Universidade de São Paulo – USP São Carlos
Instituto de Ciências Matemáticas e de Computação
SCC5900 – Projeto de Algoritmos – Prof. Marcelo Manzato
Damares Oliveira de Resende – #11022990

Relatório de Atividades – Projeto II

Nesse exercício codificou-se dois algoritmos para clusterização em grafos, o algoritmo de Prim e o Algoritmo de Kruskal.

O algoritmo de Prim é um algoritmo guloso que começa como uma árvore vazia e varre um grafo mantendo dois conjuntos de dados, os vértices já inclusos na árvore mínima e os vértices ainda a serem incluídos. A cada passo, ele considera todos os vértices contidos em cada conjunto e escolhe a aresta de menor peso para adicionar seu vértice à árvore. Os passos desse algoritmo são os seguintes:

1) Criar um conjunto que mantém os nós inclusos na árvore
2) Inicializar os valores dos pesos chaves de todos os vértices para infinito, e o do vértice raiz para zero
3) Enquanto a árvore não contiver todos os vértices:
    a. Escolher o vértice com menor peso ainda não contido na árvore
    b. Incluí-lo à árvore
    c. Atualizar os pesos chaves de todos os vértices adjacentes a ele

Esse algoritmo foi implementado com o auxílio de uma Fila de Prioridades que contém em seu nó raiz a aresta de menor valor, o que evita a ordenação O(N^2) de um array comum. Essa estrutura tem complexidade O(N log N) para inserção e remoção devido os métodos de heapfy_up e heapfy_down. Por fim as K – 1 arestas de menor peso são removidas para se obter K clusteres. A complexidade total do algoritmo é O(N) onde N é o número de vértices.

O algoritmo de Krukal é menos custoso. Ele ordena uma lista de pesos por aresta no início do algoritmo e adiciona os vértices até que N – 1 – K – 1 vértices sejam adicionados, onde N é o número total de vértices e K é o número de clusteres a serem formados.

O algoritmo codificado usa a estrutura de union find com ranqueamento e compressão de caminho, métodos que permitem que os conjuntos de nós sejam adicionados a uma árvore e sempre atualizados de acordo com seu ranking. Caso dois conjuntos sejam tenham um nó em comum, ou seja, caso um círculo se forme, ele é descartado. Do contrário há uma união de conjuntos. A complexidade desse algoritmo é O(E log E) onde E é o número de arestas.

O código foi implementado na linguagem Python e tem como única dependência a biblioteca matplotlib para plotar os resultados.

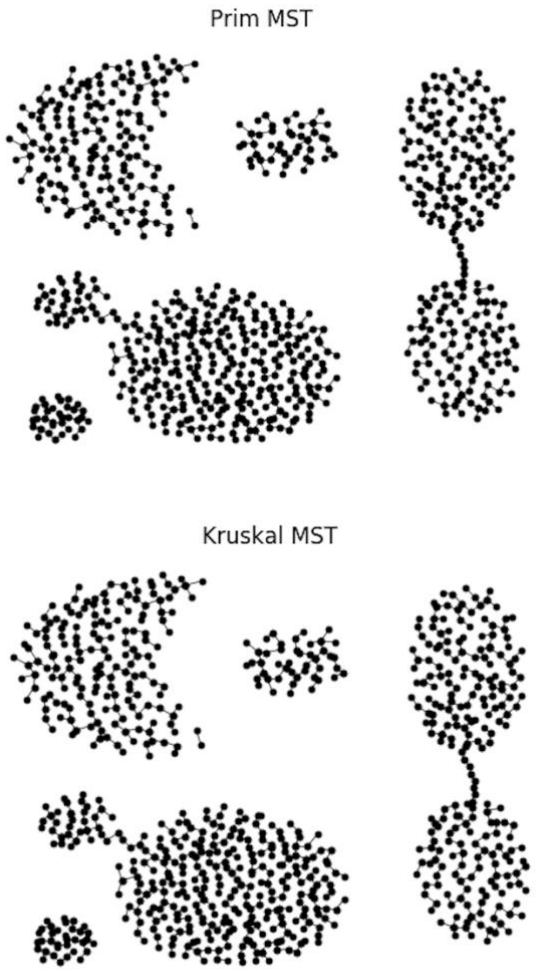
Para executa-lo basta usar o comando abaixo, onde 6 é o número de clusteres:

- python Clusterize.py 6

Como resultado uma figura será gerada com o plot das MSTs e da classificação.

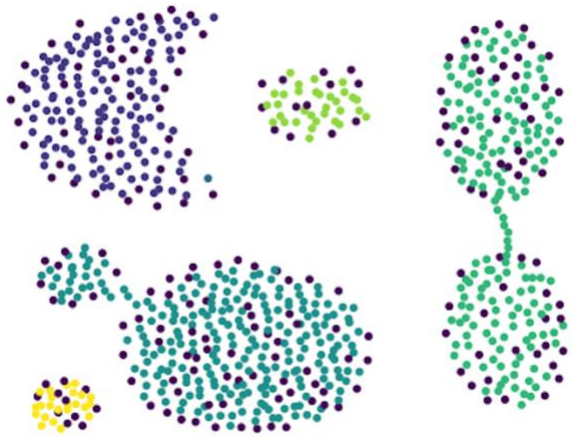
Resultados

Os algoritmos de Prim e Kruskal foram implementados com sucesso e as árvores geradas podem ser observadas abaixo. Porém, a clusterização não é igual ao do algoritmo proposto pelos autores pois para os clusteres do lado direito da figura, os vértices estão muito próximos e o peso das arestas é muito pequeno. Logo outro cluster é gerado de apenas dois componentes que pode ser observado no lado esquerdo superior da figura.



A classificação infelizmente não funcionou completamente. O DFS implementado possui falhas e deixa alguns pontos passarem apesar das árvores estarem corretas. Os pontos roxos são os pontos sem classificação.

Prim Classification



Kruskal Classification