# Atividade 4: monitoramento de São João del-Rei

**Gustavo Henriques da Cunha**
Ciência da Computação
Grafos
UFSJ

10/11/2023

# 1 Introdução

Esta atividade têm como motivação a prática dos algoritmos vistos em aulas, buscando achar uma solução para uma proposta de um problema de monitoramento de uma cidade, onde se busca colocar a menor quantidade possível de câmeras em certas esquinas para monitorar as ruas de uma região. Para isso, usaremos aproximações e heuristicas, mais especificamente sobre o problema de cobertura de vértices mínimo, em busca de cobrir as ruas. Foram desenvolvidas duas soluções, onde iremos compará-las, buscando saber qual a melhor para a situação do problema.

# 2 Proposta

A prefeitura de São João del-Rei contratou uma empresa para projetar um sistema de vigilância por câmeras na cidade. A região escolhida para o fase piloto do projeto é o centro da cidade, mais especificamente, uma região num raio de 1km do Campus Santo Antônio da Universidade Federal de São João del-Rei. A figura abaixo apresenta uma visualização da região de interesse.

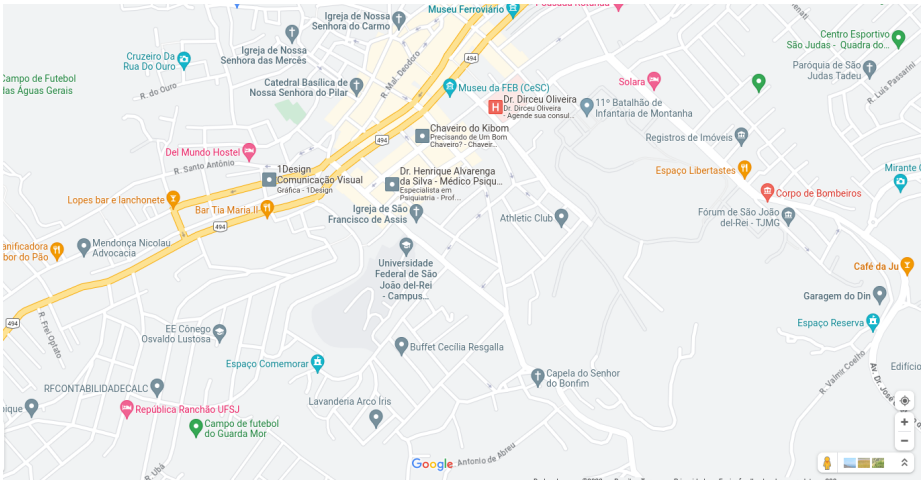


Figura 1: Mapa da região

A câmera definida para realizar a vigilância deve ser posicionada em uma esquina e realiza filmagens em 360º, sendo capaz de monitorar todas as ruas ligadas pela esquina onde está posicionada.

Um dos objetivos do projeto é reduzir o custo. Para isso, é preciso posicionar o menor número possível de câmeras capazes de monitorar todas as ruas da região.

A equipe de projeto definiu que o método de solução do problema seria através de grafos e, por isso, modelou o centro da cidade de São João del-Rei

como uma rede em que as esquinas são representadas por vértices e as ruas são representadas por arestas. A figura abaixo apresenta visualmente o grafo que representa a região de interesse.

Figura 2: Grafo que representa a região

Obtenha uma solução para o problema apresentado. Você deve fornecer a lista de esquinas nas quais serão instaladas as câmeras e as ruas monitoradas por cada uma das câmeras.

Para isso, implemente um algoritmo capaz de obter uma cobertura minimal, utilizando qualquer solução heurística que você conseguir elaborar. O arquivo "sjdr.gml"oferece uma representação do grafo da região de interesse, que pode ser lido e manipulado por qualquer biblioteca de grafos. Os nós são representados por íncices numéricos e as ruas estão representadas como atributos das arestas com o nome 'name'.

# 3 Intruções de execução

A solução, cujo detalhamento está na seção abaixo, foi dividida em duas funções, que estão ambas no arquivo 'vertex_cover.py'. Para executá-lo é necessário o arquivo 'sjdr.gml' estar na mesma pasta dele. Como resultado, será criado dois arquivos de texto, 'vca.txt', com a solução do algoritmo aproximado, e 'vcg.txt', com a solução do algoritmo guloso. Nesses arquivos será imprimido o número de vértices com câmeras, quais são esses vértices e quais ruas cada um cobre.

# 4 Solução

Para resolvermos o problema, foi utilizada a linguagem *Python*, com a biblioteca *NetworkX*, devido a facilidade de lidar com as estruturas de dados de grafos usando a biblioteca.

Como descrito na proposta, a região de interesse foi modelada como um grafo, de forma que queremos saber o menor número de esquinas, representadas como vértices, que podemos colocar câmeras de forma a cobrir todas as ruas, representadas como arestas. Isso representa um caso claro de um problema de cobertura de vértices, onde dado um grafo G = (V,E), queremos encontrar o menor conjunto $C \subset V$, para que, dado qualquer $(u, v) \in E$, $u \in C$ ou $v \in C$.

O problema da cobertura de vértices é, conhecidamente, NP-Completo, sendo assim, não existe algoritmo que o resolva em tempo polinomial se $P \neq NP$ . Assim, para uma entrada grande, é difícil conseguir computar eficientemente uma resposta ótima para o problema. Por conseguinte, devemos utilizar de soluções aproximadas e heuristicas para resolver o problema.

Nesse caso, foi decido usar duas soluções para aproximar uma resposta ao problema. A primeira consiste em um famoso algoritmo de aproximação ao problema, cujo a aproximação é fator-2. O segundo é uma heuristica gulosa que seleciona o vértice com maior grau em cada iteração.

## 4.1 Algoritmo aproximado

Essa solução é bastante simples e bem conhecida, onde dado um grafo, ela retorna uma solução que não é pior do que 2 vezes a solução ótima.

Segue o código em Python da solução:

```
1  def vertex_cover_approximation(G):
2      C = set()
3      E = [edge for edge in G.edges()]
4      while E:
5          (u, v) = random.choice(E)
6          C.add(u)
7          C.add(v)
8          for edge in E:
9              (u_line, v_line) = edge
10             if ((u_line in C) or (v_line in C)):
```

```
                E.remove(edge)
    return C
```

A partir da lista de arestas de um grafo G, a função irá selecionar sempre uma aresta, adicionar suas duas extremidades no conjunto C com a solução e remover da lista de arestas todas as arestas com um desses vértices como extremidade, finalizando quando a lista estiver vazia.

Com esse algoritmo garantimos uma solução que não é pior do que 2 vezes a solução ótima, mas não iremos provar isso aqui.

### 4.2 Algoritmo guloso

A solução gulosa é também bem simples, e pode retornar uma solução bem próxima a solução ótima.

Segue o código em Python da solução:

```
def vertex_cover_greedy(Graph):
    G = Graph.copy()
    C = set()
    while G.edges():
        mdv = max(G.nodes(), key= lambda node: G.degree(
   node))
        C.add(mdv)
        G.remove_edges_from(list(G.edges(mdv)))
    return C
```

A partir de uma cópia do grafo original, verificamos o vértice com maior grau, o adicionamos no conjunto solução C, e removemos todas as arestas que contem esse vértice em uma de suas extremidades. Repetimos isso até o grafo não ter mais vértices.

## 5 Resultados

Dado o arquivo 'sjdr.gml', o resultado do algoritmo guloso foi bem melhor do que o de aproximação. A heuristica gulosa teve um resultado de 308 esquinas com câmeras, enquanto o algoritmo de aproximação teve, geralmente, entre 450 e 480 esquinas com câmeras. A variação nos números em cada execução se deve ao fato de escolhermos de maneira aleatória a aresta em cada iteração.

Assim, apesar de não ser um resultado ótimo, conseguimos de forma eficiente uma resposta consideravelmente boa para o problema proposto, sendo que, caso fosse um problema da vida real, a prefeitura conseguiria uma grande economia de recursos em seu projeto, com o resultado esperado de vigilância no final.