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

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.

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 eficiente e 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

Fundamentação Teórica

3.1 Algoritmos Genéticos

Os Algoritmos Genéticos (GAs) são um ramo dos algoritmos evolucionários que utilizam modelos computacionais dos processos naturais de evolução para resolver problemas complexos de otimização [4]. Segundo [7], eles são definidos como uma técnica de busca baseada na metáfora do processo biológico de evolução natural, oferecendo uma abordagem heurística para otimização global. No contexto do nosso problema de cobertura de vértices para posicionamento de câmeras no bairro de Ondina, os GAs se mostram particularmente úteis devido à sua capacidade de explorar eficientemente um espaço de soluções complexo.

3.1.1 Conceitos Fundamentais e Terminologia

- **Inspiração Biológica:** Holland [4] desenvolveu os GAs inspirando-se na genética e na teoria da evolução das espécies, estabelecendo uma base teórica sólida que utiliza terminologia análoga aos conceitos biológicos.
- **Cromossomos e Genes:**
  - No nosso problema, cada cromossomo representa uma possível configuração de câmeras no bairro de Ondina
  - Os genes são bits que indicam presença (1) ou ausência (0) de câmera em cada vértice do grafo
  - O "alelo"é o valor binário que indica se há ou não câmera naquele ponto
  - O "locus"corresponde à posição específica no bairro onde pode ser instalada uma câmera
- **Genótipo e Fenótipo:**

  - O "genótipo"é a sequência binária que representa a distribuição das câmeras
  - O "fenótipo"é a cobertura efetiva alcançada por essa configuração de câmeras no bairro

- **População:** Conforme explica [7], os GAs trabalham com uma população de soluções candidatas que evoluem através de operadores genéticos, sendo cada solução avaliada quanto à sua eficácia na cobertura do bairro.

3.1.2 Funcionamento Básico

O algoritmo genético para o problema de cobertura de vértices opera através dos seguintes passos:

1. **Inicialização:** Uma população inicial de configurações de câmeras é gerada aleatoriamente, respeitando o número máximo permitido de câmeras.

2. **Avaliação:** Cada configuração é avaliada considerando:

   - Número de vértices (pontos do bairro) cobertos
   - Custo total da solução (número de câmeras utilizadas)
   - Distribuição espacial das câmeras para maximizar a cobertura

3. **Seleção de Pais:** As melhores configurações são selecionadas para reprodução, priorizando aquelas que oferecem maior cobertura com menos câmeras.

4. **Operadores Genéticos:**

   - **Crossover:** Combina características de duas boas configurações de câmeras
   - **Mutação:** Altera aleatoriamente a posição de algumas câmeras para explorar novas possibilidades

5. **Atualização da População:** As novas configurações substituem as menos eficientes da população anterior.

6. **Critério de Parada:** O processo continua até atingir uma cobertura satisfatória ou um número máximo de gerações.

3.1.3 Características Principais

Os GAs possuem características que os tornam particularmente adequados para o problema de cobertura de vértices:

- **Probabilístico:** A natureza probabilística permite explorar diferentes configurações de câmeras, evitando mínimos locais.
- **Simplicidade:** A implementação é relativamente simples, necessitando apenas de informações sobre a cobertura local de cada câmera.
- **População:** O trabalho com múltiplas soluções simultaneamente permite encontrar diferentes configurações eficientes de câmeras.
- **Codificação:** A representação binária é natural para o problema, onde cada bit representa a presença ou ausência de uma câmera.
- **Flexibilidade:** O algoritmo pode ser facilmente adaptado para diferentes critérios de otimização, como custo das câmeras ou priorização de áreas específicas do bairro.

3.1.4 Representação e Operadores

- **Representação Cromossomial:**
  - Cada cromossomo é um vetor binário de tamanho $|V|$, onde V é o conjunto de vértices do grafo
  - Um bit 1 indica a instalação de uma câmera naquele vértice
  - A soma dos bits deve respeitar o limite máximo de câmeras disponíveis
- **Função de Avaliação:**
  - Calcula a cobertura total alcançada pela configuração
  - Penaliza soluções que excedem o número máximo de câmeras
  - Considera a distribuição espacial para evitar redundância na cobertura
- **Operadores Genéticos:**
  - **Crossover:** Troca segmentos entre duas configurações promissoras
  - **Mutação:** Altera posições de câmeras para explorar novas áreas do bairro
  - **Elitismo:** Preserva as melhores configurações encontradas

3.1.5 Aplicação no Problema de Cobertura de Vértices

No contexto específico do bairro de Ondina, o algoritmo genético foi implementado considerando:

- **Representação:** Cada cromossomo representa uma possível distribuição das 40 câmeras disponíveis
- **Genes:** Indicam a presença (1) ou ausência (0) de câmeras em cada esquina ou ponto estratégico
- **Função de Avaliação:** Considera:
  - Número de ruas e pontos cobertos
  - Distribuição espacial para maximizar a área monitorada
  - Priorização de pontos críticos (próximos a escolas, comércios, etc.)
- **Restrições:**
  - Número máximo de 40 câmeras
  - Necessidade de cobrir pontos estratégicos
  - Limitações de visibilidade devido à topografia do bairro

3.2 Fundamentação Teórica de Algoritmos Gulosos

A monografia de Victor de Oliveira Colombo [1] aborda detalhadamente os algoritmos gulosos, que são uma classe de algoritmos muito importantes para a resolução de problemas de otimização. A característica principal desses algoritmos é a busca pela **solução ótima global** através de **escolhas localmente ótimas**, conhecidas como **escolhas gulosas** [1].

Conceitos Chave

- **Escolhas Gulosas:** A ideia central é que, em cada passo do algoritmo, a escolha que parece ser a melhor naquele momento é feita, sem considerar o impacto futuro dessas escolhas em subproblemas [1].
- **Propriedade da Escolha Gulosa:** Para que um algoritmo guloso funcione corretamente, é necessário que ele possua a propriedade da escolha gulosa, ou seja, que uma sequência de escolhas localmente ótimas leve à solução globalmente ótima [1].
- **Decisões Irreversíveis:** Uma vez que uma decisão é tomada em um algoritmo guloso, ela não é alterada posteriormente. Isso contrasta com outras abordagens como a programação dinâmica, onde as decisões podem ser revistas com base em subproblemas [1].

- **Eficiência e Limitações:** Algoritmos gulosos são geralmente mais eficientes em termos de complexidade computacional quando comparados com outras técnicas como programação dinâmica ou força bruta, mas eles não garantem encontrar a solução ótima para todos os tipos de problemas [1].

Formalização e Definição

Uma das dificuldades no estudo de algoritmos gulosos é a falta de uma definição precisa e formal que abranja todos os problemas que podem ser resolvidos por essa técnica [1]. A monografia cita trabalhos de autores como Edmonds e Lawler na teoria dos matroides, e de Borodin, Nielsen e Rackoff, que tentam formalizar a classe de problemas considerados "gulosos"[1].

O Papel da Intuição

A intuição e o raciocínio por trás das técnicas gulosas são tão importantes quanto a aplicação dos algoritmos. É essencial entender quando e como usar algoritmos gulosos, em vez de outras técnicas, e como adaptar a estratégia para cada problema específico. A monografia de Colombo [1] foca no desenvolvimento dessa intuição e capacidade de identificar problemas que podem ser resolvidos com uma abordagem gulosa.

Importância da Análise

Apesar de sua simplicidade, a aplicação de algoritmos gulosos exige uma análise cuidadosa. É preciso demonstrar que o algoritmo guloso específico está correto, e para isso são usados argumentos como o "argumento de troca", onde se compara a solução gulosa com uma solução ótima para provar sua corretude.

Aplicações e Exemplos

A monografia [1] explora diversos problemas onde algoritmos gulosos são aplicados, como:

- **Problema do Salto do Sapo:** Utilizado para demonstrar a técnica do argumento de troca, onde a melhor escolha é ir o mais longe possível a cada salto.
- **Escalonamento de tarefas:** A ordenação gulosa é aplicada para minimizar a multa total ao processar tarefas, demonstrando a importância da ordem de processamento para atingir a solução ótima.

- **Operações em Vetor:** Ilustra como transformar um problema de otimização em um problema de decisão que pode ser resolvido com um algoritmo guloso, usando a técnica de fixar um valor H.
- **Variação do Problema da Mochila:** Mostra a combinação de algoritmos gulosos e programação dinâmica, onde a ordenação dos itens por um critério guloso é crucial antes de aplicar a programação dinâmica.
- **Problema da Partição:** Demonstra como restrições específicas na entrada de um problema podem permitir o uso de abordagens gulosas.

Em resumo, a fundamentação teórica de algoritmos gulosos reside em sua simplicidade, eficiência e na capacidade de fazer escolhas ótimas localmente, com a condição de que essas escolhas levem a uma solução global ótima. No entanto, é crucial analisar cuidadosamente cada problema para garantir que a abordagem gulosa seja adequada e, se necessário, combinar a abordagem gulosa com outras técnicas para alcançar soluções eficazes.

Capítulo 4

Descrição Formal do Problema

4.1 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.

4.2 Restrições do Problema

O problema apresenta dois casos com diferentes conjuntos de restrições:

4.2.1 Caso 1: Cobertura Máxima

- Número máximo fixo de 40 câmeras disponíveis
- Necessidade de maximizar a cobertura com os recursos limitados
- Priorização de áreas estratégicas e de maior fluxo
- Restrições orçamentárias fixas

4.2.2 Caso 2: Cobertura Completa

- Necessidade de cobrir 100% dos vértices do grafo
- Minimização do número total de câmeras necessárias
- Garantia de monitoramento de todas as áreas
- Sem restrição prévia do número de câmeras

4.3 Função Objetivo

O problema apresenta funções objetivo distintas para cada caso:

4.3.1 Caso 1: Cobertura Máxima

Dado um número fixo de 40 câmeras, a função objetivo é:

$$\text{maximizar } f(C) = |\{v \in V : v \text{ é coberto por alguma câmera em } C\}|$$

onde:

- C é o conjunto de vértices selecionados para instalação de câmeras
- $|C| \leq 40$ (restrição do número máximo de câmeras)
- Um vértice v é considerado coberto se existe uma câmera em v ou em algum vértice adjacente a v

4.3.2 Caso 2: Cobertura Completa

Para o caso de cobertura completa, a função objetivo é:

$$\text{minimizar } f(C) = |C|$$

sujeito a:

$$\forall v \in V, \exists c \in C : (v = c) \text{ ou } (v, c) \in E$$

onde:

- $C \subseteq V$ é o conjunto de vértices selecionados para instalação de câmeras
- Todo vértice $v \in V$ deve ser coberto por pelo menos uma câmera
- Não há restrição prévia para o tamanho de $|C|$

4.4 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.

4.5 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.

4.5.1 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 4.1 apresenta a visualização inicial das ruas do bairro de Ondina, com base nos dados brutos extraídos do OSM.

Figura 4.1: Visualização inicial das ruas do bairro de Ondina.

4.5.2 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 4.2 mostra a visualização das ruas com os vértices associados.

Figura 4.2: Visualização das ruas do bairro de Ondina com os vértices associados.

4.5.3 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 4.3, mostrando a estrutura final com as simplificações implementadas.

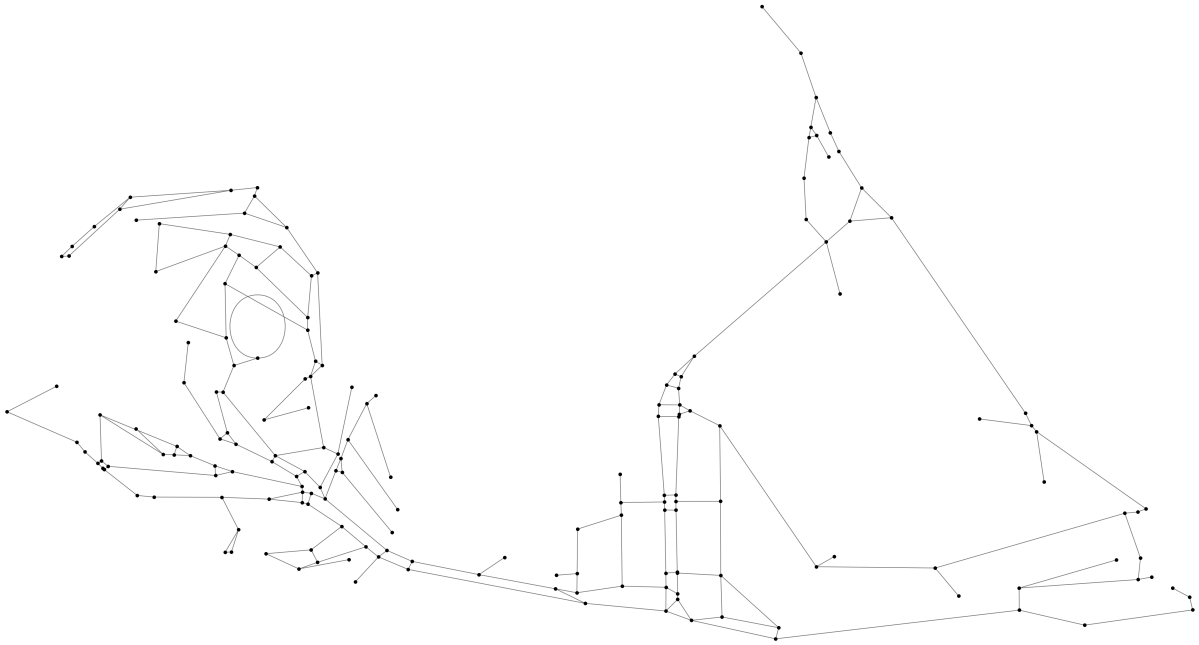

Figura 4.3: Visualização do grafo simplificado do bairro de Ondina.

4.5.4 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.

4.5.5 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".

4.6 Redução e Complexidade do Problema

Redução de Vertex Cover para Set Cover

O problema Cobertura de Vértices é um problema clássico em teoria dos grafos, onde dado um grafo $G = (V, E)$, busca-se um conjunto de vértices $S \subseteq V$ tal que toda aresta $e \in E$ tenha pelo menos uma extremidade em S [6]. Já o problema Cobertura de Conjuntos é um problema mais geral, onde dado um conjunto universo U e uma coleção de subconjuntos $S_1, S_2, \ldots, S_m$ de U, busca-se uma subcoleção de conjuntos cuja união cubra todo o conjunto U [6].

A redução de cobertura de vértices para Set Cover demonstra que o problema cobertura de vértices é um caso especial do problema Set Cover. A ideia chave desta redução é transformar uma instância do problema cobertura de vértices em uma instância do problema Set Cover, de tal forma que a solução de uma corresponde à solução da outra [6].

Construção da Instância de Set Cover

Dada uma instância do problema cobertura de vértices, definida por um grafo $G = (V, E)$ e um número k, construímos uma instância do problema Set Cover da seguinte forma:

- O conjunto universo U da instância de Set Cover é o conjunto de arestas E do grafo G.
- Para cada vértice $i \in V$ do grafo G, criamos um conjunto S_i que contém todas as arestas incidentes a i.

Equivalência entre as Instâncias

Afirmamos que existe uma cobertura de vértices de tamanho no máximo k em G se e somente se existe uma cobertura de conjuntos de tamanho no máximo k em U [6].

- **Se** $\{i_1, i_2, \ldots, i_l\}$ com $l \leq k$ é uma cobertura de vértices em G, **então** os conjuntos $S_{i_1}, S_{i_2}, \ldots, S_{i_l}$ formam uma cobertura de conjuntos em U (pois toda aresta é coberta por pelo menos um dos vértices $i_1, i_2, \ldots, i_l$, e portanto, está em pelo menos um dos conjuntos correspondentes) [6].

- **Se** $\{S_{i_1}, S_{i_2}, \ldots, S_{i_l}\}$ com $l \leq k$ é uma cobertura de conjuntos em U, **então** os vértices $\{i_1, i_2, \ldots, i_l\}$ formam uma cobertura de vértices em G (pois toda aresta em E está em pelo menos um dos conjuntos $S_{i_1}, S_{i_2}, \ldots, S_{i_l}$, e portanto, é coberta por pelo menos um dos vértices correspondentes) [6].

Esta redução estabelece que o problema cobertura de vértices é um caso especial do problema Set Cover, e que qualquer algoritmo que resolva o problema Set Cover pode ser usado para resolver o problema cobertura de vértices. No entanto, é importante notar que essa redução não necessariamente preserva a qualidade de aproximação das soluções. Por exemplo, apesar de termos um algoritmo de aproximação para Set Cover, isso não implica diretamente em um algoritmo com a mesma garantia de aproximação para o problema cobertura de vértices, especialmente quando se busca soluções ótimas. Uma análise mais cuidadosa é necessária para entender como as garantias de aproximação são preservadas ou perdidas durante reduções. [6]

Capítulo 5

Solução Algorítmica

5.1 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.

5.1.1 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)

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

Inicialização:
vertices_nao_cobertos ← V
cobertura ← ∅

while vertices_nao_cobertos ≠ ∅ **do**
    melhor_vertice ← null
    max_cobertura ← 0
    for all v ∈ V **do**
        if v ∉ cobertura **then**
            vizinhos ← N(v) ∪ {v} ▷ N(v) são os vizinhos de v
            cobertura_atual ← |vizinhos ∩ vertices_nao_cobertos|
            if cobertura_atual > max_cobertura **then**
                max_cobertura ← cobertura_atual
                melhor_vertice ← v
            end if
        end if
    end for
    if melhor_vertice = null **then**
        break
    end if
    cobertura ← cobertura ∪ {melhor_vertice}
    vertices_nao_cobertos ← vertices_nao_cobertos \ (N(melhor_vertice) ∪ {melhor_vertice})
end while
return cobertura

5.1.2 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)

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

Inicialização:
cobertura ← ∅
vertices_cobertos ← ∅

for i = 1 **to** p **do**
    melhor_vertice ← null
    max_novos_cobertos ← 0
    for all v ∈ V **do**
        if v ∉ cobertura **then**
            vizinhos ← N(v) ∪ {v}
            novos_cobertos ← |vizinhos \ vertices_cobertos|
            if novos_cobertos > max_novos_cobertos **then**
                max_novos_cobertos ← novos_cobertos
                melhor_vertice ← v
            end if
        end if
    end for
    if melhor_vertice = null **or** max_novos_cobertos = 0 **then**
        break
    end if
    cobertura ← cobertura ∪ {melhor_vertice}
    vertices_cobertos ← vertices_cobertos ∪ N(melhor_vertice) ∪ {melhor_vertice}
end for
return (cobertura, vertices_cobertos)

5.1.3 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 Genético para Cobertura Máxima

Require: Grafo G, `tamanhoPop`, `gerações`, `taxaCrossover`, `taxaMutação`, `maxCameras`
Ensure: Melhor indivíduo encontrado (lista binária indicando posições de câmeras)

1. Inicialização:
população ← GERARPOPULAÇÃO(`tamanhoPop`, `maxCameras`)
melhorSol ← ∅
melhorFitness ← $-\infty$

2. Evolução:
for g de 1 até `gerações` **do**
    AVALIARPOPULAÇÃO(população, G, `maxCameras`) ▷ calcula *fitness* de cada indivíduo
    indMelhor ← OBTERMELHOR(população)
    if FITNESS(indMelhor) > melhorFitness **then**
        melhorFitness ← FITNESS(indMelhor)
        melhorSol ← COPIAR(indMelhor)
    end if
    novaPop ← SELECIONARELITE(população) ▷ mantém os melhores indivíduos
    while |novaPop|¡`tamanhoPop` **do**
        pai1, pai2 ← SELECIONARPAIS(população)
        filho1, filho2 ← CROSSOVER(pai1, pai2, `taxaCrossover`, `maxCameras`)
        filho1 ← MUTAR(filho1, `taxaMutação`)
        filho2 ← MUTAR(filho2, `taxaMutação`)
        novaPop ← novaPop ∪$\{filho1, filho2\}$
    end while
    população ← COPIAR(novaPop)
end for

3. Resultado:
return melhorSol

5.1.4 Complexidade Computacional

A seguir, apresentamos a análise de complexidade dos algoritmos implementados:

- **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.
- Em cada iteração, são visitados todos os vértices ainda não cobertos.

- **Cobertura Máxima (Guloso):**
  - **Tempo:** $O(p \cdot |V|)$, onde p é o número máximo de câmeras.
  - **Espaço:** $O(|V|)$ para armazenar os conjuntos de vértices.
  - O loop principal executa exatamente p vezes.
  - Em cada iteração, são visitados todos os vértices ainda 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:
    - Avaliam-se P indivíduos;
    - Realizam-se operações de crossover e mutação em $O(|V|)$;
    - Calcula-se o valor de fitness para cada indivíduo em $O(|V|)$.

5.1.5 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)

Capítulo 6

Experimentos

6.1 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.

6.1.1 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.

6.2 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

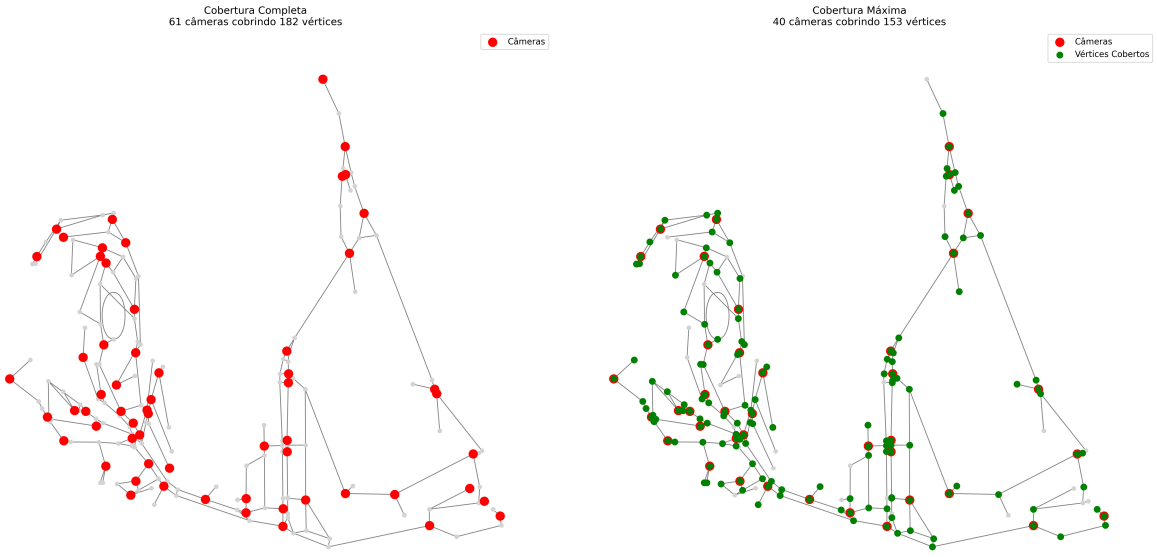

Figura 6.1: Comparação das soluções de cobertura. À esquerda: cobertura completa com 61 câmeras cobrindo 182 vértices. À direita: cobertura máxima com 40 câmeras cobrindo 153 vértices.

A Figura 6.1 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.

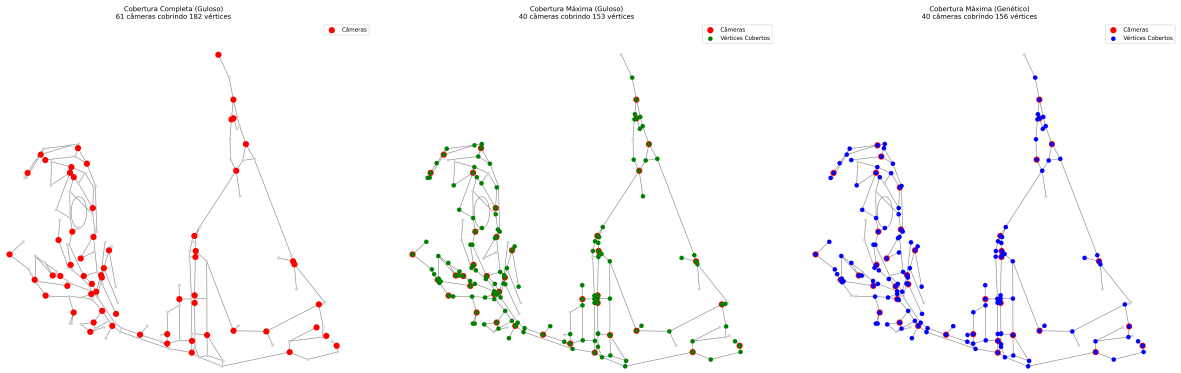

Figura 6.2: Comparação visual das três abordagens implementadas. Esquerda: cobertura completa com algoritmo guloso (61 câmeras cobrindo 182 vértices). Centro: cobertura máxima com algoritmo guloso (40 câmeras cobrindo 153 vértices). Direita: cobertura máxima com algoritmo genético (40 câmeras cobrindo 156 vértices). Pontos vermelhos indicam câmeras, pontos verdes/azuis indicam vértices cobertos.

A Figura 6.2 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 7

Considerações Finais

7.1 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.

7.1.1 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.

7.1.2 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] Victor de Oliveira Colombo. Material didático sobre algoritmos gulosos, 2018.

[2] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162:439–485, 2005.

[3] Marco Goldbarg and Elizabeth Goldbarg. *Grafos: Conceitos, algoritmos e aplicações*. Elsevier, Rio de Janeiro, 2012.

[4] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, 1975. Pioneering work on genetic algorithms and their theoretical foundations.

[5] George Karakostas. A better approximation ratio for the vertex cover problem. *Dept. of Computing and Software, McMaster University*, October 2004.

[6] Jon Kleinberg and Éva Tardos. *Algorithm Design*. Pearson Education, Inc., 2006.

[7] Ricardo Linden. *Livro algoritmos geneticos*. 2006. Introdução aos algoritmos genéticos com foco em implementação em Java.

[8] 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.