Aluna: Ana Cláudia Machado

Introdução:

Neste trabalho, desenvolveu-se uma heurística para o monitoramento das ruas de determinada parte de São João del Rei, de modo a saber em quais pontos estratégicos câmeras deveriam ser colocadas.

Definiu-se que o método de resolução do problema seria baseado em grafos e, assim, as ruas seriam arestas e, as esquinas, os vértices. Desse modo, o dever da heurística proposta é encontrar um conjunto minimal que representa a cobertura de vértices para o grafo dado, ou seja, para o sistema de vigilância, em quais esquinas as câmeras deveriam ser colocadas a fim de, juntas, cobrirem todas as ruas.

Entrada:

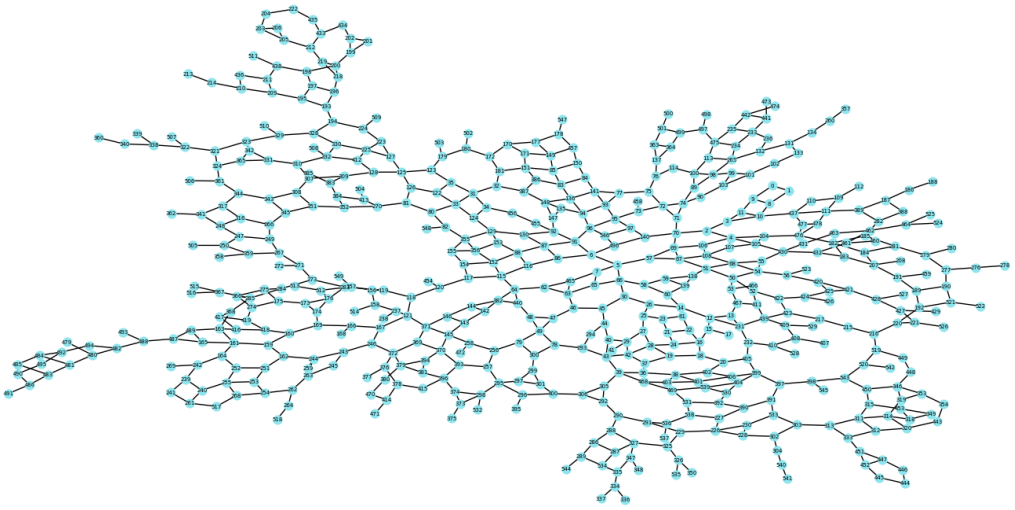
A entrada é um arquivo *.gml* fornecido para a resolução do exercício, o qual contém, a priori, todos os vértices existentes no grafo e, a posteriori, cada uma das arestas, com seu início, fim e o nome da rua que representa.

É possível ler tal entrada com a função *read_gml()* da biblioteca Networkx, de modo simples e prático. Destaca-se, ainda, que diversas funções dessa mesma biblioteca foram utilizadas nesse código, haja vista que, como o programa envolvia um processamento de mais dados, decidiu-se por utilizar funções já prontas, pois imaginou-se que essas estariam implementadas da forma mais eficiente possível.

A seguir, será explicada a lógica utilizada pela heurística adotada.

Heurística:

Em primeiro momento, é necessário analisar o grafo dado:

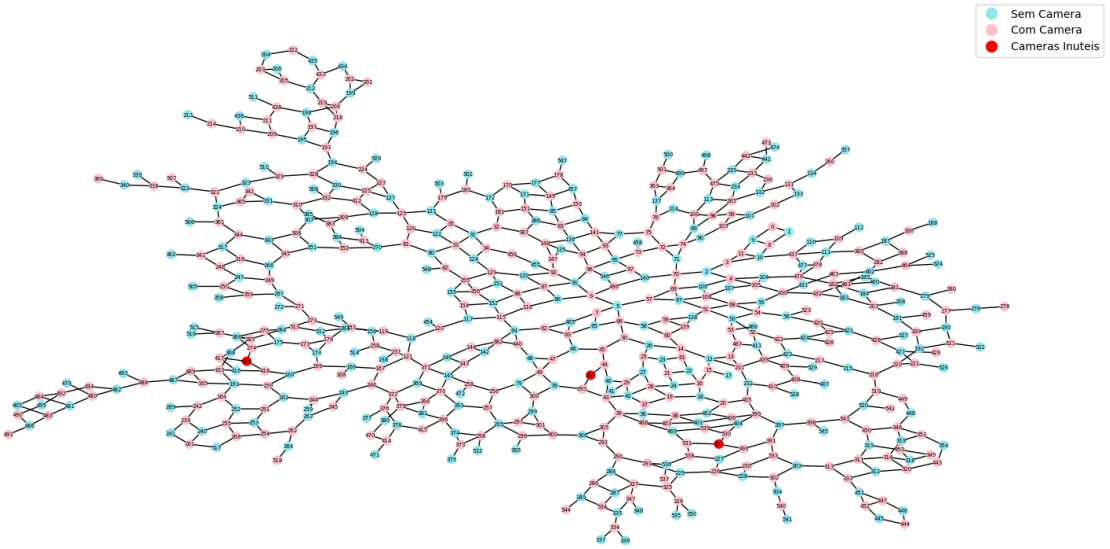


Percebe-se que, por serem esquinas na vida real, a maioria dos vértices possui poucas conexões, entretanto, é válido aproveitar os vértices com mais arestas incidentes para colocar câmeras, pois, logicamente, tais pontos cobrirão um maior número de ruas de uma só vez.

Assim, é possível modelar uma heurística que ordene os vértices de acordo com o grau em uma fila de prioridades e, a partir daí, coloque câmeras em tais vértices até que todas as ruas estejam na cobertura. Entretanto, de acordo com essa lógica, uma aresta qualquer do grafo, exemplificada por *(u, v),* seria um acréscimo no grau de ambos os vértices, *u* e *v*. Mesmo que *u* já tenha monitorado tal aresta, o grau de *v* ainda a considera como parte da cobertura que *v* pode fazer. Desse modo, ao ordená-los uma única vez, uma série de vértices, cujo grau ordenado não está mais de acordo com o número de arestas que necessitam de sua cobertura, são colocados como prioridade.

Mesmo que tal resolução seja ineficiente, ainda é possível seguir na mesma linha de raciocínio. Se os vértices forem ordenados repetidamente de acordo com o seu número de arestas adjacentes que precisam de cobertura, tal problema é solucionado. Isso porque, em uma repetição que será finalizada quando todas as arestas estiverem na cobertura, em cada iteração, o primeiro vértice da fila será aquele cuja instalação de uma câmera é mais vantajosa no momento. Portanto, o algoritmo sempre escolherá a melhor esquina para adicionar uma câmera, considerando o número de ruas não monitoradas que ela pode monitorar.

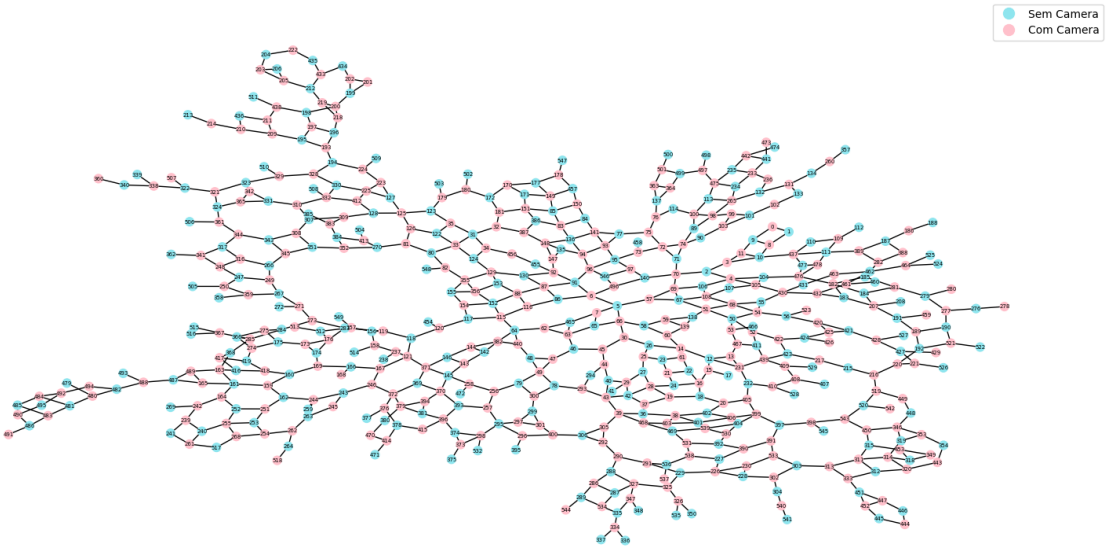
O grafo abaixo representa o funcionamento de tal abordagem:



Nesse caso, o conjunto minimal conta com 305 vértices, o que já é um resultado aceitável, entretanto, analisando a imagem, é possível perceber situações em que o funcionamento do algoritmo teve um desempenho abaixo das expectativas. Os vértices vermelhos representam esquinas que, de acordo com a resposta gerada, possuiriam câmeras, mas é visível que não há nenhuma necessidade, porque todas as esquinas adjacentes a essas possuem câmeras.

Tal falha possui uma explicaçã plausível: no nomento em que os vértices foram repetidamente ordenados, os pontos em vermelho receberam câmeras antes que seus vizinhos as recebessem. Porém, pela disposição das conexões, foi necessário colocar câmeras em arestas adjacentes, o que é comum no grafo, como pode ser visto. Quando esses dois eventos se sobreporam, a dada falha foi gerada, pois todos os seus vizinhos necessariamente receberão câmeras, o que tornou aquela esquina plausível de não receber câmera.

Nesse sentido, pode-se realizar uma verificação nos pontos marcados para receberem câmeras: dado um vértice *v,* se todos os seus adjacentes estiverem também no conjunto minimal, então *v* deve ser excluído do conjunto. Por fim, com tal verificação, chega-se a resposta de que 302 câmeras são necessárias, como é mostrado na imagem a seguir:



Por fim, ressalta-se que, nesse grafo, com esse conjunto de ruas e esquinas, diminuiu-se apenas três câmera, porém, considerando que essas câmeras possuem um custo para ser adquiridas e instaladas e, ademais, tal algoritmo pode ser utilizado para outros grafos, tal verificação se torna útil e necessária.

Saída:

A saída gerada possui 2 partes disponíveis. Primeiro, é possível visualizar os grafos gerados antes e após a execução do algoritmo, pois serão gerados os arquivos *entrada.jpg* e *resultado.jpg*. Também é gerado o arquivo *monitoramento.txt*, no qual é possível visualizar o número de câmeras necessárias e, em seguida, para cada câmera, sua esquina e as ruas que ela monitora.

Conclusão:

Heurística são ferramentas muito úteis no tratamento de problemas difíceis, como a cobertura de vértices, a qual é abordada neste trabalho de forma prática. Ao trazer tal problema com elementos reais, como ruas e esquinas, realiza-se um exercício de adaptação dos conceitos vistos em sala de aula para a realidade.

A heurística apresentada foi desenvolvida perante diversos ajustes na ideia inicial de ordenar os vértices. Tal abordagem se mostrou útil a depender da forma de implementação, pois, em alguns dos casos mostrados, não trouxe resultados tão eficientes.

Por fim, conclui-se que melhorias na forma de implementação e, até mesmo no resultado obtido, ainda podem ser desenvolvidas, haja vista que não considerou-se isso a preocupação central ao criar essa abordagem, pois o objetivo, em primeiro plano, foi elaborar uma heurística funcional.