REDES COMPLEXAS

Antônio Marcos Machado Bernardes
Ronan José Lopes

Netflix Movies & TV Shows: Estudo, análise e descoberta de conhecimento na rede

Universidade Federal de São João del-Rei

2021

Sumário

1 Introdução

No contexto da ciência da computação, a área de estudo de redes complexas fornece uma abstração para estudo de sistemas onde os elementos desse sistema realizam algum tipo de interação entre si. Essa abstração toma como base o modelo e estrutura de dados dos grafos e possibilita, através da análise de propriedades estruturais e métricas que fornecem indicadores do comportamento da rede, descrever e compreender melhor o sistema como um todo. Como uma área de aplicação multidisciplinar, pode ser utilizada em aplicações biológicas, análise de tráfego de veículos, interações em redes sociais, dentre outros vários exemplos possíveis.

A fim demonstrar um possível caso de uso, o estudo a ser aqui desenvolvido toma como objeto de estudo a base "Netflix Movies and TV Shows"[1], que contém informações relevantes sobre filmes e séries disponíveis no serviço de *streaming* e foi disponibilizada na rede de descoberta de conhecimento Kaggle. Dentre as sugestões de estudo e análise sugeridas na publicação da base, o tópico "Análise de rede de atores/diretores para encontrar insights interessantes"é um exemplo direto de aplicação de estudo da disciplina de redes complexas.

Nas seções seguintes, são detalhadas as propostas de análise e hipóteses a serem testadas na rede mencionada. O estudo desenvolvido tem como ponto de partida uma compreensão rasa acerca da rede tal como descrito no parágrafo anterior e, progressivamente, clarifica melhor o processo de geração, comportamento e predições relativos ao sistema em análise.

[1] https://www.kaggle.com/shivamb/netflix-shows

2 Objetivos e metodologia

No estudo preliminar de qualquer rede, existem algumas métricas que podem dar uma visão geral da estrutura, como por exemplo: diâmetro da rede, densidade, grau médio dos vértices, dentre outros). Aqueles indicadores que se mostrarem viáveis e relevantes serão analisados e detalhados no capítulo seguinte. Além disso, para entender o processo de formação da rede, gerou-se uma rede aleatória com propriedades semelhantes (mesmo número de vértices e probabilidades de arestas) para comparação com a rede analisada, a fim de se observar o quanto a rede em estudo se aproxima de um modelo aleatório.

Ainda com relação à rede, outra análise comumente utilizada é a detecção de comunidades, com o intuito de identificar um agrupamento onde os vértices de cada grupo sejam altamente similares entre si e dissimilares em relação aos vértices dos demais grupos. Ainda olhando para a estrutura da rede, outro objetivo é *rankear* e identificar, através de métricas de centralidade e outros parâmetros de relevância, os vértices que se destacam.

Deixando um pouco de lado as propriedades estruturais e descritivas das análises anteriores e olhando mais pro cenário da base de dados em si, outra proposta principal é tentar entender, a partir do relacionamento entre os atores da base, se é possível formular um modelo que permita a predição de sucesso com base em uma nota de avaliação. Como a base original não contém nenhum dado similar, outro pré-requisito é integrar aos registros essa informação. Para esse atributo, foi adotada a nota no IMDB como critério de avaliação.

As análises, processamento e visualização das informações em estudo foram auxiliadas pelo uso de algumas ferramentas. Na linguagem Python, foram utilizadas as bibliotecas *Igraph*[1] (que eventualmente foi abandonada por questões de performance/estouro de memória) e *Networkx*[2]. A ferramenta *Gephi*[3], desenvolvida em Java, também foi utilizada, principalmente para conferência e visualização. Para *plotagem* de gráficos, foi utilizada a biblioteca *Matplotlib*[4], também em Python.

2.1 Pré-processamento

A base original obtida contém 7.789 registros ao total. Para cada registro, os dados abaixo estão presentes:

[1] https://igraph.org/python/
[2] https://networkx.org/
[3] https://gephi.org/
[4] https://matplotlib.org/

- ID Interno
- Tipo (show, série, dentre outros)
- Título
- Diretor
- Lista do elenco de atores
- País de origem
- Data em que foi adicionado à plataforma
- Ano de lançamento
- Classíficação de idade
- Duração
- Categorias em que é listado
- Breve descrição

Aos atributos originais, foi adicionada a nota do IMDB, obtida através da API de consulta do *The Open Movie Database*[5]. Com um limite de 1.000 requisições diárias, 8 dias foram necessários para consultar toda a base. Dos 7.789 registros, 6.752 (86,6%) retornaram o valor da nota e puderam ser utilizados no estudo de predição.

Outro detalhe importante é que um ator ou atriz podem aparecer em múltiplos registros da base (e de fato ocorre frequentemente). Portanto, para atribuição individual da nota, cada integrante tem pré-processada a sua nota média de acordo com todas as ocorrências na base.

2.2 Modelagem da rede

Visando o objetivo principal do estudo de co-atuação dos elencos, a rede foi modelada tendo em mente representar essa abstração. Para tal, os vértices do grafo representam os atores, ponderados por sua nota média. Em termos de interação, existe uma aresta entre dois vértices a_1 e a_2 se a_1 co-atuou com a_2 em algum registro da base. Pela escolha de modelagem, cada elenco contido em um registro é, por definição, um clique do grafo (ou seja, todos os nós tem ligações entre si).

Para algumas análises e aplicações de algoritmos disponíveis na literatura, é necessário que todos os vértices sejam alcançáveis. Uma forma de garantir que esse pré-requisito

[5]http://www.omdbapi.com

seja atendido é tomar como objeto de estudo a componente gigante da rede (ou seja, o componente que contém o maior número de vértices). Como será mostrado no capítulo seguinte, a componente gigante da rede de co-atuação possui uma cobertura significativa dos vértices originais e foi utilizada como objeto de análise.

3 Análise da Rede

3.1 Sumarização de indicadores descritivos

De imediato, é possível aferir alguns dados básicos da rede completa com o auxílio do *Networkx* ou do *Gephi*:

- Número de nós: 32.881 (maior do que o número de registros pois cada registro contém um elenco com N atores/atrizes)
- Número de arestas: 252.055
- Grau médio: 15,33 (média de ligações/interações de cada vértice/ator)

Conforme descrito na modelagem da rede, optou-se pro obter o sub-grafo da componente gigante da rede, cuja cobertura inclui cerca de 89,5% dos vértices da rede:

- Número de vértices: 29.440
- Número de arestas: 241.744
- Grau médio: 16,42
- Densidade: 0,000557 (Razão entre número de arestas existentes/possíveis)
- Diâmetro: 17 (caminho mais longo possível entre quaisquer 2 vértices da rede)
- Coeficiente de *clustering* médio: 0,824 (Alta probabilidade devido às "aglomerações locais" do *elenco* de cada registro, onde todos os nós têm ligações entre si)
- Comprimento médio de caminho: 5,647 (Distância média, em númeo de vértices, que deveria ser percorrida para alcance entre dois vértices arbitrários)
- Fechamento triadíco: 0,39849 (Probabilidade de formação de "triângulos" - relativamente alta pelo mesmo motivo do coeficiente de *clustering*)

A figura 1 auxilia na visualização de como os graus dos vértices da rede estão distribuídos. A distribuição é exibida de forma que o eixo X representa a cardinalidade possível do grau do vértice de acordo com as ocorrências na base, enquanto no eixo Y está representado sua probabilidade, de acordo com o número efetivo de ocorrências. O grau 9 (ou seja, o ator/atriz co-atuar com 9 outros atores), por exemplo, de maior ocorrência, tem uma probabilidade um pouco maior que 16%.

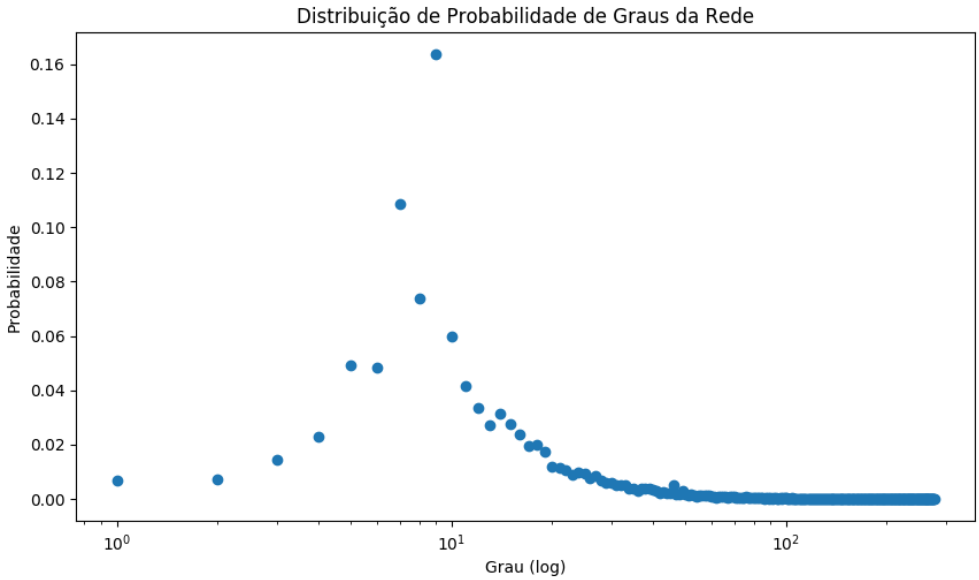


Figura 1 – Distribuição de graus da rede x probabilidade de ocorrência

3.2 Detecção de comunidades

Para detecção de comunidades na rede, foram utilizados os algoritmos de modularidade e método de Louvain (que consiste em um aprimoramento do método anterior) respectivamente no Networkx e no Gephi. Pelo método de Louvain, foram obtidas 93 comunidades, contendo a maior delas 4.823 vértices, a menor 5 vértices, e uma média de 313,19 vértices por comunidade. No Gephi, o método de modularidade gerou 99 comunidades.

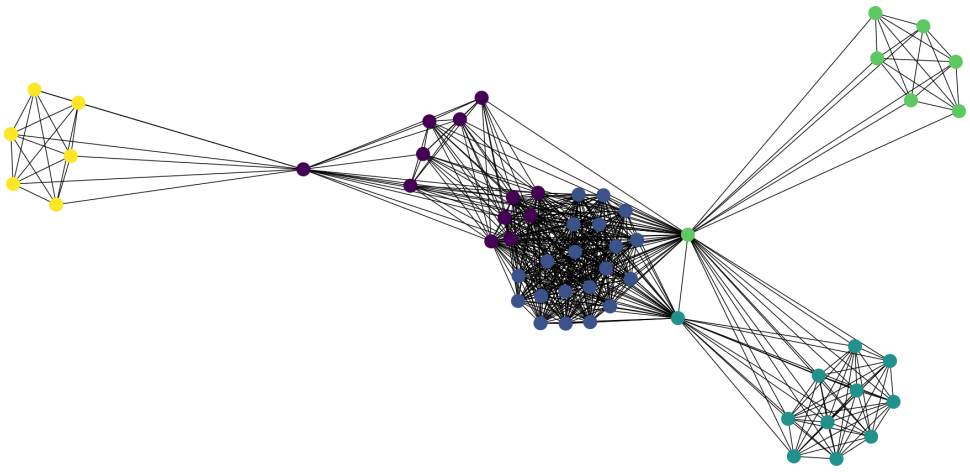


Figura 2 – Divisão de comunidades no NetworkX - Método de Louvain

A figura 2 ilustra, para uma amostra pequena, uma ideia da divisão de comunidades no NetworkX. Pela característic ada modelagem da rede, espera-se esse comportamento onde hajam vários pequenos grupos conectados (correspondentes aos elencos individualmente) e alguns vértices como é possível observar na imagem que fazem essas pontes

dentre os aglomerados. Em uma visão geral, devido ao grande número de vértices e arestas, se torna difícil fazer qualquer inferência ou dar algum significado, como mostra a figura 3, gerada com o método de modularidade do Gephi, atribuindo cores diferentes para cada comunidade.

Figura 3 – Visão geral das comunidades utilizando partição de modularidade no Gephi

3.3 Rankeamento e centralidade vértices

Do ponto de vista individual das entidades ou vértices da rede, com frequência é pertinente à aplicação estabelecer algum critério para classificar os vértices mais importantes. Essa importância denotada não é um conceito rígido e final, mas que flexivelmente depende do contexto da aplicação e passível de interpretação. Em cada cenário possível (estratégias de marketing, no estudo de disseminação de doenças, dentre outros exemplos) a escolha de uma métrica ou critério pode tomar uma conotação diferente. Para ilustrar essas possibilidades, 4 métricas são aplicadas para obter os vértices mais bem classificados segundo esses critérios.

O primeiro deles, o grau de entrada refere-se de forma direta ao número de conexões (ou, no contexto da base, quem co-atuou com mais atores diferentes). A centralidade de proximidade (*closeness*) é uma forma de detectar nós que são capazes de espalhar

informações de forma eficiente e mede sua distância média para todos os outros nós. A centralidade por intermediação (*betweenness*) refere-se ao número de menores caminhos de todos os vértices para quaisquer outros vértices que passam por aquele nó. Por fim, a centralidade por auto-vetor é baseada na noção de que a influência de um vértice é dada não de forma individual, mas existe uma influência mútua, onde um vértice que faz interação com um vértice de maior influência, é potencialmente um vértice com maior influência (ideia base do algoritmo de *pagerank*). A tabela 1 apresenta os atores melhor classificados em cada critério, juntamente com o respectivo índice.

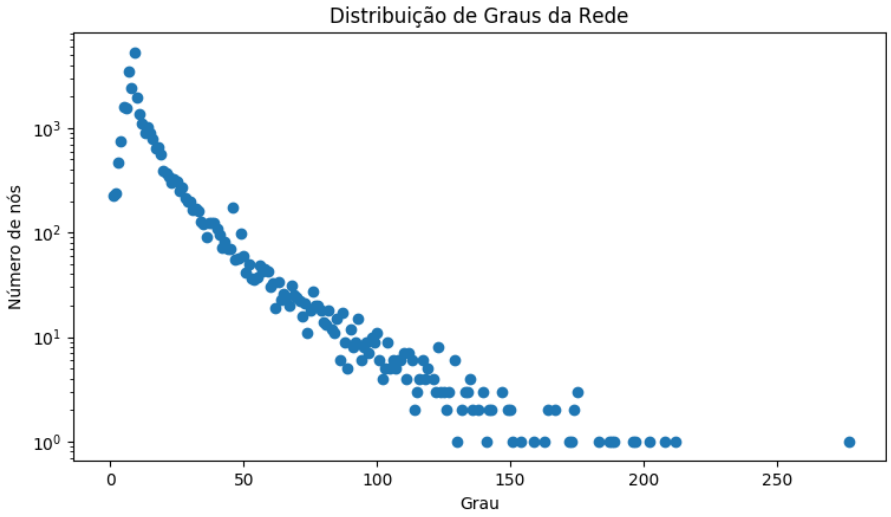
Tabela 1 – Rankeamento e centralidade dos vértices

Maior Grau	**Betweenness**	**Closeness**	**Auto-vetor**
Anupam Kher: 277	Anupam Kher: 0.059	Gerard Butler: 0.267	Takahiro Sakurai: 0.16
Shah Rukh Khan: 212	Om Puri: 0.03	Alfred Molina: 0.265	Yuichi Nakamura: 0.15
Takahiro Sakurai: 208	Sahajak Boonthanakit: 0.028	Ben Kingsley: 0.264	Yuki Kaji: 0.15
Yuki Kaji: 202	Iko Uwais: 0.026	Chloe Grace Moretz: 0.263	Jun Fukuyama: 0.14
Fred Tatasciore: 197	Ben Kingsley: 0.025	Helen Mirren: 0.262	Junichi Suwabe: 0.13
Yuichi Nakamura: 196	Cesar Montano: 0.025	James Franco: 0.262	Katsuyuki Konishi: 0.13
Fred Armisen: 189	Steven Yeun: 0.025	Samuel L. Jackson: 0.261	Kana Hanazawa: 0.12
Akshay Kumar: 188	Kari Wahlgren: 0.022	Jacki Weaver: 0.261	Eri Kitamura: 0.12
Om Puri: 187	Haluk Bilginer: 0.021	Lena Headey: 0.261	Daisuke Ono: 0.12
Boman Irani - 183	Christopher Lee - 0.021	Willem Dafoe - 0.260	Hiroshi Kamiya - 0.12

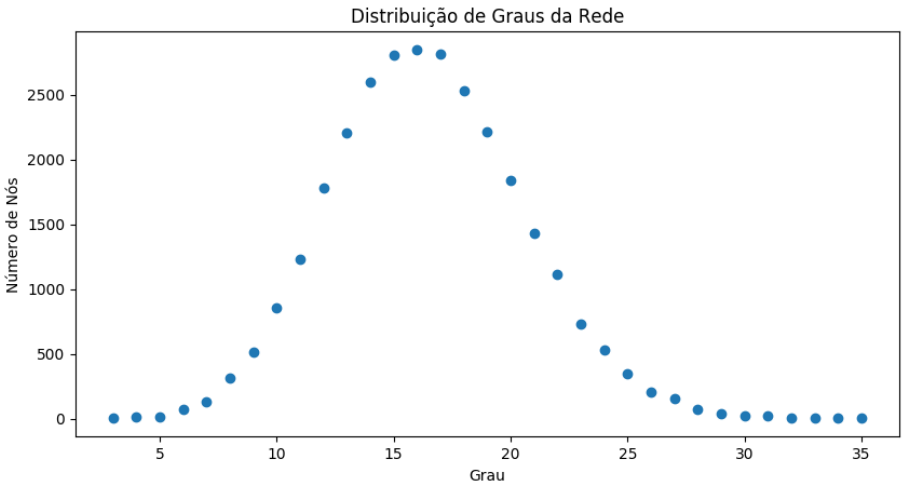
3.4 Comparação com o processo de geração aleatório (Erdõs-Rényi)

Para a etapa de inspeção e verificação com relação ao quanto a formação da rede se aproxima de um processo aleatório, gerou-se um modelo de Erdős–Rényi utilizando os parâmetros da componente principal para quantidade de nós e densidade. A rede obtida, como esperado, tem 29.440 nós e um número bem próximo de arestas (241.853, que anteriomente eram 241.744). O grafo gerado possui um único componente conectado, que por definição é também a componente gigante.

Os coeficientes de aglomeração/triangulação, por sua vez, são significativamente menores que a rede original (o que é esperado, já que não segue a definição do modelo que gera os cliques): 0.000587 e 0.000582, respectivamente. A diferença na formação das redes fica evidente ao observar a distribuição de graus dos nós na figura 4. Enquanto a rede em estudo segue uma lei de potência em sua distribuição (característica de redes do mundo real), a rede gerada aleatoriamente segue uma distribuição normal (característica de redes geradas aleatoriamente).



(a) Rede em estudo (Netflix)



(b) Rede de Erdős–Rényi

Figura 4 – Distribuição de graus

4 Predição de sucesso utilizando a nota de avaliação do IMDB

A hipótese a ser analisada no processo pode ser descrita como: "para cada nó, é possível predizer sua nota com base nas notas dos vizinhos (ou seja, aqueles que atuaram com ele em algum momento)?". Em um teste inicial de viabilidade, verifica-se uma alta correlação entre a nota do ator e a média dos vizinhos utilizando a correlação de Pearson. O cálculo apontou uma correlação de 0,96 entre o conjunto de valores de notas dos vértices e a média das notas dos seus vizinhos imediatos. Essa alta correlação é consequência também da formação dos cliques, já que o vértice possui muitos vizinhos imediatos com uma nota próxima (ainda ponderado pela quantidade de elementos da vizinhança).

Utilizando a média como forma direta de predição, e comparando com a nota efetiva, é possível obter uma estimativa da acurácia. Neste caso, como forma de avaliação, é calculado o erro quadrático entre o valor predito e o real, e para toda a rede, obtém-se o erro médio. Neste caso, o erro médio obtido foi de 0,25 em notas que variam de 0 a 10. Ainda que o erro seja baixo, uma opção de modelo para predição é utilizar os conjuntos de valores para definir uma função linear que possa predizer melhor a nota com base na média.

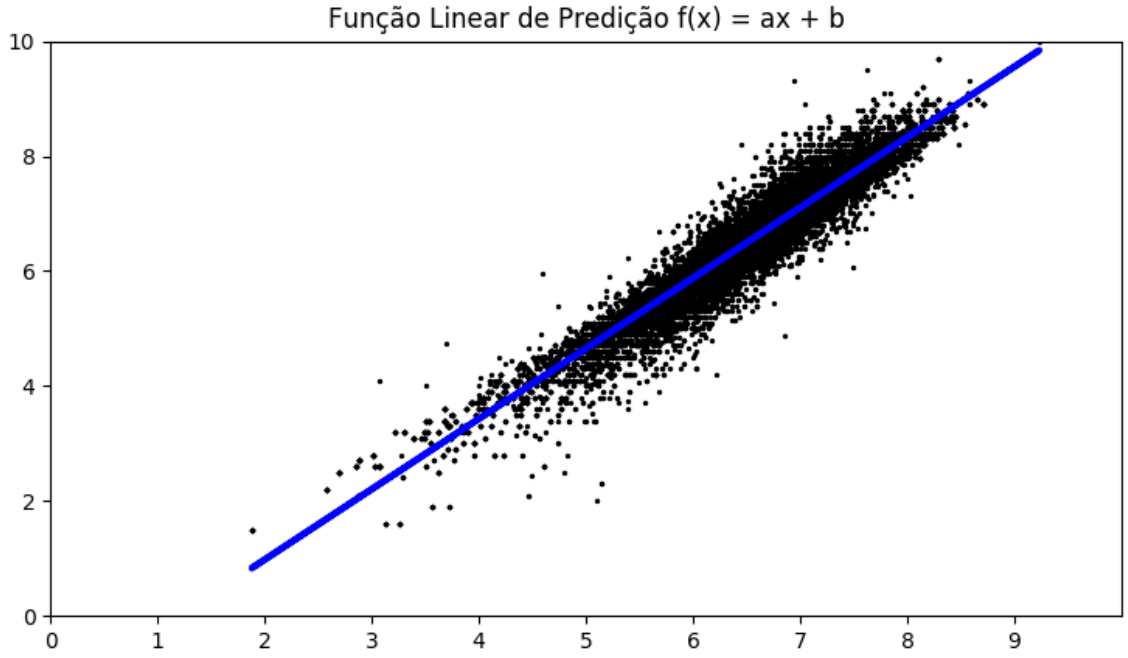


Figura 5 – Função linear para melhor aproximação do conjunto de dados

A figura 5 ilustra a dispersão dos valores e a reta que melhor estima os valores. Utilizando regressão linear, a função linear de variável simples que descreve essa reta é

dada por f(x) = 1.22x - 1.47. Utilizando essa função para fazer a predição, o erro médio cai para 0,21(...). Indo um pouco além, outra hipótese é considerar um modelo multi-variável que considere as médias de vizinhos a distâncias N do vértice. Por exemplo, para N=2, a média dos vizinhos dos vizinhos também são consideradas para tentar ajustar melhor o modelo. Efetuando o primeiro teste para modelo multi-variável com N=2, obteve-se a função $f(x) = 1.29x_1 + - 0.26x_2 - 0.19$ (onde x_1 é a média dos vizinhos diretos, enquanto x_2 é a média dos vizinhos dos vizinhos). No entando, ao efetuar o teste de predição, o erro médio obtido foi da mesma ordem de 0.21(...), não justificando seu custo computacional.

Utilizando então a função linear obtida para predição, é possível tomar o seguinte cenário hipotético para ilustrar: um ator X, não conhecido previamente, mas que se sabe que atuou com Will Smith, Adam Sandler, Mila Kunis, Keanu Reeves, Rodrigo Santoro e Carrie Fisher (ilustrado na figura 6. O valor predito da sua média é obtido substituindo a variável na função pela média desses vizinhos. Esse valor seria dado então por 1,22 * 6,31 - 1,47 = 6,23.

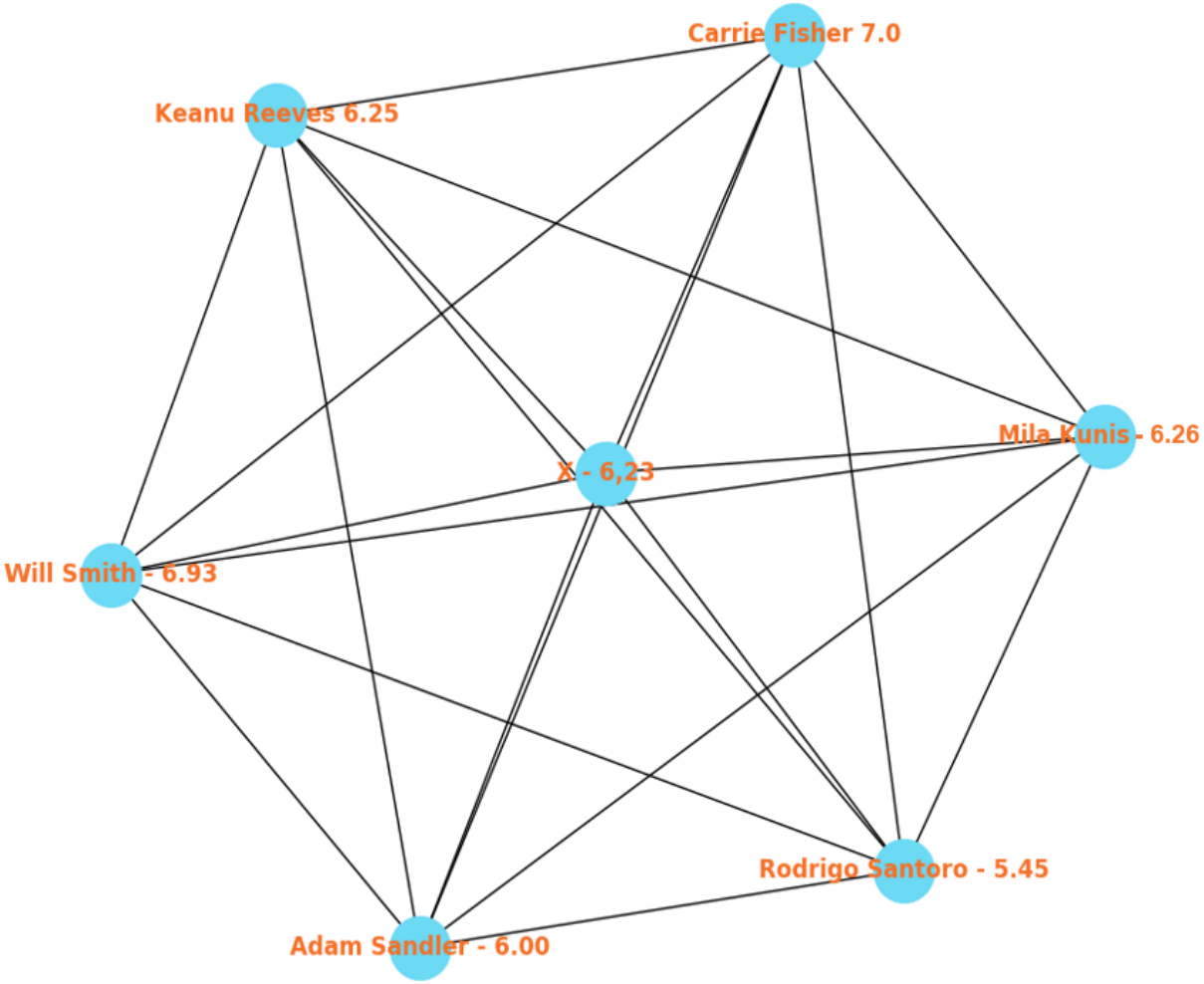


Figura 6 – Cenário utilizando na predição

Conclusão

O Tim Sort, como algoritmo híbrido, tira vantagem do fato de que, em aplicações no mundo real, frequentemente são encontrados padrões nos quais sub-conjuntos de uma lista já estão ordenados. Se beneficiando dessa característica e fazendo uso das funções de ordenação do Insertion e Merge Sort para uma estratégia de divisão e conquista, o algoritmo se mostra muito eficiente aos cenários em que é submetido na prática e acabou se tornando o método padrão das linguagens Python e Java SE 7.

Para trabalhos futuros, sugerem-se algumas otimizações que visam melhorar ainda mais o desempenho do Tim Sort, como por exemplo a ordenação por inserção com busca binária e o merge com galopeamento binário. A implementação aqui demonstrada na linguagem python, ainda que menos verbosa e de mais fácil compreensão, é menos eficiente do que uma linguagem de mais baixo nível, como C. Essa transcrição também é sugerida como melhoria em termos de performance.

Por fim, o trabalho aqui proposto é também uma demonstração de análise e comparação de diferentes algoritmos para solução de um mesmo problema. A partir da análise teórica de complexidade das rotinas propostas, a verificação prática se dá para confirmar o comportamento assintótico dessas soluções em diferentes contextos à medida que são sobrecarregados com entradas de diversas escalas. Em um caso de uso real, esse processo vem apoiar a tomada de decisão de acordo com o cenário em questão.

Referências

Insertion Sort Algorithm. Disponível em: <https://www.tutorialspoint.com/data_structures_algorithms/insertion_sort_algorithm.htm>. Acesso em março de 2021.

Merge Sort Algorithm. Disponível em: <https://www.tutorialspoint.com/data_structures_algorithms/merge_sort_algorithm.htm >. Acesso em março de 2021.

Tim Sort Algorithm. Disponível em: <https://www.geeksforgeeks.org/timsort/>. Acesso em março de 2021.

An Introduction to Algorithm Complexity Analysis. Disponível em: <https://discrete.gr/complexity/>. Acesso em março de 2021.