Redes Complexas
2017-2018
Project - Part 1

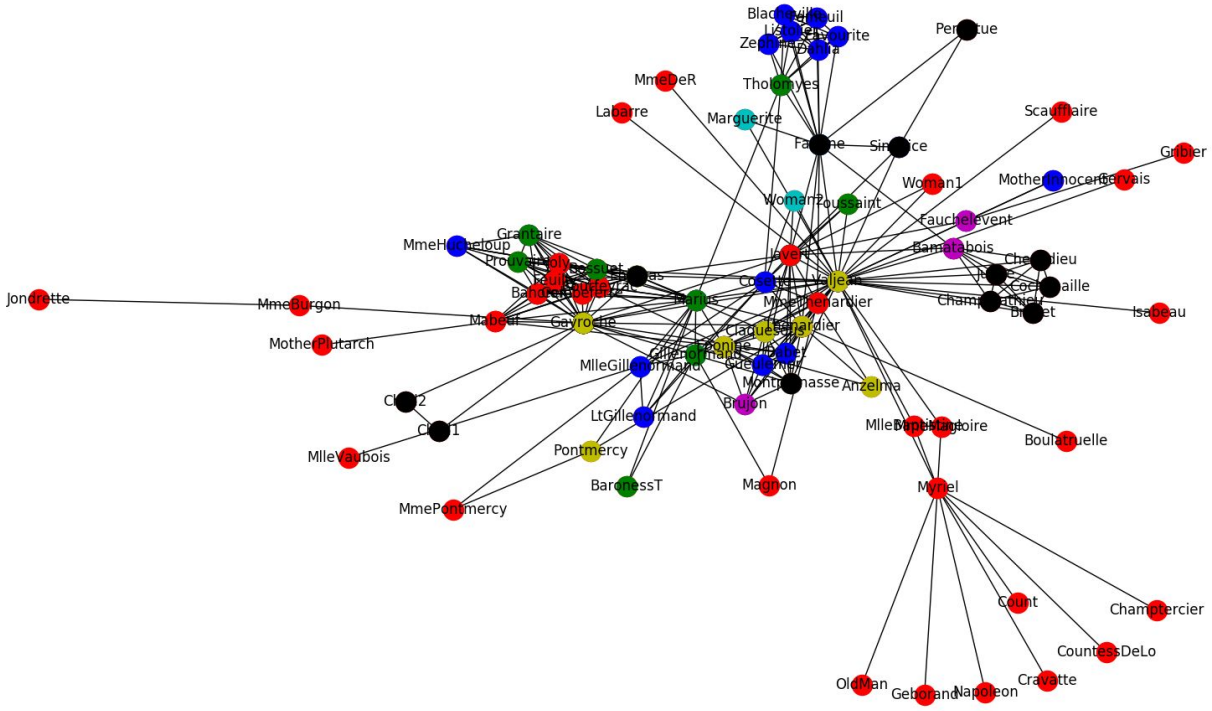

Fig 1. Cliques por cor com tamanho superior a dois, dataset Les Misérables

Grupo 3

Daniel Correia 80967

Pedro Orvalho 81151

Renato Cardoso 81530

1. Introdução

Neste projeto, decidimos explorar as funcionalidades fornecidas pela package de Python para análise e geração de grafos - **Networkx**[1] - e construir um conjunto de exemplos de utilização que possam ser facilmente utilizados, por exemplo, por colegas da cadeira que queiram obter métricas sobre um dataset sem precisarem de aprender a usar o Networkx.

A nossa exploração do Networkx incidiu sobre 3 áreas:

1. Leitura/Importação de datasets existentes (GML e JSON).
2. Cálculo de métricas utilizando os algoritmos do Networkx (clustering, cliques e assortativity).
3. Visualização personalizada de datasets em função das métricas.

A package Networkx disponibiliza ainda ferramentas de geração personalizada de grafos (random graphs, lattices, ego graphs, etc...) e muitos mais algoritmos de cálculo de métricas.

Os **objetivos principais** deste projeto foram:

- Explorar algumas das funcionalidades fornecidas pela package Networkx.
- Avaliar a package num contexto de desenvolvimento: dificuldades de utilização, pontos fortes, truques interessantes e utilidade dos resultados.
- Desenvolver um pacote de exemplos de utilização para novos utilizadores.
- Descobrir e investigar métricas interessantes que sejam disponibilizadas pelo Networkx mas que não tenham sido abordadas (ainda?) nas aulas.

Em relação às **métricas e algoritmos escolhidos**:

1. Os algoritmos de clustering permitem obter resultados sobre o número de **triângulos**, **coeficientes de clustering** e **transitividade** que permitem avaliar a existência de grupos altamente conectados na rede e extrair conclusões sobre a resiliência e robustez da rede.
2. Um **clique**[2] é um subgrafo de um grafo não dirigido em que todos os seus nós estão ligados entre si. Por exemplo, a imagem na capa do relatório representa os vários cliques de tamanho superior a 2 no dataset "Les Misérables". Existe também o **maximal clique**, os nós do maximal clique do grafo G são os cliques do grafo G e existe uma aresta entre dois nós se os cliques não forem disjuntos.
3. A ***assortativity***[3] é a preferência de um determinado nó se conectar a outros nós similares. Esta similaridade pode variar consoante a métrica escolhida, sendo que é mais comum utilizar o *degree*. Para medir a *assortativity* utiliza-se, normalmente, ou o ***assortativity coefficient***, que mede o coeficiente de correlação entre pares de nós, ou o ***neighbor connectivity***, que examina a média do *degree* dos vizinhos por cada *degree*.

O código desenvolvido será disponibilizado em anexo no zip entregável e contém um ficheiro README com uma breve explicação da estrutura e contexto dos módulos. Para além disso, cada módulo de Python desenvolvido tem alguns auxiliares de execução (help messages) relativamente aos argumentos necessários para a sua execução.

[1] Networkx Documentation Page
[2] Clique definition (Wikipedia)
[3] Assortativity definition (Wikipedia)

2. Leitura/Importação de datasets

No contexto de leitura e importação de datasets existentes, a package Networkx **fornece um conjunto de funções de leitura que recebem um dataset e devolvem um objeto manipulável** pelos restantes métodos de análise ou visualização.

Segundo a documentação disponível, **o Networkx suporta uma grande variedade de formatos** utilizados na área: GML, GEXF, JSON, YAML, GraphML, Pickle, entre outros.

Na realidade, **isto raramente foi verdade durante as nossas experiências em desenvolvimento**: por exemplo, inicialmente tínhamos planeado desenvolver um exemplo de utilização para GEXF, mas o parser de Networkx nunca conseguiu processar os datasets GEXF que nós utilizámos.

Relativamente à **importação de datasets em formato GML**, as principais dificuldades encontradas foram causadas devido à **falta de "backward-compatibility"** entre o parser do Networkx para GML e a sintaxe de GML mais antiga dos datasets utilizados.

Exemplos de problemas com GML

- O parser do Networkx só reconhece que um grafo tem pesos (weighted) se no ficheiro GML as arestas tiverem o atributo "weight", embora nos datasets fosse utilizado atributo "value" para representar o peso de uma aresta.[4]

- O parser do Networkx só aceita GML num formato específico ao nível dos espaços entre chavetas, tendo sido necessário alterar manualmente os datasets para contemplar esta restrição (solução possível: substituir "\s+\[" por " [").[5]

Relativamente ao **formato JSON**, o principal problema foi a **falta de datasets**, visto que este grafos normalmente estão associados a **projetos de visualização em D3.js.**[6]

3. Cálculo de métricas

Em relação ao cálculo de métricas, escolhemos utilizar os algoritmos fornecidos pelo Networkx sobre **clustering, cliques e assortativity**. A escolha das métricas e algoritmos foi feita de acordo com o **interesse no tema** e com a **probabilidade de obter resultados mais interessantes.**.

Relativamente ao **clustering**, desenvolvemos 2 tipos de funções: apresentação de métricas de clustering do dataset com mínimo, máximo, média e total; **apresentação de um top N dos nós em função do valor das métricas**.

Para as métricas sobre **cliques**, criámos um módulo que disponibiliza funções de descrição dos cliques do grafo, ou de um certo nó do grafo, ou de todos os nós do grafo.

No contexto da **assortativity**[7], determinamos o ***assortativity coefficient*, a *average neighbor degree* e a *average degree connectivity***. Cada um destes pode ser calculado na totalidade, ou mostrando apenas os top n elementos (variável a ser modificada).

Também pode ser calculada a **mixing matrix**, que mostra a relação entre os vários atributos tendo em atenção a métrica escolhida: esta função pode ainda mostrar os resultados como contagens ou como probabilidades conjuntas. Um dos problemas encontrados foi os **resultados aparecerem como um dicionário de dicionários** que depois tivémos de processar para facilitar a visualização.

[4] StackOverflow post on reading weighted GML graphs issue
[5] StackOverflow post on parsing errors reading GML graphs
[6] "Les Miserables" em JSON encontrado num projeto de D3.js
[7] Networkx assortativity module

4. Visualização de grafos

Para a visualização, utilizámos as funcionalidades do **NetworkX** juntamente com **Matplotlib**[8].

Desenvolvemos no nosso módulo sobre clustering funções que permitem gerar visualizações do top N por degree ou métrica de clustering com tamanho variável em função dos valores das métricas.

O nosso módulo sobre cliques disponibiliza também várias funções para a visualização de cliques:

- **draw_colored_clique_with_size_N**: atribui várias cores aos diferentes cliques com tamanho maior que "size" (default=2), pode desenhar um círculo à volta dos cliques[9] (default=False) e pode ser definido o layout de visualização do grafo[10].
- **draw_colored_maximal_clique**: visualização de um maximal clique do grafo.
- **draw_clique_bipartite**: visualização de um clique bipartido do grafo disponibilizado pelo Networkx ("A bipartite graph whose "bottom" set is the nodes of the graph G, whose "top" set is the cliques of G, and whose edges join nodes of G to the cliques that contain them").
- **draw_top_size_cliques**: devolve visualizações de todos os cliques que pertençam ao top N (default=10) por tamanho de clique no grafo.

5. Discussão

No geral, **a nossa experiência com o Networkx no contexto de leitura de datasets existentes foi negativa** e implicou alterações manuais sobre os datasets para que tudo funcionasse bem. Por outro lado, **se tivéssemos utilizado datasets escritos pelo próprio Networkx, então a leitura teria sido direta e sem problemas** (recorrendo às ferramentas de geração de grafos).

Ao utilizarmos a biblioteca NetworkX.algorithms.clique[11], verificámos que faltavam algumas funções base, por exemplo uma função que indique em quantos cliques um nó está presente (tivémos ser nós a criá-la - number_of_cliques_for_node).

Para além disso, o facto das **métricas do grupo Networkx.algorithms serem devolvidas em formato dicionário**, implica que **não têm uma ordenação bem definida**. Isto **dificultou o desenvolvimento de exemplos de visualização mais complexos**, como grafos com nós de tamanho variável em função do valor da métrica (maior valor -> tamanho maior).

Outro problema é a falta de funções de visualização dos vários cliques de um determinado grafo, tivemos de procurar e criar funções que nos permitam visualizar os cliques de um determinado grafo, funções que podem ser importantes para vários projectos.

Em relação à biblioteca Matplotlib, a nossa experiência foi desgastante, visto que a interface disponibilizada pela biblioteca foi **insuficiente para desenvolvimentos mais avançados**. Por exemplo, não é possível visualizar vários grafos numa só janela, sendo obrigados a abrir uma janela nova para cada grafo que queremos visualizar.

Embora a package Networkx seja **difícil de utilizar em contextos de leitura e visualização**, o **resultado deste projeto foi positivo** visto que os exemplos de utilização são suficientemente genéricos e portáveis para projetos de outros colegas. Por exemplo, projetos que tenham construído grafos de raiz a partir do Networkx (seja por geração ou por construção manual), podem integrar os nossos exemplos para obter métricas sem qualquer esforço adicional.

[8] Matplotlib home page
[9] circles_around_cliques
[10] Networkx layouts docs
[11] Networkx clique algorithms module