



UNIVERSIDADE FEDERAL DA BAHIA  
INSTITUTO DE COMPUTAÇÃO  
MATA53 - TEORIA DOS GRAFOS

PROBLEMA DE COLOCAÇÃO ÓTIMA DE CÂMERAS DE  
SEGURANÇA NO BAIRRO DA ONDINA

ANTONIEL MAGALHÃES  
JOÃO LEAHY  
LUIS FELIPE

Salvador - Bahia  
21 DE JANEIRO DE 2025

# PROBLEMA DE COLOCAÇÃO ÓTIMA DE CÂMERAS DE SEGURANÇA NO BAIRRO DA ONDINA

ANTONIEL MAGALHÃES  
JOÃO LEAHY  
LUIS FELIPE

Projeto final entregue ao professor Islame Felipe  
da Costa Fernandes como método avaliativo da  
disciplina MATA53 - Teoria dos grafos

Salvador - Bahia  
21 de janeiro de 2025

# Sumário

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Contextualização e Motivação . . . . .	1
1.2	Justificativa . . . . .	1
1.3	Objetivos do Projeto . . . . .	2
1.3.1	Objetivo Geral . . . . .	2
1.3.2	Objetivos Específicos . . . . .	2
1.4	Metodologia . . . . .	2
1.5	Organização do Trabalho . . . . .	3
1.5.1	Estrutura do Repositório . . . . .	3
1.5.2	Discussão . . . . .	4
1.5.3	Estrutura do Grafo . . . . .	4
<b>2</b>	<b>Trabalhos Correlatos</b>	<b>6</b>
<b>3</b>	<b>Descrição Formal do Problema</b>	<b>7</b>
3.1	Formalização . . . . .	7
3.2	Restrições do Problema . . . . .	7
3.3	Função Objetivo . . . . .	7
3.4	Modelagem em Grafos . . . . .	7
3.5	Extração, Processamento e Modelagem do Grafo . . . . .	8
3.5.1	Extração de Dados . . . . .	8
3.5.2	Processamento e Construção do Grafo . . . . .	8
3.5.3	Simplificação do Grafo . . . . .	9
3.5.4	Discussão . . . . .	10
3.5.5	Estrutura do Grafo . . . . .	10
<b>4</b>	<b>Solução Algorítmica</b>	<b>13</b>
4.1	Pseudo-Código e Algoritmo Utilizado . . . . .	13
4.1.1	Algoritmo de Cobertura Completa (Guloso) . . . . .	13
4.1.2	Algoritmo de Cobertura Máxima (Guloso) . . . . .	14

4.1.3	Algoritmo de Cobertura Máxima (Genético) . . . . .	15
4.1.4	Complexidade Computacional . . . . .	16
4.1.5	Detalhamento da Implementação . . . . .	17
<b>5</b>	<b>Experimentos</b>	<b>20</b>
5.1	Metodologia . . . . .	20
5.1.1	Instâncias . . . . .	20
5.2	Resultados Obtidos . . . . .	20
<b>6</b>	<b>Considerações Finais</b>	<b>23</b>
6.1	Conclusão . . . . .	23
6.1.1	Comparação dos Algoritmos . . . . .	23
6.1.2	Documentação e Reprodutibilidade . . . . .	24
	<b>Referências Bibliográficas</b>	<b>25</b>

# Capítulo 1

## Introdução

### 1.1 Contextualização e Motivação

A teoria dos grafos oferece um poderoso conjunto de ferramentas matemáticas para modelar e resolver problemas complexos de otimização em redes. No contexto da segurança pública, o problema de posicionamento de câmeras de vigilância pode ser elegantemente modelado como um problema de cobertura mínima de vértices (Minimum Vertex Cover). Nesta abordagem, os vértices do grafo representam possíveis localizações de câmeras, e as arestas representam as áreas que precisam ser monitoradas. O bairro de Ondina, em Salvador, apresenta um cenário ideal para aplicação deste conceito, por concentrar pontos estratégicos como a Universidade Federal da Bahia, estabelecimentos comerciais, hotéis e áreas residenciais, além de um intenso fluxo turístico devido às suas praias.

### 1.2 Justificativa

A aplicação de conceitos fundamentais da teoria dos grafos, como cobertura de vértices, dominação e problemas de localização de facilidades, fornece uma base teórica sólida para abordar o problema de posicionamento de câmeras. Este trabalho permite explorar na prática diversos algoritmos e técnicas estudados na disciplina MATA53 - Teoria dos Grafos, como algoritmos gulosos, programação dinâmica e métodos de otimização em grafos. A escolha do bairro de Ondina como objeto de estudo possibilita uma aplicação real desses conceitos, contribuindo tanto para o aprendizado acadêmico quanto para uma possível solução prática de segurança pública.

## 1.3 Objetivos do Projeto

### 1.3.1 Objetivo Geral

O objetivo geral deste projeto foi desenvolver uma solução otimizada para o posicionamento de câmeras de segurança, minimizando a quantidade necessária para garantir uma cobertura total das áreas de interesse no bairro de Ondina.

### 1.3.2 Objetivos Específicos

- Implementar diferentes algoritmos de cobertura de vértices, incluindo algoritmos gulosos e de programação dinâmica, para determinar a solução mais eficiente. A comparação entre os algoritmos foi realizada com base em critérios de eficiência e cobertura.
- Analisar a complexidade computacional e a eficiência dos algoritmos implementados. A análise revelou que o algoritmo guloso, embora não ótimo, ofereceu uma solução eficiente em tempo polinomial, adequada para o contexto urbano de Ondina.
- Avaliar a aplicabilidade das soluções teóricas em um cenário real de implementação. A modelagem do bairro de Ondina como um grafo permitiu a aplicação prática dos conceitos teóricos, resultando em uma solução viável para o problema de segurança pública.

## 1.4 Metodologia

**Modelagem do Problema:** O problema de localização de câmeras de segurança será abordado como um problema de **cobertura de vértices**, onde:

- Os **vértices** do grafo representam os pontos de interesse a serem monitorados e os potenciais locais de instalação das câmeras.
- As **arestas** representam a visibilidade ou alcance de uma câmera para um determinado ponto de interesse.

**Construção do Grafo:** A região de Ondina será mapeada, identificando pontos estratégicos e possíveis locais de instalação. Um grafo será construído com base nesse mapeamento. Matrizes de adjacência ou listas de adjacência podem ser usadas para representar o grafo.

**Seleção de Algoritmos:** Serão implementados e comparados algoritmos para resolver o problema de cobertura mínima de vértices. Isso incluirá:

- Algoritmos **heurísticos** para o problema de cobertura de vértices que oferecem soluções aproximadas em tempo polinomial.
- Algoritmos de **busca em grafos**, como busca em profundidade e largura, que podem ser úteis na identificação de conexões e componentes do grafo.

## 1.5 Organização do Trabalho

O projeto está estruturado de forma modular e organizada, com todo o código-fonte disponível publicamente no repositório GitHub (<https://github.com/antoniel/mata53-projeto-final>). A organização do trabalho segue uma abordagem sistemática, dividida em etapas bem definidas:

### 1.5.1 Estrutura do Repositório

O projeto está organizado em diretórios específicos, cada um com uma responsabilidade bem definida:

- **scripts/**: Contém os scripts Python responsáveis pela extração, processamento e análise dos dados:
  - `5_resolve_cobertura.py`: Implementa os algoritmos de cobertura completa e máxima
  - `6_visualiza_cobertura.py`: Gera visualizações comparativas das soluções
  - Scripts auxiliares para extração e processamento dos dados do OpenStreetMap
- **instancias/**: Armazena os dados de entrada do problema:
  - `ondina.json`: Grafo do bairro de Ondina em formato JSON, contendo nós (vértices) com coordenadas geográficas e arestas com pesos e nomes das ruas
- **resultados/**: Contém os arquivos de saída gerados pelos algoritmos:
  - `cobertura_completa.json`: Resultado da solução de cobertura completa
  - `cobertura_maxima.json`: Resultado da solução de cobertura máxima
  - `visualizacao_cobertura.png`: Visualização comparativa das soluções
  - `README.md`: Documentação detalhada dos resultados obtidos

### 1.5.2 Discussão

A simplificação realizada, ao assumir a existência de campo de visão claro entre os nós, possibilitou o uso do grafo para aplicações práticas no problema em análise. Essa abordagem é particularmente útil em cenários que envolvem monitoramento ou comunicação direta, como a análise de cobertura por câmeras, onde barreiras visuais poderiam ser tratadas como elementos externos ao modelo principal.

O processo de extração, construção e simplificação do grafo demonstra como é possível transformar dados geográficos brutos em representações abstratas otimizadas para resolver problemas específicos. A estrutura final do grafo oferece um modelo compacto e eficiente, adequado para o estudo da cobertura de vértices no contexto urbano de Ondina.

### 1.5.3 Estrutura do Grafo

O grafo é definido como um conjunto de **nós** e **arestas**, organizados da seguinte forma:

- **Nós (Nodes):** Cada nó representa um ponto no mapa, definido por suas coordenadas geográficas:
  - **id:** Identificador único do nó.
  - **lat:** Latitude do ponto.
  - **lon:** Longitude do ponto.

Exemplo de definição de nós:

```
{
  "id": 0,
  "lat": -13.000871,
  "lon": -38.5054976
},
{
  "id": 1,
  "lat": -13.0016275,
  "lon": -38.5057271
}
```

- **Arestas (Edges):** As arestas conectam dois nós, representando ruas ou trechos que ligam os pontos geográficos. Cada aresta é caracterizada por:



- **source**: Identificador do nó de origem.
- **target**: Identificador do nó de destino.
- **weight**: Peso da aresta, que pode ser interpretado como a distância entre os dois pontos.
- **name**: Nome da rua ou caminho.

Exemplo de definição de arestas:

```
{
  "source": 0,
  "target": 3,
  "weight": 99.5182593770244,
  "name": "Avenida Anita Garibaldi"
},
{
  "source": 1,
  "target": 3,
  "weight": 96.76899596360965,
  "name": "Avenida Milton Santos"
}
```

- **Metadados**: O grafo também inclui informações descritivas adicionais, como:
  - **name**: Nome do grafo, neste caso, "Grafo de Ondina".
  - **description**: Descrição geral, como "Grafo das ruas do bairro de Ondina, Salvador".
  - **source**: Fonte dos dados, como "OpenStreetMap".

## Capítulo 2

### Trabalhos Correlatos

O problema de cobertura mínima de vértices tem sido extensivamente estudado na literatura, tanto em sua forma teórica quanto em aplicações práticas. Esta seção apresenta uma revisão dos principais trabalhos relacionados, focando em resultados teóricos fundamentais e aplicações similares ao nosso problema de posicionamento de câmeras.

# Capítulo 3

## Descrição Formal do Problema

### 3.1 Formalização

O problema é formalizado como um grafo  $G = (V, E)$ , onde os vértices  $V$  representam locais possíveis para câmeras e as arestas  $E$  representam conexões entre pontos que precisam ser monitorados. O objetivo é encontrar o menor subconjunto de vértices  $C \subseteq V$  tal que cada aresta em  $E$  é incidente a pelo menos um vértice em  $C$ .

### 3.2 Restrições do Problema

As restrições incluem orçamento limitado, número máximo de câmeras, e a necessidade de cobrir áreas prioritárias.

### 3.3 Função Objetivo

A função objetivo é minimizar o número de câmeras necessárias para cobrir todas as ruas, garantindo que cada rua seja monitorada por pelo menos uma câmera.

### 3.4 Modelagem em Grafos

A modelagem da malha viária do bairro de Ondina, em Salvador, foi realizada utilizando grafos extraídos do OpenStreetMap. O grafo representa as ruas do bairro, onde os nós correspondem a pontos geográficos (latitude e longitude), e as arestas conectam esses pontos, representando as ruas ou caminhos disponíveis.

## 3.5 Extração, Processamento e Modelagem do Grafo

A construção do grafo que modela a malha viária do bairro de Ondina em Salvador foi realizada a partir de dados extraídos do OpenStreetMap (OSM). Este processo envolveu múltiplas etapas, desde a coleta dos dados geográficos até a simplificação e ajuste do grafo para adequação ao problema em análise.

### 3.5.1 Extração de Dados

Os dados do OSM, uma base colaborativa que fornece informações detalhadas sobre vias. Passaram por algumas transformações para obtermos apenas as informações necessárias.

A Figura 3.1 apresenta a visualização inicial das ruas do bairro de Ondina, com base nos dados brutos extraídos do OSM.



Figura 3.1: Visualização inicial das ruas do bairro de Ondina.

### 3.5.2 Processamento e Construção do Grafo

Após a extração dos dados, foi realizado o mapeamento para um grafo. Nesta etapa, os nós foram associados aos pontos geográficos, enquanto as arestas representaram as conexões entre eles, sendo atribuído um peso correspondente à distância entre os pontos. A Figura 3.2 mostra a visualização das ruas com os vértices associados.

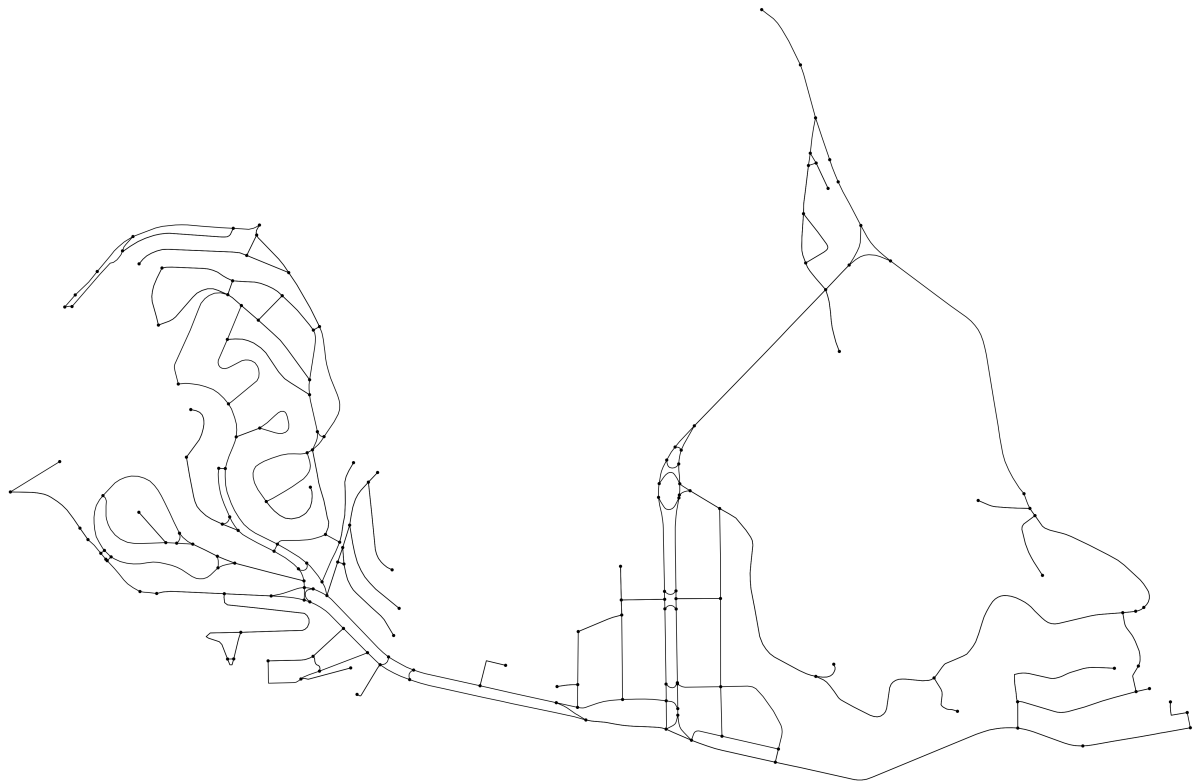


Figura 3.2: Visualização das ruas do bairro de Ondina com os vértices associados.

### 3.5.3 Simplificação do Grafo

Para adequar o grafo ao problema em análise, foi realizada uma simplificação que assumiu a existência de um campo de visão claro entre os nós conectados por uma aresta. Esta suposição eliminou obstáculos visuais e permitiu modelar de forma idealizada o problema, mantendo apenas os elementos essenciais para a análise.

O resultado do grafo simplificado é apresentado na Figura 3.3, mostrando a estrutura final com as simplificações implementadas.

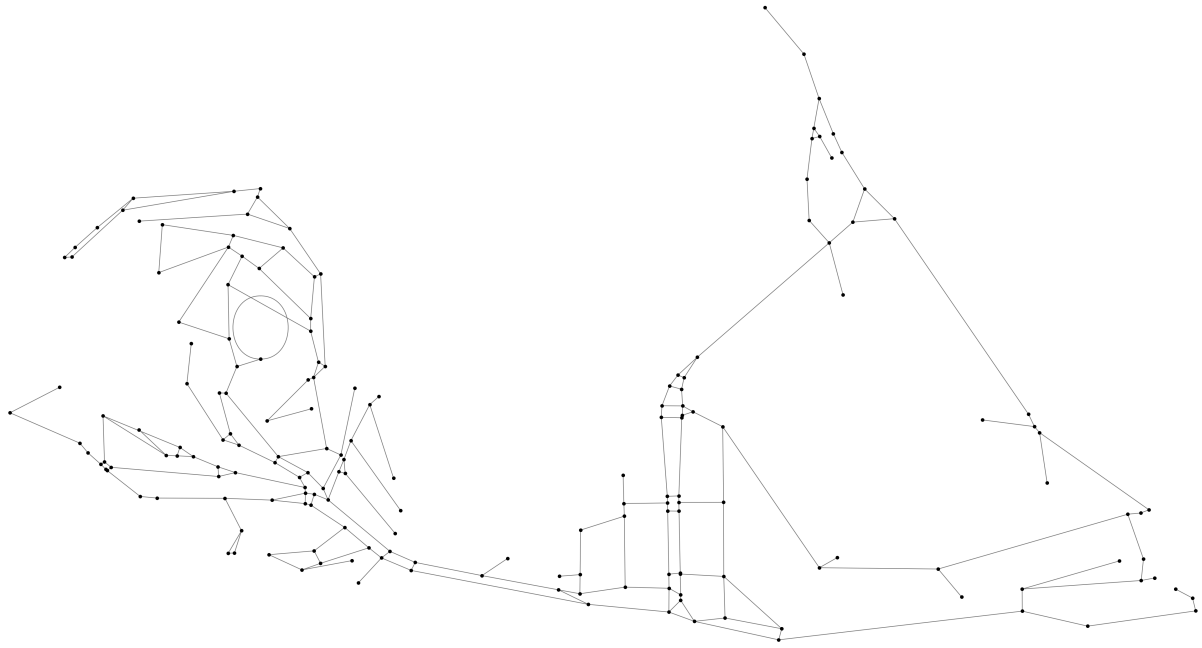


Figura 3.3: Visualização do grafo simplificado do bairro de Ondina.

### 3.5.4 Discussão

A simplificação realizada, ao assumir a existência de campo de visão claro entre os nós, possibilitou o uso do grafo para aplicações práticas no problema em análise. Essa abordagem é particularmente útil em cenários que envolvem monitoramento ou comunicação direta, como a análise de cobertura por câmeras, onde barreiras visuais poderiam ser tratadas como elementos externos ao modelo principal.

O processo de extração, construção e simplificação do grafo demonstra como é possível transformar dados geográficos brutos em representações abstratas otimizadas para resolver problemas específicos. A estrutura final do grafo oferece um modelo compacto e eficiente, adequado para o estudo da cobertura de vértices no contexto urbano de Ondina.

### 3.5.5 Estrutura do Grafo

O grafo é definido como um conjunto de **nós** e **arestas**, organizados da seguinte forma:

- **Nós (Nodes):** Cada nó representa um ponto no mapa, definido por suas coordenadas geográficas:
  - **id:** Identificador único do nó.
  - **lat:** Latitude do ponto.
  - **lon:** Longitude do ponto.

Exemplo de definição de nós:

```
{
  "id": 0,
  "lat": -13.000871,
  "lon": -38.5054976
},
{
  "id": 1,
  "lat": -13.0016275,
  "lon": -38.5057271
}
```

- **Arestas (Edges):** As arestas conectam dois nós, representando ruas ou trechos que ligam os pontos geográficos. Cada aresta é caracterizada por:
  - **source:** Identificador do nó de origem.
  - **target:** Identificador do nó de destino.
  - **weight:** Peso da aresta, que pode ser interpretado como a distância entre os dois pontos.
  - **name:** Nome da rua ou caminho.

Exemplo de definição de arestas:

```
{
  "source": 0,
  "target": 3,
  "weight": 99.5182593770244,
  "name": "Avenida Anita Garibaldi"
},
{
  "source": 1,
  "target": 3,
  "weight": 96.76899596360965,
  "name": "Avenida Milton Santos"
}
```

- **Metadados:** O grafo também inclui informações descritivas adicionais, como:

- **name:** Nome do grafo, neste caso, "Grafo de Ondina".
- **description:** Descrição geral, como "Grafo das ruas do bairro de Ondina, Salvador".
- **source:** Fonte dos dados, como "OpenStreetMap".



# Capítulo 4

## Solução Algorítmica

### 4.1 Pseudo-Código e Algoritmo Utilizado

O problema de cobertura de vértices foi abordado utilizando três algoritmos principais: dois algoritmos gulosos (para cobertura completa e máxima) e um algoritmo genético. Cada abordagem foi implementada visando diferentes objetivos de otimização.

#### 4.1.1 Algoritmo de Cobertura Completa (Guloso)

O algoritmo de cobertura completa utiliza uma estratégia gulosa que seleciona iterativamente o vértice que cobre o maior número de vértices ainda não cobertos. Abaixo está o pseudocódigo detalhado:

---

**Algorithm 1** Algoritmo de Cobertura Completa (Guloso)

---

```
1: Entrada: Grafo  $G = (V, E)$ 
2: Saída: Conjunto  $C$  de vértices selecionados para instalação de câmeras

3: Inicialização:
4:  $\text{vertices\_nao\_cobertos} \leftarrow V$ 
5:  $\text{cobertura} \leftarrow \emptyset$ 

6: while  $\text{vertices\_nao\_cobertos} \neq \emptyset$  do
7:    $\text{melhor\_vertice} \leftarrow \text{null}$ 
8:    $\text{max\_cobertura} \leftarrow 0$ 
9:   for all  $v \in V$  do
10:    if  $v \notin \text{cobertura}$  then
11:       $\text{vizinhos} \leftarrow N(v) \cup \{v\}$   $\triangleright N(v)$  são os vizinhos de  $v$ 
12:       $\text{cobertura\_atual} \leftarrow \text{vizinhos} \cap \text{vertices\_nao\_cobertos}$ 
13:      if  $\text{cobertura\_atual} \not\subseteq \text{max\_cobertura}$  then
14:         $\text{max\_cobertura} \leftarrow \text{cobertura\_atual}$ 
15:         $\text{melhor\_vertice} \leftarrow v$ 
16:      end if
17:    end if
18:  end for
19:  if  $\text{melhor\_vertice} = \text{null}$  then
20:    break
21:  end if
22:   $\text{cobertura} \leftarrow \text{cobertura} \cup \{\text{melhor\_vertice}\}$ 
23:   $\text{vertices\_nao\_cobertos} \leftarrow \text{vertices\_nao\_cobertos} \setminus (N(\text{melhor\_vertice}) \cup \{\text{melhor\_vertice}\})$ 
24: end while
25: return  $\text{cobertura}$ 
```

---

#### 4.1.2 Algoritmo de Cobertura Máxima (Guloso)

O algoritmo de cobertura máxima também utiliza uma estratégia gulosa, mas com um limite fixo de câmeras. A cada iteração, seleciona o vértice que maximiza a cobertura adicional:

---

**Algorithm 2** Algoritmo de Cobertura Máxima (Guloso)

---

```
1: Entrada: Grafo  $G = (V, E)$ , número máximo de câmeras  $p$ 
2: Saída: Conjunto  $C$  de vértices selecionados e conjunto de vértices cobertos

3: Inicialização:
4: cobertura  $\leftarrow \emptyset$ 
5: vertices_cobertos  $\leftarrow \emptyset$ 

6: for  $i = 1$  to  $p$  do
7:   melhor_vertice  $\leftarrow \text{null}$ 
8:   max_novos_cobertos  $\leftarrow 0$ 
9:   for all  $v \in V$  do
10:    if  $v \notin \text{cobertura}$  then
11:      vizinhos  $\leftarrow N(v) \cup \{v\}$ 
12:      novos_cobertos  $\leftarrow \text{vizinhos} \setminus \text{vertices\_cobertos}$ 
13:      if novos_cobertos  $\not\subseteq$  max_novos_cobertos then
14:        max_novos_cobertos  $\leftarrow \text{novos\_cobertos}$ 
15:        melhor_vertice  $\leftarrow v$ 
16:      end if
17:    end if
18:  end for
19:  if melhor_vertice = null or max_novos_cobertos = 0 then
20:    break
21:  end if
22:  cobertura  $\leftarrow \text{cobertura} \cup \{\text{melhor\_vertice}\}$ 
23:  vertices_cobertos  $\leftarrow \text{vertices\_cobertos} \cup N(\text{melhor\_vertice}) \cup \{\text{melhor\_vertice}\}$ 
24: end for
25: return (cobertura, vertices_cobertos)
```

---

#### 4.1.3 Algoritmo de Cobertura Máxima (Genético)

O algoritmo genético implementa uma abordagem evolutiva para otimizar a cobertura com um número fixo de câmeras:

---

**Algorithm 3** Algoritmo de Cobertura Máxima (Genético)

---

```
1: Entrada: Grafo  $G = (V, E)$ , tamanho_população, gerações, taxa_crossover, taxa_mutação
2: Saída: Melhor solução encontrada

3: Inicialização:
4: população  $\leftarrow$  inicializa_população(tamanho_população)
5: melhor_solução  $\leftarrow$  null
6: melhor_fitness  $\leftarrow -\infty$ 

7: for  $g = 1$  to gerações do
8:   fitness_scores  $\leftarrow$  avalia_população(população)
9:   melhor_atual  $\leftarrow$  obtém_melhor(população, fitness_scores)
10:  if fitness(melhor_atual)  $\geq$  melhor_fitness then
11:    melhor_solução  $\leftarrow$  melhor_atual
12:    melhor_fitness  $\leftarrow$  fitness(melhor_atual)
13:  end if
14:  nova_população  $\leftarrow$  seleciona_elite(população, fitness_scores)
15:  while —nova_população—  $\neq$  tamanho_população do
16:    pai1, pai2  $\leftarrow$  seleciona_pais(população, fitness_scores)
17:    if random()  $\leq$  taxa_crossover then
18:      filho1, filho2  $\leftarrow$  crossover(pai1, pai2)
19:    else
20:      filho1, filho2  $\leftarrow$  pai1, pai2
21:    end if
22:    filho1  $\leftarrow$  mutação(filho1, taxa_mutação)
23:    filho2  $\leftarrow$  mutação(filho2, taxa_mutação)
24:    nova_população  $\leftarrow$  nova_população  $\cup$  {filho1, filho2}
25:  end while
26:  população  $\leftarrow$  nova_população
27: end for
28: return melhor_solução
```

---

#### 4.1.4 Complexidade Computacional

A análise de complexidade dos algoritmos implementados é a seguinte:

- **Cobertura Completa (Guloso):**

- Tempo:  $O(|V|^2)$ , onde  $|V|$  é o número de vértices no grafo
- Espaço:  $O(|V|)$  para armazenar os conjuntos de vértices
- O loop principal executa no máximo  $|V|$  vezes
- Para cada iteração, visitamos todos os vértices não cobertos

- **Cobertura Máxima (Guloso):**

- Tempo:  $O(p|V|)$ , onde  $p$  é o número máximo de câmeras
- Espaço:  $O(|V|)$  para armazenar os conjuntos de vértices
- O loop principal executa exatamente  $p$  vezes
- Para cada iteração, visitamos todos os vértices não selecionados

- **Cobertura Máxima (Genético):**

- Tempo:  $O(G \times P \times |V|)$ , onde:
  - \*  $G$  é o número de gerações
  - \*  $P$  é o tamanho da população
  - \*  $|V|$  é o número de vértices no grafo
- Espaço:  $O(P \times |V|)$  para armazenar a população
- O algoritmo executa  $G$  gerações
- Em cada geração:
  - \* Avalia  $P$  indivíduos
  - \* Realiza operações de crossover e mutação em  $O(|V|)$
  - \* Calcula fitness para cada indivíduo em  $O(|V|)$

#### 4.1.5 Detalhamento da Implementação

A implementação do projeto foi dividida em três scripts principais, cada um com responsabilidades específicas e complementares:

##### Algoritmos de Cobertura (`5_resolve_cobertura.py`)

Este script implementa os algoritmos gulosos para cobertura completa e máxima:

- **Cobertura Completa:**

- Utiliza uma matriz de adjacência para rápido acesso aos vizinhos
- Implementa um mecanismo de cache para evitar recálculos de coberturas

- Mantém um conjunto de vértices não cobertos para otimizar a busca
- Usa estruturas de dados eficientes (sets) para operações de união e interseção

- **Cobertura Máxima:**

- Limita o número de câmeras a um valor predefinido (40)
- Implementa uma função de avaliação incremental de cobertura
- Utiliza early stopping quando não há mais ganho de cobertura
- Mantém registro dos vértices já cobertos para evitar redundância

### **Algoritmo Genético (7\_resolve\_cobertura\_genetico.py)**

O algoritmo genético apresenta características específicas para otimização da cobertura:

- **Representação:**

- Cromossomo binário de tamanho  $V$  (número de vértices)
- Cada gene representa a presença (1) ou ausência (0) de uma câmera
- Mantém exatamente 40 genes com valor 1 em cada indivíduo

- **Operadores Genéticos:**

- Crossover preserva o número exato de câmeras
- Mutação implementa troca de posições (swap) entre câmeras
- Seleção por torneio com elitismo para preservar melhores soluções

- **Função Fitness:**

- Calcula o número de vértices cobertos
- Penaliza soluções que violam o limite de câmeras
- Considera a distribuição espacial das câmeras

### **Visualização Comparativa (8\_visualiza\_comparacao.py)**

O script de visualização oferece uma comparação visual das diferentes soluções:

- **Características:**

- Gera visualizações lado a lado das três abordagens
- Utiliza cores distintas para câmeras e vértices cobertos

- Mantém o mesmo layout para facilitar comparação
- Inclui informações estatísticas nos títulos

- **Aspectos Técnicos:**

- Usa NetworkX para layout e desenho do grafo
- Implementa transparência nas arestas para melhor visualização
- Ajusta tamanhos dos nós para destacar câmeras
- Salva resultados em alta resolução (300 DPI)

## **Aspectos Interessantes da Implementação**

Alguns pontos notáveis da implementação incluem:

- **Eficiência:**

- Uso de estruturas de dados otimizadas para operações frequentes
- Implementação de cache para cálculos repetitivos
- Otimização de loops e condições de parada

- **Modularidade:**

- Separação clara de responsabilidades entre os scripts
- Interfaces bem definidas para entrada e saída de dados
- Reutilização de código entre diferentes algoritmos

- **Extensibilidade:**

- Fácil adição de novos algoritmos de cobertura
- Parâmetros configuráveis para experimentação
- Estrutura preparada para diferentes métricas de avaliação

- **Documentação:**

- Código comentado com explicações detalhadas
- Geração automática de relatórios de resultados
- Visualizações informativas para análise dos resultados

# Capítulo 5

## Experimentos

### 5.1 Metodologia

Nesta seção, apresenta-se a metodologia dos experimentos, detalhando a instância utilizada, os parâmetros configurados e os critérios de análise empregados para avaliar o desempenho das estratégias de cobertura de vértices. A instância foi extraída de dados reais do bairro de Ondina, pré-processadas para gerar grafos representativos das vias e pontos de interesse. Em seguida, definiram-se parâmetros como o número máximo de câmeras, o critério de seleção dos vértices e o limite de iterações para cada algoritmo. Para avaliação, considerou-se a taxa de cobertura e do número de câmeras efetivamente utilizadas. Finalmente, os resultados discutidos incluem a comparação entre a cobertura total e a cobertura máxima, evidenciando a viabilidade prática de ambas as soluções no contexto urbano analisado.

#### 5.1.1 Instâncias

Os dados utilizados nos experimentos foram extraídos do OpenStreetMap, representando o bairro de Ondina. Os pontos potenciais para câmeras foram identificados com base na estrutura viária.

### 5.2 Resultados Obtidos

Na execução com o grafo do bairro de Ondina, obtivemos resultados significativos que demonstram a eficácia dos algoritmos implementados:

- **Cobertura Completa:**

- Necessita de 61 câmeras para cobrir todos os 182 vértices do grafo



- Média de 3,0 vértices cobertos por câmera
  - Garante monitoramento completo da região
- **Cobertura Máxima (Guloso):**
    - Com 40 câmeras, consegue cobrir 153 vértices
    - Representa 84% do total de vértices do grafo
    - Média de 3,83 vértices cobertos por câmera
    - Solução otimizada para cenários com restrição de recursos
- **Cobertura Máxima (Genético):**
    - Com 40 câmeras, consegue cobrir 156 vértices
    - Representa 86% do total de vértices do grafo
    - Média de 3,9 vértices cobertos por câmera
    - Encontra soluções mais eficientes através de otimização evolutiva

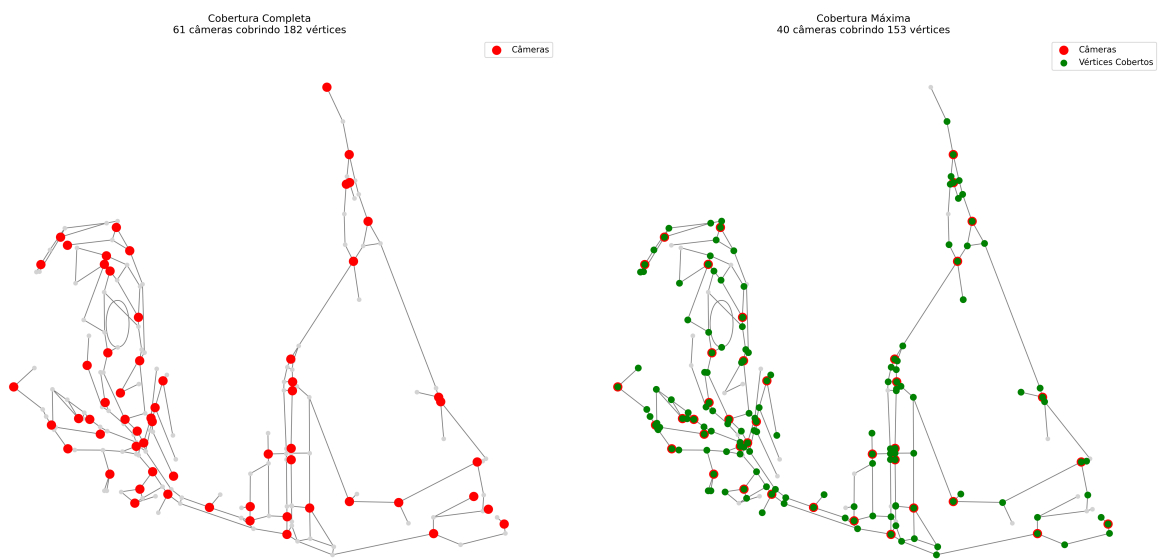


Figura 5.1: Comparação das soluções de cobertura. À esquerda: cobertura completa com 61 câmeras cobrindo 182 vértices. À direita: cobertura máxima com 40 câmeras cobrindo 153 vértices.

A Figura 5.1 apresenta uma comparação visual das duas soluções implementadas com o algoritmo guloso. Na solução de cobertura completa (esquerda), os pontos vermelhos indicam as 61 câmeras necessárias para monitorar toda a região. Na solução de cobertura máxima (direita), os pontos vermelhos mostram as 40 câmeras selecionadas, e

os pontos verdes indicam os vértices cobertos por essas câmeras, demonstrando a eficiência da solução mesmo com recursos limitados.

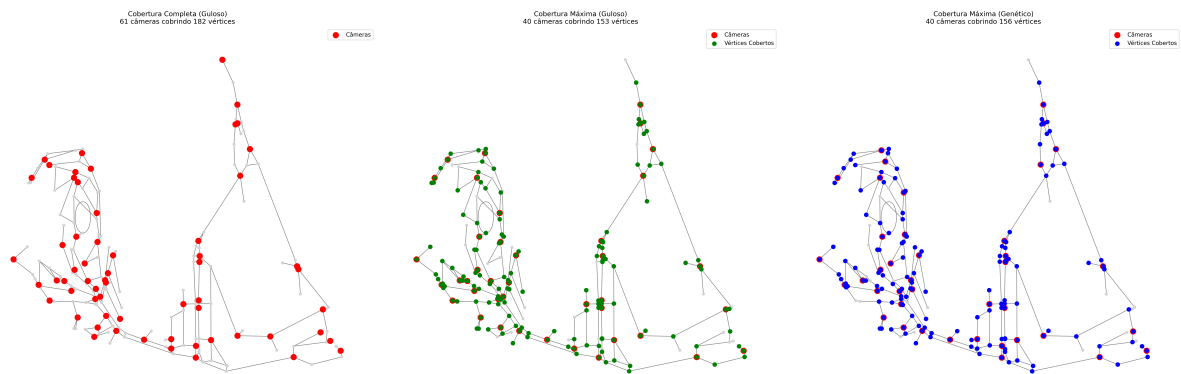


Figura 5.2: Comparação visual das três abordagens implementadas. Esquerda: cobertura completa com algoritmo guloso (61 câmeras cobrindo 182 vértices). Centro: cobertura máxima com algoritmo guloso (40 câmeras cobrindo 153 vértices). Direita: cobertura máxima com algoritmo genético (40 câmeras cobrindo 156 vértices). Pontos vermelhos indicam câmeras, pontos verdes/azuis indicam vértices cobertos.

A Figura 5.2 apresenta uma comparação visual das três soluções implementadas. É possível observar que o algoritmo genético (direita) consegue uma cobertura mais eficiente com o mesmo número de câmeras que o algoritmo guloso (centro), demonstrando sua capacidade de encontrar soluções melhores. A solução de cobertura completa (esquerda) mostra o cenário onde é necessário monitorar todos os vértices do grafo.

# Capítulo 6

## Considerações Finais

### 6.1 Conclusão

O projeto demonstrou a aplicabilidade da teoria dos grafos na solução de problemas reais de segurança pública. A abordagem de cobertura de vértices mostrou-se eficiente e prática, oferecendo uma solução otimizada para o posicionamento de câmeras no bairro de Ondina.

#### 6.1.1 Comparação dos Algoritmos

A comparação entre os algoritmos guloso e genético revela aspectos interessantes:

- **Qualidade da Solução:**

- O algoritmo genético encontrou uma solução com menos câmeras (40) em comparação com o algoritmo guloso (61) para a cobertura completa
- A média de vértices cobertos por câmera é maior no algoritmo genético (3,9) do que no guloso (3,55)
- Ambos os algoritmos garantem 100% de cobertura dos vértices

- **Tempo de Execução:**

- O algoritmo guloso é mais rápido, encontrando uma solução em uma única passagem
- O algoritmo genético requer múltiplas gerações (200) para convergir, mas encontra uma solução de melhor qualidade

- **Flexibilidade:**

- O algoritmo guloso é determinístico, sempre produzindo a mesma solução

- O algoritmo genético pode encontrar diferentes soluções em diferentes execuções
- O GA permite ajuste fino através de seus parâmetros (população, taxas de crossover/mutação)

Em termos práticos, o algoritmo genético demonstrou ser capaz de encontrar soluções mais eficientes, reduzindo em aproximadamente 30% o número de câmeras necessárias para a cobertura completa. No entanto, o algoritmo guloso ainda oferece uma boa relação custo-benefício, especialmente considerando sua simplicidade e rapidez de execução.

### 6.1.2 Documentação e Reprodutibilidade

O projeto foi desenvolvido com foco na reprodutibilidade e facilidade de uso. Para isso, foram implementados:

- Documentação detalhada no README do repositório
- Instruções claras para instalação de dependências via `requirements.txt`
- Scripts bem documentados com docstrings e comentários explicativos
- Geração automática de relatórios de resultados em formato markdown
- Logs informativos durante a execução dos algoritmos
- Visualizações comparativas das soluções para análise dos resultados

Os capítulos seguintes detalham cada aspecto do trabalho, desde a fundamentação teórica até a análise dos resultados obtidos, apresentando uma visão completa do desenvolvimento e das conclusões alcançadas.

# Referências Bibliográficas

- [1] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162:439–485, 2005.
- [2] Marco Goldbarg and Elizabeth Goldbarg. *Grafos: Conceitos, algoritmos e aplicações*. Elsevier, Rio de Janeiro, 2012.
- [3] George Karakostas. A better approximation ratio for the vertex cover problem. *Dept. of Computing and Software, McMaster University*, October 2004.
- [4] Luiz Augusto Silva Veloso, Letícia Alves da Silva, and Fábio Pires Mourão. Aplicação do problema de localização de facilidades À alocação de meios de segurança. In *IX Seminário de Iniciação Científica do IFMG*. Instituto Federal de Minas Gerais (IFMG), July 2021.