UNIVERSIDADE FEDERAL DE VIÇOSA

CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS

DEPARTAMENTO DE INFORMÁTICA

ERICK SOARES DE SOUZA

OTIMIZAÇÃO DE ROTAS URBANAS CONSIDERANDO CONGESTIONAMENTOS MODELADOS POR POLÍGONOS MÓVEIS

VIÇOSA - MINAS GERAIS

2025

ERICK SOARES DE SOUZA

OTIMIZAÇÃO DE ROTAS URBANAS CONSIDERANDO CONGESTIONAMENTOS MODELADOS POR POLÍGONOS MÓVEIS

Monografia apresentada à Universidade Federal de Viçosa como parte das exigências para a aprovação na disciplina Projeto Final de Curso.

Orientador: Salles Viana Gomes de Magalhães

VIÇOSA - MINAS GERAIS

2025

Aos meus pais, pelo apoio e incentivo dentro das nossas possibilidades, aos amigos e familiares que sempre estiveram presentes, e a Deus, fonte de força e sabedoria. Um carinho especial ao meu irmãozinho, que, com apenas sete anos, trouxe alegria e inspiração ao longo de toda a jornada.

RESUMO

SOUZA, Erick Soares, Universidade Federal de Viçosa, novembro de 2025. **OTIMIZAÇÃO DE ROTAS URBANAS CONSIDERANDO CONGESTIONAMENTOS MODELADOS POR POLÍGONOS MÓVEIS**. Orientador: Salles Viana Gomes de Magalhães.

Com o crescimento das cidades e o aumento constante do fluxo de veículos, torna-se cada vez mais necessária a melhoria da mobilidade urbana. Usuários buscam otimizar o tempo gasto no trânsito escolhendo rotas mais curtas; no entanto, é fundamental considerar também o tempo efetivamente percorrido em cada trajeto, e não apenas a distância. Este trabalho encontra caminhos entre dois pontos levando em consideração possíveis eventos de congestionamento, modelados como polígonos móveis que representam regiões de tráfego lento. Para a detecção das interseções entre esses polígonos e as arestas do grafo viário, utiliza-se uma grade uniforme. Os resultados obtidos são comparados às rotas calculadas com base apenas na distância, analisando a influência dos congestionamentos previstos no planejamento das trajetórias.

Palavras-chave: mobilidade urbana; caminho mínimo; congestionamento; polígonos móveis; grade uniforme.

ABSTRACT

SOUZA, Erick Soares, Universidade Federal de Viçosa, November, 2025. **OPTIMIZATION OF URBAN ROUTES CONSIDERING TRAFFIC CONGESTIONS MODELED AS MOVING POLYGONS**. Adviser: Salles Viana Gomes de Magalhães.

With the growth of cities and the constant increase in vehicle flow, improving urban mobility becomes increasingly necessary. Users seek to optimize the time spent in traffic by choosing shorter routes; however, it is essential to also consider the actual time taken for each trip, not just the distance. This work finds paths between two points taking into account possible congestion events, modeled as moving polygons that represent slow traffic regions. To detect intersections between these polygons and the edges of the road network graph, a uniform grid is used. The results obtained are compared to routes calculated based solely on distance, analyzing the influence of predicted congestion on route planning.

Keywords: urban mobility; shortest path; congestion; moving polygons; uniform grid.

LISTA DE ILUSTRAÇÕES

Figura 1 – Diagrama de Classes Simplificado da Arquitetura do Sistema 14
Figura 2 – Representação da Grade Uniforme sobre o Grafo Viário 15
Figura 3 – Visualização da Simulação com Agentes e Polígonos 18
Figura 4 – Visualização da Simulação com Diferentes Perspectivas 18
Figura 5 – Comparação do Espaço de Busca: Expansão Uniforme vs. Direcionada . . . 19
Figura 6 – Comparação entre as Distâncias Percorridas nas Rotas 28
Figura 7 – Distribuição da Variação de Distância 29
Figura 8 – Classificação do Desempenho das Simulações 29
Figura 9 – *Trade-off* entre Tempo de Processamento e Redução de Iterações 30

LISTA DE TABELAS

Tabela 1 – Tempo Médio de Detecção de Arestas que Intersectam um Polígono 16
Tabela 2 – Comparação de Desempenho entre Algoritmos de Caminho Mínimo 20
Tabela 3 – Definição dos Cenários de Teste 25
Tabela 4 – Comparativo de Tempo Total de Processamento (ms) 25
Tabela 5 – Análise de Replanejamento e Chamadas ao Algoritmo $A*$ 26
Tabela 6 – Comparativo de Desempenho 30

SUMÁRIO

1 INTRODUÇÃO 8
2 REVISÃO DE LITERATURA 10
3 OBJETIVOS 12
3.1 Objetivo Geral 12
3.2 Objetivos Específicos 12
4 METODOLOGIA 13
4.1 Modelagem dos Dados 13
4.2 Projeto do Sistema 13
4.2.1 Arquitetura Geral 14
4.2.1.1 Grafo Dinâmico 14
4.2.1.2 Grade Uniforme 15
4.2.1.3 Polígono 17
4.2.1.4 Agente 17
4.2.1.5 Interface Gráfica 17
4.2.2 Algoritmos Principais 18
4.2.2.1 Caminho Mínimo 19
4.2.2.2 Ponto em Polígono Convexo 21
4.2.2.3 Movimento dos Agentes 22
5 RESULTADOS E DISCUSSÃO 23
5.1 Configuração Experimental 23
5.2 Métricas de Avaliação 23
5.3 Cenários de Teste 24
5.4 Análise dos Resultados 25
5.4.1 Impacto Computacional e Tempo de Execução 25
5.4.2 Frequência de Replanejamento e Chamadas ao $A*$ 26
5.4.3 Eficiência das Rotas e Qualidade da Navegação 26
5.4.4 Impacto da Paralelização em Cenários de Alta Densidade 30
6 CONCLUSÃO 32
6.1 Trabalhos Futuros 32

REFERÊNCIAS 33

1 INTRODUÇÃO

A mobilidade urbana compreende não apenas o deslocamento de pessoas e bens no espaço urbano, mas também a efetiva possibilidade de acesso às oportunidades sociais e econômicas disponíveis na cidade. De acordo com a Lei nº 12.587/2012, que institui a Política Nacional de Mobilidade Urbana, cabe aos gestores municipais o planejamento e a execução das políticas de mobilidade, com o objetivo de assegurar a eficiência e eficácia na circulação urbana, bem como promover a equidade no acesso aos sistemas de transporte público, entre outras diretrizes (Brasil, 2012).

No entanto, a realidade brasileira revela entraves que comprometem a eficiência da circulação nos espaços urbanos. Pesquisas recentes apontam que 36% dos habitantes de grandes centros gastam ao menos uma hora diária em deslocamentos, refletindo a sobrecarga dos sistemas de transporte e da infraestrutura viária (Confederação Nacional da Indústria, 2024). No panorama internacional, o país também se destaca negativamente: em 2022, São Paulo e Belo Horizonte ocuparam, respectivamente, a 48ª e a 49ª posição no ranking global de congestionamento, segundo relatório da INRIX (2022). Esses indicadores evidenciam a urgência de intervenções que promovam maior fluidez no tráfego e ampliem as condições de mobilidade para a população.

Aplicativos de navegação por GPS, como o *Waze* e o *Google Maps*, utilizam dados em tempo real para sugerir rotas que minimizem o tempo de deslocamento dos usuários, mas sua eficácia está condicionada à quantidade de usuários ativos que alimentam o sistema com informações de tráfego, o que pode comprometer a precisão das estimativas em regiões com menor adesão. Em Joinville (SC), por exemplo, a iniciativa *Waze for Cities Data* — que compartilha dados de trânsito coletados pelo aplicativo com órgãos públicos e prestadores de serviços — apresentou resultados promissores em áreas de menor extensão, contribuindo para a formulação de políticas públicas voltadas à melhoria da mobilidade urbana. Contudo, em trechos mais longos, sua efetividade mostrou-se limitada pela escassez de pontos de coleta suficientes para uma análise mais abrangente do tráfego (Hiroki, 2021).

Este trabalho aborda um problema semelhante ao dos aplicativos de navegação, com o objetivo de encontrar a menor rota entre dois pontos, considerando possíveis áreas de congestionamento, que, por simplicidade, são modeladas como polígonos móveis. Essa abordagem abre margem para estudos futuros que possam incluir uma análise temporal de dados para gerar as trajetórias desses polígonos de forma mais precisa. Dessa forma, introduz-se a utilização de grafos dinâmicos, uma vez que essas áreas de grande intensidade de tráfego alteram o tempo médio de percurso das vias e exigem o cálculo das interseções com o grafo, tarefa que é realizada utilizando uma grade uniforme como índice para acelerar o processamento.

Com isso, esta pesquisa propõe uma abordagem que busca equilibrar distância, tempo de

percurso e atrasos causados por congestionamentos, oferecendo uma alternativa aos métodos tradicionais baseados apenas na condição do tráfego no momento da consulta. Na sequência, o Capítulo 2 apresenta a fundamentação teórica e os trabalhos relacionados que embasam a proposta. O Capítulo 3 estabelece os objetivos da pesquisa, enquanto o Capítulo 4 descreve detalhadamente a metodologia, incluindo a implementação da grade uniforme e dos agentes de navegação. A análise quantitativa da abordagem, comparando o desempenho entre rotas estáticas e dinâmicas, é apresentada no Capítulo 5. Por fim, o Capítulo 6 reúne as considerações finais e aponta perspectivas para a continuidade do projeto.

2 REVISÃO DE LITERATURA

Prever com precisão as zonas de congestionamento urbano é uma tarefa desafiadora, pois envolve uma série de variáveis complexas, que vão desde os padrões horários de fluxo intenso até eventos imprevisíveis, como acidentes e obras emergenciais. No estudo de Zhang *et al.* (2023), os autores propuseram uma abordagem baseada em redes neurais convolucionais, treinadas com dados espaço-temporais. A pesquisa foi aplicada em duas cidades distintas e os resultados indicaram uma alta precisão na previsão dos padrões de congestionamento, superando métodos tradicionais e reforçando o potencial de modelos baseados em dados para sistemas de navegação mais inteligentes.

Uma das abordagens mais fundamentais e eficientes para encontrar o caminho mínimo entre dois pontos em um grafo com pesos não-negativos foi proposta por Dijkstra (1959). Desenvolvido na década de 1950, o algoritmo de *Dijkstra* permanece como um pilar teórico na área de teoria dos grafos e no desenvolvimento de sistemas modernos de roteamento e planejamento de rotas. Sua simplicidade e eficiência garantiram sua adoção em uma ampla variedade de aplicações, servindo também como base para o aprimoramento de métodos mais sofisticados.

Um exemplo relevante é o estudo de Bast *et al.* (2016), que apresenta uma análise abrangente de algoritmos avançados para o planejamento de rotas em redes de transporte. Entre as técnicas discutidas, destaca-se a *contraction hierarchies*, que incorpora o algoritmo de *Dijkstra* em uma de suas etapas para otimizar o cálculo de rotas em grandes grafos. O trabalho também explora o equilíbrio entre custos de pré-processamento, requisitos de armazenamento e tempo de consulta, evidenciando os desafios adicionais do planejamento de rotas em sistemas de transporte público e ambientes urbanos, nos quais múltiplos fatores interagem e há demanda por respostas em tempo quase instantâneo.

A técnica de *contraction hierarchies* baseia-se na ordenação dos nós de um grafo de acordo com sua importância, de forma a construir uma hierarquia que facilita as consultas de rotas. Essa abordagem permite reduzir drasticamente o tempo de cálculo, alcançando desempenho até cinco vezes superior em relação a outras técnicas hierárquicas baseadas no algoritmo de *Dijkstra*, além de demandar menor espaço de armazenamento (Geisberger *et al.*, 2008).

Novos métodos têm sido constantemente desenvolvidos com o objetivo de alcançar desempenho igual ou superior ao do algoritmo tradicional. Um exemplo recente é o trabalho de Duan *et al.* (2025), no qual foi possível obter melhor desempenho em grafos direcionados com pesos reais não negativos, demonstrando que o algoritmo de *Dijkstra* não é, em todos os casos, a solução ótima em termos de tempo de execução.

Um dos principais desafios no cálculo de rotas em redes urbanas é a natureza dinâmica do grafo, onde os tempos de deslocamento pelas vias sofrem alterações frequentes devido a fatores como congestionamentos e incidentes. Para lidar com essa variabilidade, novos estudos foram desenvolvidos com foco em algoritmos que consideram modificações constantes nas condições do tráfego. No trabalho de Demiryurek *et al.* (2011), os autores propuseram uma variação do algoritmo *A**, adaptada para redes com tempos de viagem dependentes do tempo, simulando cenários mais realistas. De forma complementar, Jiang e Wu (2014) apresentaram algoritmos de reotimização dinâmica que conseguem ajustar as rotas em tempo real, demonstrando desempenho superior aos métodos tradicionais frente às mudanças no tempo dos arcos.

Grafos dinâmicos são estruturas em que as conexões (arestas) ou os próprios vértices podem sofrer alterações ao longo do tempo, seja pela inserção ou remoção de elementos, ou pela modificação dos pesos associados às arestas. No contexto deste trabalho, essa dinamicidade é refletida na variação do tempo médio necessário para atravessar uma via, permitindo que o grafo represente de forma mais realista as condições mutáveis do tráfego urbano.

Além disso, a representação dos fluxos de trânsito como polígonos móveis exige o suporte teórico e algorítmico da geometria computacional. Nesse contexto, o livro *Computational Geometry: Algorithms and Applications*, de Berg (2000), constitui uma referência fundamental, reunindo algoritmos eficientes para operações como interseção de polígonos, triangulações e consultas espaciais.

Para garantir eficiência na detecção de interseções entre rotas planejadas e zonas de congestionamento representadas como polígonos móveis, é essencial recorrer a técnicas de detecção de colisões. O livro de Ericson (2004) é uma referência prática nessa área, apresentando algoritmos otimizados como árvores de volumes limitadores (*bounding volumes*) e grades uniformes. Essas técnicas, amplamente utilizadas em aplicações que exigem resposta em tempo real, como jogos e simulações, oferecem soluções eficazes para o problema de sobreposição entre trajetórias e áreas congestionadas em ambientes urbanos dinâmicos.

Magalhães *et al.* (2015) propõem o uso de grades uniformes como estrutura auxiliar para acelerar a computação do caminho mínimo em redes viárias. Ao particionar o espaço em células regulares, o método restringe a busca a regiões relevantes, reduzindo o número de vértices processados e melhorando o desempenho em consultas de rotas. Essa estratégia se alinha com a abordagem adotada neste trabalho, que também utiliza grades espaciais para otimizar a detecção de interseções entre trajetórias e zonas de congestionamento, garantindo uma resposta mais eficiente em cenários urbanos dinâmicos.

3 OBJETIVOS

3.1 Objetivo Geral

Desenvolver e avaliar uma abordagem para a otimização de rotas em sistemas de navegação urbana, levando em consideração tanto a distância quanto os efeitos de congestionamentos temporários, modelados como polígonos móveis, no planejamento das trajetórias entre dois pontos.

3.2 Objetivos Específicos

- Modelar áreas de tráfego lento como polígonos móveis, simulando a dinâmica dos congestionamentos ao longo do tempo;
- Implementar uma técnica baseada em grade uniforme para detectar as interseções entre os congestionamentos e as arestas do grafo viário;
- Comparar a metodologia proposta com os métodos tradicionais baseados apenas na distância, considerando os impactos dos congestionamentos no tempo de viagem;
- Avaliar a eficácia da abordagem proposta, analisando a redução no tempo de deslocamento e o impacto na mobilidade urbana.

4 METODOLOGIA

Esta pesquisa propõe a modelagem de áreas viárias com o objetivo de identificar rotas entre diferentes pontos, incorporando fatores de atraso no cálculo dos caminhos. Tais atrasos são representados por polígonos móveis, que simulam regiões de congestionamento e serão detalhados ao longo desta seção. Dessa forma, trata-se de uma abordagem de caráter exploratório e quantitativo (Prodanov; Freitas, 2013), voltada para a busca de soluções mais eficientes do que aquelas oferecidas por métodos tradicionais, os quais, em geral, consideram apenas a menor distância entre os pontos, ignorando as dinâmicas do tráfego urbano.

Um exemplo concreto seria a modelagem de nuvens de chuva sobre uma cidade: à medida que a chuva se desloca como um polígono, ela pode interromper o tráfego nas vias que atravessa, representando um fator de atraso dinâmico.

4.1 Modelagem dos Dados

A malha viária foi modelada como um grafo direcionado. Cada vértice do grafo corresponde a uma interseção (esquina), associada às suas coordenadas geográficas (latitude e longitude), enquanto as arestas representam os trechos de rua que conectam essas interseções, respeitando o sentido do fluxo de tráfego e a distância em metros referente a cada percurso. Os dados que compõem essa rede foram extraídos utilizando a biblioteca `osmnx` da linguagem *Python* (Boeing, 2017), que facilita a obtenção e manipulação de informações do *OpenStreetMap*.

Para a composição dos cenários de simulação, os agentes de tráfego têm seus pontos de origem e destino selecionados aleatoriamente entre os vértices do grafo, garantindo que sejam distintos e que exista um caminho conexo entre eles. Essa abordagem permite avaliar o algoritmo de roteamento sob diferentes configurações de trajeto dentro da mesma malha viária.

Os congestionamentos, por sua vez, são introduzidos como polígonos móveis iniciados em células aleatórias da grade uniforme que contenham arestas, assegurando que os obstáculos surjam efetivamente sobre as vias. A movimentação desses polígonos ocorre a cada iteração do sistema através de um vetor de deslocamento predefinido, simulando a dinâmica contínua de bloqueios ou zonas de lentidão que percorrem a cidade independentemente do fluxo dos agentes.

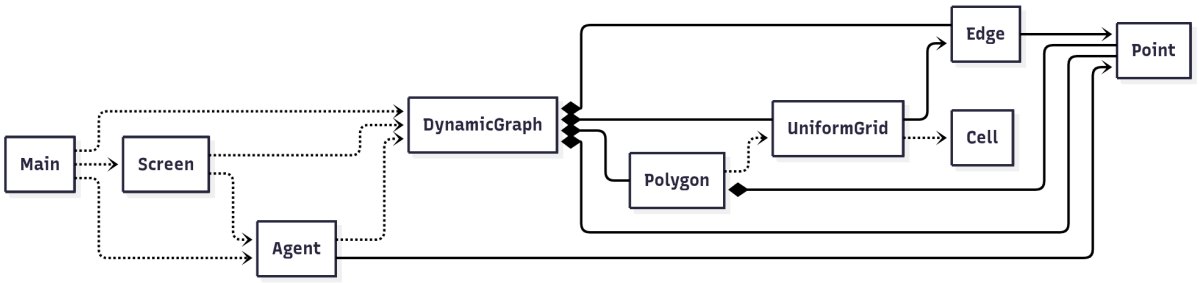
4.2 Projeto do Sistema

Para o desenvolvimento do projeto, foi estruturada uma arquitetura capaz de processar dados do grafo viário, gerenciar agentes em movimento e lidar com obstáculos dinâmicos de forma eficiente. Esta seção descreve o fluxo integrado do sistema e seus principais componentes.

4.2.1 Arquitetura Geral

O sistema foi organizado em módulos independentes, mas inter-relacionados, de forma a facilitar tanto a implementação quanto a extensibilidade do projeto. A Figura 1 apresenta uma visão geral dos componentes e das interações entre eles.

Figura 1 – Diagrama de Classes Simplificado da Arquitetura do Sistema



Fonte: Autoria própria.

O fluxo de execução segue um ciclo contínuo inspirado no *gameloop*, abordagem comum em motores de jogos descrita por Nystrom (2014). Essa estratégia foi adotada porque o sistema também possui um modo de visualização interativa, e porque a dinâmica dos congestionamentos requer atualização constante do estado interno. Cada iteração do ciclo realiza as etapas retratadas no Algoritmo 1.

Algoritmo 1: Ciclo Principal de Execução do Sistema

```
Entrada : Estruturas do sistema: polígonos móveis, agentes, grafo viário, grade uniforme
Enquanto aplicação está ativa
    Se modo_visual
        processarEventosDeInteração();
        atualizarInterface();
    atualizarPoligonos();
    moverAgentes();
    Se modo_visual
        renderizarFrame();
```

Essa arquitetura permite que o sistema reaja dinamicamente às mudanças do ambiente sem comprometer o desempenho, mesmo com múltiplas entidades em movimento. A comunicação entre os módulos ocorre por meio de interfaces estáveis; o grafo viário funciona como base espacial; a grade uniforme acelera consultas de proximidade; e os agentes utilizam ambas as estruturas para navegar de forma inteligente no ambiente.

4.2.1.1 Grafo Dinâmico

Como a duração de deslocamento nas arestas pode variar conforme a presença e o movimento dos polígonos de congestionamento, o grafo viário é tratado como um grafo dinâmico.

Nesse modelo, os pesos das arestas, que representam o custo de percorrer cada trecho, são atualizados ao longo do tempo, refletindo as condições mutáveis do tráfego urbano.

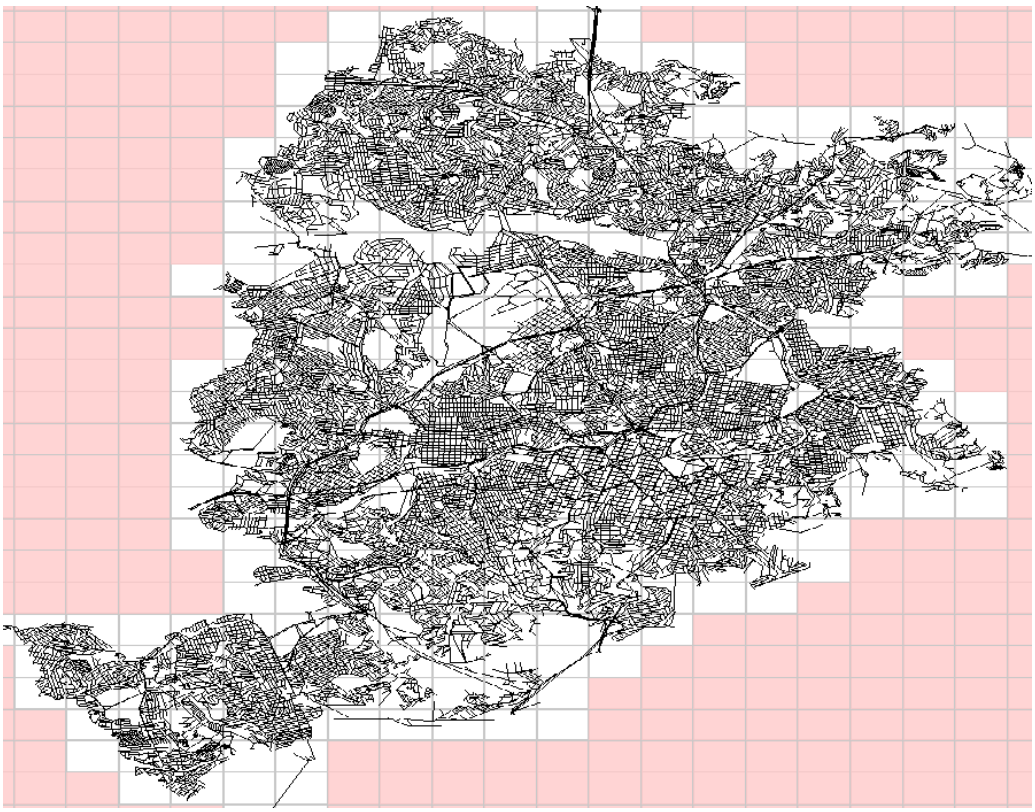
Para lidar com essas atualizações, o sistema mantém uma lista de adjacência para representar o grafo, uma grade uniforme que acelera a detecção de interseções entre arestas e polígonos, e o conjunto de polígonos móveis que se deslocam sobre a malha viária e influenciam os custos das arestas. Essa integração permite recalcular os pesos de forma eficiente a cada ciclo da aplicação.

4.2.1.2 Grade Uniforme

A grade uniforme é uma estrutura de dados bastante útil quando se trabalha com informações espaciais, como em geometria computacional ou sistemas de informação geográfica (GIS). Ela funciona como uma espécie de grade regular sobreposta ao mapa da cidade, dividindo o espaço em células iguais. Isso permite agilizar operações como encontrar interseções ou verificar a proximidade entre objetos, já que as comparações passam a ser feitas apenas entre elementos que estão na mesma célula ou em células vizinhas (Magalhães, 2017).

Uma representação visual da grade uniforme com as arestas associadas às suas células pode ser observada na Figura 2.

Figura 2 – Representação da Grade Uniforme sobre o Grafo Viário



Fonte: Autoria própria.
Nota: As células em branco indicam regiões ativas (contendo arestas/ruas), enquanto as células em rosa representam regiões vazias, descartadas para otimização de busca espacial.

Neste trabalho, a grade uniforme é usada para tornar mais rápida a detecção de interseções entre os polígonos móveis e as arestas do grafo viário. Os segmentos dos polígonos e as arestas do grafo são associados às células da grade onde estão localizados. Com isso, as verificações de interseção são feitas apenas entre elementos que ocupam a mesma região, o que evita comparações desnecessárias. Esse processo torna as simulações mais leves e permite atualizar os pesos do grafo dinâmico de forma eficiente à medida que os congestionamentos se deslocam pelas ruas.

Além do benefício estrutural, essa otimização também se refletiu no desempenho. Conforme apresentado na Tabela 1, o tempo médio para identificar todas as arestas que intersectam um polígono foi reduzido de aproximadamente 50 ms para 2 ms, considerando a média de 10 execuções. Esse resultado representa um ganho de cerca de 25 vezes, evidenciando a eficiência da grade uniforme para acelerar consultas de interseção.

Tabela 1 – Tempo Médio de Detecção de Arestas que Intersectam um Polígono

Método	**Tempo médio (ms)**
Sem grade uniforme	50
Com grade uniforme	2

Fonte: Autoria própria.
Nota: Cada execução consistiu na detecção de todas as arestas que intersectam um único polígono em um ciclo de simulação para o grafo viário da cidade de Belo Horizonte (MG).

Para inserir as arestas do grafo na grade uniforme, foi utilizada uma *bounding box*, ou caixa delimitadora, que corresponde ao menor retângulo alinhado aos eixos capaz de conter um conjunto de pontos (Heckbert, 2013), neste caso, os dois extremos de cada aresta. Essa estratégia é amplamente empregada em estruturas de aceleração por fornecer uma aproximação simples e eficiente da região ocupada pelo objeto, evitando cálculos geométricos mais complexos sempre que possível.

Como as coordenadas dos vértices estão representadas em latitude e longitude, é necessário convertê-las para unidades inteiras correspondentes às posições das células da grade. Essa conversão permite percorrer todas as células que fazem parte da *bounding box* da aresta, garantindo que ela seja corretamente associada a todas as células que intercepta, assim como apresentado no Algoritmo 2.

Algoritmo 2: Inserção de Aresta na Grade Uniforme

Entrada : Aresta $e = (u, v)$; Tamanho da célula *cellSize*; Estrutura de grade *grid*

Para $i \leftarrow \lfloor \min(u_x, v_x) \div cellSize \rfloor$ ***até*** $\lfloor \max(u_x, v_x) \div cellSize \rfloor$
  Para $j \leftarrow \lfloor \min(u_y, v_y) \div cellSize \rfloor$ ***até*** $\lfloor \max(u_y, v_y) \div cellSize \rfloor$
    Insira a aresta e em $grid[i][j]$;

4.2.1.3 Polígono

Os polígonos representam as entidades dinâmicas que interagem com a malha viária, simulando fenômenos que causam bloqueios ou atrasos, como congestionamentos severos ou condições climáticas adversas. Estruturalmente, são definidos por um conjunto ordenado de vértices (pontos com coordenadas x e y) e um centróide, que facilita a manipulação de sua posição no espaço.

No contexto da simulação, essas entidades não são estáticas e possuem a capacidade de translação sobre o grafo. A movimentação é calculada com base em um deslocamento $(\Delta x, \Delta y)$, garantindo sempre que o objeto permaneça dentro dos limites operacionais definidos pela grade uniforme.

4.2.1.4 Agente

O agente é a entidade que navega pelo grafo viário, deslocando-se de um ponto de origem até um destino. Ele gerencia seu próprio estado de navegação (posição atual, velocidade e o trajeto a ser percorrido), atuando como o elemento que é afetado pelas obstruções no grafo. O movimento entre as interseções é realizado por meio de interpolação linear (Dunn; Parberry, 2011), simulando um deslocamento contínuo ao longo da aresta com base na distância geográfica e na velocidade atribuída. O comportamento de navegação divide-se em duas estratégias distintas.

A primeira, de caráter estático, calcula a rota apenas no momento da criação, ignorando quaisquer alterações que ocorram no ambiente. Isso faz com que o agente siga estritamente o caminho original, mesmo que este se torne bloqueado; neste caso, ele permanece parado aguardando o desbloqueio do caminho.

A segunda estratégia, de caráter dinâmico, permite que o agente reavalie seu trajeto em tempo real. Durante o deslocamento, essa entidade monitora a presença de polígonos em sua rota futura e, ao detectar uma intersecção, recalcula o caminho considerando os novos obstáculos. É importante ressaltar que o agente dinâmico também pode ficar retido em uma zona de interseção.

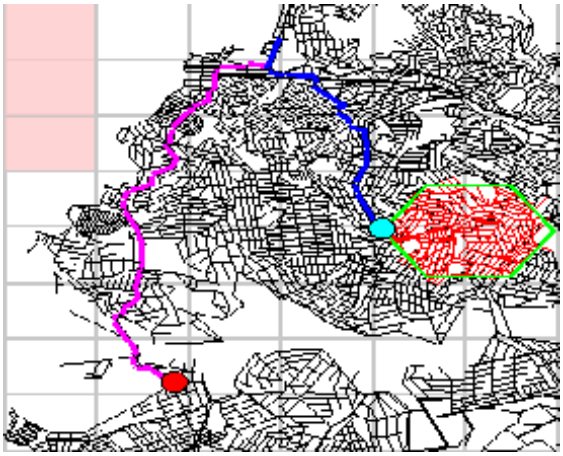
4.2.1.5 Interface Gráfica

A interface gráfica é o componente responsável pela visualização da simulação, convertendo o modelo lógico em uma representação visual. Desenvolvida sobre a biblioteca SFML (*Simple and Fast Multimedia Library*) (Haller; Hansson; Moreira, 2013), esta camada gerencia a renderização dos elementos e a interação do usuário com o ambiente simulado.

Um aspecto fundamental é a projeção de coordenadas, necessária para mapear as posições geográficas (latitude e longitude) da malha viária para o sistema de coordenadas cartesianas da tela. Essa transformação normaliza as dimensões do grafo, permitindo que a visualização mantenha a proporcionalidade correta das vias e dos obstáculos.

A renderização ocorre em camadas sobrepostas. Visando a eficiência computacional, a grade de fundo e as arestas do grafo são desenhadas inicialmente em uma textura estática, evitando o redesenho desnecessário desses elementos a cada quadro. Sobre essa base, são projetados os polígonos, que destacam em vermelho as arestas interceptadas por eles, e, por fim, são renderizados os agentes e seus respectivos trajetos, como pode ser visto na Figura 3.

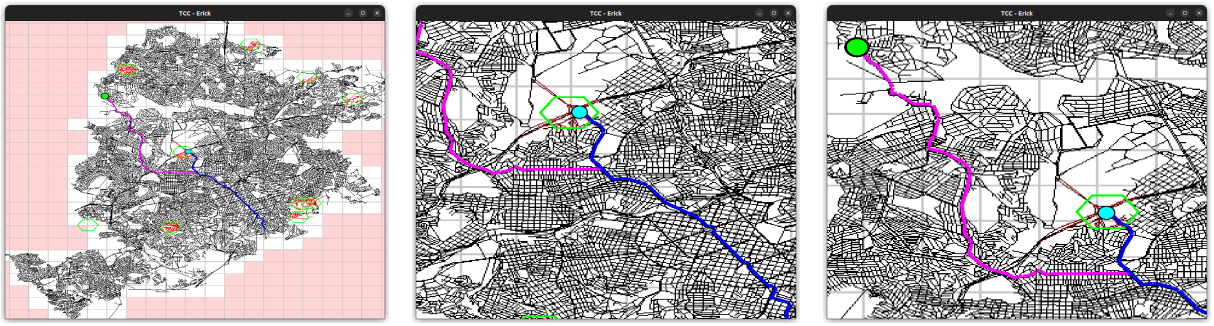
Figura 3 – Visualização da Simulação com Agentes e Polígonos



Fonte: Autoria própria.
Nota: O trajeto em magenta corresponde ao agente dinâmico, enquanto o trajeto em azul representa o agente estático.

Além da visualização estática, o sistema permite a manipulação da câmera virtual, oferecendo recursos de *zoom* e translação que facilitam a inspeção de áreas específicas durante a execução da simulação. Esses recursos são ilustrados na Figura 4, que apresenta diferentes perspectivas da cena simulada.

Figura 4 – Visualização da Simulação com Diferentes Perspectivas



(a) (b) (c)

Fonte: Autoria própria.
Nota: As imagens mostram diferentes perspectivas da simulação: (a) uma visão ampla da área, (b) uma inspeção detalhada por meio de *zoom* e (c) o movimento de translação.

4.2.2 Algoritmos Principais

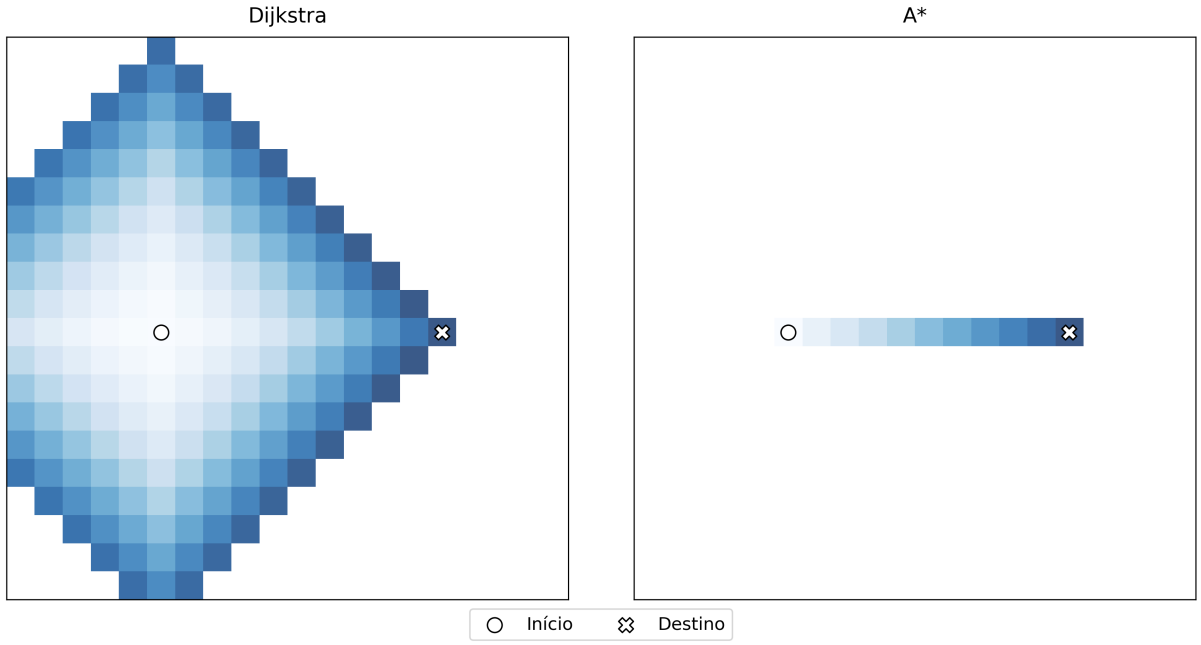
Esta seção apresenta os métodos fundamentais que compõem o sistema desenvolvido. Cada módulo algorítmico opera de forma integrada no ciclo contínuo da aplicação (Algoritmo 1),

possibilitando a simulação de rotas em uma malha viária dinâmica. A seguir, são descritas as principais técnicas utilizadas:

4.2.2.1 Caminho Mínimo

O cálculo do caminho entre os agentes é realizado por meio do algoritmo $A*$ (*A-Star*), proposto originalmente por Hart, Nilsson e Raphael (1968). Diferente do algoritmo de Dijkstra (1959), que explora uniformemente em todas as direções, o $A*$ utiliza uma heurística para guiar a busca em direção ao objetivo, tornando-o mais eficiente para rotas ponto a ponto, conforme ilustrado na Figura 5. A função de custo total é dada por $f(n) = g(n) + h(n)$, onde $g(n)$ é o custo real do início até o nó n, e $h(n)$ é a estimativa do custo de n até o destino.

Figura 5 – Comparação do Espaço de Busca: Expansão Uniforme vs. Direcionada



Fonte: Autoria própria.

Como o grafo viário utiliza coordenadas geográficas (latitude e longitude), a heurística $h(n)$ adotada foi a Fórmula de Haversine. Essa fórmula calcula a distância do "círculo máximo" entre dois pontos em uma esfera, oferecendo uma estimativa admissível da distância física restante até o alvo (Sinnott, 1984), conforme a Equação 1, onde ϕ é a latitude, λ é a longitude e r é o raio da Terra.

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{1}$$

Para o agente dinâmico, o algoritmo foi modificado para considerar os impedimentos. Durante a expansão dos vizinhos, verifica-se se a aresta intercepta algum dos polígonos ativos na

simulação. Caso haja interseção, a aresta é desconsiderada (tratada como custo infinito), forçando o algoritmo a buscar rotas alternativas, como consta no Algoritmo 3.

Algoritmo 3: $A*$ Considerando os Polígonos Móveis

Entrada : Grafo G, Origem $start$, Destino $target$, Polígonos P
Saída : Caminho recuperado ou vazio

Inicialize $gCost[\cdot] \leftarrow \infty$, $gCost[start] \leftarrow 0$;
$pq \leftarrow$ Fila de prioridade ordenada por $fCost$;
Insira $(0 + Haversine(start, target), start)$ em pq;
Enquanto *pq não está vazia*
    $u \leftarrow$ remova nó com menor $fCost$ de pq;
    Se $u == target$
        break;
    Para cada *aresta e conectando u a v em G*
        Se *IntersecaoPoligonos(e, P)*
            continue;
        $novoCusto \leftarrow gCost[u] + peso(e)$;
        Se $novoCusto < gCost[v]$
            $gCost[v] \leftarrow novoCusto$;
            $fCost \leftarrow novoCusto + Haversine(v, target)$;
            Insira $(fCost, v)$ em pq;
            $pai[v] \leftarrow u$;
return ReconstruirCaminho(pai, $target$);

A lista de adjacência é utilizada como estrutura principal de representação do grafo, sendo especialmente adequada para grafos esparsos, conforme detalhado por Cormen *et al.* (2022). Essa escolha se justifica pela economia de memória e pela eficiência nas iterações, já que, na prática, muitas ruas conectam diretamente apenas dois vértices. A estrutura permite percorrer apenas os vizinhos relevantes, otimizando o desempenho do algoritmo.

Tabela 2 – Comparação de Desempenho entre Algoritmos de Caminho Mínimo

Algoritmo	**Complexidade**	**Quantidade de Operações – Entrada 1**	**Quantidade de Operações – Entrada 2**
Dijkstra (1959)	$O(E + V \log V)$	420 232	51 867 183
Duan *et al.* (2025)	$O(E \log^{2/3} V)$	396 807	19 719 474

Fonte: Autoria própria.
Nota: V = número de vértices; E = número de arestas; Entrada 1 possui $V = 32\,433$ e $E = 83\,353$; Entrada 2 possui $V = 3\,243\,300$ e $E = 3\,243\,299$.

Embora o método de Duan *et al.* (2025) apresente melhor desempenho assintótico em grafos muito grandes, seu uso neste trabalho não ofereceu uma relação custo–benefício favorável. Como mostrado na Tabela 2, para instâncias compatíveis com a malha viária utilizada para testes

(Entrada 1), a redução no número de operações é pequena, e os ganhos relevantes só aparecem em grafos massivos (Entrada 2), fora do escopo desta pesquisa. Vale ressaltar que, no pior caso, o algoritmo $A*$ possui a mesma complexidade do algoritmo de Dijkstra, o que fundamenta a utilização deste último como base comparativa.

4.2.2.2 Ponto em Polígono Convexo

Para identificar se uma aresta possui interseção com um polígono, adotou-se uma estratégia semelhante à utilizada na geração da grade uniforme. Inicialmente, é construído um *bounding box* para o polígono, permitindo determinar quais células da grade possuem interseção com sua área. Dessa forma, a verificação de interseção é restrita apenas às arestas contidas nessas células, reduzindo significativamente o número de operações necessárias.

Como cada aresta é definida por dois pontos, considera-se que a aresta intersecta o polígono caso qualquer um desses pontos esteja localizado em seu interior. Nessa situação, a aresta é desconsiderada no cálculo do caminho mínimo, uma vez que atravessa uma região inválida.

Para realizar essa validação de forma eficiente, empregou-se o algoritmo proposto por Skala (2025), cuja complexidade é de $O(\log n)$ em função do número de vértices do polígono. Esse método, conhecido como *point-in-convex-polygon*, baseia-se na ordenação dos vértices e em operações vetoriais, e pode ser descrito pelas seguintes etapas principais:

1. **Ordenação dos vértices:** os vértices do polígono convexo devem estar organizados em sentido anti-horário, garantindo uma sequência ordenada de ângulos em relação ao centroide ou a um vértice pivô.

2. **Determinação da orientação:** utiliza-se o produto vetorial para determinar se um ponto está à esquerda ou à direita de uma aresta, possibilitando identificar a posição relativa do ponto em relação ao polígono.

3. **Busca binária:** tomando um vértice como pivô, o algoritmo divide o polígono em vários "setores" formados entre esse pivô e os demais vértices. Como esses setores estão organizados em ordem crescente de ângulo, é possível usar uma busca binária para descobrir rapidamente em qual deles o ponto está. A cada passo, compara-se a orientação do ponto com o setor do meio, decidindo se é preciso olhar para a metade anterior ou posterior.

Como os polígonos utilizados neste trabalho são construídos manualmente, seus vértices já são armazenados na ordem correta, o que dispensa uma etapa extra de processamento. Caso os vértices não sejam previamente ordenados, essa ordenação pode ser realizada uma única vez no início do algoritmo, já que o movimento do polígono não altera a sequência dos pontos.

Além disso, para fins de experimentação, o sistema implementa a geração de polígonos com formas geométricas regulares, especificamente hexágonos, que são instanciados em células aleatórias da grade que contenham arestas, assegurando que os obstáculos surjam em regiões de interesse viário.

4.2.2.3 Movimento dos Agentes

O deslocamento dos agentes pela malha viária é realizado percorrendo uma trajetória predefinida, armazenada internamente como uma sequência de identificadores de vértices. O movimento visual entre dois nós consecutivos não ocorre de forma instantânea, a posição é atualizada a cada iteração do ciclo de simulação, baseada na velocidade do agente e na distância da aresta, garantindo uma transição fluida ao longo do trecho viário.

Para os agentes configurados como dinâmicos, existe uma etapa de validação de trajeto que antecede o movimento. A fim de evitar o alto custo computacional de verificar geometricamente a interseção entre o agente e todos os polígonos a cada quadro, o sistema implementa um mecanismo de cache de células ocupadas.

Esse cache mapeia quais células da Grade Uniforme estão atualmente interceptadas por polígonos. A atualização dessa estrutura é otimizada por meio de processamento paralelo, utilizando a biblioteca *OpenMP* (Dagum; Menon, 1998). Dessa forma, o agente necessita apenas consultar se o próximo nó de seu caminho reside em uma célula marcada na cache, substituindo os cálculos geométricos de interseção por uma consulta direta à memória.

Adicionalmente, o sistema armazena o identificador da última célula onde uma interseção foi detectada e que motivou um replanejamento. Essa estratégia de "memória de colisão" impede que o algoritmo de busca de caminho seja acionado repetidamente para o mesmo obstáculo em quadros consecutivos, evitando o desperdício de recursos computacionais com recálculos redundantes.

5 RESULTADOS E DISCUSSÃO

Este capítulo apresenta a avaliação quantitativa da abordagem proposta. O objetivo dos experimentos foi medir o desempenho do algoritmo de roteamento dinâmico frente a cenários de congestionamento simulado, comparando-o com o método estático tradicional.

5.1 Configuração Experimental

A implementação foi desenvolvida em `C++`, utilizando as bibliotecas STL para a representação e manipulação dos grafos, bem como para a movimentação dos polígonos e para a estruturação da grade uniforme. Os testes e experimentos foram realizados em um notebook Dell Latitude 3 420, equipado com processador Intel® Core™ i5-1135G7 de 11ª geração, com 4 núcleos físicos e 8 threads, 32 GB de memória RAM e GPU integrada Intel® Xe Graphics (TGL GT2), operando sob o sistema Linux.

Os testes foram realizados utilizando o mapa da cidade de Belo Horizonte, MG, composto por 32 433 vértices e 83 353 arestas, extraído da base *OpenStreetMap*. A grade uniforme utilizada foi configurada com células de tamanho 0.01° em latitude e longitude, o que corresponde aproximadamente a 1 km $\times$ 1 km na região estudada. Com base na extensão geográfica do mapa utilizado, foi obtida uma grade de 21×27 células, totalizando 567 regiões indexáveis. Esse valor foi definido após uma fase de ajuste manual, baseada em testes empíricos que buscaram otimizar o desempenho computacional.

As dimensões e a quantidade de polígonos móveis foram variáveis entre os diferentes cenários experimentais, permitindo avaliar a robustez do algoritmo sob diferentes intensidades de interferência no grafo dinâmico e obter resultados relevantes para o estudo.

5.2 Métricas de Avaliação

Para analisar a eficiência da solução proposta e compreender os cenários em que o método dinâmico apresenta melhor desempenho, foram definidas algumas métricas quantitativas. Além de permitir a comparação direta com a abordagem estática tradicional, essas métricas possibilitam estimar o *overhead* introduzido pelo tratamento dos congestionamentos simulados. As principais métricas consideradas foram:

- **Número de iterações do agente**: a cada iteração, o agente tenta avançar 100 metros ao longo da rota planejada. Caso a região esteja congestionada, o deslocamento pode falhar,

resultando em uma iteração sem progresso. Essa métrica evidencia o impacto direto da presença dos polígonos móveis no avanço do agente.

- **Distância total percorrida**: representa o comprimento efetivamente percorrido pelo agente até alcançar o destino. Em cenários dinâmicos, a rota pode ser desviada múltiplas vezes, aumentando a distância final.
- **Número de replanejamentos**: indica quantas vezes o agente dinâmico precisou recalcular seu caminho devido à obstrução da rota atual por congestionamentos.
- **Tempo total de cálculo das rotas**: contabiliza o tempo computacional acumulado nos replanejamentos, incluindo chamadas ao algoritmo de caminho mínimo.

Esses indicadores, analisados em conjunto, fornecem uma visão abrangente do comportamento do sistema tanto em termos de eficiência de navegação quanto de custo computacional.

5.3 Cenários de Teste

Os cenários de teste foram construídos variando duas características principais dos polígonos móveis: a quantidade de polígonos presentes e o tamanho geométrico de cada um deles. Os congestionamentos foram modelados como hexágonos regulares, definidos a partir de seu raio circunscrito, que é a distância entre o centro do polígono e um de seus vértices. Foram executadas 500 simulações para cada configuração, totalizando 2 000 experimentos.

É importante ressaltar que, conforme detalhado na modelagem de dados (Seção 4.1), cada uma dessas simulações apresenta configurações aleatórias distintas. Os pontos de origem e destino dos agentes, bem como a posição inicial e a trajetória dos polígonos, são sorteados a cada execução. Dessa forma, os cenários descritos a seguir definem a complexidade do ambiente (regras fixas), enquanto a aleatoriedade garante a diversidade das rotas testadas.

A variação dos parâmetros resultou em quatro cenários distintos, ordenados por nível de complexidade (densidade de obstáculos e área de bloqueio). A Tabela 3 resume as configurações utilizadas. O raio especificado refere-se ao tamanho do hexágono em graus decimais (latitude/longitude). Considerando a latitude de Belo Horizonte, 0.005° equivale a aproximadamente 500 metros e 0.009° a aproximadamente 1 km.

O cenário 1 representa uma situação de tráfego leve, com poucos bloqueios e áreas de influência pequenas, de modo que o impacto sobre o caminho é geralmente limitado e a maior parte das rotas permanece válida. Já o cenário 4 funciona como um teste de estresse para o algoritmo, combinando um número maior de bloqueios com regiões mais amplas, o que aumenta significativamente a chance de o agente encontrar caminhos obstruídos e precisar replanejar.

Tabela 3 – Definição dos Cenários de Teste

Cenário	Qtd. Polígonos	Raio (graus)	Característica
1	5	0.005	Baixa densidade, obstáculos pequenos
2	5	0.009	Baixa densidade, obstáculos grandes
3	15	0.005	Alta densidade, obstáculos pequenos
4	15	0.009	Alta densidade, obstáculos grandes

Fonte: Autoria própria.

5.4 Análise dos Resultados

A análise dos resultados estrutura-se em três eixos: o custo computacional, focado no tempo de execução; a estabilidade algorítmica, indicada pela frequência de replanejamentos e uso do *A**; e a eficiência de navegação, que avalia o impacto dos desvios na distância final percorrida comparada à rota estática.

5.4.1 Impacto Computacional e Tempo de Execução

A comparação entre o método estático e o dinâmico revela o custo associado à natureza dinâmica do problema. Conforme observado na Tabela 4, o tempo de cálculo no método estático manteve-se constante e baixo (média de ≈ 31 ms) na maioria dos cenários, pois o caminho é calculado apenas uma vez, no início da simulação.

Já no método dinâmico, o tempo total cresce proporcionalmente à complexidade do cenário. No cenário 1, o tempo médio é moderado (78.8 ms), mas no cenário 4, a média salta para 732.8 ms, um aumento de quase 10 vezes em relação ao cenário mais simples.

Tabela 4 – Comparativo de Tempo Total de Processamento (ms)

Cenário	Estático				Dinâmico			
	Mediana	Média	Máx.	σ	Mediana	Média	Máx.	σ
1	28	30.62	67	8.79	48	78.82	2 994	170.27
2	29	31.76	82	9.76	52	181.91	13 229	814.41
3	30	33.00	72	9.96	64	207.43	19 191	971.25
4	31	33.41	75	9.27	81.5	732.86	39 501	3 243.25

Fonte: Autoria própria.
Nota: σ = Desvio Padrão.

A análise do desvio padrão (σ) evidencia a disparidade na previsibilidade dos dois métodos. Enquanto o método estático apresenta um desvio padrão consistentemente baixo, indicando estabilidade, o método dinâmico exibe uma variabilidade extrema. No Cenário 4, o desvio padrão (3 243 ms) supera significativamente a própria média, sugerindo uma distribuição de cauda longa (*heavy-tailed*).

Essa alta dispersão é corroborada pela presença de *outliers* significativos nos tempos máximos, como o valor de 39.5s observado no cenário 4. Tais casos evidenciam variações pontuais que destoam do comportamento mediano, ocorrendo geralmente em situações onde o agente fica "preso" em regiões de alta complexidade geométrica, o que exige múltiplos replanejamentos sucessivos em um curto intervalo de tempo.

5.4.2 Frequência de Replanejamento e Chamadas ao $A*$

A frequência com que o algoritmo $A*$ é acionado é um indicador direto da instabilidade da rota. A Tabela 5 apresenta a porcentagem de rotas que exigiram replanejamento e o número médio de chamadas ao algoritmo de busca.

Tabela 5 – Análise de Replanejamento e Chamadas ao Algoritmo $A*$

Cenário	**Mediana $A*$**	**Média $A*$**	**Máximo $A*$**	**% Replanejamento**
1	1	1.41	70	7.2%
2	1	3.99	185	14.6%
3	1	3.22	134	21.6%
4	1	7.30	150	38.2%

Fonte: Autoria própria.

Os dados revelam um crescimento na frequência de replanejamento à medida que aumenta a complexidade do ambiente. Enquanto no cenário 1 apenas 7.2% das rotas necessitaram de replanejamento, no cenário 4 este índice aumenta para 38.2%, indicando que ambientes complexos demandam ajustes contínuos de rota.

A análise das chamadas ao $A*$ mostra um padrão similar, a mediana permanece igual a 1 em todos os cenários, indicando que na maioria dos casos uma única computação de rota é suficiente. Entretanto, a média apresenta um crescimento considerável, reforçando que a presença de casos extremos demanda replanejamentos múltiplos e frequentes.

Estes casos extremos são particularmente evidentes nos valores máximos, onde o cenário 1 possui um máximo de 70 replanejamentos, enquanto no cenário 2 possui um de 185, demonstrando que mesmo em ambientes de baixa densidade, obstáculos grandes podem gerar instabilidade significativa na navegação.

5.4.3 Eficiência das Rotas e Qualidade da Navegação

Além do custo computacional, é fundamental avaliar o impacto dos desvios na distância final percorrida pelos agentes. A Figura 6 apresenta a correlação entre a distância da rota estática (eixo X) e a distância final da rota dinâmica (eixo Y) para os quatro cenários.

Observa-se que, nos cenários 1 e 2 (baixa densidade), a grande maioria dos pontos se mantém sobre a linha diagonal ($y = x$), confirmando que com poucos obstáculos raramente

interceptam o caminho ótimo. Já no cenário 4 (alta densidade e obstáculos grandes), nota-se uma dispersão significativa acima da diagonal. Esses pontos representam os agentes que precisaram realizar grandes contornos para evitar as zonas de congestionamento, resultando em trajetos consideravelmente mais longos do que o planejado originalmente. Essa variação é detalhada na distribuição estatística apresentada na Figura 7.

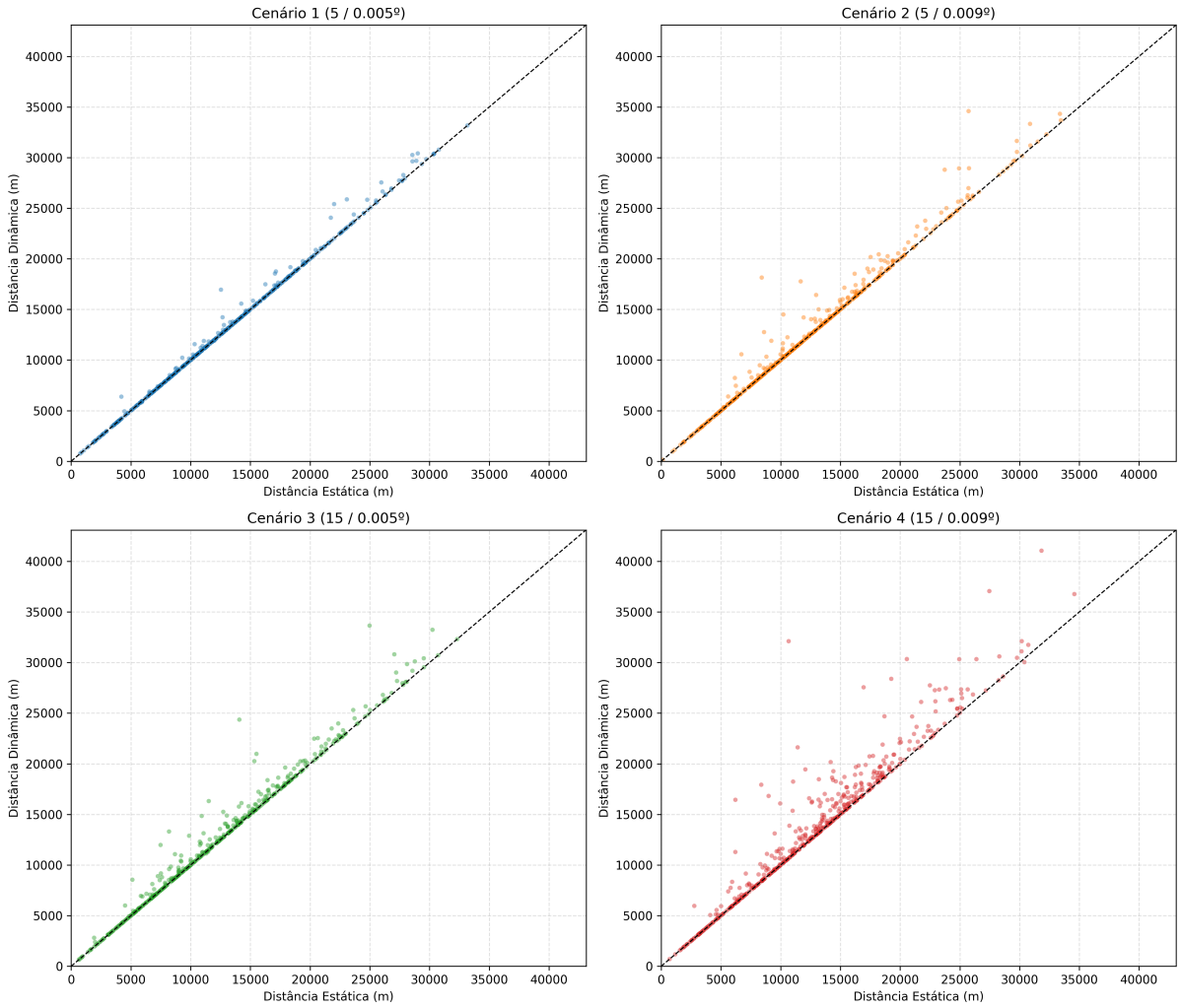
A análise dos *boxplots* revela que, embora a mediana das distâncias se mantenha estável (próxima à solução estática), a variância cresce drasticamente com a complexidade do cenário. No cenário 4, a presença de *outliers* indica casos extremos em que o agente precisou percorrer rotas muito maiores para contornar os congestionamentos modelados pelos polígonos dinâmicos.

Por fim, a Figura 8 categoriza o desempenho das rotas dinâmicas em relação às rotas estáticas. Nesta análise, uma execução é classificada como "Igual" quando ambos os agentes concluem o percurso utilizando o mesmo número de iterações. A categoria "Melhor" corresponde aos casos em que o agente dinâmico precisou de menos iterações do que o agente estático, enquanto "Pior" indica que a adaptação dinâmica resultou em um maior número de iterações.

O gráfico mostra que, conforme o cenário se torna mais complexo, aumenta também a proporção de casos em que o agente dinâmico necessita de menos iterações que o estático. Isso evidencia que o replanejamento pode ser vantajoso quando os bloqueios alteram de forma significativa a rota original. Ainda assim, a Figura 9 revela que esses ganhos costumam vir acompanhados de maior tempo de processamento, refletindo o esforço adicional exigido para recalcular trajetos em ambientes mais congestionados.

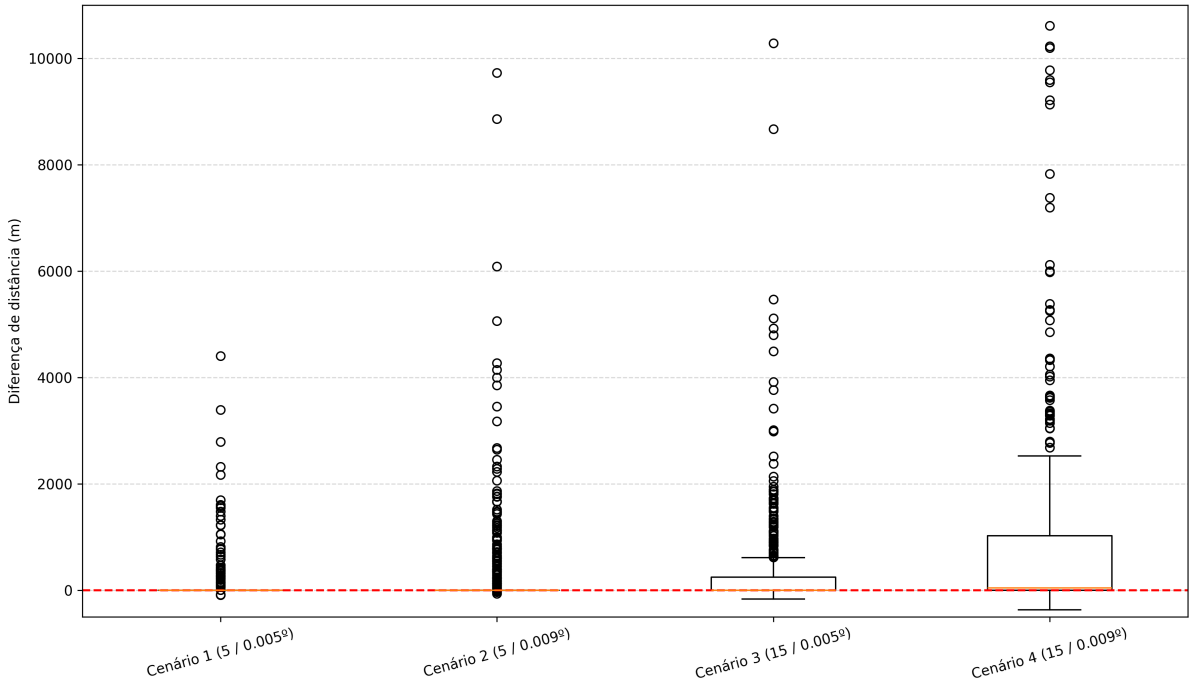
Os gráficos também evidenciam que os ganhos do replanejamento não ocorrem de forma uniforme dentro de cada cenário. A dispersão entre as execuções mostra que o benefício depende da geometria específica encontrada em cada rota: alguns percursos permanecem praticamente iguais ao estático, enquanto outros exigem desvios bem maiores. Isso reforça que o método dinâmico ajusta seu esforço conforme a dificuldade real de cada situação, aumentando a complexidade apenas quando necessário.

Figura 6 – Comparação entre as Distâncias Percorridas nas Rotas



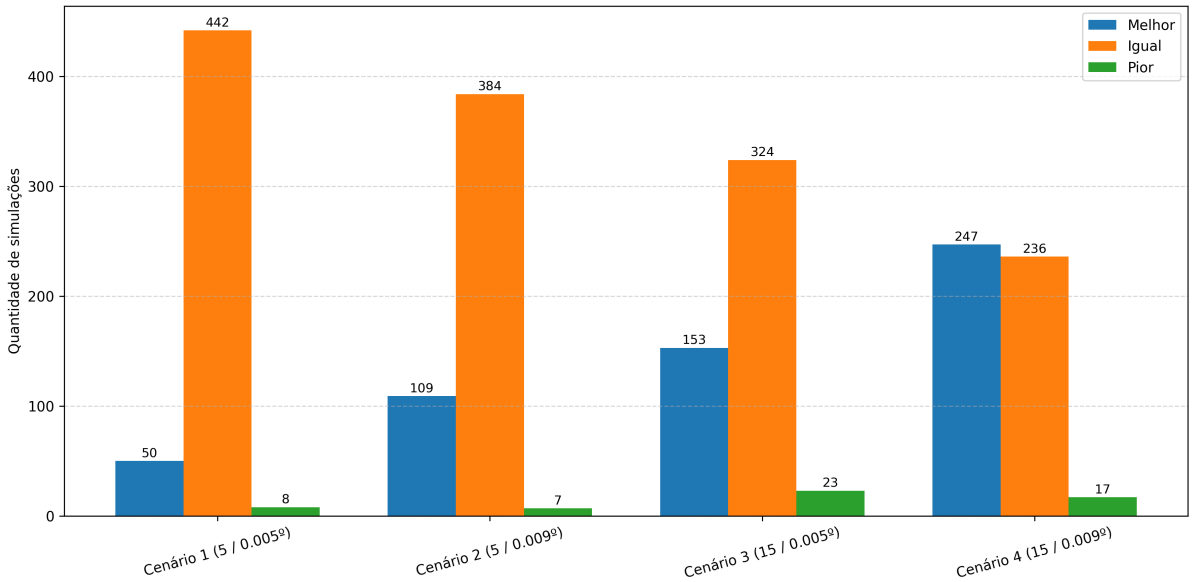
Fonte: Autoria própria.
Nota: A linha tracejada vermelha representa a igualdade ($y = x$), indicando rotas sem variação.

Figura 7 – Distribuição da Variação de Distância



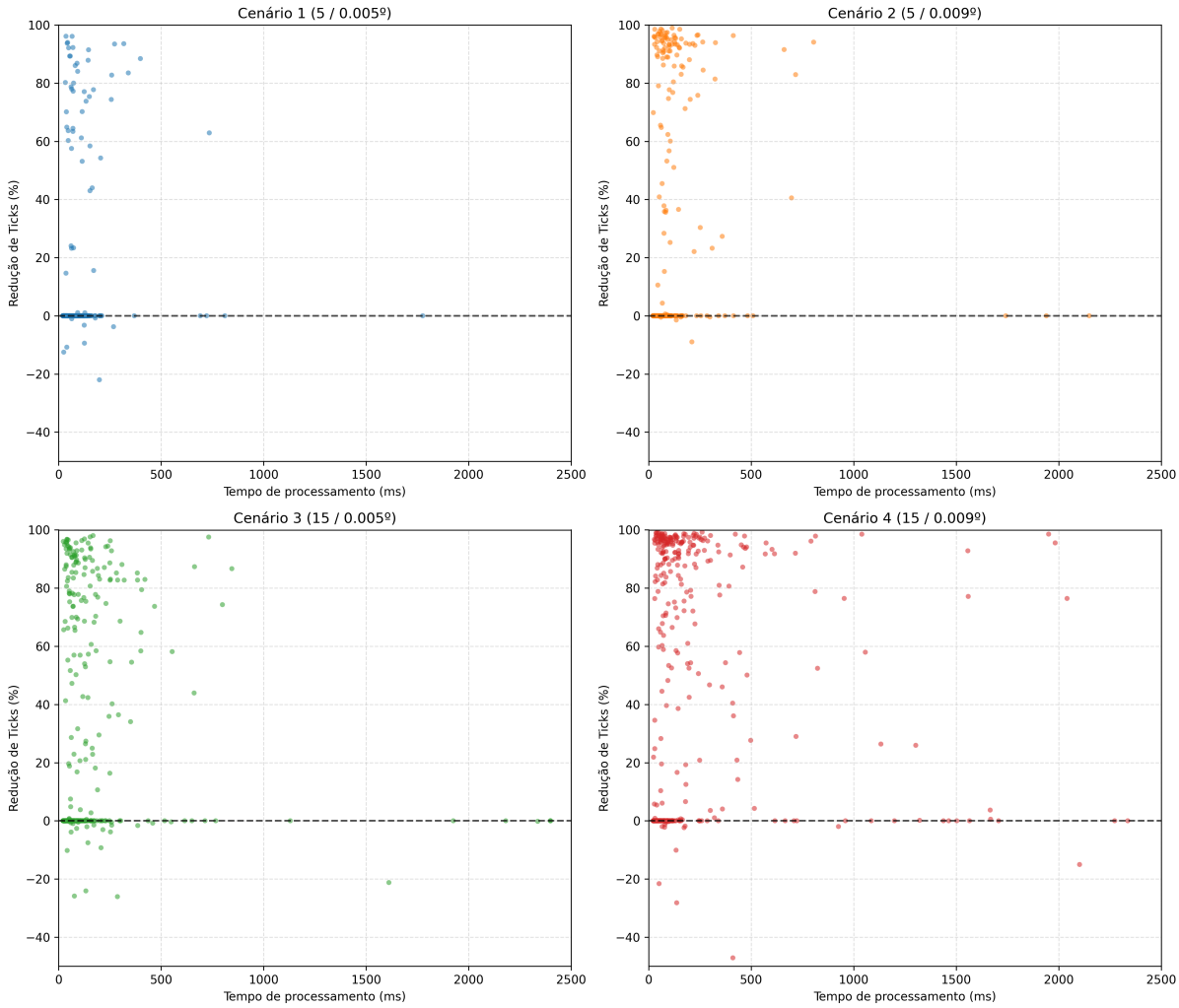
Fonte: Autoria própria.
Nota: A linha vermelha marca o desvio nulo. O eixo vertical foi limitado visualmente a 11 000m para melhor clareza da distribuição.

Figura 8 – Classificação do Desempenho das Simulações



Fonte: Autoria própria.
Nota: Classificação baseada na quantidade de iterações por execução.

Figura 9 – *Trade-off* entre Tempo de Processamento e Redução de Iterações



Fonte: Autoria própria.
Nota: *Ticks* = iterações.

5.4.4 Impacto da Paralelização em Cenários de Alta Densidade

Para avaliar a escalabilidade do sistema e a eficácia da implementação paralela com OpenMP, foi realizado um experimento de estresse contendo 50 polígonos simultâneos de raio 0.001°. A Tabela 6 compara o desempenho da execução sequencial (*single-thread*) versus a execução paralela (*multi-thread*).

Tabela 6 – Comparativo de Desempenho

Cenário	**Tempo Total (ms)**				**Métricas de Navegação**			
	Med.	**Méd.**	**Máx.**	σ	**Med. *A****	**Méd. *A****	**Máx. *A****	**% Rep.**
Sequencial	81.5	237.35	39 581	1 779.60	1	1.72	59	25.6%
Paralelo	89	211.70	5 917	420.12	1	2.14	79	26.2%

Fonte: Autoria própria.
Nota: σ = Desvio Padrão; % Rep. = Porcentagem de Replanejamentos.

Os dados revelam que a paralelização teve um impacto decisivo na estabilidade do sistema. Observa-se uma redução drástica no Desvio Padrão (σ), que caiu de aproximadamente 1 779 ms para 420 ms (redução de 76%), indicando que o tempo de resposta tornou-se muito mais previsível. Além disso, o tempo máximo de processamento foi reduzido de 39.5s para 5.9s, o que valida a hipótese de que o acúmulo de cálculos geométricos em regiões congestionadas é o principal gargalo.

Nota-se um leve aumento na mediana do tempo (de 81.5 ms para 89 ms) com a paralelização, o que é esperado devido ao *overhead* de gerenciamento das *threads*. Contudo, esse custo adicional é insignificante frente aos ganhos de robustez. Por fim, as métricas de navegação (chamadas ao A^* e taxa de replanejamento) permaneceram estatisticamente equivalentes, confirmando que a paralelização otimizou o tempo de execução sem alterar a qualidade das rotas calculadas.

6 CONCLUSÃO

Esta pesquisa desenvolveu, implementou e avaliou uma abordagem para otimização de rotas urbanas que considera congestionamentos modelados como polígonos móveis. Esse modelo permitiu analisar dinamicamente as áreas de congestionamento e replanejar rotas em tempo de execução.

Os resultados mostraram que, em cenários com baixa densidade de obstáculos, as abordagens estática e dinâmica apresentaram desempenho semelhante em termos de distância percorrida. Em ambientes com maior número e tamanho de polígonos, a estratégia dinâmica reduziu o tempo de viagem em diversos casos ao desviar de regiões congestionadas, embora isso tenha resultado em maior tempo de processamento e em um número mais elevado de replanejamentos.

A análise do equilíbrio entre eficiência computacional e ganhos na mobilidade indicou que o método dinâmico tende a ser mais vantajoso quando os congestionamentos modificam de forma significativa a viabilidade da rota original. Por outro lado, em cenários muito complexos, o custo de processamento pode se tornar excessivo, especialmente quando o agente permanece confinado em áreas com grande quantidade de obstruções ao longo do caminho.

Em síntese, o estudo confirma a viabilidade de representar congestionamentos como polígonos móveis e reforça o potencial dessa abordagem para o desenvolvimento de sistemas de navegação mais adaptativos e realistas. O código-fonte completo, dados experimentais e implementações referentes a este trabalho estão publicamente disponíveis no repositório *GitHub*[1].

6.1 Trabalhos Futuros

Considerando o caráter exploratório desta pesquisa e os desafios identificados durante o desenvolvimento, sugerem-se as seguintes extensões para trabalhos futuros:

- A integração de dados históricos e em tempo real para gerar trajetórias mais realistas dos polígonos móveis, incorporando padrões de tráfego e eventos previsíveis;
- A exploração de técnicas de aprendizado de máquina para prever a formação e dissipação de congestionamentos, melhorando a antecipação de rotas;
- A adoção de polígonos com formas irregulares e dinâmicas, capazes de representar com maior fidelidade as áreas de congestionamento observadas em ambientes urbanos reais;
- A expansão da paralelização atual para processamento em GPU, como CUDA, visando acelerar os testes de interseção geométrica.

[1] https://github.com/ericksoares13/ProjetoConclusaoCurso

REFERÊNCIAS

BAST, Hannah; DELLING, Daniel; GOLDBERG, Andrew; MÜLLER-HANNEMANN, Matthias; PAJOR, Thomas; SANDERS, Peter; WAGNER, Dorothea; WERNECK, Renato F. Route planning in transportation networks. **Algorithm engineering: Selected results and surveys**, Springer, p. 19–80, 2016.

BERG, Mark De. **Computational geometry: algorithms and applications**. [S.l.]: Springer Science & Business Media, 2000.

BOEING, Geoff. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. **Computers, Environment and Urban Systems**, Elsevier, v. 65, p. 126–139, 2017.

BRASIL. **Lei nº 12.587, de 3 de janeiro de 2012**. Institui a Política Nacional de Mobilidade Urbana. Brasília, DF: Diário Oficial [da] República Federativa do Brasil, 2012. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2011-2014/2012/lei/l12587.htm. Acesso em: 1 mai. 2025.

CONFEDERAÇÃO NACIONAL DA INDÚSTRIA. **Pesquisa Mobilidade Urbana & Trabalho**. 2024. Disponível em: https://noticias.portaldaindustria.com.br/noticias/infraestrutura/36-dos-brasileiros-de-grandes-cidades-passam-mais-de-1-hora-por-dia-no-transito/. Acesso em: 2 mai. 2025.

CORMEN, Thomas H; LEISERSON, Charles E; RIVEST, Ronald L; STEIN, Clifford. **Introduction to algorithms**. [S.l.]: MIT press, 2022.

DAGUM, Leonardo; MENON, Ramesh. Openmp: an industry standard api for shared-memory programming. **IEEE computational science and engineering**, IEEE, v. 5, n. 1, p. 46–55, 1998.

DEMIRYUREK, Ugur; BANAEI-KASHANI, Farnoush; SHAHABI, Cyrus; RANGANATHAN, Anand. Online computation of fastest path in time-dependent spatial networks. *In*: SPRINGER. **International Symposium on Spatial and Temporal Databases**. [S.l.], 2011. p. 92–111.

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

DUAN, Ran; MAO, Jiayi; MAO, Xiao; SHU, Xinkai; YIN, Longhui. Breaking the sorting barrier for directed single-source shortest paths. *In*: **Proceedings of the 57th Annual ACM Symposium on Theory of Computing**. [S.l.: s.n.], 2025. p. 36–44.

DUNN, Fletcher; PARBERRY, Ian. **3D math primer for graphics and game development**. [S.l.]: CRC Press, 2011.

ERICSON, Christer. **Real-time collision detection**. [S.l.]: Crc Press, 2004.

GEISBERGER, Robert; SANDERS, Peter; SCHULTES, Dominik; DELLING, Daniel. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. *In*: SPRINGER. **Experimental Algorithms: 7th International Workshop, WEA 2008 Provincetown, MA, USA, May 30-June 1, 2008 Proceedings 7**. [S.l.], 2008. p. 319–333.

HALLER, Jan; HANSSON, Henrik Vogelius; MOREIRA, Artur. **SFML Game Development**. [S.l.]: Packt Publishing Ltd, 2013.

HART, Peter E; NILSSON, Nils J; RAPHAEL, Bertram. A formal basis for the heuristic determination of minimum cost paths. **IEEE transactions on Systems Science and Cybernetics**, IEEE, v. 4, n. 2, p. 100–107, 1968.

HECKBERT, Paul S. **Graphics gems**. [S.l.]: Elsevier, 2013. v. 4.

HIROKI, Stella Marina Yurí. Mobilidade, participação e dados: o caso da aplicação do *Waze for Cities Data* na cidade de Joinville (SC). **urbe. Revista Brasileira de Gestão Urbana**, SciELO Brasil, v. 13, p. e20200030, 2021.

INRIX. **2022 Global Traffic Scorecard**. 2022. Disponível em: https://web.archive.org/web/20230817002047/https://inrix.com/scorecard/. Acesso em: 3 mai. 2025.

JIANG, Jincheng; WU, Lixin. A re-optimization dynamic shortest path algorithm for vehicle navigation. *In*: IEEE. **2014 IEEE Geoscience and Remote Sensing Symposium**. [S.l.], 2014. p. 3109–3112.

MAGALHÃES, Salles Viana Gomes; ANDRADE, Marcus Vinicius Alvim; FRANKLIN, W Randolph; LI, Wenli. Fast path planning under polygonal obstacle constraints. 2015.

MAGALHÃES, Salles Viana Gomes de. **Exact and parallel intersection of 3D triangular meshes**. [S.l.]: Rensselaer Polytechnic Institute, 2017.

NYSTROM, Robert. **Game programming patterns**. [S.l.]: Genever Benning, 2014.

PRODANOV, Cleber Cristiano; FREITAS, Ernani Cesar de. **Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico**. Novo Hamburgo: Feevale, 2013.

SINNOTT, Roger W. Virtues of the haversine. **Sky and telescope**, v. 68, n. 2, p. 158, 1984.

SKALA, Vaclav. A new fully projective o (log n) point-in-convex polygon algorithm: a new strategy. **The Visual Computer**, Springer, v. 41, n. 7, p. 4839–4850, 2025.

ZHANG, Kai; CHU, Zixuan; XING, Jiping; ZHANG, Honggang; CHENG, Qixiu. Urban traffic flow congestion prediction based on a data-driven model. **Mathematics**, MDPI, v. 11, n. 19, p. 4075, 2023.