenas o mais rápido, mas também o mais seguro.

Palavras-chave

GPS; Algoritmos heurísticos; VNS; Segurança Urbana; Otimização de Rotas; Algoritmo A*; Geoestatística; Krigagem; Índice de Insegurança; Teoria dos Grafos; Sistemas de Navegação;.

Abstract

Larios, Lucas; Baffa, Augusto (Advisor). **Navegação inteligente com ênfase na segurança do usuário**. Rio de Janeiro, 2023. 50p. Final Project Report - Department of Informatics. Pontifical Catholic University of Rio de Janeiro.

This computer science project addresses the development of a Vehicular Navigation System (VNS) with an emphasis on safety. A vehicular navigation system is an application that aims to solve the problem of finding the shortest path between point A and point B in a geographical area with automotive mobility. Current VNS tend to optimize only for the shortest distance or travel time, disregarding the safety of the chosen route. In urban environments, these solutions can expose users to dangerous situations. The lack of consideration for safety can lead users to follow routes that increase their vulnerability to assaults, thefts, violence, and homicides. This project proposes and prototypes a solution to this problem using the A* algorithm and the police database of the state of Rio de Janeiro, aiming to ensure that the mapped path is not only the fastest but also the safest.

Keywords

GPS; heuristic algorithm; VNS; A-Star Algorithm; Kriging interpolation; ITS; Graph Theory; Navigation Systems; Geostatistics; Insecurity Index; Urban Safety; Intelligent Transportation Systems; Route Optimization.

Sumário

1
Introdução

Um Sistema de navegação veicular(VNS) é uma ferramenta que tem como principal objetivo solucionar e otimizar o deslocamento de Veículos. No contexto automativo, esta tecnologia, traceja uma rota entre dois pontos ou mais, permitindo que o usuário consiga vizualizar um caminho.(CLAUSSEN, 1991)

A crescente popularidade desses sistemas tem revolucionado a mobilidade urbana ao proporcionar rotas otimizadas em termos de distância e tempo. No entanto, a ênfase desses sistemas na eficiência ignora um aspecto crucial: a segurança do usuário.

Em áreas urbanas, essa abordagem pode expor os usuários a perigos significativos, como assaltos, furtos e violência. Exemplos trágicos, como o incidente envolvendo um policial militar em Niterói que seguindo as instruções do gps, inadvertidamente adentrou uma comunidade perigosa e foi alvejado [1]. Da mesma forma, um casal que, por engano, entrou na comunidade Caramujo e foi recebido a tiros, resultando na trágica morte de um dos cônjuges [2]. Estes ilustram os riscos de seguir rotas sugeridas sem considerar a segurança.

Por este motivo, este projeto visa abordar essa lacuna ao desenvolver um sistema de navegação veicular que se preocupa com a segurança do usuário, utilizando soluções do clássico problema do caminho mais curto combinados com os dados de segurança dispobinibilizados pela polícia do estado do Rio de Janeiro. Desta forma, o presente projeto propõe uma evolução dos sistemas tradicionais, integrando dados de segurança no cálculo de rotas para garantir a proteção dos usuários.

Diante do exposto, torna-se fundamental analisar as soluções tecnológicas atuais que buscam resolver esse problema. Embora os sistemas atuais, como o Google Maps e o Waze, priorizem a otimização de tempo e distância, eles não contemplam a segurança dos usuários em áreas urbanas de risco.

No capítulo a seguir, será apresentada a situação atual dessas tecnologias, com foco nas iniciativas mais recentes da academia e da indústria. Serão discutidos os Sistemas de Transporte Inteligentes (ITS) e as Redes Veiculares Ad Hoc (VANETs), que oferecem uma nova perspectiva ao integrar dados de sensoriamento e comunicação em tempo real para a escolha de rotas mais eficientes e seguras.

[1]https://extra.globo.com/casos-de-policia/pm-de-folga-segue-gps-entra-em-favela-de-niteroi-por-engano-e-baleado-rv1-1-25672723.html

[2]https://g1.globo.com/rio-de-janeiro/noticia/2015/10/mulher-morre-apos-entrar-por-engano-em-comunidade-em-niteroi-rj.html

2
Situação atual

Esta seção busca mostrar o que existe em estado da arte para solucionar o problema proposto:

2.1
ITS e Redes Veiculares Ad Hoc

Sistemas de navegação veiculares frequentemente recomendam rotas baseadas a partir do conhecimento de tráfego global. Google Maps por exemplo, utiliza o algoritmo A* e técnicas de coleta de dados para inferir um conhecimento global do tráfego. No entanto VNS como a google e a waze, se limitam em apenas otimizar o tempo de deslocamento do usuário, negligenciado sua segurança.

A esfera acadêmica,sabendo desse problema, tem apresentado diversas temáticas que podem revolucionar a implementação de sistemas de navegação veiculares. Entre essas temáticas,tem se destacado o conceito de sistemas de trasporte inteligentes(ITS, do inglês Intelligent Transportation Systems), que pode ser considerado uma evolução das atuais VNS, onde este engloba uma variedade de mecanismos de sensoriamento e comunicação em tempo real, que contribuem com o sistema, para uma seleção mais eficiente de rotas. (VILLAS, 2018)

ITS se diferem dos VNS convencionais pois seu processo de inferência é oriundo de um conjunto de dados coletados de um ambiente urbano conectado. Um ambiente urbano conectado pode ser descrito como uma malha de disposivos conectados entre si, que se comunicam e trocam informações em tempo real, essa comunicação ocorre através de diversos sensores e objetos inteligentes que estarão presentes em cidades inteligentes. Esse mundo massivamente conectado, é permitido apenas pelo crescimento de áreas como Internet das coisas(IoT) e a de objetos inteligentes e trará um novo paradigma para sistemas que buscam entender de maneira aprofundada o comportamento e a complexidade do ambiente urbano, trazendo uma escolha de rotas mais eficiente e consequentemente mais seguro. Esse entendimento será alcançando pela enorme quantidade de dados gerados por esses sensores e dispositivos inteligentes, que são tecnologias advindas da área de redes, tais como as Redes Veiculares AD Hoc(Vanets - Vehicular ad-hoc networks)(GOMES, 2021).

Essa grande massa de dados permitirá que ITS tenham conhecimento sobre diversas variáveis de um ambiente urbano,como mobilidade,insegurança, pontos de interesse, níveis de poluição, etc.

Visto a importância das Redes Veiculares Ad Hoc(vanets) para a compreensão de ambientes urbanos, precisamos compreender seus diferentes módulos de comunicação. O primeiro é o V2V (Veículo para Veículo), no qual os veículos em ambientes urbanos trocam informações entre si. O segundo é o V2P (Veículo para Pedestre), onde os automóveis se conectam com dispositivos inteligentes dos pedestres, como smartphones e wearables. Já o V2I (Veículo para Infraestrutura) permite que os veículos se comuniquem com sensores e dispositivos da

cidade, como radares e semáforos. Por fim, o V2N (Veículo para Servidor) conecta os veículos a servidores remotos, possibilitando, por exemplo, o acesso a dados de segurança, como boletins de ocorrência disponibilizados pela polícia.

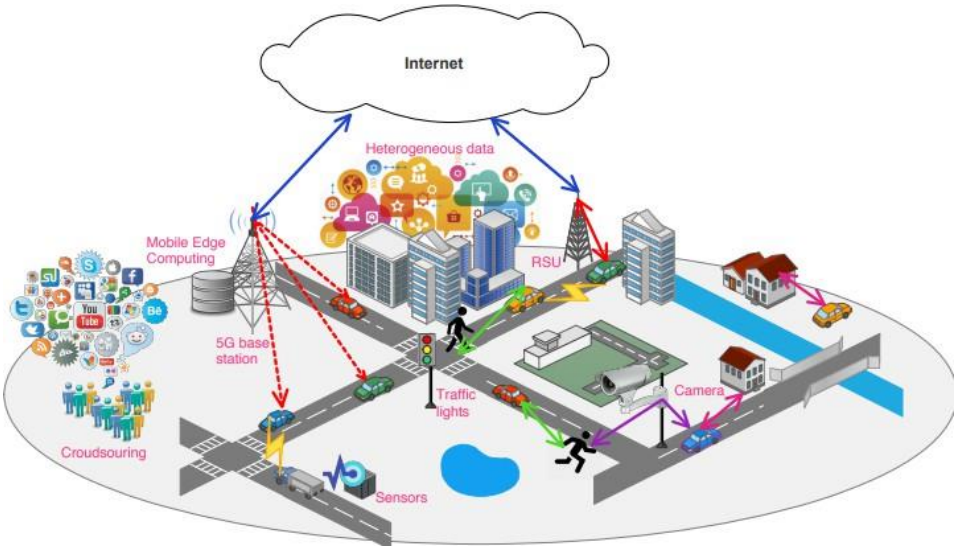


Figura 2.1: ITS landscape (SOUZA, 2021)

Essa comunicação em escala permitiria uma boa vizualização das diferentes variáveis do trânsito. Usando esses dados junto a machine learning, poderiamos predizer rotas seguras e rápidas eficientemente.

Embora promissor, o conceito de VANETs é algo novo e ainda não foi largamente implementado e apenas alguns países europeus estão desofruindo da tecnologia(ATKISON, 2020). VANETs também só mostrarão por completo seus benefícios quando todos os carros em trânsito possuirem a tecnologia (ATKISON, 2020) e no momento corrente essa não é a realidade. Além do mais embora a maturidade das tecnologia VANETs, a mesma possui diversas limitações de escalabilidade como por exemplo atrasos imprevisíveis e alcance de radio limitado. (GOMES, 2021).

Embora as tecnologias emergentes, como ITS e VANETs, representem avanços importantes no campo da navegação veicular, elas ainda enfrentam desafios como a necessidade de uma infraestrutura robusta e a limitação de sua implementação em larga escala. Além disso, essas soluções estão em estágios de desenvolvimento e não foram amplamente adotadas, principalmente em contextos urbanos complexos, como o proposto neste trabalho.

Portanto, apesar de sua relevância no estado da arte, a abordagem adotada neste projeto não se baseará diretamente nessas tecnologias emergentes. Em vez disso, será seguida uma linha mais consolidada e prática, fundamentada em conceitos clássicos e bem estabelecidos, como a teoria dos grafos e os algoritmos de caminho mais curto, que oferecem soluções robustas e eficazes para o problema de traçar rotas otimizadas e seguras.

No capítulo "fundamentos", serão apresentados os fundamentos teóricos que sustentam a implementação do sistema proposto. Conceitos como a representação do ambiente urbano por meio de grafos e o uso do algoritmo A-Estrela serão detalhados, destacando como essas ferramentas são essenciais para o cálculo de rotas eficientes e seguras em sistemas de navegação veicular.

3
Proposta e Objetivos do Trabalho

O propósito deste trabalho é projetar e desenvolver um software que auxilie os usuários a se locomoverem de forma eficiente no ambiente urbano, proporcionando uma experiência de navegação mais segura e otimizada. A principal funcionalidade do software será a capacidade de traçar rotas, levando em consideração dois critérios fundamentais: a rota mais curta em termos de distância e a mais segura, baseada em um índice de insegurança.

Para atingir esse objetivo, será desenvolvido um modelo que combina dados de mobilidade urbana com informações de segurança pública. Especificamente, o índice de insegurança será construído utilizando dados abertos fornecidos pelo Departamento de Polícia do Estado do Rio de Janeiro, que incluem estatísticas sobre ocorrências criminais em diferentes regiões da cidade. Esse índice será projetado para refletir variáveis como a frequência de crimes violentos, furtos e outros incidentes de segurança.

Uma vez criado o índice de insegurança, ele será integrado a um algoritmo de pathfinding, como o algoritmo A*, que é capaz de calcular a rota mais eficiente levando em conta tanto a distância quanto o nível de segurança das vias. O algoritmo será alimentado com dados geoespaciais, e suas rotas serão ajustadas com base nas variações de segurança. O objetivo é garantir que os usuários não apenas cheguem ao seu destino através do menor caminho, mas também evitem áreas com maior risco, promovendo uma experiência de deslocamento mais tranquila e protegida.

Esse software será uma ferramenta útil para uma grande variedade de usuários, desde pessoas que desejam evitar áreas de risco até turistas e profissionais que dependem de deslocamentos diários em ambientes urbanos complexos contribuindo para uma mobilidade urbana mais eficiente e consciente.

4
Fundamentos

Esse Capítulo visa abordar conceitos teóricos, fundamentais para a resolução do problema proposto

4.1
Teoria dos Grafos e Algoritmos de caminho mais Curto

A implementação de VNS se depara diretamente com o problema do caminho mais curto da teoria dos grafos. Quando desejamos tracejar uma rota, precisamos de uma estrutura de dados, que possa representar a área que se deseja ser mapeada. Uma cidade dificilmente será mapeada utilizando uma estrutura de dados simples, devida ao elevado grau de complexidade da estrutura urbana.Para isso precisamos de uma solução mais robusta que permita uma boa representação do ambiente que se deseja ser computado. Portanto a escolha da estrutura de dados adequada torna-se crucial para a eficácia do mapeamento e da navegação. É nesse contexto que a teoria de grafos oferece uma solução robusta e eficente por meio da estrutura matemática conhecida como grafo, que é composta por vértices e arestas. Ao modelar o problema como um grafo, é possível representar as intersecções de ruas e pontos de interesse como nós, e as próprias ruas e estradas como arestas.Outras definições importantes para o problema, é se um grafo é direcionado ou não, portanto, Quando não é direcionado, podemos percorrer de um nó para outro e voltar pela mesma aresta. Quando é direcionado podemos percorrer de um vértice para outro mas não podemos voltar, pois a aresta agora possui uma direção. Outra propriedade importante é um grafo ser ponderado ou não. Um grafo ponderado é um grafo onde uma aresta apresenta um valor atribuido a ela, conhecida como peso. No contexto automativo, um grafo precisa ser direcionado e ponderado, pois estradas possuem direção e não queremos colocar o usuário na contra-mão, quanto ao peso da aresta, será denominado a distância entre dois nós. (LANDE, 2013)

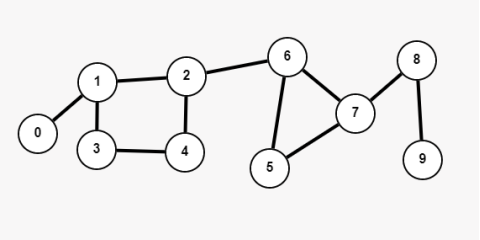


Figura 4.1: Grafo não-direcionado e não-ponderado

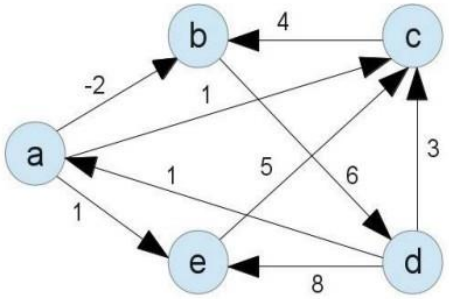


Figura 4.2: Grafo direcionado e ponderado

Dado o que é um grafo e como ele pode ser usado para computar os mapas em uma VNS, busca-se agora maneiras do sistema calcular rotas otimizadas entre dois pontos. Este conhecido pela a teoria dos grafos como problema do caminho mais curto possui diversos algoritmos robustos que solucionam esse quesito.Entre os mais convecionais pode-se ser citado: O algoritmo de Dijkstra e o algoritmo A-Estrela.(LANDE, 2013). Ao selecionar um algoritmo, devemos sempre analisar o problema dado.Ao desenvolver um sistema de nevegação veicular, os grafos que representam o mapeamento rodoviário de uma cidade, normalmente são apresentados como grafos densos, com muitos nós e arestas. Por esse fator, Algoritmos como Dijkstra podem falhar devido ao elevado aumento de tempo e complexidade de espaço. O drástico aumento do tamanho do grafo pode comprometer a eficiência do algoritmo(SYAHPUTRA, 2021).Isso se dá pelo fato de que a solução de dijkstra explora todos os nós de maneira gulosa(SYAHPUTRA, 2021). (LANDE, 2019) Contudo o algoritmo A-Estrela, contorna essa perda de perfomance, ao utilizar funções heurísiticas que auxiliam na escolha dos nós explorados dando prioridade a vértices melhores, tornando-a escolha ideal entre as duas soluções citadas(SYAHPUTRA, 2021).

Dado isso, vamos agora explorar com mais detalhes o funcionamento do Algoritmo A-Estrela e entender por que ele é tão eficiente em aplicações de navegação veicular.

4.2 Algoritmo A-Estrela

O algoritmo A* é um algoritmo de busca que se destaca no planejamento de caminhos em grafos.(SYAHPUTRA, 2021) Sua excepcionalidade está em utilizar uma estratégia diferente dos algoritmos de busca convencionais, conhecida como busca informada.(NETTO, 2018) Esse método utiliza uma informação adicional conhecida sobre o problema para auxiliar o algoritmo na busca. Esse pedaço de informação, transmitido para o algoritmo, pode ser chamado de função heurística(RUSSEL, 2013), ou **h(n)**, e seu principal papel é orientar o algoritmo na busca. Analogamente, poderíamos dizer que essa função age como uma bússola para o algoritmo.

4.2.1
Heurística e condições para otimalidade

Um algoritmo completo e ótimo é aquele que sempre encontra uma solução, caso ela exista, e garante que essa solução será a melhor possível. No caso do A*, ele é considerado ótimo se as soluções apresentadas corresponderem ao caminho mais curto entre o nó inicial e o nó objetivo. No entanto, a otimalidade do A* depende diretamente da implementação de sua função heurística e das propriedades que ela satisfaz. Para que o A* seja ótimo, sua heurística deve cumprir duas condições essenciais: ser admissível e consistente.(RUSSEL, 2013)

A admissibilidade da heurística significa que o valor estimado pela heurística para alcançar o objetivo, partindo de qualquer nó *n*, nunca pode superar o custo real mínimo necessário para alcançar esse objetivo. Em termos práticos, isso garante que a heurística subestima ou, no máximo, iguala o custo real. Essa condição pode ser expressa matematicamente como:

$$h(n) \leq h'(n) \tag{4-1}$$

onde:

- **h(n)** é o valor estimado pela heurística para alcançar o objetivo a partir do nó *n*;
- **h'(n)** é o custo mínimo real para alcançar o objetivo a partir do nó *n*.

A consistência da heurística (também conhecida como monotonicidade) exige uma propriedade adicional: para qualquer nó **n** e seu sucessor **n'**, alcançado por meio de uma ação **a**, o valor estimado pela heurística em **n** não deve ser maior que o custo de transição para **n'** somado ao valor estimado da heurística em **n'**.(RUSSEL, 2013) Essa condição assegura que o valor de $f(n) = g(n) + h(n)$ não diminua ao longo do caminho, garantindo a correção do algoritmo. Formalmente, temos:

$$h(n) \leq c(n, a, n') + h(n'). \tag{4-2}$$

onde:

- **c(n, a, n')** é o custo para transitar do nó *n* ao nó n' pela ação *a*;
- **h(n')** é o valor estimado pela heurística para o sucessor n'.

É importante destacar que toda heurística consistente é admissível.(RUSSEL, 2013) Isso significa que, ao satisfazer a consistência, a admissibilidade também será garantida.

4.2.2
Heuristíca euclidiana

Ao definir uma função heuristíca, podemos citar como uma da mais comuns, a distância euclidiana. Essa métrica sempre descreve a menor distância entre dois pontos qualquer e é definido (Y.CHEN, 2022) pela fórmula mostrada na figura 4.3.

A distância euclidiana é um exemplo de heuristica admissível e consitente (Y.CHEN, 2022) e será utilizada na implementação do protótipo do VNS.

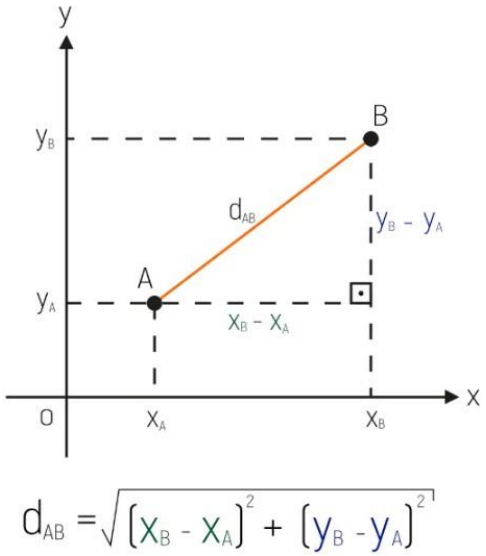


Figura 4.3: Fórmula da distância Euclidiana.(EDUCAçãO, 2024)

4.2.3
Função f(n)

Para entender o algoritmo A* precisamos antes entender como que o algoritmo seleciona os vértices que ele deseja expandir. Diferente do algortimo de Dijkstra, o A*, expande apenas os nós que ele acha mais promissores.E isso é alcançado através de uma função de avaliação **f(n)** que avalia quais serão os nós que o algoritmo escolherá para ser expandido.

A função de avaliação do A* é composta por duas outras funções **g(n)** e **h(n)** onde: (LANDE, 2019)

- **g(n)** é o custo acumulado até **n** a partir da origem.
- **h(n)** é o custo aproximado de **n** até o objetivo final.

Portanto, quando o algoritmo for selecionar um vértice para ser expandido, este buscará expandir o vértice que possuir o menor valor de g(n) + h(n). (RUSSEL, 2013)

4.2.4
Exemplo do Funcionamento do Algoritmo A*

O livro (RUSSEL, 2013) apresenta um excelente exemplo visual do funcionamento do algoritmo A*. Neste exemplo, o algoritmo mostra, passo a passo, os nós expandidos enquanto traça uma rota entre as cidades de Arad e Bucareste.

Ao demonstrar o funcionamento do A*, precisamos definir dois pontos: a origem e o destino. Arad será a origem e Bucareste o destino. Assim, o objetivo do algoritmo é traçar o caminho entre esses dois pontos. No estado inicial, não há nenhum nó selecionado, e o único disponível é Arad, onde $f(n) = g(n) + h(n) = 0 + 366$. Aqui, $g(n) = 0$, pois Arad é o ponto inicial, e $h(n) = 366$, que corresponde à distância euclidiana de Arad até Bucareste.

Como Arad é o único nó disponível, ele será expandido.

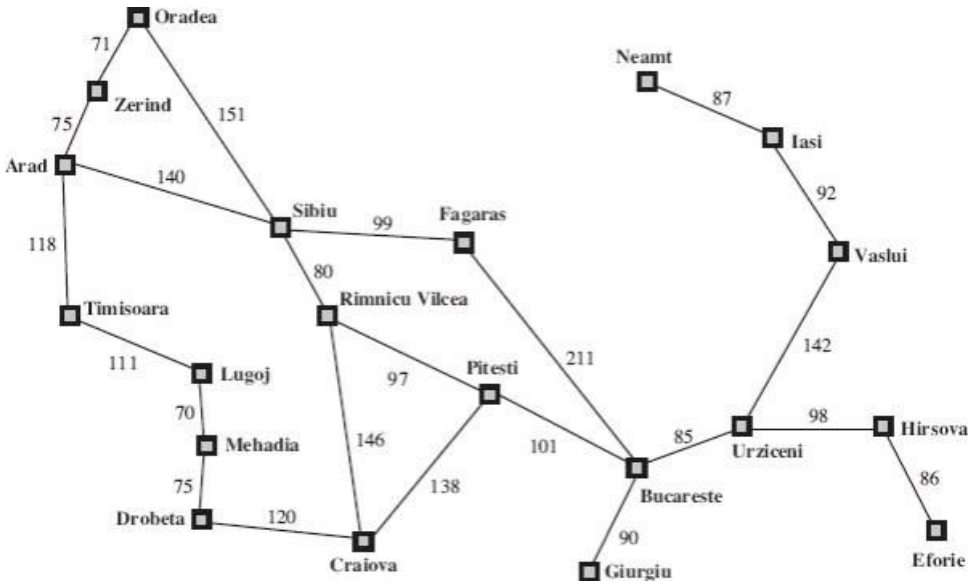


Figura 4.4: Grafo Rodoviário da Romênia (RUSSEL, 2013)

Arad	366	**Mehadia**	241
Bucareste	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Figura 4.5: Tabela com distâncias euclidianas de cada cidade até Bucareste (RUSSEL, 2013)

Figura 4.6: A* - Estado Inicial

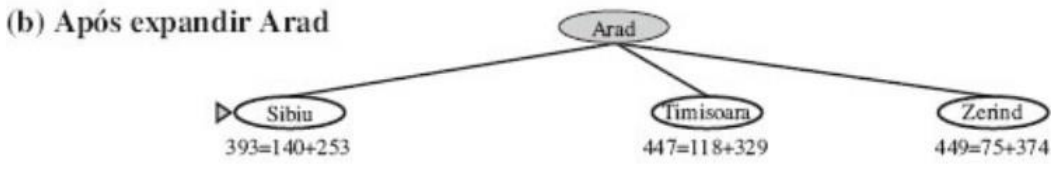


Figura 4.7: Expansão do nó Arad

Após a expansão de Arad, o algoritmo verifica qual será o próximo nó a ser expandido. Arad possui três vizinhos: Sibiu ($f(n) = 393$), Timisoara ($f(n) = 329$) e Zerind ($f(n) = 374$). Lembrando que $f(n) = g(n) + h(n)$, onde $g(n)$ é o custo acumulado do nó inicial até n, e $h(n)$ é a distância euclidiana de

n até Bucareste. Como Sibiu possui o menor valor de $f(n)$, ele será expandido.

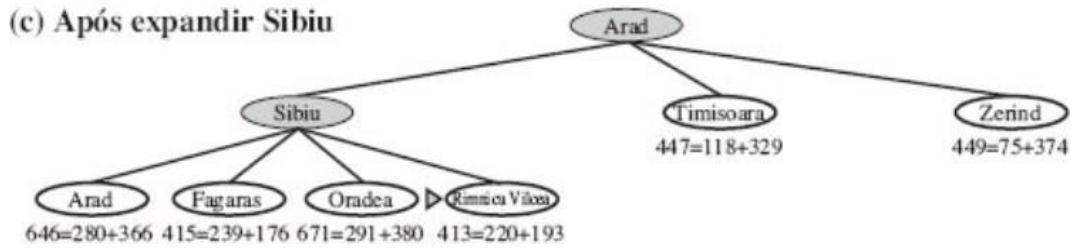


Figura 4.8: Expansão do nó Sibiu

Após expandir Sibiu, os nós adjacentes a ele entram na fronteira de escolha do algoritmo: Arad, Fagaras, Oradea e Rimnicu Vilcea. É importante destacar que os nós não expandidos na iteração anterior (Timisoara e Zerind) permanecem na fronteira de escolha. O algoritmo seleciona o nó com o menor $f(n)$, que, nesse caso, é Rimnicu Vilcea.

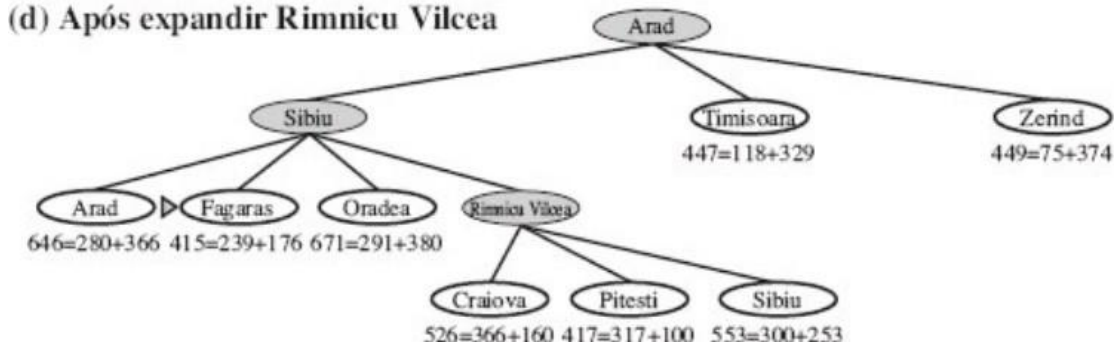


Figura 4.9: Expansão do nó Rimnicu Vilcea

Ao expandir Rimnicu Vilcea, seus nós adjacentes (Craiova, Pitesti e Sibiu) são adicionados à fronteira de escolha, mantendo também os nós previamente não expandidos. Dessa vez, o menor $f(n)$ não pertence a nenhum nó adjacente de Rimnicu Vilcea, mas sim a Fagaras, que será o próximo nó a ser expandido.

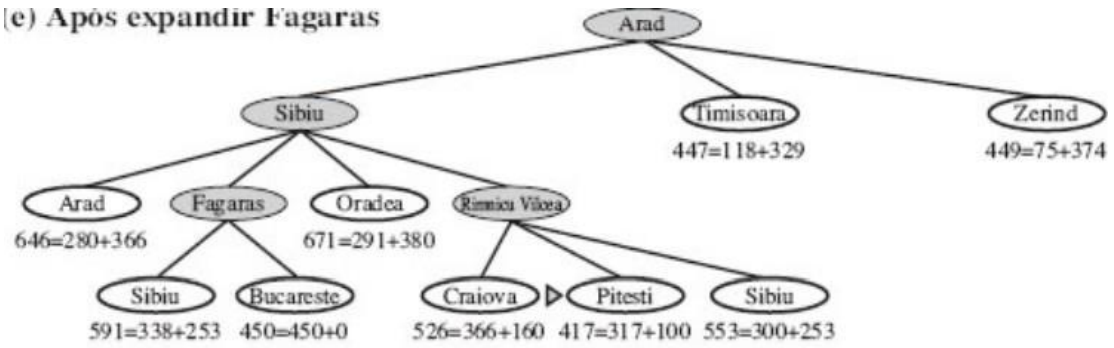


Figura 4.10: Expansão do nó Fagaras

Esse processo se repete até que o nó destino, Bucareste, seja alcançado. No exemplo apresentado, após Fagaras, o próximo nó a ser expandido é Pitesti, seguido por Bucareste, que é adjacente a Pitesti e o nó objetivo.

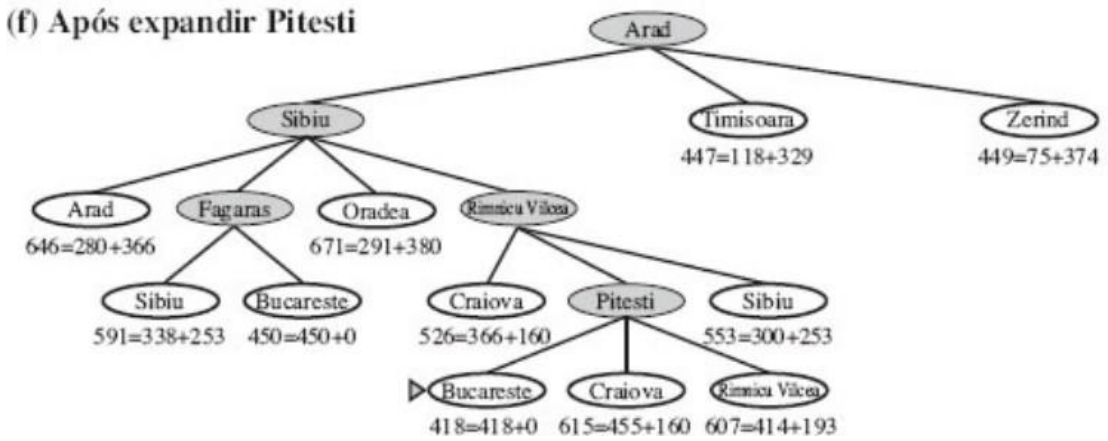


Figura 4.11: Expansão do nó Pitesti (RUSSEL, 2013)

4.2.5 Pseudo-Código

Segue um pseudo-código do algoritmo a* retirado do (LANDE, 2019). Importante ressaltar que o pseudo-código utiliza uma lista aberta para computar os nós não examinados e uma lista fechada para os nós examinados.

1. Iniciar duas listas:
    a) lista_aberta
    b) lista_fechada
2. Inserir o nó inicial na lista_aberta.
3. Manter o custo total do nó inicial como zero.
4. Enquanto lista_aberta não for nula:(While)
    a) Checar o nó com o mínimo custo total(menor f(x)) presente dentro de lista_aberta.
    b) Renomear o nó com o mínimo custo total como X.
    c) Remova X da lista_aberta
    d) Gere os sucessores de X (expandir o nó X).
    e) Coloque X como nó pai de todos os nós sucessores
5. Loop pelo conjunto de sucessores:(for)
    a) Se nó sucessor é igual ao nó de destino:
        a.1) Pare.
    b) Senão:
        b.1) sucessor.g(n) = X.g(n) + distancia entre X e o sucessor.
        b.2) sucessor.h(n) distância do sucessor até o objetivo (dado pela função heuristica)
        b.3) sucessor.f(n) = sucessor.g(n) + sucessor.h(n).
    c) Se um nó com a mesma posição do sucessor estiver presente na lista_aberta, mas tem um custo f(n) menor, então:
        c.1) pular esse sucessor.
    d) Se um nó com a mesma posição do sucessor estiver presente na lista_fechada, mas tem um custo f(n) menor, então:
        d.1) pular esse sucessor
    e) Senão:
        e.1) insira o nó em lista_aberta
6. fim do loop (FOR)
7. Adicionar o nó X em lista_fechada
8. fim do loop(while)
9. A rota é determinada pela lista_fechada
10. Parada do Algoritmo

4.3
Sistema de Refêrencia de coordenadas

Um sistema de referência de coordenadas é uma metodologia para descrever as posições de um objeto.

Quando se deseja representar posições na superfície da terra, podemos representa-las de duas maneiras, através de coordenadas geodésicas e coordenadas projetadas.

As coordenadas geodésicas são coordenadas tridimensionais sobre um elipsoidal da terra, e leva em conta o achatamento dos polos e o abaulamento equatorial. Suas coordenadas são representadas por latitude,longitude e altitude.

Diferentemente das geodésicas, as coordenadas projetadas utilizam transformações matemáticas para representar o espaço tridimensional, em um espaço bidimensional, como um plano. A projeção é muito importante, pois simplifica calculos de distância, área, e volumes reduzindo consideravelmente custo computacional. (GALO, 2002; JANSSEN, 2009)

Dentre os diversos sistemas de referência, destacam-se o WS84 que é comum a aplicações GPS e é um sistema de referência geodésico elipsoidal e a projeção UTM que é um sistema de referência projetado que foi elaborada de maneira a reduzir distorções em uma zona.(G.MANCHUK, 2009)

4.4
Geoestatística e Krigagem

A geoestatística é uma área da estatística que estuda variáveis regionalizadas(LANDIM, 2006), ou seja, variáveis cujos valores apresentam dependência espacial. Diferentemente da estatística clássica, que assume independência entre as observações, a geoestatística é desenvolvida para resolver problemas que requerem a interpretação de fenômenos geográficos.(YASOJIMA, 2020) Exemplos típicos incluem a estimativa da quantidade de um recurso mineral em uma área de exploração e a determinação da extensão de uma contaminação no solo ou na água subterrânea.

Tais problemas são difíceis de serem adequadamente descritos pelas técnicas da estatística clássica, uma vez que estas não consideram a correlação espacial entre os dados.(CAMARGO, 1998) Essa limitação pode levar a interpretações distorcidas da realidade. Para superar esse desafio, a geoestatística incorpora métodos que analisam e modelam a dependência espacial dos dados, utilizando variáveis regionalizadas como base para suas técnicas. (YASOJIMA, 2020)

4.4.1
Objetivos da Geoestatística

A geoestatística tem como objetivo analisar, modelar e interpretar a variabilidade espacial de dados georeferenciados. Com ela podemos realizar predições da distribuição de uma propriedade e mapear superficies contínuas produzindo estimativas de (regiões não-medidas) através de um conjunto de dados discretos (regiões medidas). Uma das técnicas para a realização dessa

predição é a krigagem que (YASOJIMA, 2020) será detalhado nas próximas seções após uma breve introdução geral a geoestatística e suas propriedades.

4.4.2
Variáveis regionalizadas

Uma variável regionalizada é uma função que varia de forma contínua no espaço, e seus valores dependem diretamente do local onde são observados. Essa ideia é fundamental na análise de fenômenos espaciais, já que a posição influencia diretamente o comportamento da variável. (CAMARGO, 1998; UZUMAKI, 1994)

Segundo a teoria das variáveis regionalizadas, podemos descrever essa variação espacial como a soma de três componentes distintas: (CAMARGO, 1998)

$$z(x) = m(x) + \epsilon'(x) + \epsilon'' \tag{4-3}$$

- $m(x)$ representa uma tendência espacial determinística, como uma variação gradual de um dado ao longo de uma área. Essa componente é geralmente modelada por uma função matemática que descreve a variação média da variável na região.(CAMARGO, 1998; .MCBRATNEY, 1981)
- $\epsilon'(x)$ Corresponde a flutuações locais aleatórias que apresentam dependência espacial. Essa componente é modelada por um processo estocástico espacial, como um semivariograma, que descreve a correlação espacial entre os valores da variável em diferentes locais.
- ϵ'' Representa a variabilidade aleatória não explicada pelos outros dois componentes e é geralmente assumida como um ruído.

4.4.3
Representação de Valores Medidos e Estimados

Após a definição dos três componentes que formam uma variável regionalizada ($m(x)$, $\epsilon'(x)$ e ϵ''), é relevante apresentar como os valores dessa variável são representados no espaço, diferenciando os pontos medidos dos estimados.

A variável regionalizada $Z(x)$ é definida em função de x, que representa uma posição no espaço. Essa posição pode ser unidimensional, bidimensional ou tridimensional, dependendo do contexto. Para distinguir entre os valores conhecidos e os valores estimados, utiliza-se a seguinte notação:

- $Z(x_i)$: representa os valores medidos (ou conhecidos) da variável em pontos específicos do espaço ($x_1, x_2, \ldots, x_n$), onde $i = 1, 2, 3, \ldots, n$.
- $Z^*(x_0)$: indica o valor estimado da variável em um ponto x_0, que não foi medido, representando uma estimativa em locais não amostrados.

Essa notação permite organizar os dados conhecidos ($Z(x_i)$) e estimar valores em locais não medidos ($Z^*(x_0)$), o que é fundamental na aplicação de modelos de análise espacial e interpolação.

4.4.4
Semivariograma

O variograma ou semivariograma é uma ferramenta que representa quantitivamente a variação de um um fenômeno regionalizado no espaço.(CAMARGO, 1998; SANTOS, 2018) Em outras palavras o variograma proporciona uma imagem da dissimilaridade das variáveis regionalizadas, a medida que a distância entre elas aumenta.(PEREIRA, 2011)

Para melhor entendimento, Considere duas variáveis regionalizadas, sendo a variável A = Z(x), onde Z(x) representa o valor medido de um atributo qualquer na posição x detonado por uma posição de duas dimensões, e B = Z(x+h), onde Z(x+h) é o valor de medição do mesmo atributo em outra posição, separada por uma distância h. O variograma captura como a diferença entre Z(x) e Z(x+h) varia com h, proporcionando uma visão quantitativa da continuidade espacial do fênomeno.

Um detalhe é que na literatura, o semivariograma é amplamente utilizado, Contudo, por convenção e praticidade, o termo variograma é frequentemente usado como sinônimo de semivariograma.(SANTOS, 2018) No entanto, tecnicamente, o variograma corresponde ao dobro do semivariograma.

Logo podemos representar a correlação e depêndecia entre as duas variáveis A e B pelo semivariograma $2\gamma(h)$ onde $\gamma(h)$ é definido como sendo a esperança matemática do quadrado da diferença entre os valores de pontos no espaço separados por uma distância h (CAMARGO, 1998; PEREIRA, 2011; SANTOS, 2018), isto é:

$$\gamma(h) = \frac{1}{2}\hat{1}[Z(x) - Z(x+h)]^{2'} \tag{4-4}$$

Ao ter uma amostra de variáveis regionalizadas dada por (x_i) com i variando num intervalo de 1 até n, podemos estimar o semivariograma através de

$$\gamma(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} [Z(x_i) - Z(x_i + h)]^2 \tag{4-5}$$

onde

- $\gamma(h)$ é o semivariograma estimado
- $N(h)$ é o valor que representa o número de pare medidos, $Z(x_i)$ e $Z(x_i + h)$ separados pela distância h
- $Z(x_i)$ e $Z(x_i + h)$ são a i-nésima variável regionalizada separadas pela distâcia h.

4.4.5
Interpretação do gráfico do semivariograma

O semivariograma é uma função representada por uma curva crescente que descreve como a variabilidade entre os valores de uma variável regionalizada aumenta com a distância h. Quando h=0, o gráfico geralmente se aproxima da a origem, indicando um alto grau de correlação entre as amostras. A medida que h aumenta, a depêdencia espacial entre os valores diminui, refletindo um maior valor entre as amostras.

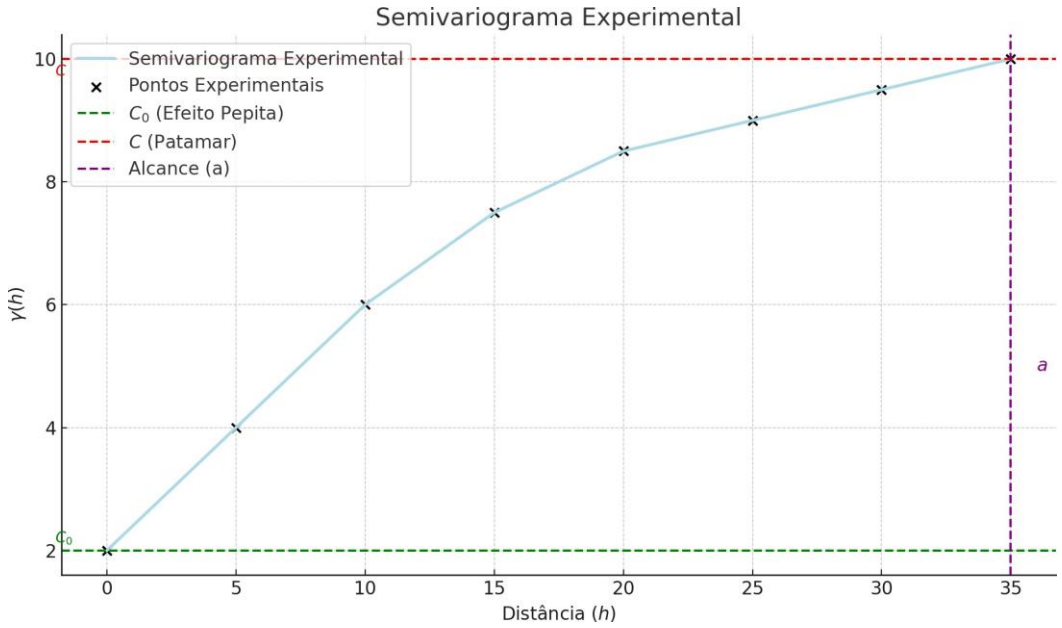


Figura 4.12: Exemplo de semivariograma

A imagem acima mostra um exemplo de um gráfico de um semivariograma experimental(valores discretos), no entanto, apesar da descrição dada anteriormente, entender os atributos e propriedades do gráfico torna-se essencial para uma completa compreensão da dependência dos dados, logo as propriedades podem ser definidas como (CAMARGO, 1998):

- Patamar(sill) ou $C_0 + C$: É o valor máximo que o semivariograma pode chegar, onde as amostras continuam com uma depêndencia espacial. Do patamar em diante as amostras não estão mais correlacionadas espacialmente.
- Alcance(Range): É a distância h em que o semivariograma alcança seu patamar.Ou seja para distâncias maiores que o alcance as amostras são independentes espacialmente.
- Efeito pepita(Nugget) ou C_0: É o valor no semivariograma quando h tende a zero, representando uma variabilidade inicial que pode ser causado por variações em pequenas escalas ou até por erros de medição. Ele nos indica que mesmo para distância muito pequenas existe algum tipo de variabilidade,frequentemente interpretada como ruído.
- A distância entre C_0 e C é conhecida como contribuição adicional.

4.4.6 Modelos Variográficos

O semivariograma experimental é baseado em amostras discretas, no entanto para algoritmos como o de interpolação como a krigagem, precisamos ajustar o semivariograma de maneira que seus dados sejam representados de maneira contínua. Esse ajuste é feito atráves da escolha interativa do intérprete de um conjunto de modelos variográficos téoricos que mais se adequa ao semivariograma experimental.

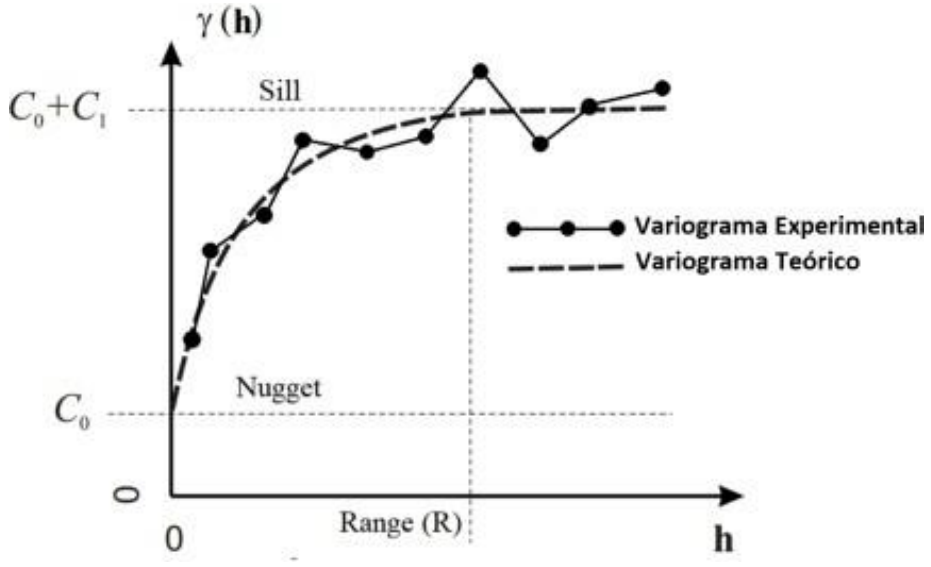


Figura 4.13: Exemplo de ajuste do semivariograma experimental utilizando um modelo (YASOJIMA, 2020)

Dentre os modelos mais básicos a literatura menciona o exponencial, gaussiano,esférico, etc. A seguir será listados esses modelos o qual suas variáveis serão definidas para todos os modelos como:

- C_0 representa o efeito pepita
- $C_0 + C_1$ É o patamar.
- a representa o alcance.
- h É a distância.
- d equivale a distância máxima na qual o semivariograma é definido.

4.4.6.1
Modelo Exponencial

O modelo exponencial (PEREIRA, 2011) possui comportamento linear próximo a origem e seu patamar é atingindo assintóticamente. O seu alcance é definido como a distância correspondente a 95% do patamar e sua equação é dada por:

$$\gamma(h) = C_0 + C_1 \left(1 - \exp\left(-\frac{3h}{a}\right)\right), \quad 0 < h < d \tag{4-6}$$

4.4.6.2
Modelo Gaussiano

O modelo gaussiano (PEREIRA, 2011) possui comportamento parabólico perto da origem e seu patamar é atingindo assintóticamente.

Esse modelo é parecido com o modelo anterior se diferenciando em seu comportamento parabólico nas aproximidades da origem. Esse modelo normalmente é utilizado para modelar fenômenos extramente contínuos, em outras

palavras, no semivariograma gaussiano, o crescimento inicial de *gamma*(h) é muito mais lento, indicando uma depêndencia espacial mais forte em distâncias curtas. Podemos representar este modelo téorico pela a equação:

$$\gamma(h) = C_0 + C_1 \left[1 - \exp\left(-3\left(\frac{h}{a}\right)^{B_2}\right)\right], \quad 0 < h < d \tag{4-7}$$

4.4.6.3
Modelo Esférico

O modelo esférico (PEREIRA, 2011) possui comportamento linear próximo a origem e diferente do modelos anteriores, esse modelo atinge o patamar (C_0+C_1),portanto quando h=a, os valores da amostra se tornam espacialmente independentes. Esse modelo representa de maneira efieciente fenômenos com dependência espacial bem definido, sua equação é definida como:

$$\gamma(h) = \begin{cases} C_0 + C_1 \left[1.5\left(\frac{h}{a}\right) - 0.5\left(\frac{h}{a}\right)^3\right], & \text{se } 0 \le h < a, \\ C_0 + C_1, & \text{se } h \ge a. \end{cases} \tag{4-8}$$

4.4.7
Vizualização gráfica dos modelos

Dado as características matemáticas de cada modelo, é útil vizualizá-los graficamente para reforçar a compreensão de suas diferenças. A seguir, será apresentado uma figura contendo os três modelos descritos, para melhor vizualização de seus comportamentos em termos de crescimento inicial,establização e alcance.

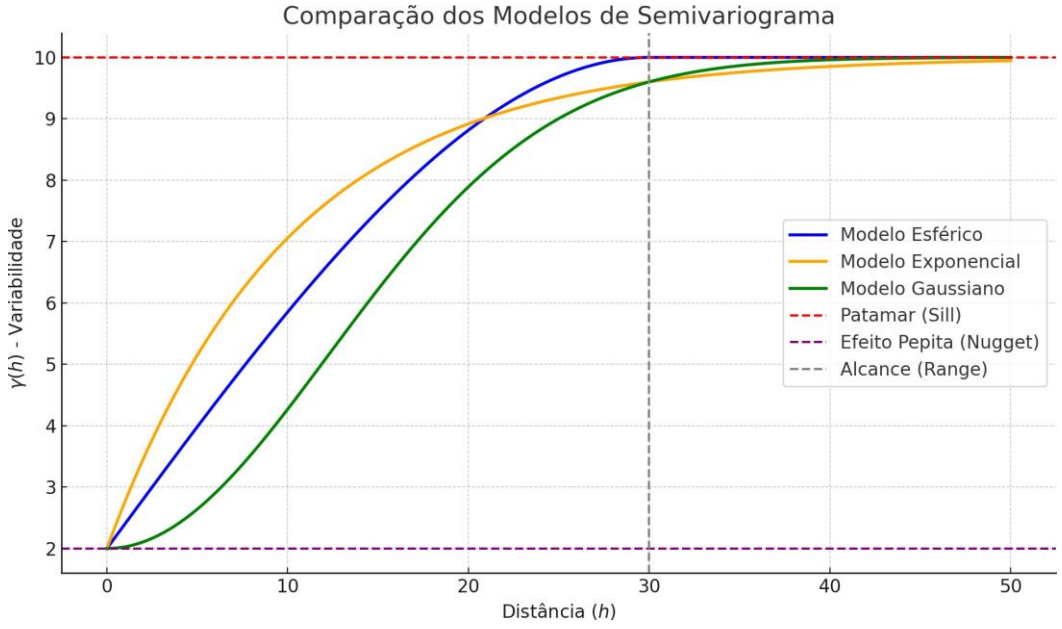


Figura 4.14: Comparação dos modelos teóricos

4.4.8
Krigagem

A krigagem (YASOJIMA, 2020; LANDIM, 2006; CAMARGO, 1998; PEREIRA, 2011) é uma técnica de interpolação da geoestatística que tem como objetivo predizer dados espaciais não amostrados. Esse interpolador considera as caracteriscticas de correlação espacial das váriaveis medidas para estimar pontos onde o valor da variável é desconhecida. Krigagem não é a única técnica de interpolação na geoestatística, existe maneiras mais simples como a interpolação linear, o inverso do quadrado da distância, entre outros. Porém esses métodos, diferente da krigagem, defini o peso das amostras de maneira simples, podendo trazer resultados errôneos para a solução. A krigagem por outro lado, se assemelha a interpolação média móvel ponderada, no entanto os pesos λ_i de suas amostras são determinados a partir de uma análise espacial, baseada no semivariograma experimental.Dessa maneira a krigagem proporciona, estimativa não tedenciosas e com variância mínima. Na literatura, a krigagem é abrangida por um diferente conjunto de variações para o cálculo de estimação. Entre elas estão: A krigagem simples, krigagem ordinária, krigagem universal, Co-kriagem, krigagem disjuntiva, entre outros. A variação mais utilizada é é krigagem ordinária, sendo amplamente aplicada devido à sua simplicidade e eficiência.

4.4.8.1
Krigagem Simples

A krigagem simples (PEREIRA, 2011) é a forma mais básica de krigagem e sua técnica parte da premissa de que a média da variável regionalizada é conhecida e costante em toda a área de estudo. Essa abordagem simplifica os cálculos, pois a média global μ é utilizada diretamente no modelo como um parâmetro fixo. A estimativa de um valor no ponto x_0 é obtida por uma combinação linear do valores conhecidos nas amostras, ajustada pela média global. A equação que define a krigagem simples é:

$$Z^*(\mathbf{x}_0) = \sum_{i=1} \lambda_i Z(\mathbf{x}_i) + \left(1 - \sum_{i=1} \lambda_i\right) \mu \qquad (4\text{-}9)$$

onde:

- $Z^*(x_0)$ é o valor estimado no ponto não amostrado;
- $Z(x_i)$ são os valores observados nos pontos de amostragem x_i;
- λ_i são os pesos calculados para a amostra;
- μ É a média global da variável, assumida como conhecida.

Essa abordagem é vantajosa em situações onde a média global é bem definida e estável, mas apresenta em limitações em contextos onde a média varia localmente ou não é confiável.

4.4.8.2
Krigagem Ordinária

A krigagem ordinária (YASOJIMA, 2020; LANDIM, 2006; CAMARGO, 1998; PEREIRA, 2011) é uma das variações mais utilizadas na geoestatística, destacando-se por não exigir o conhecimento prévio da média global da variável. Diferentemente da krigagem simples, este método assume que a média é constante apenas em uma escala local e que seu valor exato é desconhecido.

A estimativa de um valor no ponto $\mathbf{x}_0$ é feita por uma combinação linear dos valores conhecidos nos pontos amostrados. A fórmula que descreve essa estimativa é:

$$Z^*(\mathbf{x}_0) = \sum_{i=1}^{n} \lambda_i Z(\mathbf{x}_i), \tag{4-10}$$

com a restrição de que:

$$\sum_{i=1}^{n} \lambda_i = 1. \tag{4-11}$$

Nesta fórmula:

- $Z^*(\mathbf{x}_0)$ é o valor estimado no ponto não amostrado $\mathbf{x}_0$;
- $Z(\mathbf{x}_i)$ são os valores conhecidos nos pontos de amostragem $\mathbf{x}_i$;
- λ_i são os pesos atribuídos a cada ponto de amostragem, determinados com base na correlação espacial entre os pontos.

A krigagem ordinária também permite calcular a variância do erro de estimativa em cada ponto, oferecendo uma medida da incerteza associada às predições.

5
Implementação

Nesta seção, são descritas as etapas de implementação do sistema, abordando o desenvolvimento do protótipo em suas diferentes fases. Os repositórios do código-fonte estão disponíveis nos seguintes links:

- Repositório da API/Backend[1]
- Repositório da Aplicação Android[2]

O protótipo é composto por dois componentes que se comunicam entre si: um endpoint de API, implementado em Python utilizando o framework Django REST, e uma aplicação Android, desenvolvida em Java com o Android Studio. A aplicação Android é projetada para consumir e enviar dados ao endpoint, possibilitando a interação e o processamento de informações.

Antes da implementação dos componentes mencionados, foi realizada a extração e o tratamento dos dados, fundamentais para o funcionamento do algoritmo de pathfinding no endpoint.

As etapas de desenvolvimento do projeto foram:

- Extração, análise e tratamento dos dados rodoviários do município do Rio de Janeiro.
- Extração, análise e tratamento dos dados de segurança do município do Rio de Janeiro e criação do índice de insegurança
- Algoritmo A* modificado
- Implementação do endpoint com o algoritmo A* modificado.
- Desenvolvimento da aplicação Android para consumo do endpoint.

Nos tópicos subsequentes, cada uma das etapas mencionadas será detalhada.

5.1
Extração, análise e tratamento dos dados rodoviários do município do Rio de Janeiro

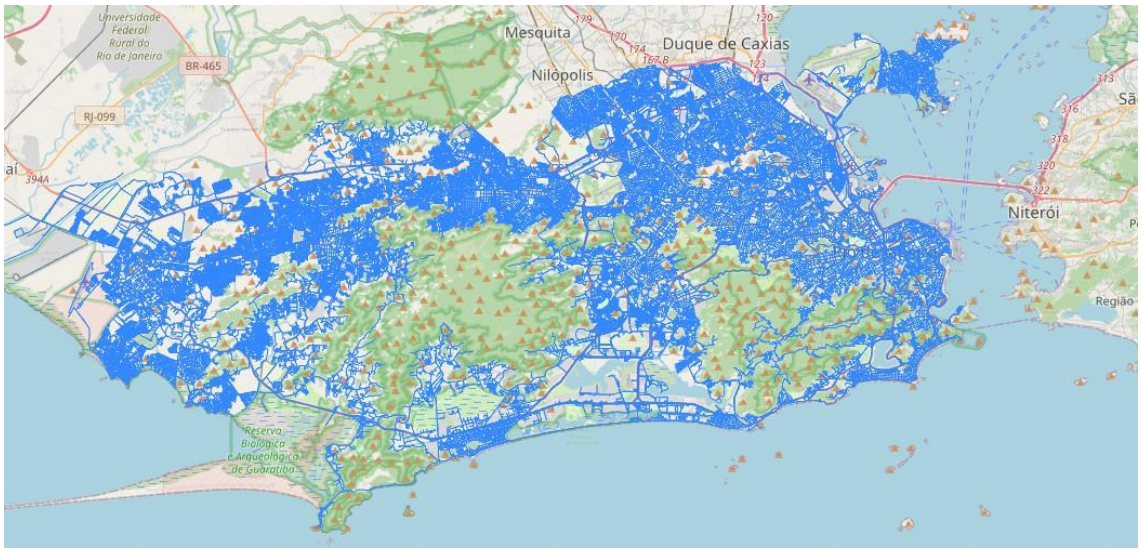
[1] https://github.com/lucaslarios/VNS$_{BACKEND}$
[2] https://github.com/lucaslarios/VNS$_{ANDROIDAS}$

5.1.1
Extração do grafo rodoviário do Rio de Janeiro

A primeira etapa do projeto consistiu na extração do grafo rodoviário do município do Rio de Janeiro. Para isso, foram utilizados dados geográficos do OpenStreetMap, uma plataforma de código aberto que disponibiliza informações detalhadas sobre rodovias, trilhas e demais trajetos veiculares. A extração dos dados foi facilitada pelo uso da biblioteca OSMnx, um pacote em Python que permite o download, análise e visualização de dados geoespaciais do OpenStreetMap.

A configuração inicial da biblioteca OSMnx foi feita em um notebook Jupyter, com o objetivo de realizar o download do grafo rodoviário do município do Rio de Janeiro. Devido a limitações de recursos computacionais, a extração de grafos de áreas maiores foi descartada. Após o download, os dados foram armazenados em um arquivo no formato GraphML, uma escolha convecional para grafos, que permite a representação estruturada de grafos e facilita a manipulação e análise dos dados. É importante destacar que o osmnx permite a escolha do sistema de coordenadas do grafo, por padrão o osmnx entrega um documento com coordenadas geodésicas. Como explicado na seção 4.3 do capítulo fundamentos, este projeto optou em baixar um grafo com coodenadas projetadas.

A visualização inicial do grafo foi feita utilizando a biblioteca Folium, garantindo a confirmação visual de que o grafo correspondia corretamente à malha rodoviária do município.



5.1.2
Estrutura do Graphml

O graphml é uma estrutura de arquivo baseado em XML, desenvolvido para descrever grafos. Este formato possui uma linguagem core que permite que o usuário descreva diferentes tipos de grafos. Este formato estrutura-se em um elemento raiz <graphml> que pode conter múltiplos grafos onde cada grafo é representando pelo elemento <graph>. Para cada grafo os vértices e

arestas são representados pelos elementos <node> e <edge>, identificados por ids únicos.

Um diferencial do GraphML é a possibilidade de personalização por meio dos elementos <key> e <data>. Os elementos <key> são declarados no nível do <graphml> e têm a função de especificar os atributos personalizados, incluindo o nome,identificador e o tipo de dado. Por sua vez, os elementos <data> contêm o valor real do dado e fazem referência ao <key> correspondente, permitindo assim a atribuição de informações adicionais ao grafo ou a qualquer nó ou aresta. Como no exemplo abaixo temos um grafo simples não direcionado descrito em um arquivo graphml. Este grafo tem 3 atributos: x e y para os nós e length para as arestas.

```
<?xml version="1.0" encoding="UTF-8"?>
<graphml>

    <key id="x" for="node" attr.name="x" attr.type="double"/>
    <key id="y" for="node" attr.name="y" attr.type="double"/>
    <key id="length" for="edge" attr.name="length" attr.type="double"/>

    <graph id="G" edgedefault="undirected">

        <node id="n0">
            <data key="x">1.0</data>
            <data key="y">2.0</data>
        </node>


        <node id="n1">
            <data key="x">3.0</data>
            <data key="y">4.0</data>
        </node>

        <!-- Aresta com atributo length -->
        <edge id="e0" source="n0" target="n1">
            <data key="length">10.0</data>
        </edge>
    </graph>
</graphml>
```

5.1.3
Atributos relevantes no Graphml do openStreetMap

O openstreetmap como citado anteriormente utiliza o arquivo graphml. Seu arquivo contém um conjunto enorme de atributos, no entanto, não precisamos de todos eles para o protótipo. Tornando essencial listar os atributos relevantes para a aplicação.

Abaixo temos uma lista com os atributos essenciais para o desenvolvimento da VNS proposta.

- "length"é um atributo de aresta e representa o tamanho das arestas em metros. Em outras palavras, define a distância entre dois nós.
- "x"e "y", responsável pelas coordenadas geográficas dos nós no grafo.Esse dois atributos representam a coordenada geográfica do nó.
- Atributo CRS, é um atributo do grafo que representa o sistema de coordenadas utilizado no grafo.

5.2 Extração, análise e tratamento dos dados de segurança do município do Rio de Janeiro.

A segunda etapa do projeto foca na extração de dados da polícia do estado do Rio de Janeiro e sua análise e tratamento para a criação de um indíce de insegurança. Esses dados foram extraídos do iSPDados que é uma página de dados abertos do instituto de segurança pública que compõe uma série de documentos CSV contendo diversos registros criminais e atividade policial no Estado do Rio de janeiro.

Para elaborar um índice de insegurança foram usados três tabelas. A primeira tabela contabiliza os número de crimes e o tipo de crime cometido separados por região num período de 2001 até 2024, a segunda tabela proporciona os mesmo dados, no entanto suas informações apenas proporcionam informações criminais em comunidades cariocas pacificadas através de uma unidade de polícia pacificadora(UPP).

5.2.1 Índice de Insegurança: Discreto x Interpolado

Inicialmente, o índice de insegurança foi gerado em uma representação discreta, atribuindo um único valor de insegurança para cada bairro e comunidade. Nessa abordagem inicial, cada unidade territorial possui um índice único que representa a insegurança da região como um todo. Isso implica que todas as ruas de um mesmo bairro ou comunidade compartilham o mesmo valor de índice, o que pode mascarar diferenças significativas dentro da área analisada.

Entretanto, considerando que o objetivo deste trabalho é alimentar o algoritmo A* com informações detalhadas de insegurança, é essencial estimar o índice em pontos significativos para o problema através de uma técnica de interpolação, permitindo valores distintos para diferentes ruas dentro de uma mesma unidade territorial. Essa interpolação possibilita uma análise mais granular da insegurança, refletindo as variações existentes dentro de cada bairro ou comunidade.

Conforme descrito no capítulo de fundamentos, a krigagem é uma técnica de interpolação que requer um conjunto de valores conhecidos para realizar estimativas nos pontos desconhecidos. Assim, nas próximas seções, será detalhado o processo de preparação dos dados necessários para alimentar o algoritmo de krigagem, incluindo a definição dos pontos de partida e as etapas realizadas para gerar os valores interpolados.

5.2.2
Indíce discreto de insegurança por bairro

A elaboração de um índice de insegurança por bairro foi realizada com base em dados históricos de criminalidade extraidos da tabela BaseDPEvolucaoMensalCisp.csv que contém os registros mensais de ocorrências criminais classificados por tipo de crime e são associados por unidades CISP. CISP representa uma subdivisão geográfica usada pelas forças de segurança para monitorar e registrar ativiades criminais. Como o objetivo é desenvolver um índice de insegurança por bairro, uma segunda tabela CSV complementar (Relacao_RISPxAISPxCISP.csv) foi utilizada para compreender a correspondência geográfica entre as unidades administriativas de segurança pública e os bairros do município do estado do Rio de Janeiro.

5.2.2.1
Composição das tabelas

A tabela BaseDPEvolucaoMensalCisp.csv possui a seguinte composição:

- CISP,AISP e RISP que são as colunas utilizadas como identificador da circurscrição de segurança pública
- Ano e Mês são as colunas com informações temporais que permitem a análise de tendências criminais ao longo do tempo.
- Nunic: Essa coluna representa o município.
- Um conjunto de colunas de crimes, onde cada coluna contabiliza um tipo de crime.

Este conjunto de dados é essencial para identificar os padrões criminais em áreas específicas e será uma tabela essencial para criação de um indíce de insegurança por bairro.Já a Relacao_RISPxAISPxCISP.csv é composta pelos seguintes atributos:

- CISP,AISP e RISP que são as colunas utilizadas como identificador da circurscrição de segurança pública
- Unidade territorial: É a coluna que representa a equivalência geográfica dos identificadores acima. Alguns identificadores representam mais de um bairro, portanto unidade territorial, apresenta linhas com mais de um bairro descrito.
- Munícipio: Essa coluna representa o múnicipio em que o CISP está localizado.
- Região do governo: Essa coluna representa qual região governamental se refere, como por exemplo a metropolitana ou a serrana.

Este arquivo é o elo entre os dados de segurança pública e a malha urbana, permitindo a consolidação dos dados criminais em unidades territoriais compreensíveis.

5.2.2.2
Pré-processamento e Limpeza dos dados

Inicialmente, para garantir uma leitura correta dos dados pelo pandas, utilizou-se a biblioteca chardet para detectar a codificação utilizada pelo o arquivo CSV. A codificação foi identificada como ISO-8859-1, uma escolha que busca garantir a compatibilidade com caracteres especiais em português. Após a importação dos dados, foi realizado uma análise inicial para verificar o número de colunas e linhas, bem como identificar a presença de valores nulos. Essa etapa teve um papel de compreender quais informações estavam incompletas e quais precisavam ser tratadas ou removidas antes de prosseguir com a análise.

O conjunto de dados abrangia 63 colunas e 34437 linhas, onde 9 dessas colunas possuiam valores nulos e 12 não representavam dados significativos para um índice de insegurança. Com análise inicial, foi adotado o primeiro tratamento, realizado pelo método de remoção de colunas e linhas não informativas. Nessa etapa foram excluídas do conjunto de dados, todas as colunas que apresentavam valores nulos ou que não agregavam valor significativo ao cálculo de indice de insegurança.Essas colunas incluíam tipos de ocorrências como 'apf','cmp','veiculos recuperados pela polícia','estelionato', entre outras que não eram diretamente relacionadas a criminalidade na região. Outro tratamento, importante foi filtrar os dados num período de 2007 até 2021, esse tratamento foi necessário, pois posteriormente esses dados serão agregados aos dados de crimes registrados nas comunidades com unidade de polícia pacificadora, e esta tabela dos crimes cometidos em comunidade não abrange dados do período de 2022 até 2024 como a tabela BaseDPEvolucaoMensalCisp.csv, portanto reduzimos a tabela para esse período. Além do período, reduziu-se o escopo regional dos dados do dataframe apenas para crimes no Munícipio do Rio de Janeiro,pois o protótipo trabalhado está utilizando apenas o grafo das rodovias do municipio do Rio de Janeiro devido a limitações computacionais, tornando irrelevante a análise da criminalidade em outros municipios. Isso se deu através da coluna "munic"do dataframe. Após a filtragem dos dados, "munic"foi removido.

5.2.3
Transformação dos Dados

Com os dados limpos, foi necessário manipulá-los de forma que cada coluna de crime representasse o número total de ocorrências registradas naquela região no período de 2007 a 2021. Como o dataframe original possui dados mensais, uma mesma região CISP aparece 12 vezes a cada ano, contabilizando os crimes mensalmente.

Para consolidar essas informações e evitar duplicações, realizou-se uma operação de groupby utilizando as colunas **CISP**, **AISP** e **RISP**. Com essa transformação, cada linha do dataframe passou a representar uma região única, e suas colunas passaram a indicar o total de crimes registrados por tipo ao longo dos 14 anos analisados.

Em seguida, foi necessário identificar as regiões representadas pelas combinações de CISP, AISP e RISP. Para isso, realizou-se um merge com

o dataframe Relacao_RISPxAISPxCISP.csv, que associa cada combinação de RISP, AISP e CISP ao nome do respectivo bairro.

Como resultado, obteve-se um dataframe em que:

- Cada linha representa um bairro único.
- Uma coluna identifica o nome do bairro.
- As demais colunas contabilizam o número total de crimes cometidos por tipo no período de 2007 a 2021.

Após essas etapas de transformação, o dataframe encontra-se preparado para a criação do índice de insegurança.

5.2.4
Indice de insegurança

O índice de insegurança é uma métrica criada com o objetivo de mensurar, o nível de risco e vulnerabilidade a segurança pública em diferentes regiões. A sua construção envolve a consideração tanto da frequência quanto da gravidade dos diferentes tipos de crimes registrados. O cálculo do índice de insegurança é baseado em um somatório ponderado, onde cada ocorrência de crime é multiplicada por um peso que reflete a sua gravidade. Dessa forma, o índice é construido de modo a garantir que crimes de maior impacto, como homicídios, sejam devidamente considerados de forma mais significativa do que crimes de menor gravidade. A formulação do índice de insegurança I_r pode ser expressa pela a seguinte equação:

$$I_r = \sum_{i=1}^{n} (C_i \times P_i) \tag{5-1}$$

Onde:

- I_r é o índice de insegurança para a unidade territorial r;
- C_i é o número de ocorrências do crime i registrado na unidade r;
- P_i é o peso atríbuido ao crime i, variando de acordo com a sua gravidade;
- n é o numero total de diferentes tipos de crimes considerados na análise.

5.2.4.1
Atribuição dos pesos

A atribuição dos pesos aos diferentes tipos de crimes é baseada no manual de "Categorização dos Crimes Violentos no Brasil"de (SILVA, 2016), que sugere a seguinte classificação dos crimes:

- **Peso 5:Crimes classificados como Violentos Letais**,como homicídios dolosos e latrocínios, representam o maior risco, pois envolvem perda de vida.
- **Peso 4:Crimes classificados como Violentos não letais graves**, como sequestros e extorsão, causam grande impacto físico ou psicológico na vítima.

- **Peso 3:Crimes Violentos não letais moderados**,como roubo, envolvem coerção ou violência, mas não ameaçam diretamente a vida.
- **Peso 2:Crimes não letais leves**,como ameaças e arrombamentos, causam danos psicológicos ou patrimoniais menores.
- **Peso 1:Crimes patrimoniais menores**,como furtos, que afetam o bem-estar patrimonial mas têm impacto reduzido na integridade física.

5.2.4.2
Fundamentação

A escolha do somátorio ponderado como método para o cálculo do índice de insegurança encontra respaldo na literatura sobre construção de indicadores compostos. Conforme discutido em (CORTES ADELAR FOCHEZATTO, 2018), a ponderação dos diferentes tipos de crime é fundamental para garantir que o índice final seja representativo do risco real enfrentado pela comunidade. A técnica de ponderação pemite que crimes mais graves tenham um impacto maior no valor final do índice, evitando que uma alta quantidade de ocorrências de menor gravidade ofusque a severidade de crimes violentos.(NARDO, 2005) reforçam a importância da ponderação em indicadores compostos, destacando que essa técnica contribui para uma avaliação mais precisa e equilibrada dos diversos aspectos que influenciam o fenômeno em estudo. No caso da segurança pública, a ponderação assegura que o índice reflita não apenas a quantidade de ocorrências, mas também o impacto diferenciado de cada crime, oferecendo uma visão mais precisa do risco envolvido.

5.2.4.3
Normalização

Para padronixar os valores do índice, foi aplicada uma normalização min-max, que transformou os valores para uma escala de 0 a 1, facilitando a comparação entre os índices nas diferentes unidades territoriais. O índice de insegurança normalizado permite identificar as regiões de maior insegurança, com valores próximos de 1, e aquelas de menor insegurança, com valores próximos de 0.

5.2.5
Indíce discreto de insegurança por comunidades

A elaboração de um índice de insegurança por comunidade foi realizada com base em dados históricos de criminalidade extraídos da tabela UppEvolucaoMensalDeTitulos.csv que contém os registros mensais de ocorrências criminais classificados por tipo de crime e são associados por unidades de polícia pacificadora instaladas por comunidade.

5.2.5.1
Composição da Tabela

A tabela UppEvolucaoMensalDeTitulos.csv possui a seguinte composição:

- upp é o atributo que representa a comunidade o qual a UPP permenece instalada
- Ano e Mês são as colunas com informações temporais que permitem a análise de tendências criminais ao longo do tempo.
- O restante é um conjunto de colunas de crimes, onde cada coluna contabiliza um tipo de crime.

5.2.5.2
Tratamento dos dados e cálculo do índice de insegurança

O CSV contendo os dados históricos de criminalidade em comunidades, segue o mesmo formato que o CSV dos dados históricos de crime separados POR CISP. A sua principal diferença, é que ela não precisa de uma tabela auxiliar para identificar as regiões CISP denominadas pela polícia. Essa tabela já entrega diretamente o nome da comunidade, permitindo uma rápida interpretação de qual região os dados condizem. Logo o tratamento de dados e o cálculo de índice de insegurança seguiram com a mesma lógica de tratamento e cálculo da tabela de criminalidade anterior. Agora possuímos dois dataframes, um contendo os índices de criminalidade por bairro e outro contendo os índices de criminalidade por comunidade. Ambas são normalizadas e possuem valores de 0 a 1, sendo 0 mais seguro e 1 menos seguro.

5.2.6
Índice discreto de insegurança Global

A criação do índice global de insegurança tem como objetivo integrar as informações de insegurança tanto das comunidades quanto dos bairros do município do Rio de Janeiro em um único indicador. Para isso precisamos ajustar os dataframes, combina-los e renormaliza-los.

5.2.6.1
Ajustes e combinação dos dataframes

Para combinar os dois dataframes, antes precisamos ajustar seus valores. O dataframe de bairros varia de 0 a 1 e o dataframe das comunidades também, no entanto de acordo com (SILVA, 2010), as comunidades são frequentemente associadas a níveis mais altos de criminalidade e insegurança quando comparadas a outras áreas urbanas, destacando que comunidades são historicamente áreas com maior vulnerabilidade, em grande parte devido á ausência de poder público, ao tráfico de drogas e às condições socioeconômicas desfavoráveis. Para ajustar essa discrepância, foi implementada uma modificação no índice de insegurança das comunidades. Inicialmente, cada dataframe (comunidades e bairros) recebeu um atributo adicional, chamado **is_favela**.

Esse atributo foi definido como **True** para as comunidades e **False** para os bairros, permitindo que, ao combinar os dois dataframes, fosse possível identificar facilmente se a unidade territorial correspondia a uma comunidade ou a um bairro.Combinando os dataframes, agora precisamos ajustar o índice de forma a refletir adequadamente o risco diferenciado, foi então somado um valor de 1 ao índice de todas as linhas do dataframe combinado que tivessem o atributo **is_favela** como **True**. Com essa modificação, os valores dos índices passaram a variar de 0 a 2, onde os bairros variam de 0 a 1 sendo 0 o bairro mais seguro e 1 menos seguro, enquanto as favelas mais seguras passaram a variar entre 1 e 2, sendo 1 a comunidade mais segura e 2 a comunidade menos segura. Dessa Forma, foi possível garantir que mesmo as favelas mais seguras fossem consideradas mais vulneráveis do que os bairros menos seguros. Após essa alteração, os índices foram renormalizados para uma escala de 0 a 1, considerando o novo intervalo obtido de 0 a 2. Esse procedimento de renormalização garantiu a comparabilidade entre todas as unidades territoriais dentro de uma mesma escala, ao mesmo tempo em que preserva a distinção entre bairro e comunidades, evidenciando o maior risco associado as favelas.

5.2.7
Interpolação do índice de insegurança

O índice global de insegurança, em sua forma inicial, era representado por valores discretos associados às regiões de bairros e comunidades. Esses valores discretos, embora adequados para análises gerais, representavam a insegurança de toda a unidade territorial com um único dado. Para realizar análises mais detalhadas e atribuir valores específicos a diferentes pontos de uma mesma região, foi necessário criar um índice com uma representação espacial mais detalhada do Rio de Janeiro através da interpolação.

O objetivo era estimar os valores do índice de insegurança para cada nó presente no grafo urbano do município do Rio de Janeiro, extraído do OpenStreetMap utilizando a biblioteca OSMNX.A interpolação desses valores permitiria que cada nó do grafo, identificado por suas coordenadas x e y,fosse associado a um índice de insegurança específico. Essa abordagem viabiliza a análise da insegurança em uma escala muito mais detalhada, fornecendo um valor estimado para cada nó presente no grafo rodoviário do município. O processo de estimativa foi realizado por meio de técnicas de geostatística, utilizando a krigagem ordinária para interpolar os índices a partir de pontos medidos (representados pelos bairros e comunidades). A seguir, são descritas as etapas do processo.

5.2.7.1
Identificação de Pontos Representativos: Adição de geometria e cálculo de centroides

Para que fosse possível realizar a interpolação espacial, foi necessário identificar um ponto representativo para cada região, considerando que o índice discreto correspondia a bairros e comunidades inteiras. A solução adotada foi o uso dos centroides geográficos dessas regiões, que representam o ponto central de cada polígono delimitado pela geometria das unidades territoriais.

Primeiramente, as geometrias espaciais dos bairros e comunidades foram adicionadas ao dataframe contendo os índices globais de insegurança. Em seguida, foram calculados os centroides dessas geometrias, gerando as coordenadas x e y de cada região. Esses centroides passaram a ser os pontos medidos a serem usados no modelo de krigagem. Os valores do índice global normalizado foram associados a esses centroides como a variável z. Dessa forma, cada centroide forneceu a localização e o valor de insegurança para alimentar o modelo de interpolação.

5.2.8
Semivariograma Experimental e Modelo de Variograma

O semivariograma experimental é uma ferramenta essencial para compreender a dependência espacial entre os dados geográficos, sendo um passo crucial no processo de interpolação por krigagem, como descrito no Capítulo de Fundamentos 4.4.4. Neste estudo, o semivariograma experimental foi construído a partir das coordenadas dos centroides dos bairros e comunidades, utilizando o índice de insegurança como a variável de interesse (Z).

Para o cálculo do semivariograma experimental, foi adotado um número de lags igual a 10 (n_lags = 10), o que permitiu dividir a distância máxima entre os pontos em intervalos uniformes, garantindo um equilíbrio entre granularidade e confiabilidade estatística. Essa configuração permitiu capturar a variabilidade espacial sem comprometer a robustez das estimativas de semivariância em cada bin.

O semivariograma experimental revelou um padrão típico de dependência espacial, com a semivariância aumentando conforme a distância (h) cresce, até atingir um patamar (sill), onde a dependência espacial torna-se insignificante. Após a análise dos dados, verificou-se que o comportamento do semivariograma era melhor representado pelo modelo gaussiano, uma vez que a curva ajustada por este modelo mostrou-se bem alinhada com os pontos experimentais. O modelo gaussiano foi escolhido devido à sua capacidade de capturar transições suaves na dependência espacial, refletindo com precisão o comportamento gradual observado no semivariograma experimental.

A Figura 5.2.8 ilustra o semivariograma experimental ajustado com o modelo gaussiano.

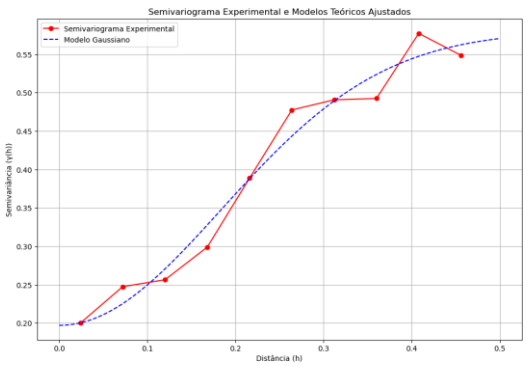


Figura 5.1: Semivariograma experimental x modelo gaussiano

5.2.9
PyKrige e Krigagem Ordinária

A biblioteca *PyKrige* foi utilizada para implementar a krigagem ordinária, uma técnica de interpolação que se baseia na dependência espacial identificada pelo modelo de variograma. A *PyKrige* suporta diferentes tipos de krigagem e permite ajustar diversos modelos de semivariograma, proporcionando uma abordagem prática e eficiente para análise espacial.

No presente estudo, o modelo gaussiano foi ajustado ao semivariograma experimental utilizando a funcionalidade integrada da biblioteca *PyKrige*. Este modelo foi escolhido devido à sua capacidade de capturar transições suaves na dependência espacial, como evidenciado na curva ajustada apresentada na Figura 5.2.9. Os parâmetros ajustados para o modelo gaussiano foram:

- **Patamar (Sill)**: 0.581, representando a variância total dos dados após a estabilização da dependência espacial;
- **Alcance (Range)**: 0.471, indicando a distância máxima em que há correlação espacial significativa;
- **Efeito Pepita (Nugget)**: 0.206, correspondendo à variância residual ou ruído nos dados.

O semivariograma ajustado revelou que os dados apresentam uma forte dependência espacial até aproximadamente $h \approx 0.5$. Após essa distância, os valores tornam-se mais dispersos, indicando a ausência de correlação espacial.

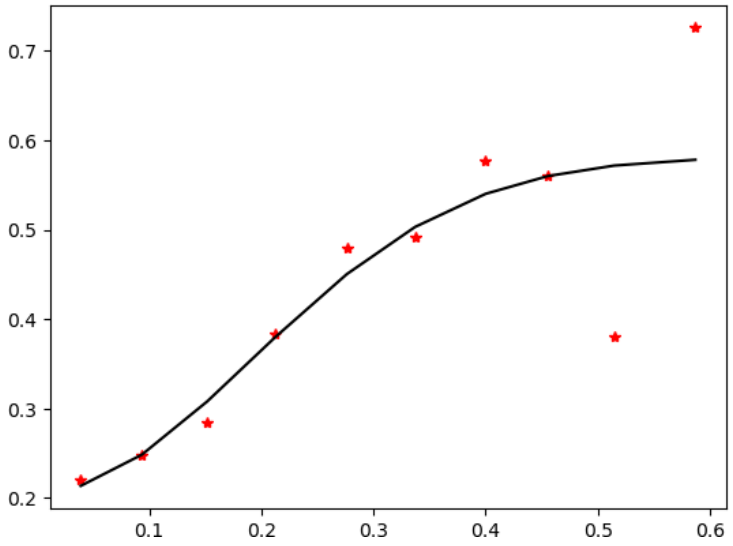


Figura 5.2: Plot do modelo gaussiano ajustado pelo pykrige

A aplicação da krigagem ordinária com o modelo ajustado permitiu estimar o índice de insegurança em pontos não medidos, distribuídos ao longo dos nós do grafo do município do Rio de Janeiro, obtidos com o *OSMnx*.

5.2.10
Integração do Índice de Insegurança ao GraphML

Após a obtenção do índice de insegurança interpolado por krigagem ordinária, os valores estimados foram integrados ao grafo do município do Rio de Janeiro, representado no formato *GraphML*. Esse processo consistiu na criação de um novo atributo associado a cada nó do grafo, o qual foi denominado *indice_inseguranca*. Esse atributo foi adicionado diretamente ao arquivo *GraphML*, de forma que cada nó passou a conter essa propriedade.

Para preencher o atributo *indice_inseguranca_estimado* em cada nó, foram utilizados os valores estimados pela a krigagem. A associação entre os valores estimados e os nós foi realizada com base na correspondência espacial entre as coordenadas dos nós do grafo e os pontos de predição utilizados na krigagem. Dessa forma, cada nó do grafo recebeu um valor de índice de insegurança, refletindo as condições de risco estimadas para sua localização específica.

Esse processo de integração permitiu que o grafo do município fosse enriquecido com informações espaciais detalhadas sobre a insegurança. O resultado final é um *GraphML* no qual cada nó contém o atributo *indice_inseguranca*.

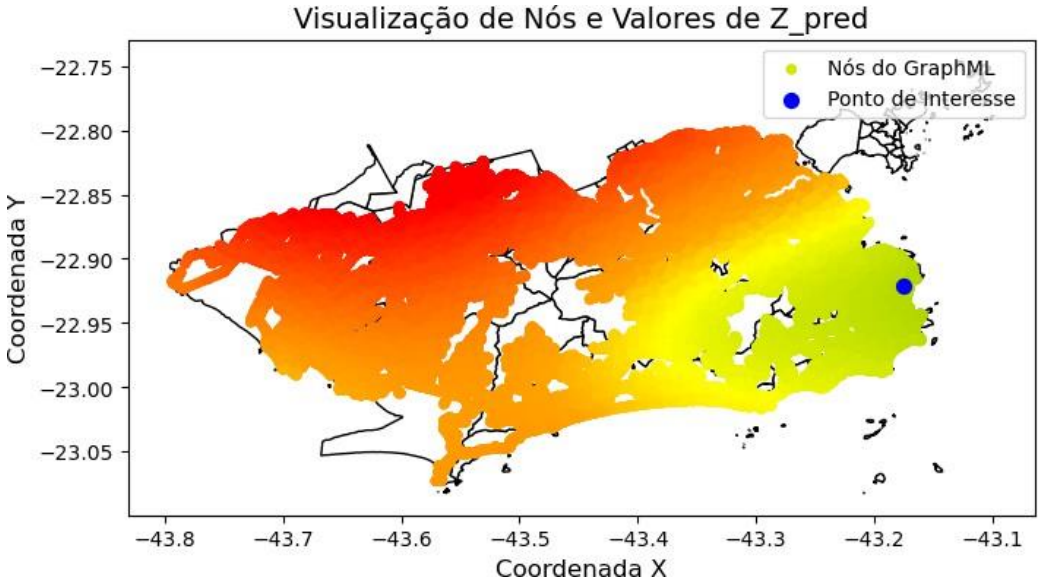


Figura 5.3: Plot índice de insegurança por nó

5.3
Alteração do Algoritmo A* para Considerar o Índice de Insegurança

O algoritmo A* foi detalhado previamente na seção de fundamentos 4.2, onde foram apresentados seus princípios básicos e funcionamento. Nesta seção, discute-se a modificação realizada no algoritmo A* para que ele considere o índice de insegurança no cálculo de rotas. Essa alteração foi implementada de forma a incorporar a variável de insegurança como um fator de penalização nas decisões do algoritmo, influenciando as rotas escolhidas.

O A* baseia-se na função $f(n)$, que é responsável por avaliar e priorizar os nós a serem expandidos durante a busca. No A* convencional, a função é definida como:

$$f(n) = g(n) + h(n)$$

onde:

- $g(n)$ representa o custo acumulado do caminho desde o nó inicial até o nó atual n,
- $h(n)$ é uma heurística que estima o custo do caminho do nó atual n até o nó objetivo.

5.3.1 Modificação em $g(n)$

A modificação introduzida no algoritmo A* concentrou-se na forma como o $g(n)$ é calculado. No A* original, $g(n)$ é dado por:

$$g(n) = g(n-1) + c \tag{5-2}$$

onde:

- $g(n-1)$ é o custo acumulado até o nó $n-1$,
- c é o custo associado à aresta que conecta o nó $n-1$ ao nó n.

Para incorporar o índice de insegurança no cálculo do custo, o termo c foi multiplicado por um fator exponencial que considera o índice de insegurança i associado ao nó n. A nova fórmula para $g(n)$ é dada por:

$$g(n) = g(n-1) + \left[c \times e^{i \times 10}\right]^2 \tag{5-3}$$

Essa modificação implica que:

- Quanto maior o valor do índice de insegurança em um nó, maior será o impacto no custo da aresta, penalizando rotas que passam por regiões inseguras;
- Em casos onde o índice de insegurança é zero (representando segurança), o fator exp(índice_insegurança × 10) será 1, resultando em um comportamento similar ao A* convencional;
- Já em regiões de alta insegurança, o fator exponencial faz com que o custo da aresta aumente significativamente, desincentivando a escolha dessa rota.

Por exemplo, no caso limite em que o índice de insegurança é igual a zero, o cálculo do custo permanece inalterado, pois exp(0) = 1. Nesse cenário, a função $g(n)$ reduz-se ao formato original do A* convencional:

$$g(n) = g(n-1) + c$$

5.3.2
Função $f(n)$ proposto

Com a alteração em $g(n)$, a nova função de avaliação $f(n)$ do algoritmo A* é expressa como:

$$f(n) = g(n-1)+[\text{custo\_aresta} \times \exp(\text{índice\_insegurança} \times 10)]+h(n) \quad (5\text{-}4)$$

Essa formulação reflete a integração do índice de insegurança no cálculo das rotas, garantindo que o algoritmo priorize caminhos que não apenas sejam curtos, mas também seguros.

5.4
Backend Rest API

Após a construção do índice de insegurança e a modificação do A-estrela através de arquivos jupyter, foi necessário integrá-los de maneira que qualquer aplicação pudesse requisitar o caminho mais curto de forma robusta e em tempo real. Assim, a próxima etapa envolveu o desenvolvimento de uma API que se comunicaria com a aplicação Android.

Para isso, foi utilizada a biblioteca Django Rest Framework, uma solução Python excelente e de fácil manuseio. Nesta etapa, foi criado um endpoint onde a aplicação Android envia para o servidor Django, através de requisições HTTP, a localização atual do usuário e o endereço de destino. Quando o endpoint recebe essas informações, ele realiza os devidos tratamentos nos parâmetros recebidos e chama o algoritmo A* para calcular o caminho mais curto e seguro, retornando para a aplicação Android uma lista de nós e coordenadas que representam o trajeto entre a origem e o destino.

5.4.1
Organização dos módulos

Configuração do Django Rest Framework: Após seguir as configurações iniciais do Django Rest Framework, foi criado um módulo chamado ‘algorithms.py‘ na árvore de módulos do Django. Este módulo contém a classe com os métodos responsáveis pelo tratamento dos dados recebidos pela requisição e pelo cálculo do caminho mais curto e mais seguro.

Módulo ‘algorithms.py‘: Em ‘algorithms.py‘, foi criada a classe ‘Graph‘, que é singleton e responsável por todos os métodos utilizados na aplicação para a manipulação do grafo como por exemplo: O carregamento do grafo, o cálculo do A-estrela, etc. Entre os métodos principais, destacam-se:
- ‘**load_graph**‘: responsável por ler o arquivo GraphML e converter os dados serializados em estruturas de dados adequadas. Este método encapsula a função ‘load_graphml‘ da biblioteca OSMnx.
- ‘**geocode_finder**‘: traduz uma string que representa um endereço físico em sua respectiva coordenada cartesiana, encapsulando a função ‘geocode‘ da API Nominatim.
- ‘**nearest_node**‘: encapsula a função ‘nearest_node‘ da OSMnx, retornando o nó mais próximo a uma coordenada cartesiana específica. Este método é crucial para calcular o caminho mais curto entre as coordenadas de origem e destino do usuário, pois o parametro ’origem’ da requisição e o parametrô

'destino' tratado pelo nomatim, ambos nos retornam coordenadas cartesianas, e para executar o shortestpath precisamos dos respectivos nós ou nós mais próximos dessas coordenadas no grafo.
- '**shortest_path**': encapsula o algoritmo A* implementado na etapa anterior.

Módulo 'apps.py': Este módulo é utilizado para configurar a aplicação e definir suas propriedades. Nela foi criada a classe 'CoreConfig', que possui o método 'ready()', executado quando a aplicação está pronta para ser usada. Este método inicializa a instância da classe 'Graph' (singleton) de 'algorithms.py', carregando o grafo (GraphML) necessário. Após esta etapa, todas as chamadas à instância da classe 'Graph' são únicas, pois ela é singleton.

Módulo 'views.py': O módulo 'views.py' define como as requisições são tratadas pelos endpoints da API. Dentro dele, a view 'RotaView' recebe uma requisição GET com os parâmetros de origem e destino. A origem consiste nas coordenadas da localização atual do usuário, enquanto o destino é uma string coletada do input de pesquisa da aplicação Android. A função 'geocode_finder' de 'algorithms.py' é chamada para converter o destino em coordenadas, e a função 'nearest_node' é utilizada para encontrar os nós correspondentes no grafo. Em seguida, a função 'shortest_path' é chamada para calcular e retornar a lista de nós e coordenadas representando o caminho mais curto.

Módulo 'urls.py': Este módulo define o roteamento das URLs da aplicação, mapeando cada URL para sua respectiva view. A configuração seguiu as recomendações da documentação do Django Rest Framework, com a adição da view 'RotaView' na URL '/path'.

Esta etapa consistiu na criação do servidor backend, proporcionando uma interface robusta para a comunicação entre a aplicação Android e o servidor Django.

5.5
Aplicação Android

Para disponibilizar ao usuário o caminho calculado pelo algoritmo A* modificado, foi desenvolvida uma aplicação Android utilizando a linguagem Java e o Android Studio como ambiente de desenvolvimento. Essa aplicação tem como objetivo obter a localização do usuário, permitir a busca de um destino e solicitar ao backend a rota correspondente, já levando em conta o índice de insegurança.

5.5.1
Arquivo AndroidManifest.xml

O arquivo *AndroidManifest.xml* é responsável pelas configurações essenciais da aplicação Android, incluindo permissões, componentes (activities, serviços) e outras propriedades. Para este projeto, foram adicionadas as seguintes permissões:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Essas permissões garantem que o aplicativo possa realizar requisições HTTP ao backend (INTERNET), bem como acessar a localização precisa (ACCESS_FINE_LOCATION) ou aproximada (ACCESS_COARSE_LOCATION) do dispositivo. Dessa forma, a aplicação pode obter a posição atual do usuário e solicitar rotas ao servidor.

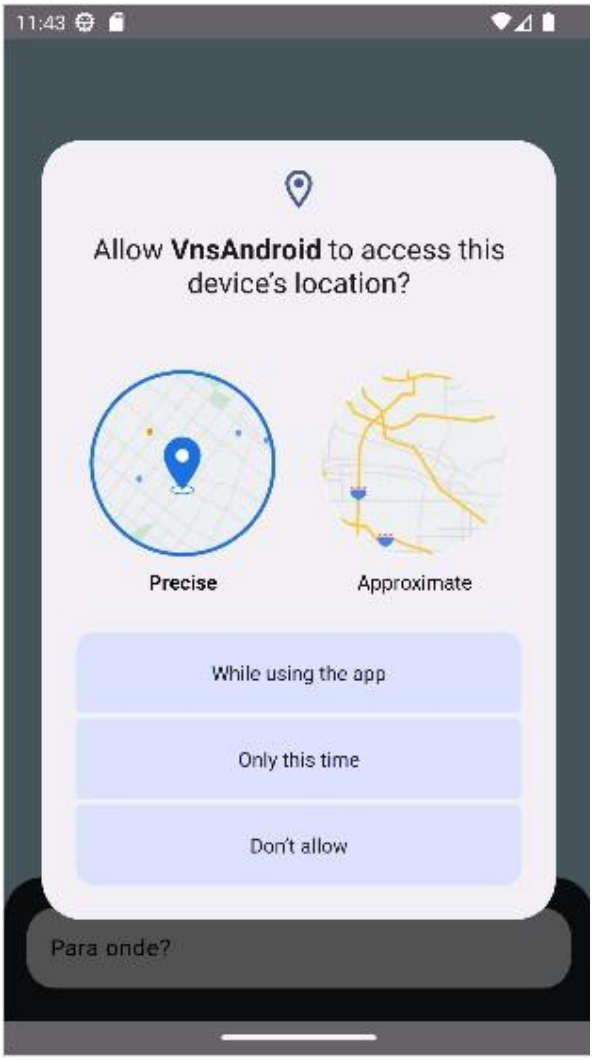


Figura 5.4: Tela de Permissão

5.5.2 Activities

A aplicação Android é organizada em (*Activities*), cada uma representando uma tela ou funcionalidade principal. Nesta aplicação, foram utilizadas duas Activities:

1. **MainActivity**: Esta é a tela inicial da aplicação. Ao ser iniciada, a *MainActivity* exibe um mapa e uma barra de busca (campo de pesquisa). Nesta Activity:

- As permissões de localização são solicitadas ao usuário. Assim que o usuário as concede, o aplicativo obtém as coordenadas atuais do dispositivo e as exibe no mapa.
- Ao inserir um endereço de destino na barra de busca, o aplicativo envia uma requisição HTTP ao endpoint do backend, incluindo a posição atual do usuário e a string referente ao endereço de destino.
- Quando o backend retorna a rota calculada, já considerando o índice de insegurança, a *MainActivity* abre a segunda Activity para exibição da rota.

Em resumo, a *MainActivity* inicia o aplicativo, solicita permissões, obtém a posição do usuário, permite a busca por um destino e envia a requisição ao servidor para obter a rota.

2. **RotaActivity**: Esta é a segunda tela, exibida após o cálculo da rota pelo backend. Sua função é simples: mostrar o mapa com a rota traçada entre a origem (posição do usuário) e o destino especificado. Nesta Activity:

   - Os dados recebidos do backend são uma lista ordenada de nós, cada um com suas coordenadas (x, y).
   - O mapa é atualizado para exibir o caminho como uma linha sobreposta, indicando o trajeto a ser seguido.

   Diferentemente da *MainActivity*, a *RotaActivity* não solicita permissões nem faz novas requisições ao servidor. Sua única responsabilidade é exibir a rota já obtida.

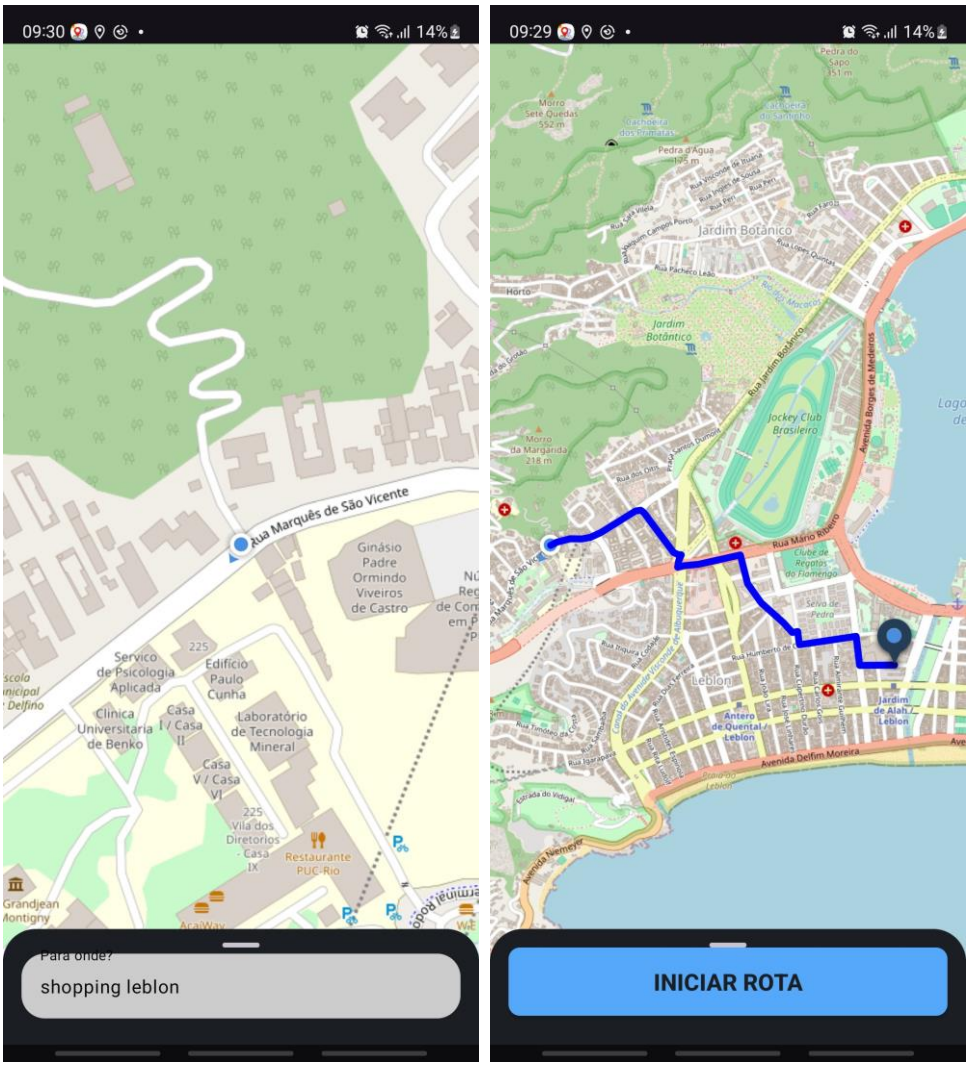


Figura 5.5: MainActivity e RotaActivity

5.5.3
Fluxo de Operação da Aplicação

O funcionamento da aplicação pode ser descrito em etapas:

1. O usuário inicia a aplicação, carregando a *MainActivity*, que solicita as permissões de localização. Assim que o usuário as concede, a posição atual é obtida e mostrada no mapa.

2. O usuário insere um destino na barra de busca. A *MainActivity* envia uma requisição ao backend com a coordenada atual e a string do destino.

3. O backend, por meio do algoritmo A* modificado, calcula uma rota que considera tanto o percurso quanto a segurança do trajeto.

4. A *MainActivity* recebe a lista de nós retornada pelo backend e, em seguida, inicia a *RotaActivity*.

5. Na *RotaActivity*, a rota é desenhada no mapa, permitindo que o usuário visualize o caminho seguro e eficiente entre a origem e o destino.

5.6
Resultados

Para avaliar a eficácia do algoritmo A* modificado, foram comparadas duas rotas geradas pelo sistema: uma utilizando o A* convencional, que considera apenas o custo tradicional (distância), e outra utilizando o A* modificado, que incorpora o índice de insegurança. Ambas as rotas têm como origem a PUC-Rio e destino o Shopping Rio Sul.

A Figura 5.6 apresenta as rotas geradas, destacando a rota vermelha (calculada pelo A* convencional) e a rota azul (calculada pelo A* modificado). A rota vermelha atravessa a região de Copacabana, enquanto a rota azul passa pelo Jardim Botânico.

Figura 5.6: Comparação entre rotas geradas pelo A* convencional (vermelho) e A* modificado (azul).

5.6.1
Análise do caminho seguro

A escolha do caminho seguro (rota azul) foi baseada no índice de insegurança calculado a partir dos dados de criminalidade disponíveis nos arquivos CSV. No índice de insegurança discreto calculado para bairros e comunidades pacificadas, o bairro de Copacabana já apresentava um índice de insegurança maior em comparação ao Jardim Botânico. Essa diferença inicial foi reforçada pelo processo de krigagem, que estimou valores mais detalhados para os nós do grafo com base na distribuição espacial do índice.

No caso da rota que atravessa Copacabana, é provável que o índice de insegurança tenha sido ainda mais influenciado pela proximidade da comunidade do Vidigal, incluída nos dados e que, historicamente, apresenta valores altos de insegurança. Essa influência aumenta o índice de insegurança das áreas adjacentes, impactando significativamente o caminho por Copacabana.

Por outro lado, a rota que passa pelo Jardim Botânico manteve um índice de insegurança consistentemente mais baixo. Isso demonstra que o algoritmo A* modificado, ao incorporar o índice de insegurança no cálculo, foi capaz de identificar e priorizar uma rota mais segura, evitando áreas de maior risco.

5.7
Conclusão

Este trabalho teve como objetivo propor uma solução que integrasse dados sobre segurança urbana ao cálculo de rotas, buscando criar um sistema capaz de considerar não apenas critérios tradicionais de custo, mas também o índice de insegurança associado a diferentes regiões. Para isso, foram processados dados oficiais da Polícia do Estado do Rio de Janeiro, empregando métodos de geoestatística, como a krigagem, e modificando o algoritmo A* para incorporar o índice de insegurança ao cálculo da rota.

A utilização da krigagem permitiu estimar o índice de insegurança em áreas não cobertas pelos dados originais, ampliando a compreensão da distribuição espacial do risco. Dessa forma, obteve-se uma melhor visualização do perigo urbano, o que pode auxiliar no planejamento de rotas mais seguras. No entanto, algumas limitações emergiram devido à ausência de dados sobre determinadas comunidades de risco e bairros nos arquivos CSV analisados. Isso restringiu a abrangência da análise, evidenciando que a qualidade e a precisão das estimativas dependem diretamente da disponibilidade e da abrangência dos dados.

Como perspectiva para trabalhos futuros, sugere-se incorporar novas fontes de informação, buscando cobrir as regiões não contempladas pelos dados iniciais. Além disso, a criação de um sistema de *crowdsourcing*, no qual usuários forneçam feedbacks de segurança em tempo real sobre determinadas áreas, poderia enriquecer a base de informações. Esses feedbacks, validados por outros usuários (por exemplo, por meio de um sistema de curtidas), formariam uma fonte contínua e dinâmica de dados. Com o acúmulo desses dados ao longo do tempo, seria possível atualizar o índice de insegurança de forma mais fiel e, eventualmente, integrá-los a modelos de aprendizado de máquina para aprimorar a previsão da segurança de rotas específicas.

Em síntese, o estudo demonstrou que técnicas clássicas da computação, combinadas a métodos de geoestatística e a um algoritmo de busca modificado, podem fornecer um ponto de partida para incorporar índices de segurança no planejamento de rotas urbanas. Com a possibilidade de integrar dados adicionais e abordagens colaborativas, há um caminho promissor para melhorar a qualidade e a confiabilidade das estimativas, resultando em rotas mais seguras e adequadas às condições reais encontradas nas cidades.

6
Referências bibliográficas

ATKISON, M. L. T. Vanet applications: Past, present, and future. 2020. Disponível em: <https://www.sciencedirect.com/science/article/abs/pii/S2214209620300814?fr=RR-2&ref=pdf_download&rr=81b50449e9fc2553>.

CAMARGO, E. C. G. Geoestatística: Fundamentos e aplicaÇÕes. **dpi.inpe**, 1998.

CLAUSSEN, H. **Modern Cartography Series**. [s.n.], 1991. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780080402772500192>.

CORTES ADELAR FOCHEZATTO, P. d. A. J. R. X. Crimes nos municípios do rio grande do sul: Análise a partir de um Índice geral de criminalidade. 2018.

EDUCAçãO, M. **Distância entre dois pontos**. 2024. Disponível em: <https://mundoeducacao.uol.com.br/matematica/distancia-entre-dois-pontos.htm>.

GALO, C. R. de Aguiar; Paulo de O. C. M. TransformaÇÃo de coordenadas e datum com propagaÇÃo de covariÂncias. **Anais do Simpósio Brasileiro de Geomática**, 2002.

G.MANCHUK, J. Conversion of latitude and longitude to utm coordinates. **CCG Annual Report**, 2009.

GOMES, M. B. de A. M. S. L. Dino: Roteamento dinâmico para its considerando recursos da rede. 2021. Disponível em: <https://periodicos.univali.br/index.php/acotb/article/view/17601/10069>.

JANSSEN. Understanding coordinate reference systems, datums and transformations. **t International Journal of Geoinformatics**, 2009.

LANDE, H. M. K. A review and evaluations of shortest path algorithms. 2013. Disponível em: <https://www.researchgate.net/profile/Magzhan-Kairanbay/publication/310594546_A_Review_and_Evaluations_of_Shortest_Path_Algorithms/links/5ac49631a6fdcc1a5bd06106/A-Review-and-Evaluations-of-Shortest-Path-Algorithms.pdf>.

LANDE, H. M. K. Google maps. 2019. Disponível em: <https://www.researchgate.net/profile/Pratik-Kanani/publication/333117435_Google_Maps/links/5eb6cc7da6fdcc1f1dcb10aa/Google-Maps.pdf>.

LANDIM, P. M. B. Sobre geoestatística e mapas. **UNICAMP**, 2006.

.MCBRATNEY, A. The design of optimal sampling schemes for local estimation and mapping of regionalized variables–i. 1981.

NARDO, M. Tools for composite indicators building. 2005.

NETTO, M. L. S. N. F. Análise e comparação dos algoritmos de dijkstra e a*estrela na descoberta de caminhos mínimos em mapas de grade. 2018. Disponível em: <https://sol.sbc.org.br/index.php/etc/article/download/9852/9750/>.

PEREIRA, W. C. Simulação geoestatítica via métodos convolutivos. 2011.

RUSSEL, S. **Inteligência Artificial**. [S.l.: s.n.], 2013.

SANTOS, D. dos. Uma introduÇÃo ao semivariograma: E sua importÂncia dentro da geoestatistica. 2018.

SILVA, F. R. da. **Categorização dos Crimes Violentos no Brasil**. [S.l.], 2016.

SILVA, L. A. M. da. "violÊncia urbana", seguranÇa pÚblica e favelas - o caso do rio de janeiro atual. 2010.

SOUZA, A. M. de. Towards a personalized multi-objective vehicular traffic re-routing system. 2021. Disponível em: <https://repositorio.unicamp.br/acervo/detalhe/1166113>.

SYAHPUTRA, M. R. W. S. H. N. G. D. Greedy, a-star, and dijkstra'salgorithms in findingshortest path. **International Journal of Advances in Data and Information Systems**, 2021. Disponível em: <http://ijadis.org/index.php/ijadis/article/view/greedy-a-star-and-dijkstras-algorithms-in-finding-shortest-path/22>.

UZUMAKI, E. T. GeoestatÍstica multivariada: Estudo de mÉtodos de prediÇÃo. 1994.

VILLAS, A. M. de S. L. C. B. I. C. G. L. A. Por aqui é mais seguro: Melhorando a mobilidade e a segurança nas vias urbanas. 2018. Disponível em: <https://sol.sbc.org.br/index.php/sbrc/article/view/2475>.

YASOJIMA, C. T. K. Modelo de krigagem automática baseada em agrupamento. 2020.

Y.CHEN, K. An improved a* search algorithm for road networks using new heuristic estimation. **https://arxiv.org/**, 2022. Disponível em: <https://arxiv.org/abs/2208.00312>.