Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Marcela Megumi Terakado
Marcos Kazuya Yamazaki

Modelagem e Análise dos Dados do Portal de Revistas da USP

São Paulo
Janeiro de 2017

Modelagem e Análise dos Dados do Portal de Revistas da USP

Monografia Final da Disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisora: Profa. Dra. Kelly Rosa Braghetto

São Paulo
Janeiro de 2017

Resumo

Atualmente, o Portal de Revistas da USP, que é gerenciado pelo Sistema Integrado de Bibliotecas da USP (SIBi-USP), utiliza um modelo de dados relacional para armazenar as informações dos artigos publicados nas revistas da USP. Esse banco de dados contém os dados de artigos, revistas, autores, métricas sobre os artigos, entre outras informações. Entretanto, ainda não existem análises sobre estes dados, como por exemplo, a detecção de comunidades, que possibilita a identificação automática de grupos de autores de uma mesma área de pesquisa. Assim, neste trabalho propõe-se uma modelagem de um banco de dados orientado a grafos com o objetivo de viabilizar análises mais sofisticadas sobre os dados do Portal de Revistas da USP, como a detecção de comunidades, e de disponibilizar visualizações das informações de acesso aos artigos do portal, considerando, inclusive a geolocalização dos acessos.

Palavras-chave: banco de dados orientado a grafos, detecção de comunidades, grafos.

Abstract

Currently, the USP's Journals Portal, which is managed by the Integrated Library System of USP (SIBi-USP), uses a relational data model to store the information of articles published in USP journals. This database contains data on articles, journals, authors, metrics on articles, and other information. However, there are no analyzes of these data, such as community detection, which allows the automatic identification of groups of authors from the same research area. Thus, this work proposes a new modeling using a graph database with the objective of making feasible sophisticated analyzes on the data of USP's Journals Portal, such as of community detection, and facilitating the vizualization of access information to the portal's articles, including the geolocation of accesses.
Keywords: graph database, community detection, graph.

Sumário

Capítulo 1

Introdução

1.1 Contexto e Motivação

As informações sobre os artigos publicados em revistas científicas da Universidade de São Paulo (USP) são mantidas em um sistema web chamado Portal de Revistas, gerenciado pelo Sistema Integrado de Bibliotecas (SIBi) da USP. O Portal de Revistas é implementado sobre o sistema eletrônico de editoração de revistas *Open Journal System* (OJS), um software livre que usa um banco de dados relacional para armazenar dados de artigos, revistas, autores, métricas sobre artigos, entre outras informações.

O modelo relacional do banco de dados do Portal de Revistas da USP é adequado para auxiliar as operações de manipulação de dados, como inclusão, alteração, remoção e consultas simples de dados. Entretanto, essa estrutura dificulta a realização de consultas mais elaboradas sobre os dados das publicações científicas. Uma evidência desta dificuldade é que o portal ainda não disponibiliza aos seus usuários análises sobre os dados contidos em seu bancos.

Desde sua criação nos anos 70, o modelo de dados relacional é o mais utilizado no desenvolvimento de bancos de dados. Porém, ele não é o mais adequado para representar dados nas aplicação em que os relacionamentos entre as entidades são mais importantes que os atributos das mesmas. Diversas aplicações atuais tem essas características, como é o caso das redes sociais e das próprias publicações científicas, com suas relações de coautoria. Para esses tipos de aplicações, a representação e armazenamento de dados segundo o modelo de dados orientado a grafos geralmente é o mais indicado.

Em um banco de dados orientado a grafos, as entidades do domínio de aplicação são representadas como vértices (nós) do grafo, enquanto os relacionamentos entre elas são representados como arestas. Nesse modelo, consultas que buscam por entidades associadas por meio de relacionamentos (de qualquer grau) são executadas muito mais rapidamente do que em um bancos de dados relacional, onde as informações sobre essas entidades estariam espalhadas em diferentes tabelas interligadas por meio de chave estrangeiras.

Segundo Penteado *et al.* (2014), apesar dos bancos de dados orientado a grafos terem surgido nos anos 80, eles só ganharam força recentemente, sendo um dos motivos a expansão das redes sociais. Além disso, a grande quantidade de dados e o interesse nas análises sobre eles também impulsionaram o uso deste modelo de dados.

Em uma rede social, as pessoas são representadas por nós e a relação de amizade entre duas pessoas é representada por uma aresta entre dois nós. Uma das análises que pode ser realizada sobre o grafo de uma rede social é a busca de amigos de amigos, em diferentes níveis de profundidade, a partir de uma dada pessoa. Vukotic *et al.* (2014) realizaram experimentos que mostram que um banco de dados orientado a grafos tem um melhor desempenho que

um banco de dados relacional quando se deseja realizar este tipo de análise.

Uma outra análise que pode ser realizada sobre bancos de dados orientados a grafos é a detecção de comunidades. Segundo Fortunato (2009), "comunidades, também chamadas de *clusters* ou módulos, são grupos de nós que provavelmente compartilham propriedades comuns e/ou desempenham funções similares no grafo"[1].

O objetivo deste tipo de análise é identificar grupos de entidades a partir da topologia do grafo na qual estão inseridas. No caso dos dados do Portal de Revistas da USP, é possível obter os grupos de autores de uma dada área de pesquisa a partir do grafo de coautoria.

Como as técnicas de detecção de comunidades são definidas sobre grafos, armazenar estes dados em um banco de dados orientado a grafos facilita a realização de tais análises, visto que os algoritmos podem ser aplicados diretamente.

1.2 Objetivos

Neste trabalho, propõe-se um modelo de banco de dados orientado a grafos para os dados contidos no banco de dados relacional do Portal de Revistas da USP. Esse novo modelo visa atender dois objetivos principais:

- viabilizar análises mais sofisticadas sobre os dados do portal, como a de detecção de comunidades, capaz de identificar as áreas de pesquisa dos autores e possíveis redes de colaboração entre eles;
- facilitar a organização e a visualização das informações de acesso aos artigos do portal, considerando, inclusive, a geolocalização dos acessos.

Para atingir esses objetivos, primeiramente, foi necessário estudar a estrutura do banco de dados relacional do Portal de Revistas da USP e os dados contidos neste banco.

Os dados extraídos do banco do Portal de Revistas da USP tiveram que ser limpos e padronizados pois muitos continham erros de ortografia, estavam incompletos e/ou não tinham padrão. Os dados extraídos e utilizados foram os seguintes:

- artigos: título, palavras-chave, resumo, DOI, revista em que foi publicado e a data em que foi submetido;
- autores: nome, país a qual o autor pertence, artigos que publicou e nome da instituição a qual o autor é afiliado;
- métricas (foram utilizados apenas as métricas do tipo acesso): tipo de acesso ao artigo, artigo sobre o qual estes acessos estão relacionadas, data do acesso, país e cidade de origem do acesso;
- revistas: título, descrição, palavras-chave e instituição do editor.

Além disso, estudou-se e modelou-se um banco de dados orientado a grafos. Neste banco, foram criados sete tipos de nó: artigo, autor, revista, cidade, país, instituto e universidade.

Para as análises, foram estudados quatro métodos de detecção de comunidades: Girvan-Newman, Louvain, autovalores e autovetores da matriz de modularidade e passeios aleatórios. Além disso, foram construídos quatro grafos a partir do banco modelado para cada um dos experimentos elaborados.

[1] Tradução livre

Em todos os grafos construídos, os nós representavam os autores e as arestas representavam a publicação de um artigo entre dois autores, ou seja, foram utilizados, do banco de dados orientado a grafos, os nós do tipo autor e a relação entre os autores que publicaram algum artigo junto.

Com este trabalho, foi possível realizar análises de detecção de comunidade, o que permitiu verificar que é possível identificar os grupos de autores de uma mesma área de pesquisa quando o grafo tem uma estrutura que facilita este tipo de análise.

Além disso, observou-se que o método de Louvain é o que tem melhor desempenho e que o tempo de execução do método de Girvan-Newman é afetado pela quantidade de arestas no grafo (quanto mais denso o grafo, mais demorado este método é).

Também verificou-se que os gráficos dos acessos a um artigo facilitam a visualização da difusão e do impacto de artigos nos níveis regional, nacional e internacional.

As etapas realizadas neste trabalho estão descritas na Figura 1.1.

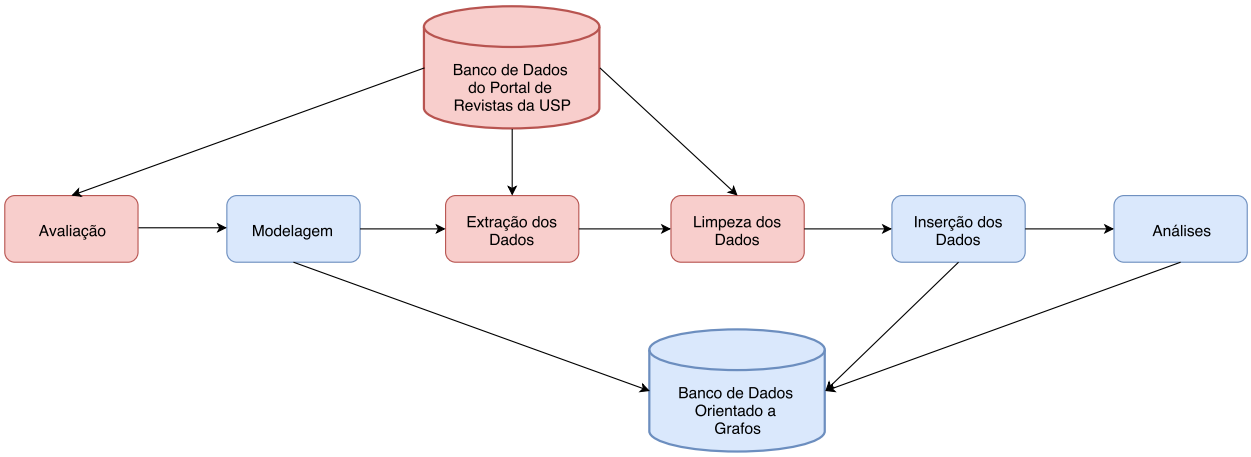


Figura 1.1: *Descrição das etapas realizadas neste trabalho*

1.3 Organização do Trabalho

O Capítulo 2 apresenta toda teoria utilizada neste trabalho para a modelagem do banco de dados orientado a grafos e para as análises de detecção de comunidade. O Capítulo 3 contém as informações sobre o banco de dados do Portal de Revistas da USP. O Capítulo 4, descreve a modelagem do banco de dados orientado a grafos, o tratamento dos dados e a inserção deles no banco orientado a grafos.

O Capítulo 5 contém as informações e resultados das análises realizadas sobre os dados do banco de dados de grafos modelado. E, finalmente, o Capítulo 6 apresenta as considerações finais e os trabalhos futuros relacionados.

Capítulo 2

Conceitos

Este capítulo aborda os conceitos utilizados neste trabalho. São definidas as estruturas chamadas grafos, discute-se brevemente o que é banco de dados e, na sequência, introduz-se o modelo de dados baseado em grafos. Além disso, são apresentados os métodos de detecção de comunidades utilizados e a ferramenta escolhida para as análises e visualizações realizadas neste trabalho.

2.1 Grafos

Grafo é um par ordenado $G = (V, E)$, onde V é um conjunto de vértices (nós) e E é um conjunto de arestas que interconectam os vértices. Um grafo pode ser direcionado ou não-direcionado, ou seja, ter arestas direcionadas ou não.

Neste trabalho, também utilizamos as seguintes definições:

Definição 2.1.1 (Subgrafo)**.** Um *subgrafo* $G' = (V', E')$ do grafo $G = (V, E)$ é um grafo tal que: $V' \subseteq V$, $E' \subseteq E$ e toda aresta em E' tem início e fim em nós de V'.

Definição 2.1.2 (Componente Conexa)**.** Uma *componente conexa* do grafo G é um subgrafo maximal de G tal que existe um caminho entre todos os pares de nós da componente.

Definição 2.1.3 (Ponte)**.** Uma *ponte* é uma aresta do grafo G que quando removida do grafo são geradas duas componentes conexas.

Definição 2.1.4 (Grau de um vértice)**.** O *grau de um vértice* de um grafo é o número de arestas que incidem neste vértice.

2.2 Bancos de Dados

Segundo Elmasri e Navathe (2010): "Um banco de dados é uma coleção de dados relacionados. Por dados, nos referimos aos fatos que podem ser armazenados e que tem um significado implícito."[1]

Um modelo de dados é um tipo de modelo que determina como os dados estarão estruturados dentro de um banco de dados. Alguns exemplos de modelos de dados lógicos (ou seja, de implementação de bancos de dados) são: relacional, orientado a objetos, orientado a documento e orientado a grafos.

[1] Tradução livre.

Um sistema gerenciador de banco de dados ou SGBD, segundo Elmasri e Navathe (2010), "é um sistema de software de uso geral que facilita os processos de definição, construção, manipulação e compartilhamento de um banco de dados entre vários usuários e aplicações"[2].

Outras funções de um SGBD são controle de concorrência, manutenção da integridade do banco de dados e proteção de segurança contra acessos não autorizados e proteção de sistema contra problemas de hardware ou *software*. Como exemplo de SGBDs temos: MySQL, PostgreSQL, Neo4j, MongoDB, entre outros.

O modelo mais utilizado e difundido hoje é o relacional. Nos bancos de dados relacionais, os dados são armazenados em tabelas que contêm linhas e colunas. As linhas das tabelas são chamadas de tuplas e as colunas de atributos. Cada tupla é única, ou seja, cada linha possui uma sequência única de valores para atributos. A linguagem padrão para a interação com bancos de dados relacionais é a SQL (*Structured Query Language*).

2.2.1 Banco de Dados Orientado a Grafos

O modelo de dados orientado a grafos classifica-se como um banco de dados NoSQL (*Not only SQL*). Neste modelo, as entidades são representadas como nós e as relações entre elas como arestas de um grafo.

A principal diferença entre o modelo de dados orientado a grafos e o modelo relacional, é que o foco da informação no primeiro está nos relacionamentos enquanto no segundo está nos atributos. Um breve comparativo pode ser visto na Tabela 2.1.

Modelo de dados	**Estrutura de dados base**	**Foco da informação**
Relacional	Tabela (relações/entidades)	Atributos
Orientado a Grafos	Grafo	Relacionamentos

Tabela 2.1: *Comparação entre os modelos relacional e orientado a grafos apresentada por Angles e Gutiérrez (2008)*

O modelo de dados orientado a grafos mais utilizado contém nós, que têm propriedades e podem ter um ou mais rótulos, e relacionamentos, que possuem nomes, são direcionados (tem nó de início e fim) e podem ter propriedades.

As vantagens de se usar um banco de dados de grafos ao invés de um relacional são: melhor desempenho, ou seja, as consultas não dependem do tamanho do banco, ou seja, o banco pode crescer sem afetar o desempenho; maior flexibilidade, ou seja, é possível adicionar nós, relacionamentos e rótulos sem afetar os dados e as consultas existentes; por fim, maior agilidade no desenvolvimento do banco, pois não são necessárias etapas de abstrações como no modelo relacional (modelo conceitual e lógico).

2.2.1.1 Neo4j

Neste trabalho, utilizamos o SGBD de grafos Neo4j, pois é o SGBD de grafos mais utilizado segundo o site DB-engines[3], está bem documentado e é gratuito.

Neo4j é um banco de dados transacional, logo, atende as propriedades ACID (atomicidade, consistência, isolamento e durabilidade). Para comparar o desempenho de um banco de dados relacional (MySQL) com um banco de dados de grafos (Neo4j), Vukotic *et al.* (2014) realizaram alguns experimentos que consistiam em buscar todos os amigos de amigos de uma

[2]Tradução livre.

[3]http://db-engines.com/en/ranking/graph+dbms

dada pessoa em bancos de dados de redes sociais. Em um dos experimentos, a rede social continha um milhão de pessoas, sendo que cada pessoa tinha, em média, cinquenta amigos.

O resultado desse experimento pode ser visto na Tabela 2.2, que mostra que a consulta utilizando o Neo4j foi mais rápida que a do MySQL. A profundidade, utilizada no experimento, indica a distância máxima considerada entre dois nós pela relação de amizade.

Profundidade	**Tempo de execução (em segundos) MySQL**	**Tempo de execução (em segundos) Neo4j**
2	0,016	0,010
3	30,267	0,168
4	1543,505	1,359
5	Não terminou	2,132

Tabela 2.2: *Tempo de execução da busca de amigos em diferentes profundidades, usando um banco de dados relacional (MySQL) e um banco de dados de grafos (Neo4j) Robinson et al. (2015); Vukotic et al. (2014)*

Este e os demais resultados dos experimentos feitos por Vukotic *et al.* (2014) evidenciam os benefícios, discutidos nesta seção, dos bancos de dados orientados a grafos para aplicações que, como as redes sociais, têm o foco nos relacionamentos.

2.2.1.2 Cypher

Cypher[4] é uma linguagem de consulta declarativa para banco de dados orientados a grafos usada no Neo4j. Algumas das cláusulas que compõem a linguagem podem ser vistas na Tabela 2.3.

Cláusula	**Descrição**
CREATE	Cria nós e relacionamentos
DELETE	Apaga nós, relacionamentos e caminhos
MATCH	Especifica o padrão a ser buscado no grafo
WHERE	Filtra ou restringe uma busca no grafo
RETURN	Devolve uma consulta realizada no banco
SET	Adiciona propriedades e rótulos nos nós
REMOVE	Remove propriedades e rótulos nos nós e relacionamentos
UNION	Combina o resultado de uma ou mais consultas

Tabela 2.3: *Principais cláusulas da linguagem Cypher*

2.3 Detecção de Comunidade

Comunidades, também conhecidas como *clusters*, são conjunto de nós que possuem propriedades similares dentro do grafo.

Detecção de comunidade é o agrupamento de um conjunto de nós de uma rede (grafo) de acordo com a similaridade por algum critério.

Segundo Newman e Girvan (2004), a detecção de comunidade auxilia a compreensão e visualização da estrutura da rede (no nosso caso, a relação de coautoria).

[4]http://neo4j.com/docs/developer-manual/current/cypher/?ref=gdb-book

2.3.1 Girvan-Newman

O trabalho de Newman e Girvan (2004) discute técnicas de *clustering* hierárquico para detecção de comunidade, pois o algoritmo desenvolvido por eles é baseado em uma dessas técnicas.

Segundo os autores, "As técnicas [de *clustering* hierárquico] têm como objetivo encontrar divisões naturais na rede (social) em grupos com base em métricas de similaridade ou força de conexões entre os vértices. Elas são classificadas em duas grandes categorias: de aglomeração e de divisão, dependendo se focam em adicionar ou remover arestas da rede"[5].

Com o resultado desta técnica é possível construir um dendrograma, que é uma estrutura em forma de árvore hierárquica que pode ser vista na Figura 2.1. Nesta figura, cada circunferência representa um nó na rede e, conforme, subimos na estrutura, os nós são agrupados em comunidades. Um 'corte' na horizontal nesta árvore mostra as comunidades daquele nível, a linha pontilhada vermelha é um exemplo de um 'corte' em um nível que fornece 4 comunidades.

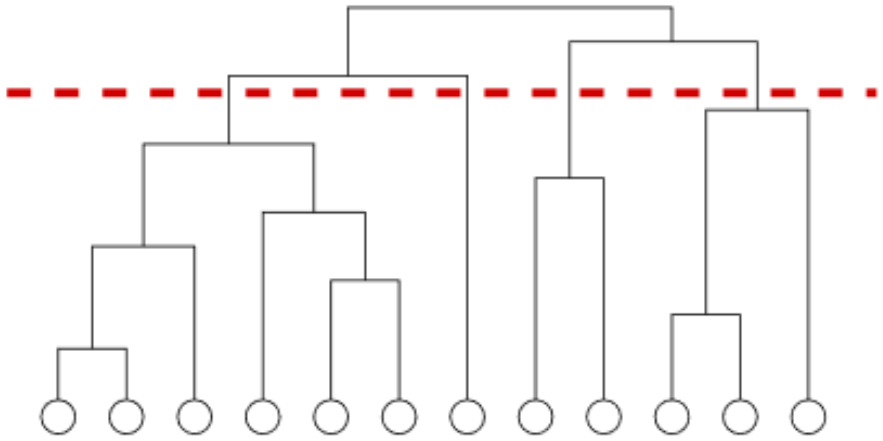


Figura 2.1: *Exemplo de um dendrograma retirado do trabalho de Newman e Girvan (2004)*

Além disso, os autores utilizam a definição de que o método de aglomeração insere arestas com maior valor de similaridade em um grafo vazio de acordo com a similaridade entre pares de vértices. E, em contrapartida, o método de divisão remove as arestas entre os nós com menor valor de similaridade. Os métodos aglomerativos tendem a errar a classificação dos nós mais periféricos pois estes têm baixo valor de similaridade.

O método de Girvan-Newman é baseado no método de divisão mas, ao invés de remover as arestas entre os vértices com menos similaridade, são removidas as arestas com maior valor de *betweenness*.

Segundo os autores "*betweenness* é uma medida que favorece arestas entre as comunidades e desfavorece aquelas que se encontram dentro das comunidades"[6]. Além disso, eles também afirmam que este valor pode ser calculado de diversas formas, entretanto a forma escolhida não tem grande influência sobre o resultado final.

Uma das formas de se calcular o valor de *betweenness* de uma aresta é contando o número de vezes que a aresta faz parte de um caminho mínimo entre dois nós, ou seja, quanto maior este valor, maior a probabilidade desta aresta conectar duas comunidades (ou, em outras

[5]Tradução livre
[6]Tradução livre

palavras, maior a probabilidade desta aresta ser uma ponte no grafo atual). Desta forma, as arestas de uma mesma comunidade possuem um valor baixo de *betweenness*, enquanto as arestas entre as comunidades têm valores altos.

O método de Girvan-Newman é composto por duas etapas, a primeira consiste na geração do dendrograma utilizando o caminho mínimo para o cálculo do valor de *betweenness* para cada aresta e, a segunda, consiste na escolha do particionamento do dendrograma que otimiza a modularidade, uma métrica que indica o quão bem estruturado é um grafo em termos dos *clusters*.

A fórmula de modularidade Q é basicamente:

$Q =$ (número de arestas internas de uma comunidade) $-$ (número esperado destas arestas)

A fórmula de modularidade utilizada nos trabalhos de Newman e Girvan (2004) e de Pons e Latapy (2005) é:

$$Q = \sum_i (e_{ii} - a_i^2) = \text{Tr } e - ||e^2||$$

Considerando a divisão do grafo em k comunidades, e é uma matriz kxk,

$$\text{Tr } e = \sum_i e_{ii}$$

$$a_i = \sum_j e_{ij}$$

$$e_{ij} = \frac{\text{número de arestas entre as comunidades i e j}}{\text{número total de arestas}}$$

e $||x||$ representa a soma dos elementos da matriz x.

Nos trabalhos de Newman (2006) e de Blondel *et al.* (2008), utiliza-se a seguinte fórmula de modularidade:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

onde A_{ij} representa o peso da aresta entre os nós i e j (A é chamada de matriz de adjacência), c_i é a comunidade à qual o nó i foi atribuído,

$$k_i = \sum_j A_{ij}$$

é o grau do nó i,

$$m = \frac{1}{2} \sum_{ij} A_{ij}$$

é o número total de arestas no grafo e

$$\delta(c_i, c_j) = \begin{cases} 1 & \text{se } i = j, \\ 0 & \text{caso contrário.} \end{cases}$$

Segundo Newman e Girvan (2004), o valor da modularidade geralmente varia entre $0,3$ e $0,7$, sendo raros os valores maiores que $0,7$. Sendo que, quanto mais próximo de 1, melhor

o particionamento dos nós em comunidades.

Como descrito por Newman e Girvan (2004), os passos do algoritmo da geração do dendrograma são:

1. para cada aresta do grafo é calculado seu valor de *betweenness*
2. é removida a aresta com o maior valor de *betweenness*
3. calcula-se novamente o valor de *betweenness* das arestas que sobraram no grafo
4. repetir a partir do passo 2 até que não hajam mais arestas a serem removidas

Em seguida, para cada nível do dendrograma é feito o cálculo da modularidade e é escolhido o nível com maior valor de modularidade. Os *clusters* do nível escolhido correspondem às comunidades.

O problema deste método é que ele é caro computacionalmente. Para um grafo com n nós e m arestas sua complexidade computacional é $O(nm^2)$.

2.3.2 Louvain

No trabalho de Blondel *et al.* (2008), foi criado um algoritmo guloso que tem como objetivo detectar comunidades em uma rede otimizando o valor da modularidade. Nesse trabalho, os autores consideram uma rede com N nós e m arestas com peso. Segundo eles, o algoritmo é realizado em dois passos, como pode ser visto na Figura 2.2 (retirada do mesmo trabalho).

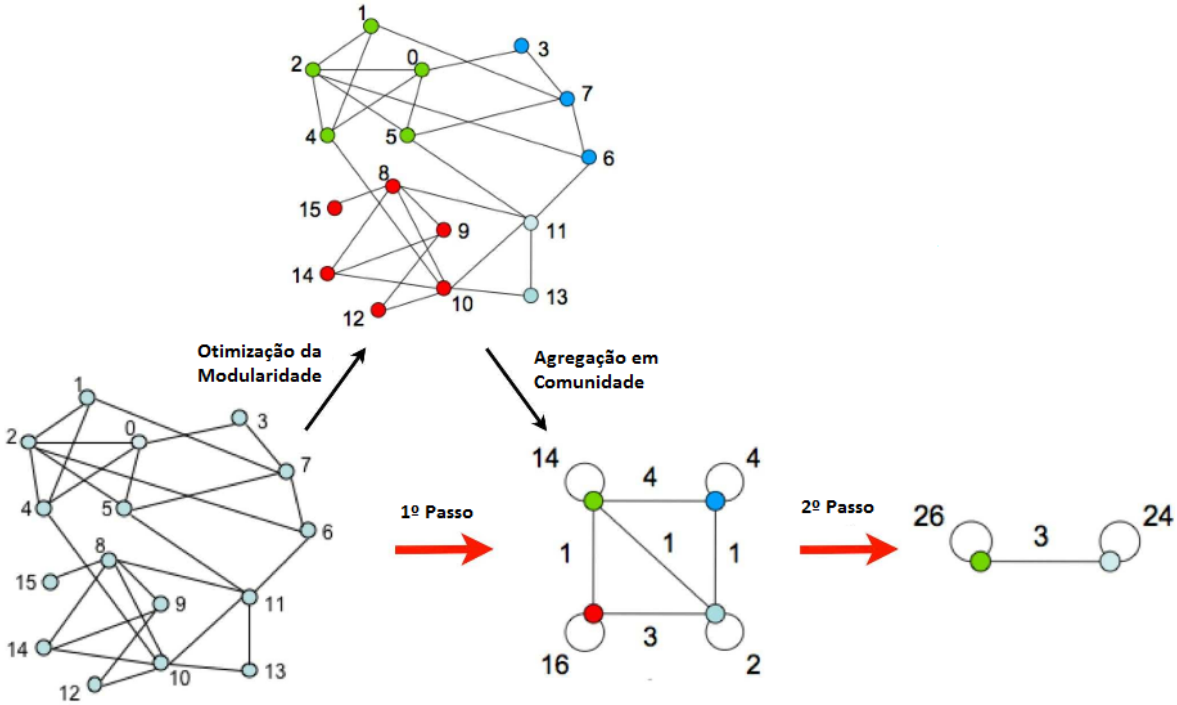


Figura 2.2: *Passos do algoritmo Louvain, imagem retirada do trabalho de Blondel et al. (2008)*

Inicialmente, cada nó tem sua própria comunidade, ou seja, existem N comunidades. Em seguida, para cada nó i calcula-se o valor da modularidade entre i e seus vizinhos j. O nó i mudará para comunidade de j cujo valor de modularidade é o maior positivo. Se nenhum

valor for positivo, o nó i continuará na sua comunidade. Este processo é aplicado iterativamente e sequencialmente até que não seja possível melhorar mais, ou seja, até atingir um máximo local em que nenhuma mudança de comunidade melhora o valor da modularidade.

A segunda parte é construir uma rede em que os nós são as comunidades detectadas no passo anterior. Os pesos das arestas entre os nós atuais são a soma do peso das arestas entre os nós de cada comunidade. As arestas dos nós de uma mesma comunidade são representadas como uma aresta para esta mesma comunidade. Então, a primeira fase é repetida até que não haja mudança no valor da modularidade. Assim, na segunda parte, é encontrado um máximo global.

Nesse mesmo trabalho, os autores aplicaram seu algoritmo, descrito acima, sobre algumas redes de tamanhos variados e comparam o tempo de execução e o resultado com os de outros algoritmos. Nesses testes, o algoritmo de Louvain teve um desempenho melhor que os demais.

O único problema do algoritmo é a falha em detectar comunidades com poucos nós.

2.3.3 Autovalores e Autovetores da Matriz de Modularidade

O método proposto por Newman (2006) para detecção de comunidades também tem por objetivo maximizar a medida de modularidade, mas, diferentemente dos métodos anteriores, ele utiliza técnicas baseadas em autovalores e autovetores sobre a matriz de modularidade.

Inicialmente, o autor descreve todo o método para o caso em que o número de comunidades é dois. No final, ele explica que, quando o número de comunidades é maior que dois, basta repetir o método em cada comunidade detectada até que o valor de modularidade da próxima iteração seja menor do que o da iteração atual.

Nesse trabalho, o grafo é representado por sua matriz de adjacência A, em que:

$$A_{ij} = \begin{cases} 1 & \text{se existe uma aresta entre os nós } i \text{ e } j, \\ 0 & \text{caso contrário.} \end{cases}$$

Além disto, é definida a matriz de modularidade como sendo

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

em que $k_i = \sum_j A_{ij}$ é o grau do nó i e $m = \frac{1}{2}\sum_{ij} A_{ij}$ é o número total de arestas no grafo.

A ideia desse método é calcular os autovalores e autovetores da matriz de modularidade até que algum autovalor seja negativo. Quando isto ocorre, o maior valor de modularidade é encontrado.

Como esse algoritmo utiliza o cálculo de autovetor da matriz, ele não funciona em alguns casos particulares, por exemplo, quando o grafo não tem arestas.

2.3.4 Passeios Aleatórios

O trabalho de Pons e Latapy (2005) propõe um algoritmo, denominado *Walktrap*, baseado em passeios aleatórios. Intuitivamente, um passeio aleatório é um conjunto de nós visitados a partir da escolha aleatória e uniforme de um nó vizinho a partir do nó atual. Segundo os autores desse artigo, "a sequência dos nós visitados é uma cadeia de Markov, em que os estados são os nós do grafo"[7].

[7] Tradução livre

A cada passo, a probabilidade de transição do nó i para o nó j é $P_{ij} = \frac{A_{ij}}{d(i)}$, onde A_{ij} é o peso da aresta entre os nós i e j (caso não exista aresta entre estes dois nós, $A_{ij} = 0$) e $d(i) = \sum_j A_{ij}$, ou seja, $d(i)$ é o grau do nó i.

A probabilidade de ir de um nó i para um nó j através de um passeio aleatório de tamanho t é denotado por P_{ij}^t e atende as seguintes propriedades:

Propriedade 2.3.1. *Num passeio aleatório qualquer, a probabilidade de se estar no nó j só depende do grau deste nó.*

Propriedade 2.3.2. *Dados dois nós quaisquer do grafo, a chance de um passeio aleatório ir de um nó para o outro depende apenas do grau de cada um.*

Os autores de Pons e Latapy (2005) utilizam as seguintes observações que permitem comparar dois nós a partir de P^t:

- Se dois vértices i e j estão na mesma comunidade, a probabilidade P_{ij}^t será certamente alta. Mas o fato que P_{ij}^t é alta não implica necessariamente que i e j estão na mesma comunidade.
- A probabilidade P_{ij}^t é influenciada pelo grau $d(j)$ porque o andarilho (entidade abstrata que caminha pelo grafo e gera o passeio aleatório) tem alta probabilidade de ir para nós com alto grau.
- Dois nós da mesma comunidade tendem a ver todos os outros nós da mesma maneira. Então, se i e j estão na mesma comunidade, eles provavelmente terão $\forall k$, $P_{ik}^t \simeq P_{jk}^t$. [8]

Nesse artigo, também é definido o conceito de distância entre dois nós para medir a similaridade entre eles. Quanto maior a distância entre dois nós, menor a similaridade entre eles, ou seja, eles provavelmente pertencem a comunidades diferentes. O cálculo da distância r entre os nós i e j é dado pela fórmula a seguir:

$$r_{ij} = \sqrt{\sum_{k=1}^{n} \frac{(P_{ik}^t - P_{jk}^t)^2}{d(k)}}$$

Inicialmente, o algoritmo *Walktrap*, descrito no trabalho de Pons e Latapy (2005), considera que cada nó está em uma comunidade distinta. Em seguida, todas as distâncias dos nós adjacentes são calculadas e, então, os seguintes passos são realizados iterativamente:

- Escolhem-se duas comunidades de acordo com o método de Ward.
- Substituem-se estas duas comunidades pela união delas.
- Atualizam-se as distâncias entre as comunidades.

Após $(n-1)$ iterações, o algoritmo termina e um dendrograma é gerado. Finalmente, basta escolher o ponto cujo valor da modularidade é o maior.

Considerando um grafo com n nós e m arestas, a complexidade computacional desse algoritmo é, no pior caso, $O(mn^2)$ e, no caso médio, $O(n^2 \log n)$.

[8] Tradução livre

2.4 Ferramenta para Análise e Visualização

Neste trabalho, foi utilizada a linguagem de programação R[9] para realizar as análises de detecção de comunidade, as visualizações destas análises e as visualizações dos acessos aos artigos por região geográfica. Essa linguagem foi escolhida pelos seguintes motivos: riqueza da linguagem para geração de gráficos, sua facilidade de integração com o Neo4j (SGBD orientado a grafos usado no trabalho) e a existência de uma biblioteca chamada *igraph*, com algoritmos de detecção de comunidade implementados.

2.4.1 R

R é uma linguagem de programação de código aberto e de alto nível. É voltada principalmente para análises estatísticas, geração de gráficos e relatórios.

2.4.1.1 Bliblioteca s do R

Neste trabalho, foram utilizadas seis bibliotecas do R:

- RNeo4j[10]

  Esta biblioteca é basicamente uma API que conecta o ambiente R com o SGBD de grafos, permitindo interação rápida e fácil com o Neo4j. Com ela, é possível realizar consultas usando a linguagem Cypher diretamente do ambiente R.

- igraph[11]

  Esta biblioteca contém os algoritmos de detecção de comunidade utilizados neste trabalho. As funções para os algoritmos utilizados foram:

  - Girvan-Newman: cluster_edge_betweenness()
  - Louvain: cluster_louvain()
  - Autovalores e autovetores da matriz de modularidade: cluster_leading_eigen()
  - Passeios aleatórios: cluster_walktrap()

- visNetwork[12]

  Esta biblioteca permite a visualização dos grafos.

- ggplot2[13]

  Esta biblioteca foi utilizada para geração dos gráficos.

- plotly[14]

  Esta biblioteca cria versões interativas na *web* de gráficos gerados pela *ggplot2*.

- microbenchmark[15]

  Esta biblioteca possibilita a medição do tempo de execução dos algoritmos de detecção de comunidade com maior precisão.

[9] https://www.r-project.org/
[10] https://cran.r-project.org/web/packages/RNeo4j/
[11] http://igraph.org/
[12] https://cran.r-project.org/web/packages/visNetwork/
[13] https://cran.r-project.org/web/packages/ggplot2/
[14] https://cran.r-project.org/web/packages/plotly/
[15] https://cran.r-project.org/web/packages/microbenchmark/

Capítulo 3

Portal de Revistas da Universidade de São Paulo

Este capítulo apresenta o Portal de Revistas da USP, a estrutura de seu banco de dados relacional e os dados selecionados para a utilização no modelo de dados proposto neste trabalho.

3.1 Sobre

O Portal de Revistas da USP é um sistema de gerenciamento de publicações científicas que é mantido pelo órgão chamado Sistema Integrado de Bibliotecas da Universidade de São Paulo (SIBi-USP).

Ele é implementado sobre um sistema eletrônico de editoração de revistas, o *Open Journal System (OJS)*, que é um sistema de código aberto e tem como objetivo a expansão dos acessos às pesquisas científicas.

3.2 Banco de Dados do Portal de Revistas

O banco de dados do Portal de Revistas da USP é relacional e está mantido em um SGBD PostgreSQL[1].

Como não há documentação sobre os dados e sua organização dentro do banco do Portal de Revistas, pois ele é criado e mantido por meio da ferramenta OJS, foi necessário fazer uma análise de todas as 127 tabelas do banco para encontrar as informações necessárias para este trabalho.

A partir desta análise, foi construído um diagrama, que pode ser visto na Figura 3.1, com as tabelas que foram utilizadas. Mais informações sobre estas tabelas podem ser vistas no Apêndice A.

Neste banco de dados, os artigos são referenciados como *article* ou *submission*, as revistas como *journal* ou *context* e os autores como *author*.

[1] https://www.postgresql.org/

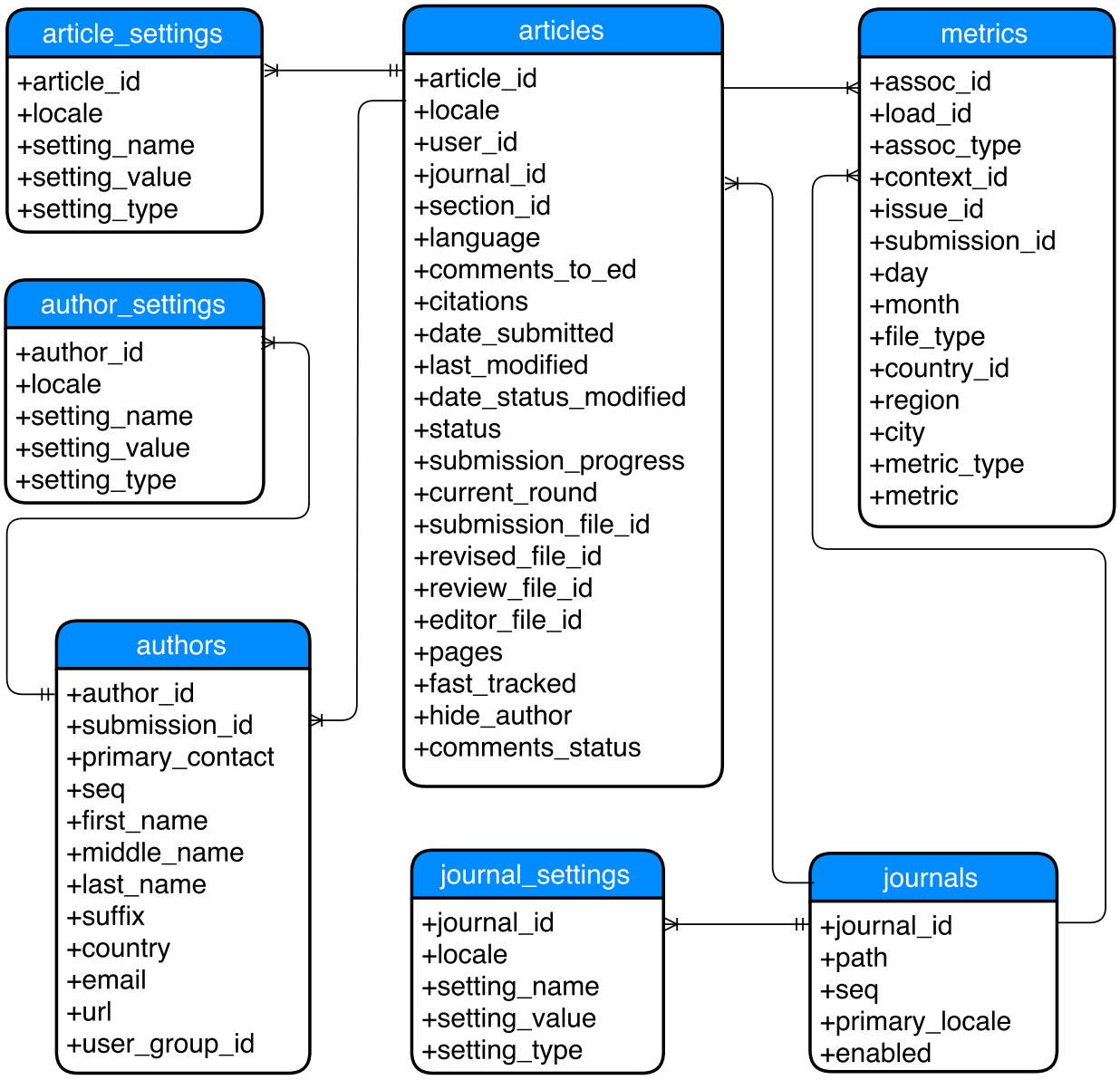


Figura 3.1: *Tabelas do banco de dados do Portal de Revistas que foram utilizadas neste trabalho*

3.2.1 Análise do Banco de Dados

Mesmo sendo um banco de dados relacional, o banco de dados do Portal de Revistas da USP não segue o modelo tradicional. Como é possível ver na Figura 3.2, as tabelas contêm pares de colunas com a seguinte característica: uma das colunas, *setting_name*, é usada para caracterizar o tipo de dado do valor em outra coluna, *setting_value*. Ou seja, a coluna *setting_value* é utilizada para armazenar os valores para os múltiplos tipos de *setting_name*. No modelo relacional usual, cada coluna tem valores de apenas um tipo de atributo e não haveria necessidade de uma coluna auxiliar para realizar esta diferenciação.

article_id	locale	setting_name	setting_value	setting_type
1	pt_BR	title	Groups generated by 3-state automata over a 2-letter alphabet, I	string
1	pt_BR	cleanTitle	Groups generated by 3state automata over a 2letter alphabet I	string
1	pt_BR	abstract	<U+FEFF>Groups generated by 3-state automata over a 2-letter alphabet, I	string
1	pt_BR	sponsor		string
2	pt_BR	title	Self-similar groups and their geometry	string
2	pt_BR	cleanTitle	Selfsimilar groups and their geometry	string
2	pt_BR	abstract	Self-similar groups and their geometry	string
2	pt_BR	sponsor		string
3	pt_BR	title	Classification of algebras with minimal quadratic growth of identities	string
3	pt_BR	cleanTitle	Classification of algebras with minimal quadratic growth of identities	string

Figura 3.2: *Resultado de uma consulta na tabela article_settings do Portal de Revistas da USP*

As tabelas *author_settings*, *article_settings* e *journal_settings* são tabelas verticais, ou seja, o atributo *setting_name*, de todas estas tabelas, assume alguns valores já definidos. Existem 5, 29 e 228 valores diferentes de *setting_name* nas tabelas *author_settings*, *article_settings* e *journal_settings*, respectivamente.

Encontraram-se dois grandes problemas no banco de dados do Portal de Revistas da USP: dados incompletos e sem padronização e chaves estrangeiras não definidas.

Em relação aos dados, foi necessário limpá-los e normalizá-los antes de inseri-los no novo banco de dados orientado a grafos, mais detalhes serão apresentados no próximo capítulo.

E, quanto às chaves estrangeiras das tabelas, foi necessário investigar de quais tabelas os atributos eram chaves estrangeiras. Abaixo temos as chaves estrangeiras identificadas:

- *article_id* da tabela *article_settings* é chave estrangeira de *article_id* da tabela *articles*
- *submission_id* da tabela *authors* é chave estrangeira de *article_id* da tabela *articles*
- *assoc_id* da tabela *metrics* é chave estrangeira de *article_id* da tabela *articles*
- *author_id* da tabela *author_settings* é chave estrangeira de *author_id* da tabela authors *authors*
- *journal_id* da tabela *journal_settings* é chave estrangeira de *journal_id* da *journals*
- *journal_id* da tabela *articles* é chave estrangeira de *journal_id* da tabela *journals*
- *context_id* da tabela *metrics* é chave estrangeira de *journal_id* da tabela *journals*

Sem a declaração das chaves estrangeiras, quando alguma tupla de uma tabela que possui um atributo que é chave estrangeira de outra tabela é removida ou alterada, a mudança só afetará a primeira tabela. Por exemplo, se um artigo for removido da tabela *articles*, as informações deste artigo não serão removidos das tabelas *article_settings*, *authors* e *metrics*.

Além destes problemas, um outro encontrado foi em relação aos identificadores dos autores. Estes identificadores são representados pelo atributo *author_id*, que é a chave primária da tabela *authors*. Um autor, deveria ter apenas um identificador, porém, como nota-se na Figura 3.3, isto não ocorre.

author_id	first_name	middle_name	last_name
155043	Maria	Eliete de	Queiroz
190829	Maria	Eliete de	Queiroz
204582	Maria	Isaura Pereira de	Queiroz
26773	Maria	Isaura Pereira de	Queiroz
66022	Maria	Isaura Pereira de	Queiroz
70046	Maria	Isaura Pereira de	Queiroz
121038	Maria	Veraci Oliveira	Queiroz
103550	Maria	Veraci Oliveira	Queiroz
107295	Maria	Veraci Oliveira	Queiroz
107361	Maria	Veraci Oliveira	Queiroz
5517	Maria	Veraci Oliveira	Queiroz
186766	Maria	Veraci Oliveira	Queiroz
245413	Maria	Veraci Oliveira	Queiroz

Figura 3.3: *Resultado de uma consulta na tabela authors do Portal de Revistas da USP*

Como um autor pode ter mais de um identificador na tabela *authors* e não há padronização dos dados, o valor para o atributo *middle_ name* de um autor pode aparecer escrito de formas diferentes nos vários registros que podem corresponder ao autor. Isto pode ser visto na Figura 3.4.

author_id	first_name	middle_name	last_name
61110	Afonso	D. C.	Passos
80229	Afonso	D.C.	PASSOS
64681	Afonso	DC	Passos
62253	Afonso	Dinis	Costa Passos
59716	Afonso	Dinis	Costa- Passos
60360	Afonso	Dinis	Costa-Passos
87839	Afonso	Dinis Costa	Passos
1194	Afonso	Dinis Costa	Passos
58634	Afonso	Dinis Costa	Passos
1007	Afonso	Dinis Costa	Passos
112301	Afonso	Dinis Costa	Passos

Figura 3.4: *Outro resultado de uma consulta na tabela authors do Portal de Revistas*

Estes problemas, de um autor não ter um identificador único e os nomes estarem escritos de formas diferentes para um mesmo autor, afetaram as análises, pois um mesmo autor pode aparecer no banco de dados orientado a grafos mais de uma vez, mas como nós distintos. Então, se um autor publicar em duas áreas diferentes e seu nome estiver escrito de duas formas diferentes, este autor estaria em duas comunidades ao mesmo tempo e não seria possível detectar a informação de que o autor é uma ponte entre duas áreas de pesquisa.

3.3 Extração dos Dados

Os códigos em SQL utilizados para extrair os dados do Portal de Revistas da USP estão no Apêndice B.

3.3.1 Artigos

As tabelas *articles* e *article_settings* contêm as informações de 93.010 artigos. Da tabela *articles*, os seguintes atributos foram utilizados:

- *article_id*: identificador do artigo
- *journal_id*: identificador da revista em que o artigo foi publicado
- *data_submitted*: data em que o artigo foi submetido

A Tabela 3.1 apresenta os valores utilizados de *setting_name* da tabela *article_settings*. Selecionaram-se apenas as palavras-chave e os resumos em português, ou seja, quando o valor do atributo *locale* = 'pt_BR'. Além disso, selecionaram-se os títulos em português e em inglês, pois nem todos os artigos têm o título em português.

Valor do *setting_name*	**Descrição**
cleanTitle	título do artigo
subject	palavras-chave do artigo
abstract	resumo do artigo
pub-id::doi	DOI do artigo

Tabela 3.1: *Valores utilizados de setting_name da tabela article_settings*

Para o desenvolvimento deste trabalho, foi considerado que o valor da variável *setting_value* para o atributo *locale* estava correto, independentemente da linguagem em que o texto foi escrito.

3.3.2 Autores

As tabelas *authors* e *author_settings* contêm as informações de 216.463 autores, incluindo as replicações. Da tabela *authors*, os atributos utilizados foram:

- *author_id*: identificador do autor
- *submission_id*: identificador do artigo que o autor publicou
- *first_name*: primeiro nome do autor
- *middle_name*: nome do meio do autor
- *last_name*: sobrenome do autor
- *country*: sigla do país de onde o autor é

Da tabela *author_settings* vamos utilizar somente a informação da afiliação dos autores. A Tabela 3.2 apresenta o valor de *setting_name* e sua descrição.

Valor do *setting_name*	**Descrição**
affiliation	nome da instituição a qual o autor é afiliado

Tabela 3.2: *Valores utilizados de setting_name da tabela author_settings*

3.3.3 Métricas

A tabela *metrics* contém 25.776.074 valores de métricas correspondentes às informações de um acesso à um artigo. Da tabela *metrics*, utilizaram-se os seguintes atributos:

- *assoc_type*: tipo do acesso
- *assoc_id*: identificador do artigo associado
- *day*: dia, mês e o ano em que o acesso foi realizado
- *country_id*: sigla do país de onde veio o acesso
- *city*: nome da cidade de onde veio o acesso
- *metric_type*: identifica a forma de contagem

O valor 260 em *assoc_type* corresponde ao acesso do tipo '*download*'; os demais valores são considerados do tipo 'visualização'.

Neste trabalho, utilizou-se apenas o valor 'ojs::counter' do atributo *metric_type*. Filtrando as métricas com *metric_type* = 'ojs::counter', tem-se 25.216.843 registros, sendo que 14.380.727 são relacionados a *download* e 10.836.116 a visualização.

3.3.4 Revistas

Os dados das 211 revistas são armazenados nas tabelas *journals* e *journal_settings*. Da tabela *journals*, utilizou-se somente o atributo *journal_id* que representa o valor do identificador da revista.

A Tabela 3.3 apresenta os valores utilizados de *setting_name* da tabela *journal_settings*. Assim como foi feito na seleção dos títulos e das palavras-chave dos artigos, selecionaram-se apenas os títulos e as palavras-chave em português, ou seja, quando o valor do atributo *locale* = 'pt_BR'.

Valor do *setting_name*	**Descrição**
title	título da revista
searchDescription	descrição da revista
searchKeywords	palavras-chave da revista
publisherInstitution	instituição do editor

Tabela 3.3: *Valores utilizados de setting_name da tabela journal_settings*

Capítulo 4

Modelagem do Banco de Dados de Grafos

Neste capítulo, são apresentadas as etapas realizadas na modelagem do banco de dados de grafos. Estas etapas consistem em: levantamento de requisitos, descrição dos dados que serão armazenados, limpeza e padronização dos dados extraídos do banco de dados relacional do Portal de Revistas da USP e a construção do banco de dados orientado a grafos no Neo4j.

4.1 Levantamento de Requisitos

O primeiro passo para modelar o banco de dados orientado a grafos consiste no levantamento de requisitos, isto é, determinar quais dados serão armazenados no banco e quais questões o banco de dados deve ser capaz de responder. Esta etapa tem por objetivo guiar a modelagem para que o banco desenvolvido atenda às necessidades dos usuários da aplicação final.

As questões que foram consideradas relevantes para a modelagem do banco foram:

- Com quais autores um determinado autor publicou um artigo?
- Quantos artigos dois autores publicaram juntos?
- Quantos acessos um artigo teve?
- Quais artigos foram os mais acessados?
- Qual a localização dos acessos?

Dadas estas perguntas, é possível definir os nós, os relacionamentos e os rótulos do banco de dados a ser modelado.

4.2 Descrição dos Dados

Após o levantamento de requisitos, é realizada a descrição dos dados conforme os itens a seguir:

- Um **artigo** tem um título
- Um **artigo** tem um doi

- Um **artigo** tem um resumo
- Um **artigo** tem uma data de publicação
- Um **autor** tem um nome
- Uma **revista** tem um nome
- Uma **revista** tem uma lista de palavras-chave
- Uma **revista** tem uma descrição
- Um **local** tem o nome de um país
- Um **local** tem o nome de uma cidade
- Um **local** tem a sigla de um país
- Uma **afiliação** tem nome de uma universidade
- Uma **afiliação** tem nome de um instituto
- Um **artigo** *tem visualização vindo de* um **local**
- Um **artigo** *tem download vindo de* um **local**
- Um **artigo** *foi publicado por* um **autor**
- Um **artigo** *foi publicado em* uma **revista**
- Um **autor** *é de* um **local**
- Um **autor** *é afiliado a* uma **instituição**
- Uma **revista** *pertence a* uma **afiliação**

Os termos em negrito serão representados no banco como nós, os termos em itálico como os relacionamentos entre estes nós e os termos sublinhados como as propriedades dos nós.

Uma representação dos nós e relacionamentos descritos acima pode ser vista na Figura 4.1. Nesta figura, os nós (Artigo, Cidade, País, Autor, Revista, Instituto, Universidade) correspondem aos rótulos que serão utilizados no Neo4j para identificar o tipo de cada nó no grafo.

Os relacionamentos estão representados no diagrama em letra maiúscula e estão próximos às flechas que ligam os nós. Entre chaves tem-se as propriedades associadas aos nós e aos relacionamentos.

Foram retirados todos os acentos dos rótulos, relacionamentos e propriedades antes de serem criados no Neo4j.

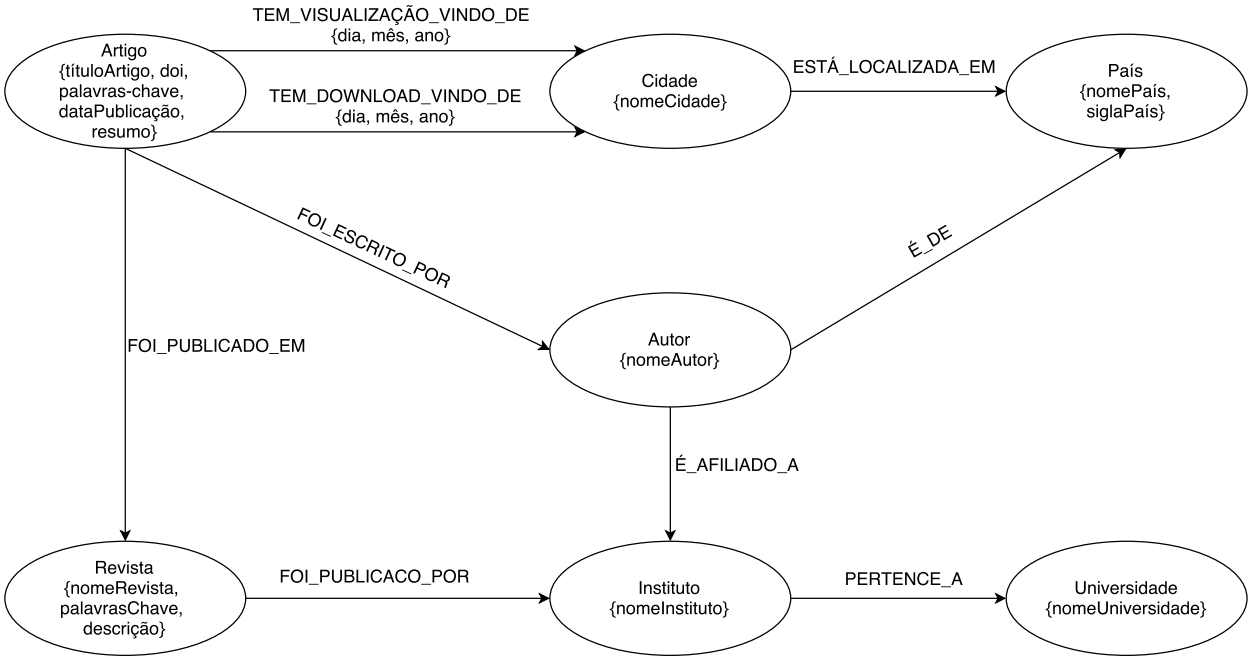


Figura 4.1: *Diagrama contendo os nós, relacionamentos e propriedades do banco de dados*

4.3 Limpeza e Padronização dos Dados

Como foi dito no capítulo anterior, os dados do banco do Portal de Revistas não estão completos e nem padronizados.

Se estes dados fossem inseridos no banco de dados orientados a grafos desta forma, existiriam alguns problemas como, por exemplo, dois autores de um mesmo instituto/universidade estarem ligados a dois institutos/universidades diferentes e uma revista estar duplicada por ter mais de um título.

Devido a isto, foi necessário limpar e normalizar estes dados antes de inseri-los no banco orientado a grafos modelado. Os códigos utilizados nesta seção encontram-se no Apêndice C.

4.3.1 Afiliações

Os nomes das afiliações dos autores, contidos na tabela *author_settings*, não estavam padronizados. Algumas afiliações apresentavam *link* para *sites*, endereços, números antes do nome da afiliação e várias representações para a mesma afiliação. Além disso, a afiliação pode ser representada pela universidade, instituto, faculdade ou departamento, sendo que nem todos os registros contidos no banco de dados do Portal de Revistas da USP contêm todas estas informações.

Um exemplo dos valores do atributo *setting_value* da tabela *author_settings* quando o atributo *setting_name* é 'affiliation' pode ser visto na Tabela 4.1.

Nome das afiliações
Veterinary Cancer Society (VCS) www.vetcancersociety.org
1 Departamento de Cirurgia, Faculdade de Medicina Veterinária, Universidade de São ...
2 Faculdade de Medicina Veterinária, Faculdades Metropolitanas Unidas, R. Ministro ...
2. Universidade Federal do RS
/Universidade de São Paulo; Faculdade de Filosofia, Letras e Ciências Humanas; ...
usp; Faculdade de Filosofia, Letras e Ciências Humanas
usp; Faculdade de Direito; Departamento de Direito Internacional
usp; Faculdade de Direito
universidade de sao paulo
universidade de São Paulo

Tabela 4.1: *Nome de algumas afiliações na tabela author_ settings*

Para padronizar o nome das afiliações, selecionaram-se apenas os nomes das universidades e o nome dos institutos (faculdades, escolas ou departamentos), ou seja, o campo *affiliation* foi dividido em outros dois: *university* e *institute*.

Os nomes de todas as afiliações foram tratados, convertendo todos os caracteres para maiúsculo e retirando acentuações e caracteres não alfanuméricos, com exceção de vírgula (,), ponto e vírgula (;) e ponto (.), que foram usados para detectar possíveis distinções entre universidades e institutos.

Também foi necessário padronizar o nome das universidades, pois elas estavam expressas ora por seu nome por extenso, ora por sua sigla. Para isto, consultou-se a tabela do Índice Geral de Cursos – IGC[1] do INEP para obter o mapeamento das siglas e dos nomes das universidades. Um trecho desta tabela pode ser visto na Figura 4.2.

Ano	Cód.IES	Nome da IES	Sigla da IES
2014	1	UNIVERSIDADE FEDERAL DE MATO GROSSO	UFMT
2014	2	UNIVERSIDADE DE BRASÍLIA	UNB
2014	3	UNIVERSIDADE FEDERAL DE SERGIPE	UFS
2014	4	UNIVERSIDADE FEDERAL DO AMAZONAS	UFAM
2014	5	UNIVERSIDADE FEDERAL DO PIAUÍ	UFPI
2014	6	UNIVERSIDADE FEDERAL DE OURO PRETO	UFOP
2014	7	UNIVERSIDADE FEDERAL DE SÃO CARLOS	UFSCAR
2014	8	UNIVERSIDADE FEDERAL DE VIÇOSA	UFV
2014	9	UNIVERSIDADE ESTADUAL DE LONDRINA	UEL
2014	10	PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ	PUCPR
2014	11	UNIVERSIDADE CATÓLICA DE PERNAMBUCO	UNICAP
2014	12	UNIVERSIDADE FEDERAL DO RIO GRANDE	FURG
2014	13	UNIVERSIDADE DE CAXIAS DO SUL	UCS
2014	14	UNIVERSIDADE DO VALE DO RIO DOS SINOS	UNISINOS
2014	15	UNIVERSIDADE CATÓLICA DE PETRÓPOLIS	UCP
2014	16	UNIVERSIDADE GAMA FILHO - DESCREDENCIADA (DESPACH(	UGF
2014	17	UNIVERSIDADE FEDERAL DE UBERLÂNDIA	UFU
2014	18	UNIVERSIDADE CATÓLICA DE PELOTAS	UCPEL

Figura 4.2: *Trecho da tabela do IGC utilizado para mapear as siglas e os nomes das universidades*

[1] http://portal.inep.gov.br/educacao-superior/indicadores/indice-geral-de-cursos-igc

4.3.2 Artigos

Realizaram-se a limpeza e a padronização das palavras-chave dos artigos da seguinte forma: os acentos de todos caracteres foram removidos; os caracteres maiúsculos foram substituídos por minúsculos; e os caracteres ponto, vírgula, hifens e travessão foram todos substituídos por ponto-vírgula, já que o caractere mais utilizado para separação das palavras-chaves é o ponto-vírgula.

4.3.3 Autores

Como no banco de dados um autor pode ter mais de um identificador, foi necessário gerar um número único para representá-los. Desta forma, considerou-se que não há homônimos neste banco de dados.

Além disso, como existem autores que não têm nome do meio, ou seja, o atributo *middle_name* é *NULL*, foi feita a junção de espaços entre o *first_name* e o *last_name*.

Outros problemas enfrentados foram em relação aos nomes dos autores. Encontraram-se caracteres não-alfanuméricos nos atributos *first_name*, *middle_name* e *last_name*. Além disso, um mesmo autor poderia ter seu nome escrito de diversas formas. Como existem 216.463 registros na tabela *authors* e há muitos casos particulares, não foi viável tratar cada um deles.

4.3.4 Métricas

Em relação às métricas, o atributo *day* foi dividido em três outros atributos: dia, mês e ano. Isto foi feito pois este atributo guardava a informação do ano, mês e dia numa única *string*, por exemplo '20160403', e isto dificulta a busca envolvendo datas no banco de dados orientado a grafos.

Além disso, nesta tabela, existem várias tuplas sem a informação da cidade de origem dos acessos aos artigos. Quando isto ocorre, colocou-se a cidade como sendo 'Desconhecida', foi feito o mesmo com o campo *country*, pois vários também estavam vazios ou com siglas que não correspondem a algum país.

Foi dada uma atenção especial às cidades brasileiras, já que a maioria dos acessos tem origem no Brasil. As cidades dos demais países foram ignoradas, pois não seriam usadas neste trabalho.

Alguns nomes das cidades brasileiras estão com erros de grafia e, para resolver isto, utilizou-se uma tabela que contém o nome de todas as cidades do Brasil. Um exemplo de uma cidade que está com nome incorreto no banco de dados do Portal de Revistas da USP é 'FOZ DO IGUA', neste caso, o nome foi facilmente corrigido para 'FOZ DO IGUACU'. Porém, existem casos em que não foi possível corrigir o nome, como as cidades que têm em seu nome apenas a letra 'S'.

4.3.5 Países

Foi necessário corrigir e normalizar os nomes dos países contidos no arquivo dos autores, pois os valores do atributo *country* da tabela *authors* são representados de duas formas: através da sigla do país ou do nome do país. Para resolver este problema, foi mapeado, em um arquivo CSV, o nome de cada país (por extenso) com sua respectiva sigla.

Além disso, muitos valores (nome dos países) estavam com erros de ortografia, um exemplo pode ser visto na Tabela 4.2.

Representações do Nome do país Brasil
, Brasil
- Brasil
BRA
BRASIL
BRAZIL
Barzil
Basil
Br
Brasi
Brasil
Brazi
Brasill

Tabela 4.2: *Representações do nome do país Brasil na tabela authors*

4.3.6 Revistas

O campo *journal_ institution*, do arquivo 'journalsInstitution.csv', foi dividido em *university* e *institute*, como foi feito com as afiliações. E as palavras-chave das revistas foram tratadas da mesma forma que as dos artigos.

Além disso, no banco de dados do Portal de Revistas, algumas revistas tinham mais de um título, como pode ser visto na Tabela 4.3. Nestes casos, as seguintes regras foram aplicadas para selecionar somente um deles:

- se os títulos estiverem em línguas diferentes, selecionar o título em português
- se um dos títulos possuir acentos e o outro não, selecionar o título sem acentos
- se os títulos diferirem em algumas palavras, buscou-se, na internet, o título correto da revista

journal_id	journal_title
1	Acolhendo a Afabetização em Países de Língua Portuguesa
1	Acolhendo a Alfabetização nos Países de Língua Portuguesa
6	Boletim de Botânica
6	Boletim de Botânica da Universidade de São Paulo
9	ARS
9	ARS (São Paulo)
19	Estilos da Clinica
19	Estilos da clínica
28	Archives of Clinical Psychiatry
28	Revista de Psiquiatria Clínica

Tabela 4.3: *Algumas revistas com mais de um título*

4.4 Construção do Banco de Dados no Neo4j

Construiu-se o banco de dados no Neo4j importando os arquivos CSV gerados a partir da extração e do tratamento (limpeza e padronização) dos dados do banco de dados relacional do Portal de Revistas.

4.4.1 Criação de Restrições e Índices

De acordo com a especificação da estrutura do banco de dados orientado a grafos, foram criadas restrições e índices sobre as propriedades dos nós. As restrições são utilizadas para evitar a criação de nós distintos com uma mesma propriedade, pois não é desejável, por exemplo, a criação de dois nós :Autor com o mesmo valor para propriedade *nomeAutor*.

```
1 CREATE CONSTRAINT ON (a:Artigo) ASSERT a.artigoID IS UNIQUE;
2 CREATE CONSTRAINT ON (a:Autor) ASSERT a.nomeAutor IS UNIQUE;
3 CREATE CONSTRAINT ON (p:Pais) ASSERT p.nomePais IS UNIQUE;
4 CREATE CONSTRAINT ON (p:Pais) ASSERT p.siglaPais IS UNIQUE;
5 CREATE CONSTRAINT ON (r:Revista) ASSERT r.revistaID IS UNIQUE;
6 CREATE CONSTRAINT ON (u:Universidade) ASSERT u.nomeUniversidade IS UNIQUE;
```

Os índices nas propriedades dos nós são criados para que a busca por estes nós seja mais eficiente. Isto melhora o tempo da criação dos relacionamentos, pois para criar um relacionamento é feita uma busca pelos nós que este relacionamento liga.

```
1 CREATE INDEX ON :Revista(nomeRevista);
2 CREATE INDEX ON :Artigo(tituloArtigo);
3 CREATE INDEX ON :Artigo(doi);
4 CREATE INDEX ON :Cidade(nomeCidade);
5 CREATE INDEX ON :Instituto(nomeInstituto);
```

4.4.2 Criação dos Nós

Para criação dos nós com rótulos :Artigo e :Revista foram necessárias mais de uma consulta (*query*), pois as informações das propriedades estão em arquivos distintos devido à forma não tradicional das tabelas do banco de dados relacional do Portal de Revistas da USP.

Para o nó :Artigo, primeiramente, cria-se um nó com o título e o identificador deste artigo no banco de dados do Portal de Revistas. Após a criação do nó, um índice sobre o identificador do artigo é criado, pois será necessário buscar os nós por estes identificadores para adicionar as outras propriedades (palavraChave e descricao) dos nós. Um processo similar é feito com os nós com rótulo :Revista.

4.4.2.1 Artigos

A criação e inserção das propriedades dos nós com rótulo :Artigo é feita em 4 passos. Primeiro, os nós com rótulo :Artigo são criados a partir do arquivo articlesTitle.csv, um trecho deste arquivo pode ser visto na Figura 4.3.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///articlesTitle.csv" AS row
3 CREATE (artigo:Artigo {tituloArtigo: row.article_title,
4                        artigoID: toInt(row.article_id),
5                        dataPublicacao: row.date_submitted});
```

```
article_id,journal_id,date_submitted,article_title
1,61,2011-12-16 08:38:15,Groups generated by 3state automata over a 2letter alphabet I
2,61,2011-12-16 08:40:48,Selfsimilar groups and their geometry
3,61,2011-12-16 08:44:29,Classification of algebras with minimal quadratic growth of identities
5,61,2011-12-16 08:52:12,On semiperfect rings of injective dimension one
7,61,2011-12-16 09:08:24,The primitives of the Hopf algebra of noncommutative symmetric functions
8,61,2011-12-16 09:13:09,Stability in Nonautonomous Dynamics: A Survey of Recent Results
```

Figura 4.3: *Primeiras linhas do arquivo articlesTitle.csv*

Em seguida, inserem-se os títulos em 'en_US' dos artigos que não tinham título em 'pt_BR'. Isto é feito a partir do arquivo articlesTitleEN.csv, um trecho deste arquivo pode ser visto na Figura 4.4.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///articlesTitleEN.csv" AS row
3 MATCH (artigo:Artigo {artigoID: toInt(row.article_id)})
4 WHERE artigo.tituloArtigo = 'TITULO NULO' OR artigo.tituloArtigo = ''
5 SET artigo.tituloArtigo = row.article_titleen;
```

```
article_id,article_titleen
85,Cultural Forests: Sociocultural Handling territorialities and sustentability
89,The population of Concórdia district  Tuneiras do Oeste  Paraná  Brasil The ...
90,Rural Traditions at the party from divine eternal father in Trindade GO: ...
91,The Lourde's Jamamadi territorialities: tradition and modernity
95,Agriculture and tecnification: notes for a discussion
```

Figura 4.4: *Primeiras linhas do arquivo articlesTitleEN.csv*

No terceiro passo, o DOI de cada artigo é inserido de acordo com o arquivo articlesDoi.csv, um trecho deste arquivo pode ser visto na Figura 4.5.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///articlesDoi.csv" AS row
3 MATCH (artigo:Artigo {artigoID: toInt(row.article_id)})
4 SET artigo.doi = row.article_doi;
```

```
article_id,article_doi
1,10.11606/issn.2316-9028.v1i1p1-39
2,10.11606/issn.2316-9028.v1i1p41-95
3,10.11606/issn.2316-9028.v1i1p97-109
5,10.11606/issn.2316-9028.v1i1p111-132
7,10.11606/issn.2316-9028.v1i2p175-202
8,10.11606/issn.2316-9028.v1i2p133-173
```

Figura 4.5: *Primeiras linhas do arquivo articlesDoi.csv*

O quarto passo é inserir as palavras-chave de cada artigo conforme o arquivo gerado vf_articlesKeywords.csv, um trecho deste arquivo pode ser visto na Figura 4.6.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_articlesKeywords.csv" AS row
3 MATCH (artigo:Artigo {artigoID: toInt(row.article_id)})
4 SET artigo.palavrasChave = row.article_keywords;
```

```
"article_id","article_keywords"
72,""
73,"reforma agraria; agricultura; politica agraria; estado; campo;"
74,"reforma agraria; movimentos sociais; estado; cidadania"
75,"reforma agraria; agricultura camponesa; agricultura familiar; questao agraria e estado"
76,"bairro rural; campesinato; identidade; territorio;"
```

Figura 4.6: *Primeiras linhas do arquivo vf_articlesKeywords.csv*

Por fim, inserem-se os resumos dos artigos que estão armazenados no arquivo gerado vf_articlesAbstract.csv, um trecho deste arquivo pode ser visto na Figura 4.7.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_articlesAbstract.csv" AS row
3 MATCH (artigo:Artigo {artigoID: toInt(row.article_id)})
4 SET artigo.resumo = row.article_abstract;
```

```
"article_id","article_abstract"
1,"Groups generated by 3-state automata over a 2-letter alphabet, I"
2,"Self-similar groups and their geometry"
3,"The main goal of this paper is to prove that the five ... growth."
```

Figura 4.7: *Primeiras linhas do arquivo vf_articlesAbstract.csv*

Após a criação dos nós com rótulo :Artigo, realizou-se uma consulta no banco de dados Neo4j buscando por artigos, com limite de 9 nós na resposta. O resultado desta consulta pode ser visto na Figura 4.8.

Figura 4.8: *Imagem do Neo4j após inserção das artigos*

4.4.2.2 Autores

A criação dos nós :Autores é feita em uma única etapa utilizando o arquivo vf_authorsInfo.csv, gerado após a limpeza e padronização dos dados. Um trecho deste arquivo pode ser visto na Figura 4.9.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_authorsInfo.csv" AS row
3 MERGE (autor:Autor {nomeAutor: row.author_name});
```

```
"submission_id","author_name","university_name","institute_name","country_code"
1,"Ievgen Bondarenko","TEXAS AM UNIVERSITY","Desconhecido","US"
1,"Rostislav Grigrchuk","TEXAS AM UNIVERSITY","Desconhecido","US"
1,"Rostyslav Kravchenko","TEXAS AM UNIVERSITY","Desconhecido","US"
1,"Yevgen Muntyan","TEXAS AM UNIVERSITY","Desconhecido","US"
1,"Volodymyr Nekraschevych","TEXAS AM UNIVERSITY","Desconhecido","US"
1,"Dmytro Savchuk","TEXAS AM UNIVERSITY","Desconhecido","US"
1,"Zoran Sunié","TEXAS AM UNIVERSITY","Desconhecido","US"
```

Figura 4.9: *Primeiras linhas do arquivo vf_ authorsInfo.csv*

A Figura 4.10 apresenta uma consulta dos autores (com limite de 25) após a inserção dos nós com rótulo :Autor.

Figura 4.10: *Imagem do Neo4j após inserção das autores*

4.4.2.3 Países

Assim como os autores, os nós com rótulo :Pais são criados e as suas propriedades inseridas em uma única etapa, a partir do arquivo tab_country.csv. Um trecho deste arquivo pode ser visto na Figura 4.11.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///tab_country.csv" AS row
3 CREATE (pais:Pais {nomePais: row.country_name, siglaPais: row.country_code
      });
```

```
country_name,country_code
Desconhecido,Desconhecido
Afghanistan,AF
Aland Islands,AX
Albania,AL
Algeria,DZ
American Samoa,AS
Andorra,AD
```

Figura 4.11: *Primeiras linhas do arquivo tab_ country.csv*

A Figura 4.12 mostra o resultado de uma consulta no banco de dados do Ne4j após a inserção dos nós com rótulo :Pais

Figura 4.12: *Imagem do Neo4j após inserção dos países*

4.4.2.4 Revistas

Para criar os nós com rótulo :Revista e inserir suas propriedades, são necessárias 3 etapas. A primeira utiliza o arquivo journalsTitle.csv. Um trecho deste arquivo pode ser visto na Figura 4.13.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///journalsTitle.csv" AS row
3 CREATE (revista:Revista {nomeRevista: row.journal_title,
4                          revistaID: toInt(row.journal_id)});
```

```
journal_id,journal_title
1,Acolhendo a Alfabetização nos Países de Língua Portuguesa
2,Almanack Braziliense
3,Anais do Museu Paulista: História e Cultura Material
4,Arquivos de Zoologia (São Paulo)
6,Boletim de Botânica da Universidade de São Paulo
7,Brazilian Journal of Oceanography
8,Brazilian Journal of Veterinary Research and Animal Science
```

Figura 4.13: *Primeiras linhas do arquivo journalsTitle.csv*

Em seguida, as palavras-chave de cada revista são inseridas no banco de dados utilizando o arquivo vf_journalsKeywords.csv. Um trecho deste arquivo pode ser visto na Figura 4.14.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_journalsKeywords.csv" AS row
3 MATCH (revista:Revista {revistaID: toInt(row.journal_id)})
4 SET revista.palavraChave = row.journal_keywords;
```

```
"journal_id","journal_keywords"
1,"lingua portuguesa"
2,"historia; brasil"
3,"historia; museus"
4,"anatomia; biogeografia; biologia evolutiva; comportamento; ecologia; ..."
7,"oceanografia; oceanografia biologica; oceanografia fisica; geologia"
```

Figura 4.14: *Primeiras linhas do arquivo vf_journalsKeywords.csv*

Por último, são inseridas as propriedades, que descrevem as revistas e estão no arquivo journalsDescription.csv. Um trecho deste arquivo pode ser visto na Figura 4.15.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///journalsDescription.csv" AS row
3 MATCH (revista:Revista {revistaID: toInt(row.journal_id)})
4 SET revista.descricao = row.journal_description;
```

```
journal_id,journal_description
1,Acolhendo a Alfabetização nos Países de Língua Portuguesa
2,Almanack Braziliense
3,Anais do Museu Paulista
4,Arquivos de Zoologia (São Paulo)
7,Brazilian Journal of Oceanography
9,ARS
```

Figura 4.15: *Primeiras linhas do arquivo journalsDescription.csv*

A Figura 4.16 apresenta o resultado de uma consulta no Neo4j, buscando os nós com rótulo :Revista após a inserção dos mesmos.

Figura 4.16: *Resultado de uma consulta no Neo4j após inserção das revistas*

4.4.2.5 Universidades

Assim como os autores e os países, os nós com rótulo :Universidades são criados e as suas propriedades inseridas em uma única etapa. Isto é feito a partir do arquivo tab_institute.csv. Um trecho deste arquivo pode ser visto na Figura 4.17.

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///tab_institute.csv" AS row
3 MERGE (uni:Universidade {nomeUniversidade: row.university_name});
```

```
"university_name","institute_name"
"TEXAS AM UNIVERSITY","Desconhecido"
"UNIVERSIDADE FEDERAL DE MINAS GERAIS","Desconhecido"
"UNIVERSIDADE CATOLICA DE MINAS GERAIS","Desconhecido"
"KIEV NATIONAL TARAS SHEVCHENKO UNIVERSITY","Desconhecido"
"Desconhecido","Desconhecido"
"DEPARTAMENTO DE MATEMATICA; INSTITUTO SUPERIOR TECNICO","INSTITUTO SUPERIOR TECNICO"
"UNIVERSIDADE DE SAO PAULO","INSTITUTO DE MATEMATICA E ESTATISTICA"
"INSTITUTE OF MATHEMATICS","INSTITUTE OF MATHEMATICS"
"UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL","Desconhecido"
```

Figura 4.17: *Primeiras linhas do arquivo tab_institute.csv*

A Figura 4.18 mostra o resultado de uma consulta no banco de dados Neo4j buscando pelos nós com rótulo :Universidade, após a inserção dos mesmos.

Figura 4.18: *Imagem do Neo4j após inserção das universidades*

4.4.3 Criação dos Relacionamentos

Para criação dos relacionamentos, buscam-se os nós entre os quais quer-se adicionar um relacionamento (aresta) de acordo com as propriedades desejadas destes nós. A cláusula 'MERGE' cuida da criação do elemento adequado no banco de dados. Abaixo estão os códigos utilizados para a criação dos relacionamentos.

- Relacionamento [:ESTA_LOCALIZADA_EM] entre os nós cidade –> país

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///tab_cities.csv" AS row
3 CREATE (cidade:Cidade {nomeCidade: row.city_name})
4 WITH cidade, row
5 MATCH (pais:Pais {siglaPais: row.country_code})
6 MERGE (cidade)-[:ESTA_LOCALIZADA_EM]->(pais);
```

- Relacionamento [:PERTENCE_A] entre os nós instituto –> universidade

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///tab_institute.csv" AS row
3 CREATE (instituto:Instituto {nomeInstituto: row.institute_name})
4 WITH instituto, row
5 MATCH (universidade:Universidade {nomeUniversidade: row.
      university_name})
6 MERGE (instituto)-[:PERTENCE_A]->(universidade);
```

- Relacionamento [:FOI_PUBLICADO_EM] entre os nós artigo –> revista

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///articlesTitle.csv" AS row
3 MATCH (artigo:Artigo {artigoID: toInt(row.article_id)})
4 MATCH (revista:Revista {revistaID: toInt(row.journal_id)})
5 MERGE (artigo)-[:FOI_PUBLICADO_EM]->(revista);
```

- Relacionamento [:FOI_ESCRITO_POR] entre os nós artigo -> autor

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_authorsInfo.csv" AS row
3 MATCH (autor:Autor {nomeAutor: row.author_name})
4 MATCH (artigo:Artigo {artigoID: toInt(row.submission_id)})
5 MERGE (artigo)-[:FOI_ESCRITO_POR]->(autor);
```

- Relacionamento [:E_DE] entre os nós autor -> país

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_authorsInfo.csv" AS row
3 MATCH (autor:Autor {nomeAutor: row.author_name})
4 MATCH (pais:Pais {siglaPais: row.country_code})
5 MERGE (autor)-[:E_DE]->(pais);
```

- Relacionamento [:E_AFIALIADO_A] entre os nós autor -> instituto

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_authorsInfo.csv" AS row
3 MATCH (autor:Autor)
4 MATCH (inst:Instituto)-[:PERTENCE_A]->(uni:Universidade)
5 WHERE inst.nomeInstituto = row.institute_name AND
6       uni.nomeUniversidade = row.university_name AND
7       autor.nomeAutor = row.author_name
8 MERGE (autor)-[:E_FILIADO_A]->(inst);
```

- Relacionamento [:FOI_PUBLICADO_POR] entre os nós revista -> instituto

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_journalsInstitution.csv" AS
    row
3 MATCH (revista:Revista)
4 MATCH (inst:Instituto)-[:PERTENCE_A]->(uni:Universidade)
5 WHERE inst.nomeInstituto = row.institute_name AND
6       uni.nomeUniversidade = row.university_name AND
7       revista.revistaID = toInt(row.journal_id)
8 MERGE (revista)-[:FOI_PUBLICADO_POR]->(inst);
```

A Figura 4.19 mostra uma parte do banco de dados após a inserção das relações descritas acima no Neo4j.

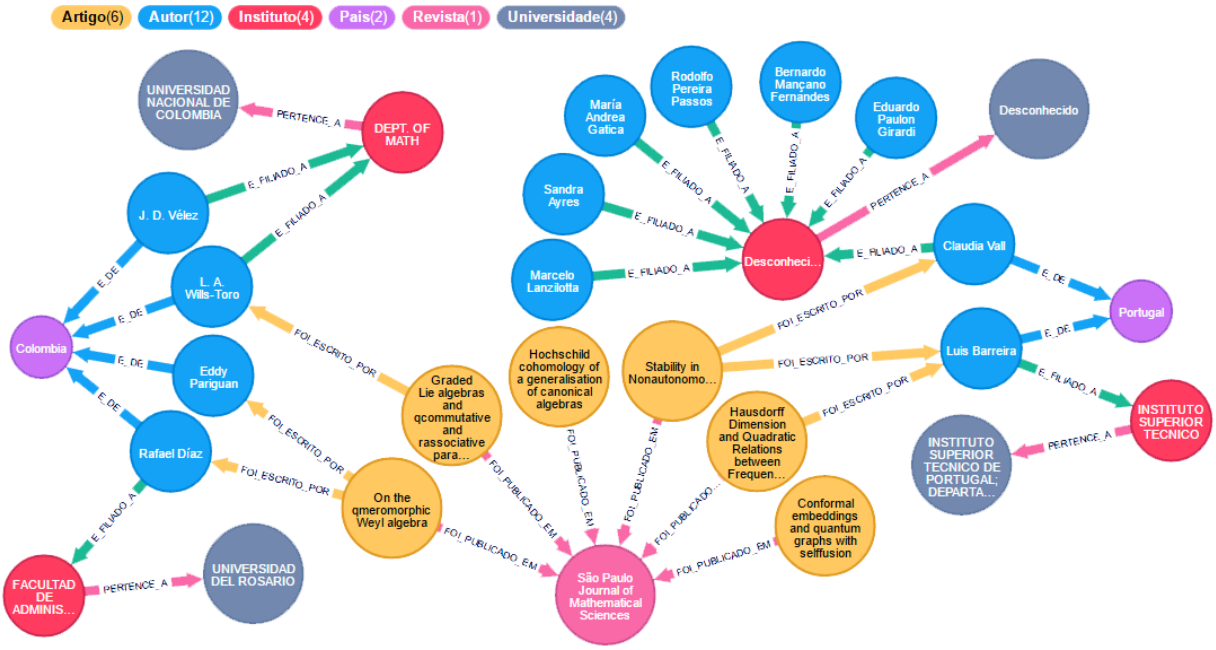


Figura 4.19: *Banco de dados após a criação das relações, com exceção dos relacionamentos de acesso (visualização e download)*

- Relacionamento [:TEM_VISUALIZACAO_VINDO_DE] entre os nós artigo –> cidade

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_visualizInfo.csv" AS row
3 MATCH (artigo:Artigo {artigoID: toInt(row.assoc_id)}),
4       (cidade:Cidade)-[:ESTA_LOCALIZADA_EM]->(pais:Pais)
5 WHERE cidade.nomeCidade = row.city_name AND
6       pais.siglaPais = row.country_code
7 CREATE (artigo)-[:TEM_VISUALIZACAO_VINDO_DE {dia: row.day, mes: row.
      month, ano: row.year}]->(cidade);
```

- Relacionamento [:TEM_DOWNLOAD_VINDO_DE] entre os nós artigo –> cidade

```
1 USING PERIODIC COMMIT
2 LOAD CSV WITH HEADERS FROM "file:///vf_downloadInfo.csv" AS row
3 MATCH (artigo:Artigo {artigoID: toInt(row.assoc_id)}),
4       (cidade:Cidade)-[:ESTA_LOCALIZADA_EM]->(pais:Pais)
5 WHERE cidade.nomeCidade = row.city_name AND
6       pais.siglaPais = row.country_code
7 CREATE (artigo)-[:TEM_DOWNLOAD_VINDO_DE {dia: row.day, mes: row.month
      , ano: row.year}]->(cidade);
```

A Figura 4.20 apresenta uma consulta dos acessos a dois artigos após a criação dos relacionamentos TEM_VISUALIZACAO_VINDO_DE e TEM_DOWNLOAD_VINDO_DE.

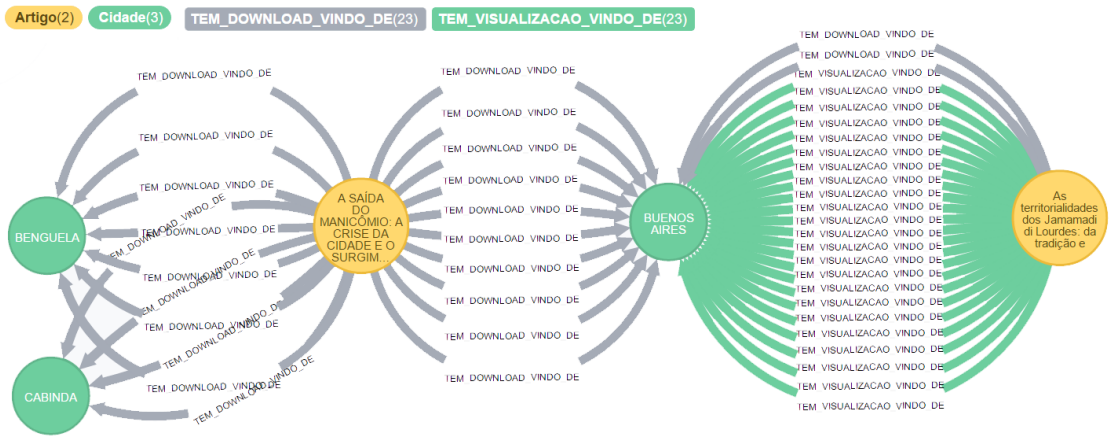


Figura 4.20: *Exemplo dos relacionamentos de acesso (visualização e download) a dois artigos, após a inserção destes relacionamentos no banco de dados Neo4j*

Capítulo 5

Análises

Neste capítulo, são apresentadas as análises de detecção de comunidades realizadas sobre os dados inseridos no banco de dados orientado a grafos modelado e, em seguida, as visualizações dos acessos aos artigos, separados por região.

5.1 Detecção de Comunidade

Para detecção de comunidade, utilizou-se a biblioteca igraph[1] do R. Além disso, utilizaram-se as bibliotecas RNeo4j[2], para conexão com o banco de dados Neo4j, e visNetwork[3], para visualização dos grafos.

Antes de executar os algoritmos, foi necessário conectar com o banco Neo4j a fim de realizar consultas para selecionar os nós e as arestas dos grafos desejados.

No grafo completo, ou seja, contendo todos os autores e seus relacionamentos, existem 115.361 nós e 503.688 arestas. Para cada método, quatro experimentos foram realizados: um com o grafo completo e os outros três com um subgrafo do grafo completo. Os nós destes grafos são representados pelos autores e as arestas pelas relações de coautoria entre eles.

No método de passeios aleatórios, o tamanho do passeio aleatório considerado em todos os experimentos foi igual a 4.

O código utilizado para realizar a detecção de comunidade pode ser visto no Apêndice D.

5.1.1 Primeiro Experimento

O primeiro experimento utilizou o grafo completo, 115.361 nós e 503.688 arestas. Como o conjunto inteiro é muito grande, não foi possível obter uma imagem dos nós e das comunidades detectadas. O tempo limite considerado para execução dos algoritmos foi de 3h.

Como pode ser visto na Tabela 5.1, o método de Louvain é o que apresenta maior valor de modularidade, ou seja, melhor particionamento dos nós em comunidades. E o método que utiliza autovetores é o que possui menor valor de modularidade, $0,5372661$.

[1] http://igraph.org/

[2] https://cran.r-project.org/web/packages/RNeo4j/RNeo4j.pdf

[3] https://cran.r-project.org/web/packages/visNetwork/vignettes/Introduction-to-visNetwork.html

	Número de Comunidades Detectadas	Modularidade em [0,1]	Tempo de Execução (em segundos)
Girvan-Newman	-	-	tempo excedido
Louvain	30.600	0,9589542	1,55
Autovetores	30.484	0,5372661	23,62
Passeios Aleatórios	33.495	0,8993574	104,14

Tabela 5.1: *Tabela com as informações do primeiro experimento*

A maior comunidade detectada pelo método de Louvain possui 2.823 nós e as demais comunidades possuem uma distribuição uniforme. A Figura 5.1 apresenta a distribuição dos nós nas 8 comunidades com mais nós.

Figura 5.1: *Número de nós nas 8 comunidades com mais nós após a aplicação do método de Louvain no grafo do experimento 1*

Na Figura 5.2 temos a distribuição da quantidade de nós nas 8 comunidades com maior número de autores aplicando o método de passeios aleatórios no grafo completo. É possível ver que a maioria dos nós encontra-se numa mesma comunidade.

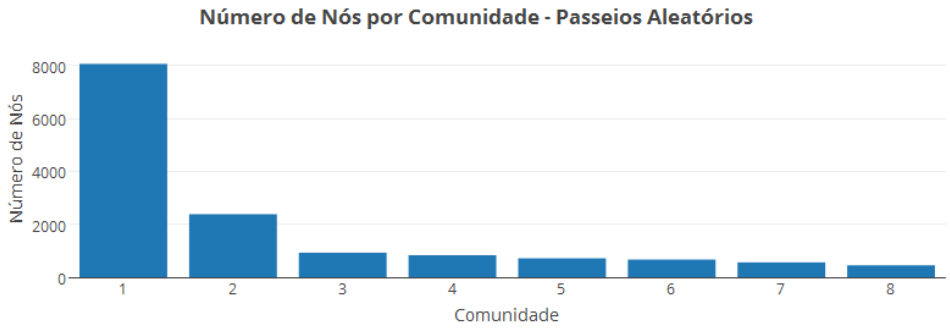


Figura 5.2: *Número de nós nas 8 comunidades com mais nós após a aplicação do método de passeios aleatórios no grafo do experimento 1*

Como a modularidade encontrada pelo método de autovetores é a menor dentre os outros dois métodos (Louvain e passeios aleatórios), a detecção da comunidade de cada nó não é tão precisa. Como nota-se na Figura 5.3, a grande maioria dos nós, em torno de 50.000, está concentrada na mesma comunidade.

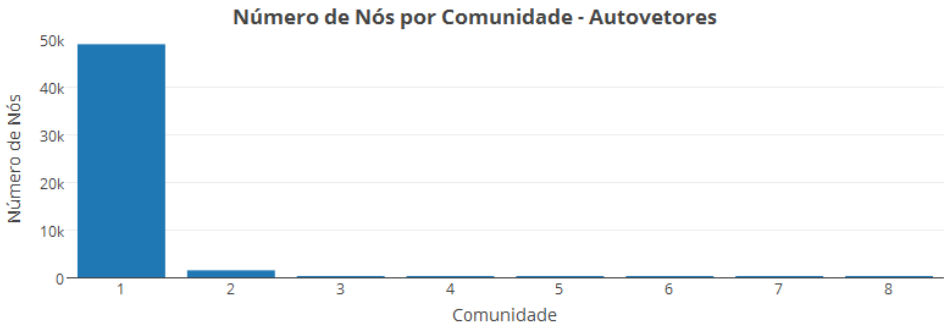


Figura 5.3: *Número de nós nas 8 comunidades com mais nós após a aplicação do método de autovetores no grafo do experimento 1*

Com os gráficos apresentados, observa-se que, mesmo que o número de comunidades detectadas pelos métodos Louvain, autovetores e passeios aleatórios sejam, relativamente, próximos, as comunidades detectadas não são exatamente as mesmas. Nos métodos de autovetores e de passeios aleatórios, os autores estão concentrados em uma única comunidade, enquanto no método de Louvain, os autores estão distribuídos de modo mais uniforme entre as comunidades.

5.1.2 Segundo Experimento

No segundo experimento, escolheram-se os 10 autores que publicaram mais artigos e, a partir destes autores, selecionaram-se todos os autores com quem eles publicaram. No total, esta amostra contém 1.100 autores, 1.118 arestas entre eles e 2.137 artigos.

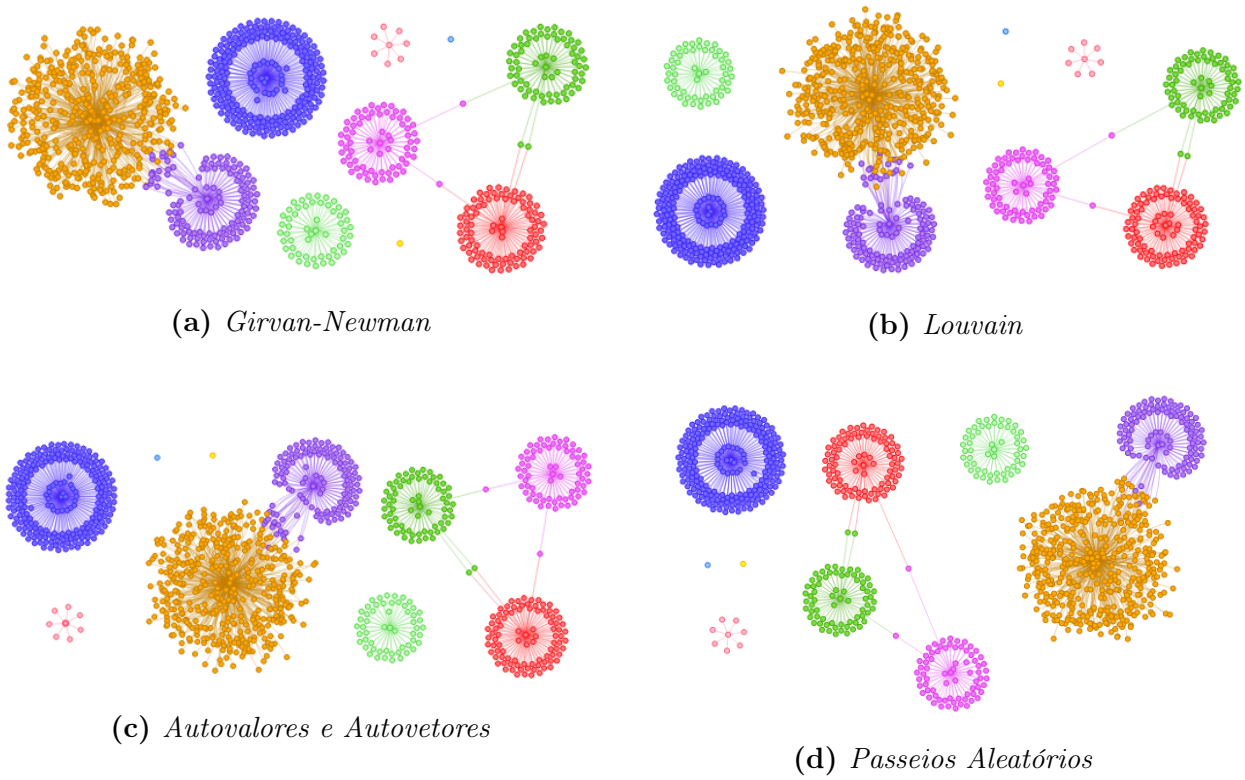
Os 4 métodos detectaram 10 comunidades, sendo 2 destas comunidades compostas apenas de 1 nó. Em todos os métodos, uma das comunidades detectadas era composta apenas do autor "Os Editores" e uma outra comunidade do autor "O Editor".

Como é possível ver na Tabela 5.2, o valor da modularidade de todos os métodos é bem próximo, só mudando a partir da terceira casa decimal. O que possui maior modularidade é o método de Louvain e o com menor modularidade é o método de passeios aleatórios.

	Número de Comunidades Detectadas	**Modularidade em [0,1]**	**Tempo de Execução (em segundos)**
Girvan-Newman	10	0,7741037	20,3500
Louvain	10	0,7757297	0,0087
Autovetores	10	0,7750811	0,0264
Passeios Aleatórios	10	0,7723873	0,5936

Tabela 5.2: *Tabela com as informações do segundo experimento*

Como também pode ser visto nesta tabela, os 4 métodos detectam 10 comunidades, sendo que 8 delas foram as mesmas em todos os métodos. A Figura 5.4 apresenta os nós do subgrafo deste experimento em que cada cor representa uma comunidade.



(a) *Girvan-Newman*

(b) *Louvain*

(c) *Autovalores e Autovetores*

(d) *Passeios Aleatórios*

Figura 5.4: *Resultado da aplicação dos métodos de detecção de comunidades sobre o subgrafo com 1.100 autores utilizado no segundo experimento*

Na Tabela 5.3 nota-se que as comunidades 1 e 3 são as únicas diferentes em todos os métodos e que o número de nós na comunidade 1 é maior no método de passeios aleatórios.

Na Figura 5.4, a comunidade 1 é representada pelos nós em amarelo escuro e a comunidade 3 pelos nós na cor roxa. Além disso, é possível confirmar que o número de nós da comunidade 1 (nós em amarelo escuro) é maior no método de passeios aleatórios.

	Girvan-Newman	**Louvain**	**Autovetores**	**Passeios Aleatórios**
Comunidade 1	444	449	445	463
Comunidade 2	227	227	227	227
Comunidade 3	127	122	126	108
Comunidade 4	92	92	92	92
Comunidade 5	73	73	73	73
Comunidade 6	72	72	72	72
Comunidade 7	55	55	55	55
Comunidade 8	8	8	8	8
Comunidade 9	1	1	1	1
Comunidade 10	1	1	1	1

Tabela 5.3: *Tabela com as distribuições da quantidade de nós em cada comunidade*

As comunidades detectadas representam conjuntos de autores de uma mesma área. De acordo com a Figura 5.4 e a Tabela 5.4, observa-se que as comunidades 1 (amarelo escuro) e 3 (roxo) representam áreas que têm alguma relação. Já as áreas das comunidades 7 (verde claro) e 8 (rosa claro) não tem relação com outras áreas.

	Área
Comunidade 1	Agropecuária
Comunidade 2	Medicina
Comunidade 3	Ciências Agrárias
Comunidade 4	Psicologia
Comunidade 5	Saúde Pública
Comunidade 6	Enfermagem
Comunidade 7	Botânica
Comunidade 8	Sociologia

Tabela 5.4: *Tabela com a área de cada comunidade detectada no experimento 2*

5.1.3 Terceiro Experimento

Para construir o subgrafo deste experimento, primeiro, foram selecionados os autores que publicaram o artigo com mais colaboradores. Este artigo, "Mudança dos critérios Qualis!", foi publicado por 64 autores da área médica.

Após selecionar estes autores, também foram selecionados os autores com quem estes publicaram algum outro artigo. No final, o subgrafo utilizado neste experimento contém 634 nós e 4.614 arestas e pode ser visto na Figura 5.5. Nesta figura, é possível ver que o grafo só tem 1 componente conexa, ou seja, existe um caminho entre todos os nós deste subgrafo.

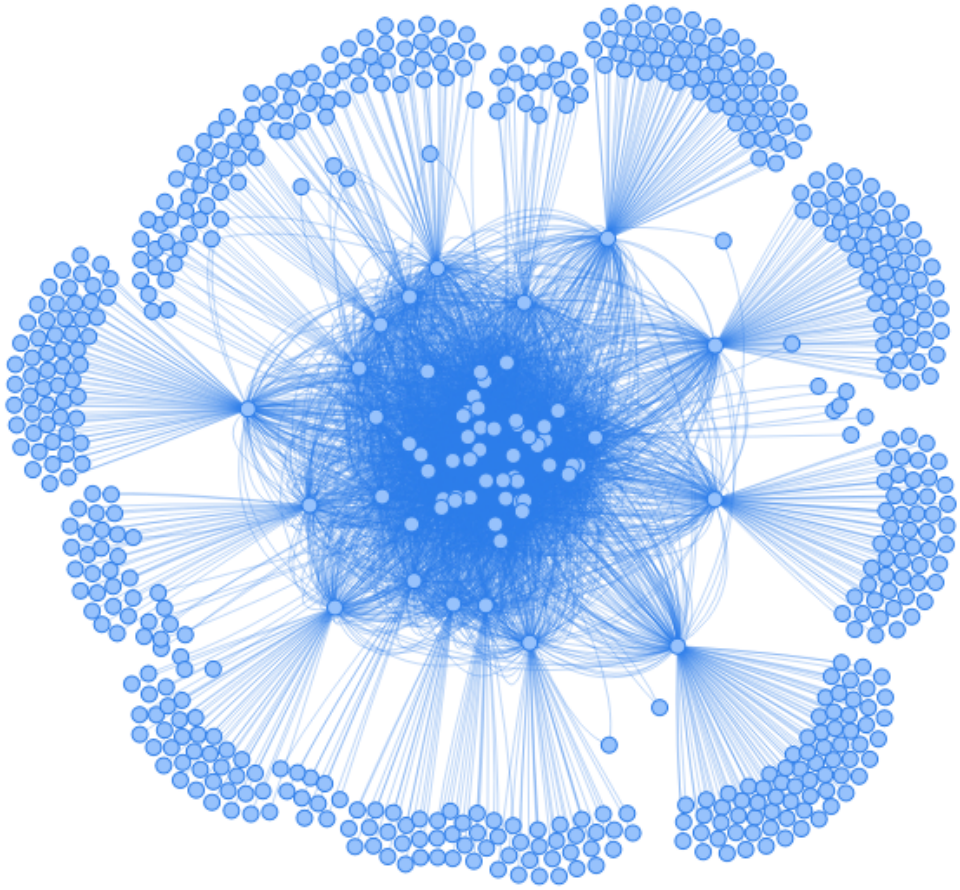


Figura 5.5: *Subgrafo, com 634 nós e 4.614 arestas ,utilizado no terceiro experimento*

Como é possível ver na Tabela 5.5, o valor da modularidade em todos os métodos é menor que 0,2, ou seja, a detecção de comunidade não foi boa em nenhum dos métodos. Portanto, o grafo gerado neste experimento não possui uma estrutura bem definida em termos dos *clusters*.

	Número de Comunidades Detectadas	Modularidade em [0,1]	Tempo de Execução (em segundos)
Girvan-Newman	578	0,0576789	54,1900
Louvain	15	0,1999809	0,0047
Autovetores	7	0,1395753	0,0506
Passeios Aleatórios	3	0,0709824	0,0369

Tabela 5.5: *Tabela com as informações do terceiro experimento*

5.1.4 Quarto Experimento

Neste experimento, buscou-se pelos autores cujo instituto filiado continha a palavra 'MATEMATICA' e este instituto pertencesse à USP, e pelos autores que publicaram artigos nas revistas destes institutos. Em seguida, foram selecionados todos estes autores e os autores com quem eles publicaram algum outro artigo. Este subgrafo tem 994 nós e 1.151 arestas.

Como é possível ver na Tabela 5.6, os métodos de Louvain e o de autovetores têm um valor bem próximo de comunidades detectadas. Além disso, observa-se que os valores de modularidade dos 4 métodos também são bem próximos, sendo o mais baixo o de Girvan-Newman e o mais alto o de Louvain.

	Número de Comunidades Detectadas	Modularidade em [0,1]	Tempo de Execução (em segundos)
Girvan-Newman	288	0,6941664	2,1400
Louvain	291	0,7187849	0,0154
Autovetores	290	0,7100749	0,0412
Passeios Aleatórios	300	0,7151536	0,0503

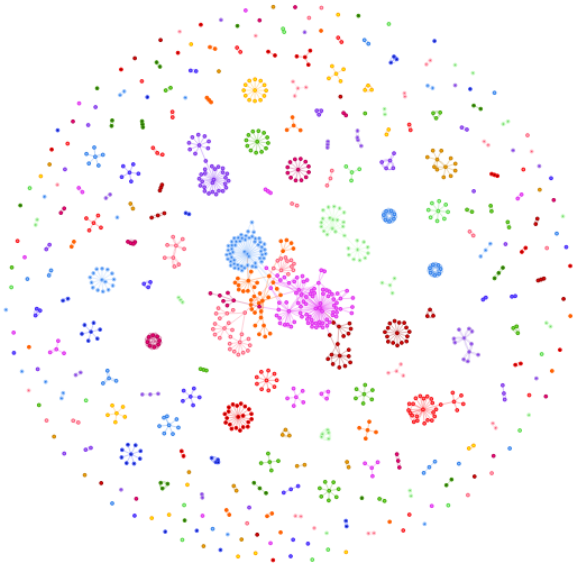
Tabela 5.6: *Tabela com as informações do quarto experimento*

	Girvan-Newman	Louvain	Autovetores	Passeios Aleatórios
Comunidade 1	81	83	85	83
Comunidade 2	55	68	68	69
Comunidade 3	35	35	35	26
Comunidade 4	35	35	35	22
Comunidade 5	35	28	28	21
Comunidade 6	28	27	27	18
Comunidade 7	27	21	21	16
Comunidade 8	21	19	17	15
Comunidade 9	17	16	16	14
Comunidade 10	16	15	15	13

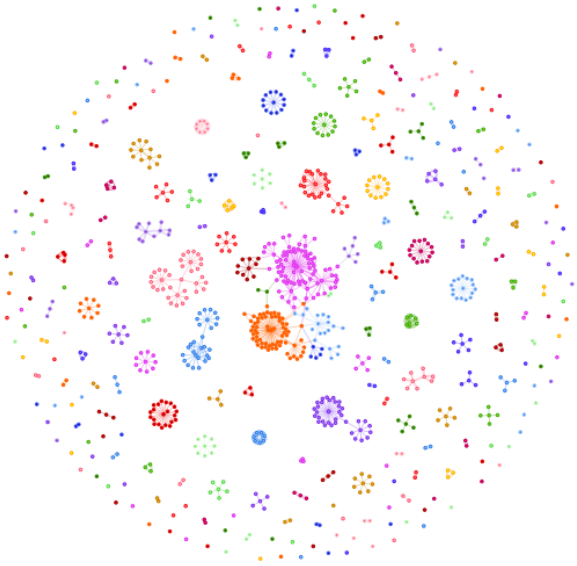
Tabela 5.7: *Tabela com as distribuições da quantidade de nós nas 10 comunidade com mais nós*

Como pode ser visto na Tabela 5.7, os 4 métodos apresentam uma distribuição bem parecida em termos da quantidade de nós em cada comunidade. E nota-se que a quantidade de nós nas comunidades detectadas pelos métodos de Louvain e de Autovetores são bem próximas.

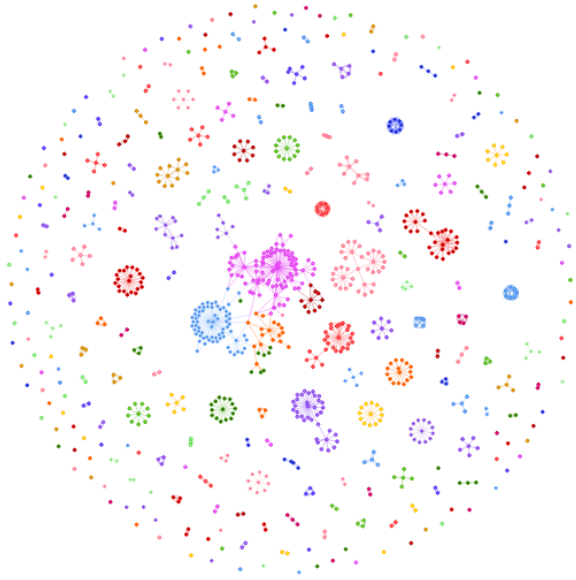
Na Figura 5.6 são apresentadas as comunidades detectadas por cada um dos métodos sobre o subgrafo deste experimento. Cada comunidade é representada por uma cor diferente.



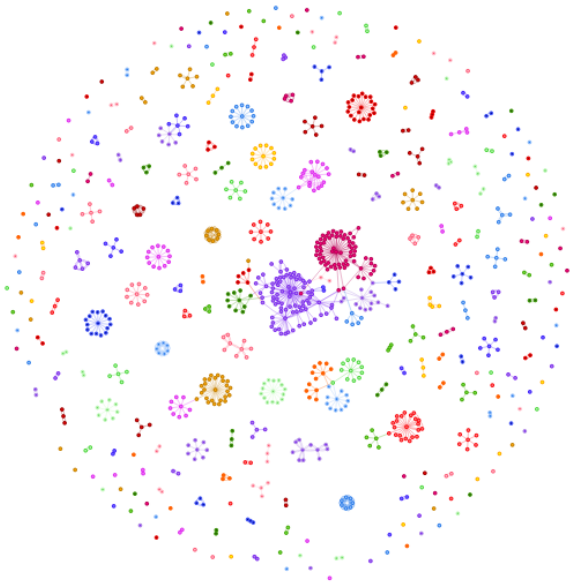
(a) *Girvan-Newman*



(b) *Louvain*



(c) *Autovalores e Autovetores*



(d) *Passeios Aleatórios*

Figura 5.6: *Resultado da aplicação dos métodos de detecção de comunidades sobre o subgrafo com 994 autores utilizado no quarto experimento*

5.1.5 Resultados

A seguir, são apresentados os gráficos dos valores de modularidade dos experimentos descritos nesta seção. O eixo x do gráfico representa o método de detecção de comunidade utilizado e o eixo y apresenta o valor da modularidade.

Como é possível ver na Figura 5.7, o método de Louvain é o que tem melhor resultado, ou seja, maior valor de modularidade. E, conforme a Tabela 5.1, tem o menor tempo de execução. Além disso, como o grafo utilizado neste experimento é denso, ou seja, tem um grande número de arestas, 503.688, o método de Girvan-Newman não tem um bom desempenho, pois a cada aresta removida é necessário re-calcular o valor de *"betweenness"* para cada aresta.

Figura 5.7: *Modularidade obtida na aplicação dos métodos de detecção de comunidade no grafo do experimento 1*

A construção utilizada no segundo experimento produz um subgrafo, com 1.100 nós e 1.118 arestas, cuja estrutura favorece a detecção de comunidades. Desta forma, todos os métodos utilizados têm um bom desempenho, como pode ser visto na Figura 5.8.

Como foi dito anteriormente, para este experimento, foram escolhidos os 10 autores que publicaram mais artigos e todos os autores com quem eles também publicaram. Verificou-se que cada um dos 10 autores inicialmente escolhidos pertence a uma comunidade distinta, o que resulta em 10 comunidades.

Figura 5.8: *Modularidade obtida na aplicação dos métodos de detecção de comunidade no grafo do experimento 2*

Relembrando a construção do subgrafo do terceiro experimento, primeiro, buscou-se pelo artigo com mais autores e, então, foram selecionados todos os autores que publicaram este artigo e os autores com quem eles publicaram algum outro artigo. Este subgrafo possui 634 nós e 4.614 arestas.

Como este subgrafo é denso, é possível ver na Tabela 5.5 que o método de Girvan-Newman é o mais demorado, 54,19 segundos, enquanto os outros métodos levam menos de 0,1 segundo.

Ainda nesta tabela, pode-se ver que o número de comunidades detectadas varia muito em todos os métodos. Além disso, é possível ver na Figura 5.9 que os valores de modularidade são menores do que 0,2, isto significa que o subgrafo associado não tem uma estrutura modular bem definida.

O problema com a estrutura deve-se ao fato dos 64 autores iniciais não pertencerem à mesma área da medicina (Cirurgia Cardiovascular, Cirurgia Plástica, Radiologia, Clínica Médica, Ortopedia, Dermatologia, entre outras) e do subgrafo ter somente 1 componente conexa, ou seja, existe um caminho entre todos os nós deste subgrafo.

Desta forma, autores de áreas diferentes da medicina estão ligados entre si devido a este artigo, o que dificulta a detecção de comunidades destes autores.

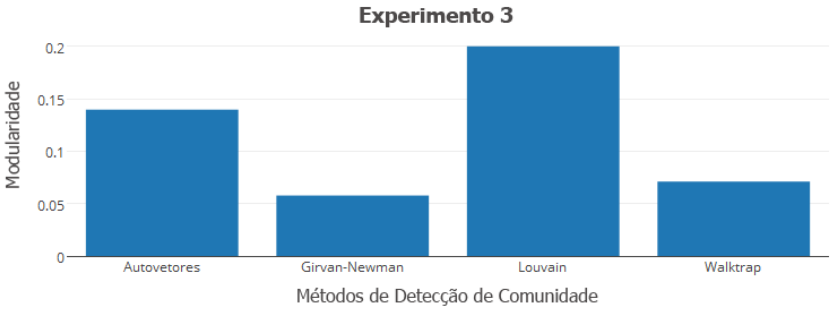


Figura 5.9: *Modularidade obtida na aplicação dos métodos de detecção de comunidade no grafo do experimento 3*

Conforme é possível ver na Figura 5.10, todos os métodos têm um bom desempenho com o subgrafo do quarto experimento, que possui 994 nós e 1.151 arestas mas em estrutura que facilita a detecção de comunidade.

Este subgrafo contém os autores que pertencem a algum instituto de matemática da USP, ou seja, contém autores das áreas de matemática, estatística e computação.

Figura 5.10: *Modularidade obtida na aplicação dos métodos de detecção de comunidade no grafo do experimento 4*

Como é possível ver em todas as tabelas e gráficos, o método de Louvain é o que tem melhor resultado do valor de modularidade e o que tem melhor desempenho em todos os experimentos. Além disso, também verificou-se que, quando o grafo é denso, o método de Girvan-Newman é o mais afetado em relação ao tempo, dada a forma como é feita a otimização da modularidade.

5.2 Visualizações dos Acessos aos Artigos

A motivação para que fossem feitas outras formas de visualização dos acessos (visualizações e *downloads*) aos artigos se deu pelo fato que no Portal de Revistas da USP a única informação disponível para os usuários é a quantidade de acessos e de *downloads* em cada artigo agrupados por mês ou ano.

Ao analisar os dados que estão presentes no banco de dados, observou-se que haviam informações relevantes que não estavam sendo utilizadas, entre elas, tem-se o nome das cidades e dos países de onde vêm os acessos.

As consultas utilizadas nesta seção podem ser vistas no Apêndice E.

5.2.1 Impactos Regionais

Para verificar a influência de um determinado artigo, autor ou revista no mundo ou em alguma região específica do Brasil, foram gerados alguns gráficos que facilitam a visualização desta influência regional.

Como o processo de geração dos gráficos é o mesmo para todos os artigos do banco de dados, os exemplos a seguir apresentam as análises dos acessos a um único artigo. O artigo escolhido, "On the qmeromorphic Weyl algebra", é o que possui maior número de acessos, ou seja, maior número de visualizações e *downloads*.

Foram identificadas um total de 19.502 acessos a este artigo, dos quais 14.583 tinham como origem cidades brasileiras e, destas cidades, muitas eram desconhecidas. A Tabela 5.8 mostra o nome das 5 cidades brasileiras de onde vieram mais acessos ao artigo selecionado.

Cidade	**Número de acessos**
Desconhecida	1.266
SAO PAULO	745
RIO DE JANEIRO	555
BELO HORIZONTE	512
BRASILIA	475

Tabela 5.8: *Tabela com nome de 5 cidades de onde mais vieram os acessos ao artigo "On the qmeromorphic Weyl algebra"*

Para visualizar os acessos ao artigo escolhido foi necessário obter as localizações geográficas das cidades brasileiras. Para isto, utilizou-se uma biblioteca do R chamada *maps*, pois ela possui uma tabela com os nomes das cidades de diversos países do mundo e suas localizações.

Após a obtenção destes dados geográficos, agrupou-se a quantidade de acessos de acordo com cada cidade brasileira. Como o mapa gerado é interativo, para verificar o nome da cidade e a quantidade de acessos vindo dela, basta passar o mouse em cima dos itens contidos no mapa. Um exemplo pode ser visto na Figura 5.11.

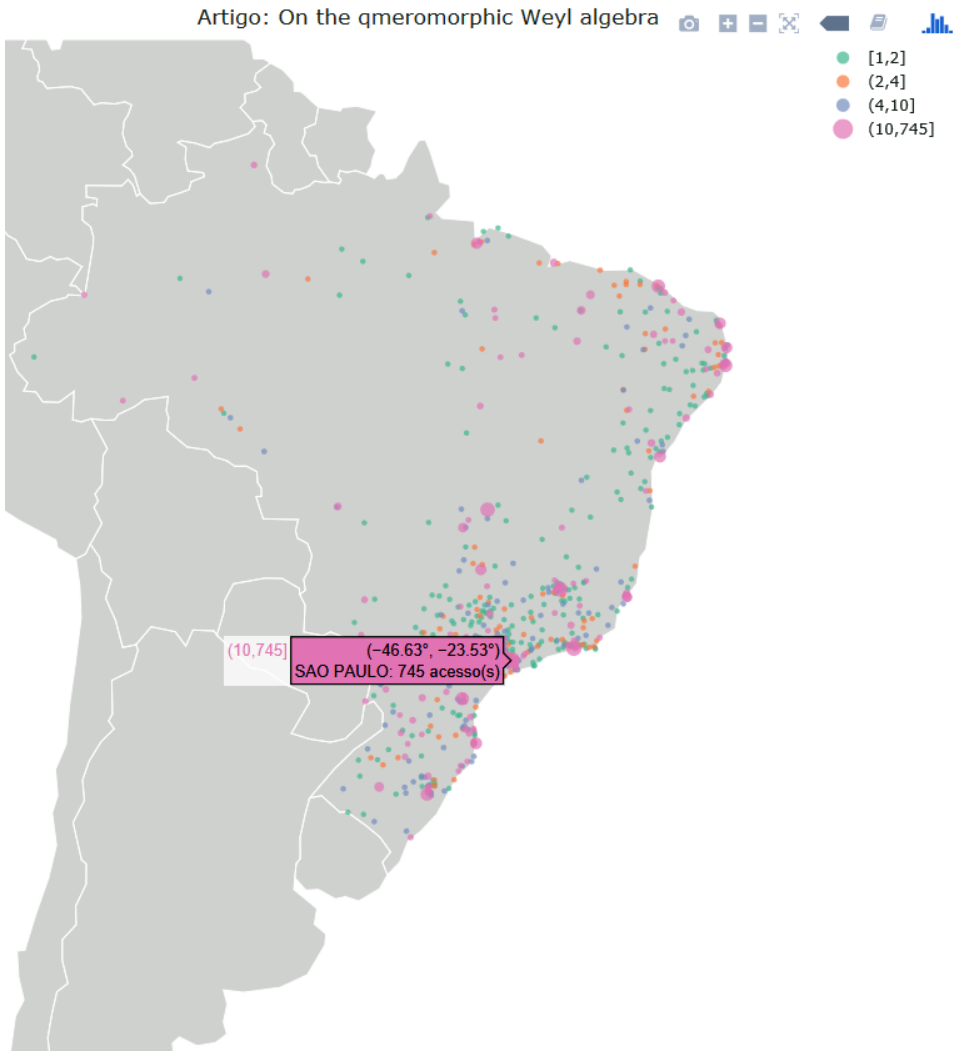


Figura 5.11: *Métricas do artigo com ID 48 nas cidades brasileiras*

Como foi dito anteriormente, uma outra visualização que pode ser feita com os dados contidos no banco de dados orientado a grafos é a dos acessos realizados em outros países aos artigos das revistas da USP. Isto permite uma análise do impacto dos trabalhos acadêmicos brasileiros no exterior.

A Tabela 5.9 contém os 8 países de onde mais vieram acessos ao artigo "On the qmeromorphic Weyl algebra", a sigla de cada um destes países e o número de acessos provenientes deles.

countryName	**country**	**count**
Brazil	BR	14.580
Argentina	AR	1.411
Angola	AO	1.329
Australia	AU	1.278
Belgium	BE	1.268
United Arab Emirates	AE	1.268
Bolivia	BO	1.265
Austria	AT	1.265

Tabela 5.9: *Tabela com o nome dos 8 países de onde mais vieram acessos ao artigo "On the qmeromorphic Weyl algebra"*

Antes de gerar o mapa mundial com as informações dos acessos, foi necessário converter a sigla de cada país para seu código de 3 dígitos correspondente, pois a função da biblioteca *plotly* que gera o mapa só identifica os países por este código.

A Figura 5.12 apresenta as informações dos acessos ao artigo "On the qmeromorphic Weyl algebra" vindo de todos países do mundo.

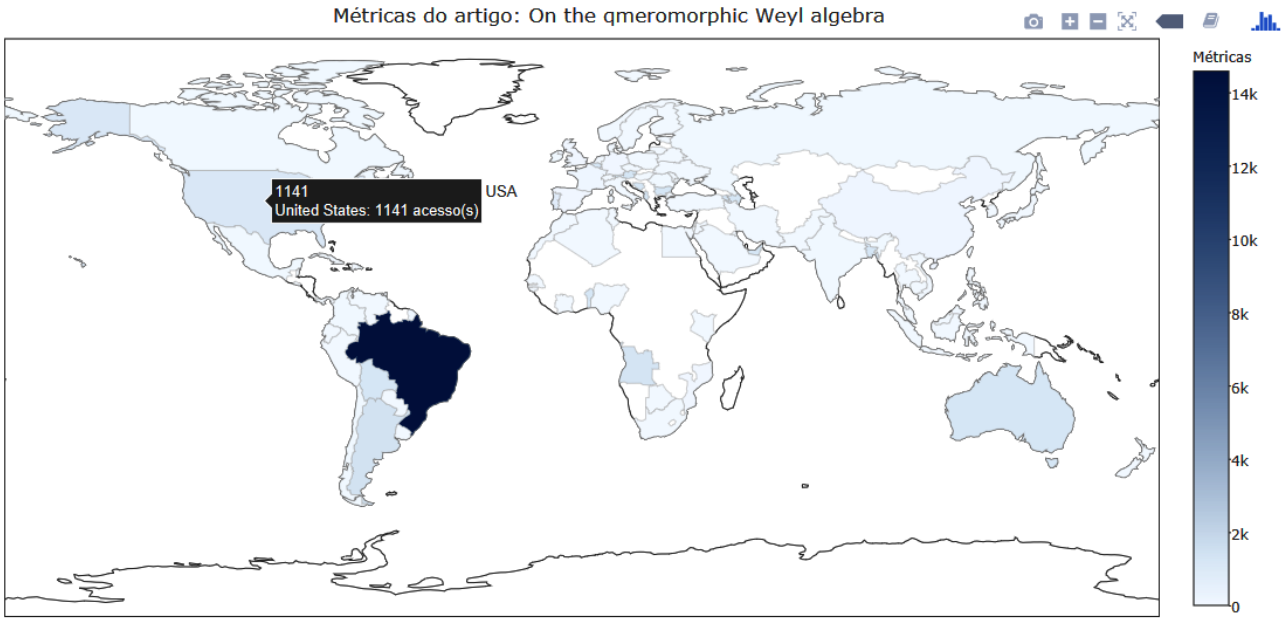


Figura 5.12: *Informações dos acessos, provenientes de todos os países, ao artigo "On the qmeromorphic Weyl algebra"*

Além das visualizações regionais dos acessos aos artigos, é possível fazer este mesmo tipo de visualização para avaliar a influência de um autor. Para isto, basta contabilizar todos os acessos a todos os artigos publicados por este autor. E, em seguida, aplicar as mesmas técnicas desta seção para gerar os mapas das quantidades de visualizações e *downloads*.

5.2.2 Quantidade dos Acessos

As informações disponibilizadas pelo Portal de Revistas da USP em seu site podem ser exploradas como mostraremos a seguir. A Figura 5.13 apresenta um gráfico da quantidade de acessos de um artigo.

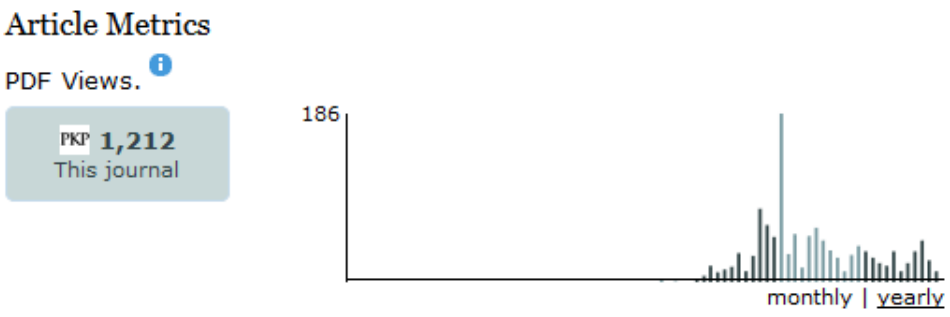


Figura 5.13: *Gráfico da quantidade de acessos de um artigo apresentado no site do Portal de Revistas da USP*

Para comparar o crescimento de visualizações e *downloads* de um determinado artigo, foram feitas análises deste artigo em relação a outros artigos, por um período de três semanas, ou seja, 21 dias.

Devido à enorme quantidade de relações existentes no banco de dados, foram comparados apenas 250 artigos. Para cada um destes artigos selecionados, contabilizou-se a quantidade de acessos (visualização e *download*) nos últimos 21 dias.

Na Figura 5.14, cada ponto representa um artigo, o eixo vertical representa a média de todos acessos nas últimas três semanas e o eixo horizontal caracteriza a porcentagem de aumento de acessos na última semana em comparação com as três últimas semanas.

Foram selecionados 3 pontos nesta figura. Ao selecionar um ponto, as informações do nome e identificador do artigo, representado por este ponto, aparecem no gráfico. Com estas informações, é possível gerar um outro gráfico para acompanhar os acessos de cada artigo, separadamente.

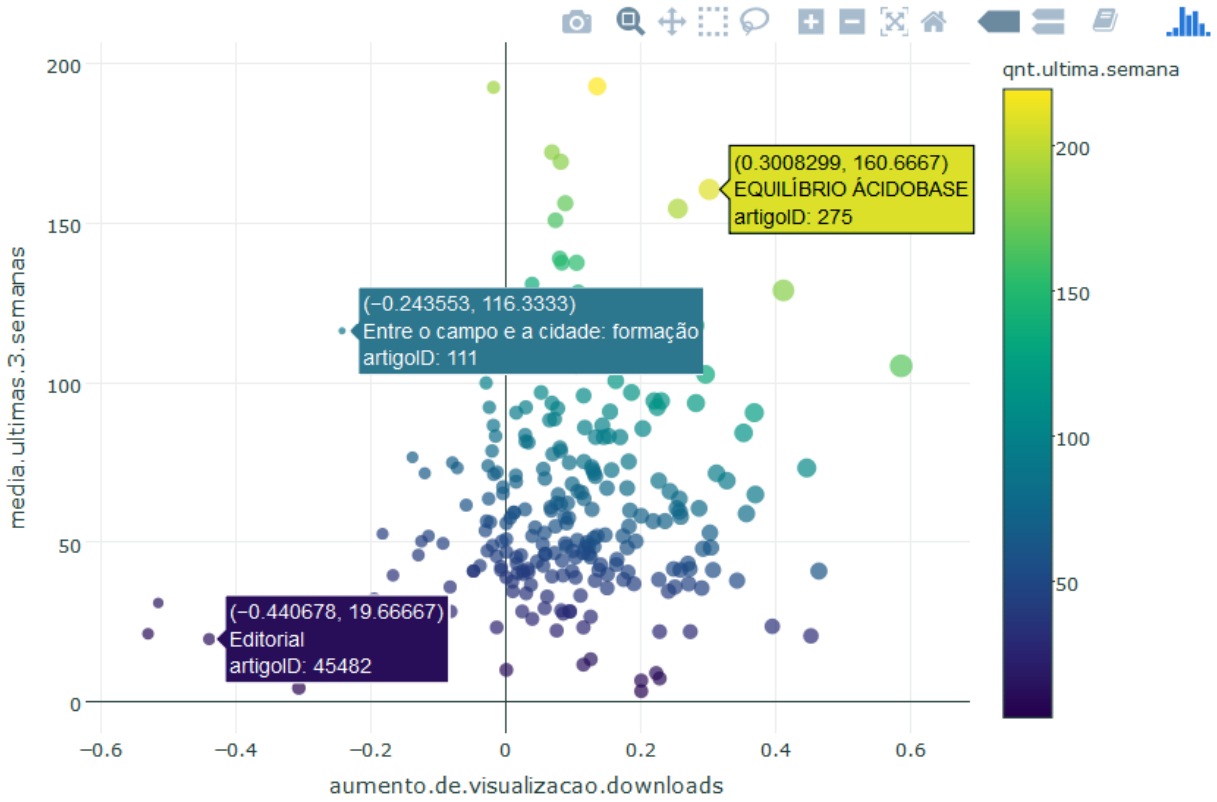


Figura 5.14: *Comparação dos acessos entre os 250 artigos selecionados, com 3 artigos escolhidos, para a visualização das informações de cada um destes 3 artigos*

As informações da quantidade de acessos realizados nas últimas 3 semanas ao artigo com identificador 275, que é representado pelo ponto amarelo selecionado na Figura 5.14, podem ser vistas na Figura 5.15.

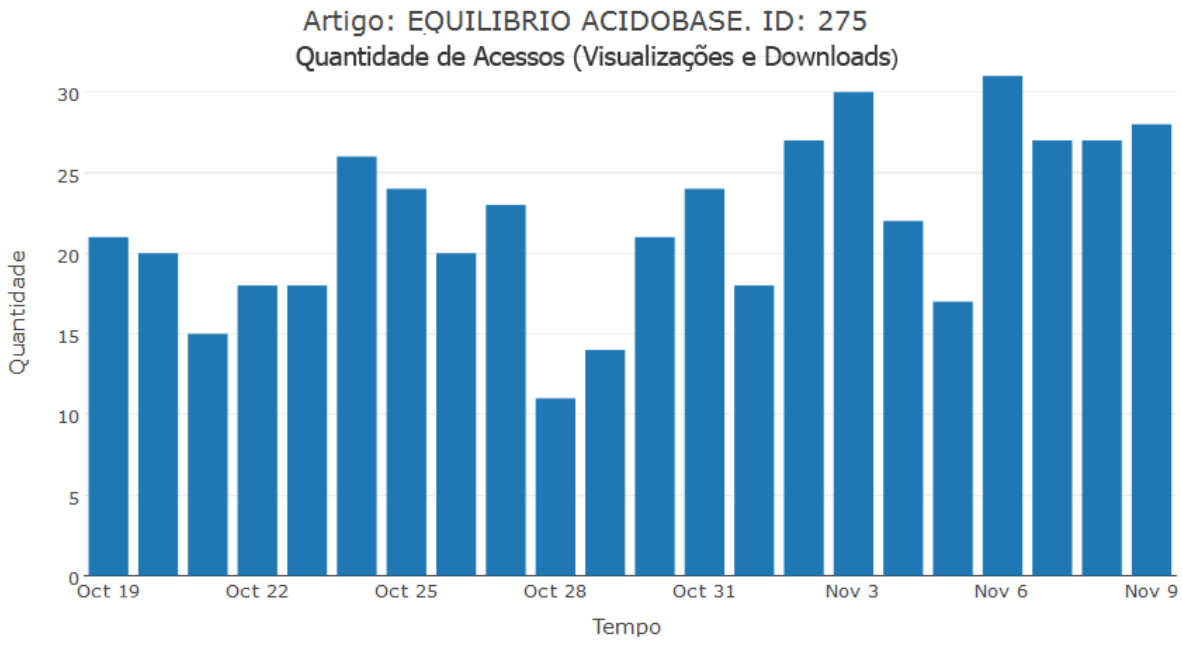


Figura 5.15: *Quantidade de acessos ao artigo cujo artigoID é 275*

As informações da quantidade de acessos, realizados nas últimas 3 semanas, ao artigo, com identificador 111, que é representado pelo ponto verde selecionado na Figura 5.14, podem ser vistas na Figura 5.16.

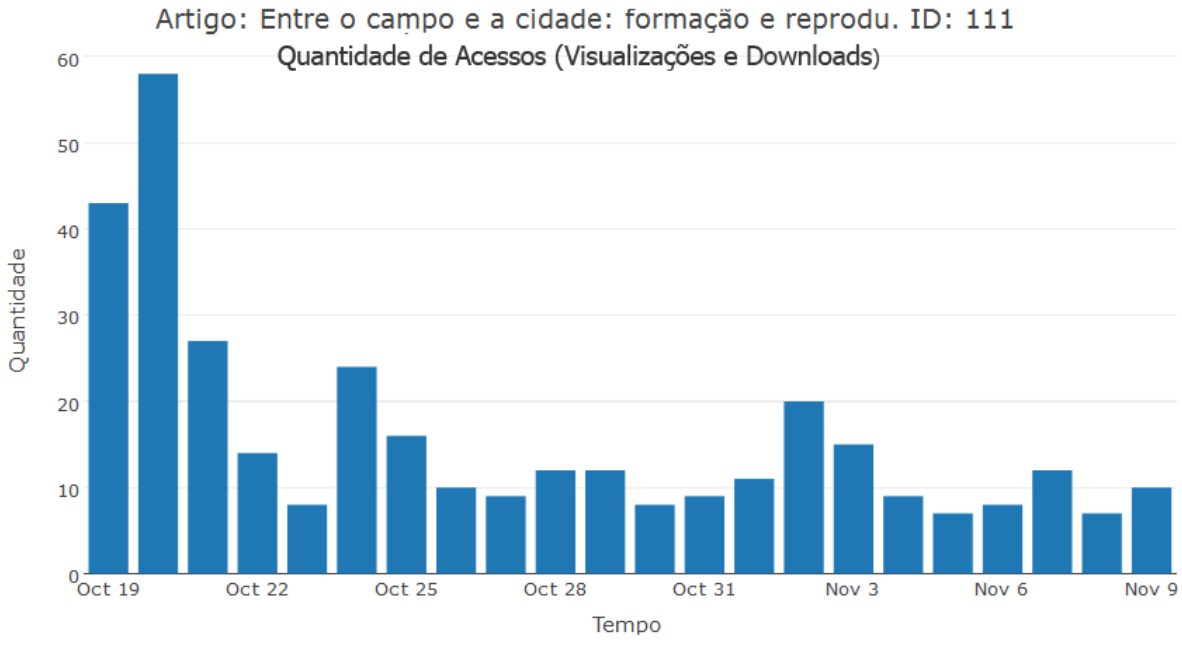


Figura 5.16: *Quantidade de acessos ao artigo cujo artigoID é 111*

As informações da quantidade de acessos realizados nas últimas 3 semanas ao artigo com identificador 45482, que é representado pelo ponto azul escuro selecionado na Figura 5.14, podem ser vistas na Figura 5.17.

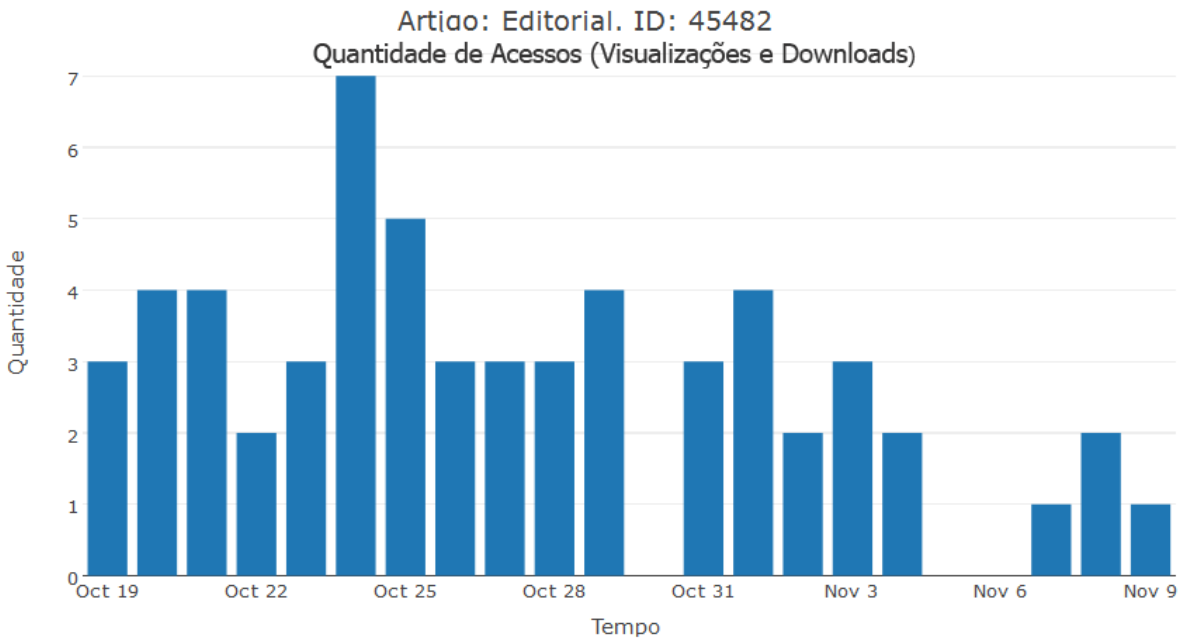


Figura 5.17: *Quantidade de acessos ao artigo cujo artigoID é 45482*

Obteve-se o resultado esperado, pois artigos que estão localizados à direita no gráfico na Figura 5.14 tiveram um aumento nos acessos, enquanto artigos que estão à esquerda deste mesmo gráfico tiveram uma diminuição de acessos na última semana em comparação às três últimas semanas da data atual.

Capítulo 6

Considerações Finais

Neste trabalho, foi proposto uma modelagem de um banco de dados orientado a grafos com o objetivo de viabilizar análises mais sofisticadas sobre os dados do Portal de Revistas da USP, como a detecção de comunidades, e de disponibilizar visualizações das informações de acesso aos artigos do portal, considerando, inclusive a geolocalização dos acessos.

Para isso, inicialmente, foi feito um estudo do banco de dados relacional do Portal de Revistas da USP e dos dados contidos nele. Em seguida, realizou-se o levantamento de requisitos do banco de dados a ser modelado, levando em consideração as análises e as visualizações desejadas.

A partir deste levantamento, modelou-se o banco de dados orientado a grafos, como descrito no Capítulo 4. Em seguida, extraíram-se os dados do banco de dados original, que tiveram que ser tratados antes de inseri-los no banco modelado, caso contrário, haveriam muitas informações com erros e dados replicados.

As análises de detecção de comunidades foram realizadas com o intuito de identificar as áreas de pesquisa dos autores e possíveis redes de colaboração entre eles, pois não é possível obter estas informações a partir do banco original. Foram estudados e utilizados quatro métodos de detecção de comunidade: Girvan-Newman, Louvain, autovalores e autovetores da matriz de modularidade e passeios aleatórios.

Com nossos experimentos, foi possível verificar que o método de Girvan-Newman não tem bom desempenho quando o grafo é denso, enquanto que o método de Louvain é o que tem melhor desempenho em geral. Além disso, quando o grafo tem uma estrutura bem definida, o resultado de todos os métodos é bem similar quanto ao valor da modularidade e é possível detectar as áreas de pesquisa dos autores.

Ressalta-se que, de fato, o modelo de dados orientado a grafos facilitou a aplicação dos métodos de detecção de comunidades, conforme afirmado no início deste trabalho. A obtenção dos grafos sobre os quais os algoritmos de detecção são executados é praticamente direta a partir do banco.

Por fim, os gráficos dos acessos a um artigo facilitam a visualização da difusão e do impacto de artigos em diversos níveis (regional, nacional e internacional). Novamente, o modelo de dados de grafo e as ferramentas de software disponíveis tornaram esta tarefa muito mais simples.

6.1 Trabalhos Futuros

Como trabalhos futuros, é possível realizar análises temporais de detecção de comunidade, ou seja, aplicar outros métodos que permitem a classificação e a predição de autores ou artigos por ano.

Além disso, também pode-se aplicar análises sobre os artigos utilizando técnicas da área de recuperação de informação. Uma destas técnicas é a utilização das palavras-chave dos artigos, informação que já está disponível no banco de dados de grafos modelado. A Figura 6.1 mostra as palavras que mais aparecem entre as palavras-chave dos artigos.

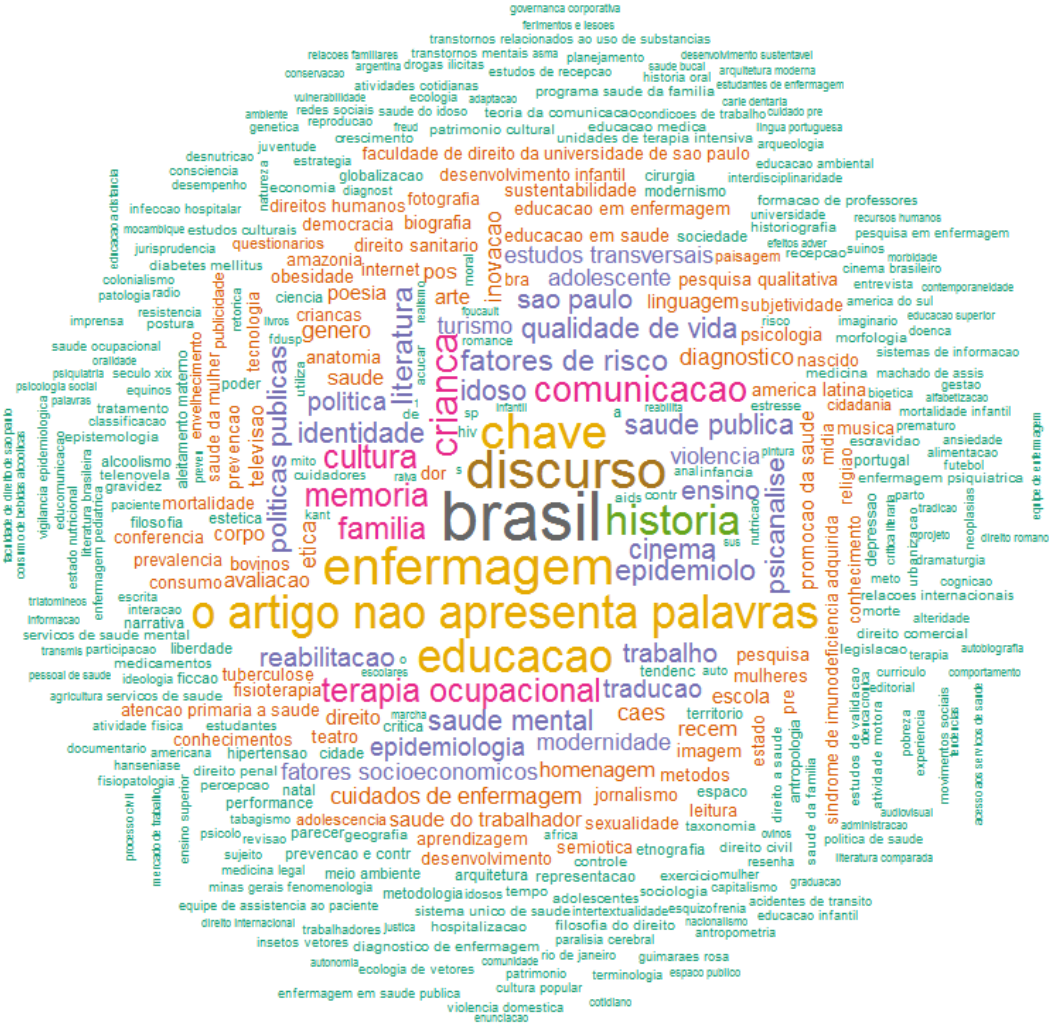


Figura 6.1: *Worldcloud das palavras-chaves dos artigos*

Apêndice A

Tabelas do Banco de Dados do Portal de Revistas da USP

As tabelas do banco de dados do Portal de Revistas da USP que utilizamos neste trabalho foram: *articles*, *article_settings*, *authors*, *author_settings*, *journals*, *journal_settings* e *metrics*.

- articles

Coluna	**Tipo**
article_id	integer
locale	character varying(5)
user_id	bigint
journal_id	bigint
section_id	bigint
language	character varying(10)
comments_to_ed	text
citations	text
date_submitted	timestamp without time zone
last_modified	timestamp without time zone
date_status_modified	timestamp without time zone
status	smallint
submission_progress	smallint
current_round	smallint
submission_file_id	bigint
revised_file_id	bigint
review_file_id	bigint
editor_file_id	bigint
pages	character varying(255)
fast_tracked	smallint
hide_author	smallint
comments_status	smallint

Tabela A.1: *Tabela "articles"*

Índices:

"articles_pkey" PRIMARY KEY, btree (article_id)

"articles_journal_id" btree (journal_id)

"articles_section_id" btree (section_id)

"articles_user_id" btree (user_id)

- article_settings

Coluna	Tipo
article_id	bigint
locale	character varying(5)
setting_name	character varying(255)
setting_value	text
setting_type	character varying(6)

Tabela A.2: *Tabela "article_settings"*

Índices:

"article_settings_pkey" UNIQUE, btree (article_id, locale, setting_name)

"article_settings_article_id" btree (article_id)

"article_settings_name_value" btree (setting_name, setting_value) WHERE setting_name::text = ANY (ARRAY['indexingState'::character varying, 'medra::registeredDoi'::character varying, 'datacite::registeredDoi'::character varying]::text[])

- journals

Coluna	Tipo
journal_id	integer
path	character varying(32)
seq	double precision
primary_locale	character varying(5)
enabled	smallint

Tabela A.3: *Tabela "journals"*

Índices:

"journals_pkey" PRIMARY KEY, btree (journal_id)

"journals_path" UNIQUE, btree (path)

- journal_settings

Coluna	Tipo
journal_id	bigint
locale	character varying(5)
setting_name	character varying(255)
setting_value	text
setting_type	character varying(6)

Tabela A.4: *Tabela "journal_settings"*

Índices:

"journal_settings_pkey" UNIQUE, btree (journal_id, locale, setting_name)

"journal_settings_journal_id" btree (journal_id)

- metrics

Coluna	**Tipo**
load_id	character varying(255)
assoc_type	bigint
context_id	bigint
issue_id	bigint
submission_id	bigint
assoc_id	bigint
day	character varying(8)
month	character varying(6)
file_type	smallint
country_id	character varying(2)
region	character varying(2)
city	character varying(255)
metric_type	character varying(255)
metric	integer

Tabela A.5: *Tabela "metrics"*

Índices:

"metrics_load_id" btree (load_id)

"metrics_metric_type_assoc_type_submission_id" btree (metric_type, assoc_type, submission_id)

"metrics_metric_type_context_id_assoc_type" btree (metric_type, context_id, assoc_type)

"metrics_metric_type_journal_id" btree (metric_type, context_id)

- authors

Coluna	**Tipo**
author_id	integer
submission_id	bigint
primary_contact	smallint
seq	double precision
first_name	character varying(40)
middle_name	character varying(40)
last_name	character varying(90)
suffix	character varying(40)
country	character varying(90)
email	character varying(90)
url	character varying(255)
user_group_id	bigint

Tabela A.6: *Tabela "authors"*

Índices:

"authors_pkey" PRIMARY KEY, btree (author_id)

"authors_submission_id" btree (submission_id)

- author_settings

Coluna	**Tipo**
author_id	bigint
locale	character varying(5)
setting_name	character varying(255)
setting_value	text
setting_type	character varying(6)

Tabela A.7: *Tabela "author_ settings"*

Índices:

"author_settings_pkey" UNIQUE, btree (author_id, locale, setting_name)

"author_settings_author_id" btree (author_id)

Apêndice B

Código para de Extração dos Dados do Banco do Portal de Revistas da USP

Para extrair os dados do banco de dados do Portal de Revistas da USP foram realizadas consultas SQL.

B.1 Artigos

Para artigos, foram necessárias três consultas:

- *setting_name* = 'cleanTitle', *locale* = 'pt_BR'

```
1 \copy (
2   SELECT distinct(art.article_id), journal_id, date_submitted,
        setting_value AS article_title
3     FROM articles art JOIN article_settings arts
4       ON arts.article_id = art.article_id
5       WHERE setting_name = 'cleanTitle' AND arts.locale = 'pt_BR'
6 ) TO './articlesTitle.csv' DELIMITER ',' CSV HEADER;
```

- *setting_name* = 'cleanTitle', *locale* = 'en_US'

```
1 \copy (
2   SELECT DISTINCT(article_id), setting_value AS article_titleEN
3       FROM article_settings
4       WHERE setting_name = 'cleanTitle' AND locale = 'en_US' AND
5             setting_value <> '' AND setting_value <> 'TITULO NULO'
6 ) TO './articlesTitleEN.csv' DELIMITER ',' CSV HEADER;
```

- *setting_name* = 'subject'

```
1 \copy (
2   SELECT distinct(art.article_id), setting_value AS article_keywords
3     FROM articles art JOIN article_settings arts
4       ON arts.article_id = art.article_id
5       WHERE setting_name = 'subject' AND arts.locale = 'pt_BR'
6 ) TO './articlesKeywords.csv' DELIMITER ',' CSV HEADER;
```

- *setting_name* = 'abstract'

```
\copy (
  SELECT distinct(art.article_id), setting_value AS article_abstract
    FROM articles art JOIN article_settings arts
      ON arts.article_id = art.article_id
        WHERE setting_name = 'abstract' AND arts.locale = 'pt_BR')
  TO './articlesAbstract.csv' DELIMITER ',' CSV HEADER;
```

- *setting_name* = 'pub-id::doi'

```
\copy (
  SELECT distinct(art.article_id), setting_value AS article_doi
    FROM articles art JOIN article_settings arts
      ON arts.article_id = art.article_id
        WHERE setting_name = 'pub-id::doi')
  TO './articlesDoi.csv' DELIMITER ',' CSV HEADER;
```

B.2 Autores

Para autores só utilizamos uma consulta, pois só foi utilizado um valor de *setting_name*:

- *setting_name* = 'affiliation'

```
\copy (
  SELECT distinct(aut.author_id), submission_id, first_name::text ||
      ' ' || middle_name::text || ' ' || last_name::text AS
      author_name, country, setting_value AS affiliation
    FROM authors aut JOIN author_settings auts
      ON auts.author_id = aut.author_id
        WHERE setting_name = 'affiliation')
  TO './authorsInfo.csv' DELIMITER ',' CSV HEADER;
```

B.3 Métricas

Na tabela *metrics*, existem 25814022 registros. Inicialmente, extraíram-se todos os registros da tabela *metrics* com metric_type = 'ojs:counter'. Mas inserir metade destes dados no Neo4j demorava mais de 30 minutos. Então, como a quantidade de registros era muito grande, selecionou-se os 250 artigos com o maior número de métricas.

A *query* utilizada para extrair os dados das métricas é composta por três consultas encadeadas. A *query* mais interna, em que são selecionados os *article_id* distintos da tabela *articles*, é necessária devido ao fato do banco de dados do Portal de Revistas da USP não possuir integridade entre os dados, ou seja, não possuem chaves estrangeiras. Assim quando um artigo é excluído da tabela *articles* do banco de dados, as informações relacionadas a este artigo em outras tabelas são mantidas.

```
\copy (
  SELECT assoc_id, day, country_id, city, assoc_type, metric_type
    FROM metrics
      WHERE assoc_id IN (
        SELECT assoc_id
        FROM metrics
        WHERE assoc_id IN (
          SELECT DISTINCT article_id
          FROM articles)
```

```
10          GROUP BY assoc_id
11          ORDER BY count(*) DESC
12          LIMIT 250))
13   TO './metricsInfo.csv' DELIMITER ',' CSV HEADER;
```

B.4 Revistas

Para revistas, utilizamos quatro consultas pois usamos quatro valores de *setting_name*:

- *setting_name* = 'title'

```
1 \copy (
2   SELECT distinct(j.journal_id), setting_value AS journal_title
3     FROM journals j JOIN journal_settings js
4       ON js.journal_id = j.journal_id
5         WHERE setting_name = 'title' AND locale='pt_BR'
6         ORDER BY j.journal_id)
7   TO './journalsTitle.csv' DELIMITER ',' CSV HEADER;
```

- *setting_name* = 'searchKeywords'

```
1 \copy (
2   SELECT DISTINCT(j.journal_id), setting_value AS journal_keywords
3     FROM journals j JOIN journal_settings js
4       ON js.journal_id = j.journal_id
5         WHERE setting_name = 'searchKeywords' AND locale='pt_BR'
6         ORDER BY j.journal_id)
7   TO './journalsKeywords.csv' DELIMITER ',' CSV HEADER;
```

- *setting_name* = 'searchDescription'

```
1 \copy (
2   SELECT distinct(j.journal_id), setting_value AS journal_description
3     FROM journals j JOIN journal_settings js
4       ON js.journal_id = j.journal_id
5         WHERE setting_name = 'searchDescription'
6         ORDER BY j.journal_id)
7   TO './journalsDescription.csv' DELIMITER ',' CSV HEADER;
```

- *setting_name* = 'publisherInstitution'

```
1 \copy (
2   SELECT distinct(j.journal_id), setting_value AS journal_institution
3     FROM journals j JOIN journal_settings js
4       ON js.journal_id = j.journal_id
5       WHERE setting_name = 'publisherInstitution'
6       ORDER BY j.journal_id)
7   TO './journalsInstituion.csv' DELIMITER ',' CSV HEADER;
```

Apêndice C

Código para Limpeza dos Dados

C.1 Arquivo articleAbstract.csv

```
1 abs <- read.csv('articlesAbstract.csv'  , sep = ',', encoding='UTF-8')
2
3 abs$article_abstract <- gsub("<[^>]*>","",abs$article_abstract)
4 abs$article_abstract <- to.plain(abs$article_abstract)
5
6 abs$article_abstract <- gsub("[\r\t]", "", abs$article_abstract)
7 abs$article_abstract <- gsub("[\r\n]", "", abs$article_abstract)
8 abs$article_abstract <- gsub(" [ ]+", " ", abs$article_abstract)
9
10 abs$article_abstract <- gsub('"', '', abs$article_abstract)
11 abs$article_abstract <- gsub("^nulo$", '', abs$article_abstract)
```

C.2 Arquivo authorsInfo.csv

```
1 home = '/'
2 author <- read.csv(paste0(home, "authorsInfo.csv")  , sep = ",", encoding=
      "UTF-8")
3 uni    <- read.csv(paste0(home, "tab_universidades.txt"), sep = "\t")
4
5 uni$Nome <- to.plain(uni$Nome)
6 paises <- author
7
8 # affiliation
9 #
10 ##################################################
11 paises$affiliation <- gsub("[\r\n]", "; ", paises$affiliation)
12
13 yo_backup <- data.frame(paises$affiliation)
14 yo <- yo_backup
15
16 yo$paises.affiliation <- gsub('[^[:alnum:] ;,().]', "", yo$paises.
      affiliation)
17 yo$paises.affiliation <- to.plain(yo$paises.affiliation)
18 yo$paises.affiliation <- toupper(yo$paises.affiliation)
19
20 # universidade
21 #
22 ##################################################
```

```
23 yo$paises.affiliation <- gsub("^USP$", "UNIVERSIDADE DE SAO PAULO", yo$
     paises.affiliation)
24 yo$paises.affiliation <- gsub("UNIVERSIDADE DE SAO PAULO \\(USP\\)", "
     UNIVERSIDADE DE SAO PAULO", yo$paises.affiliation)
25 yo$paises.affiliation <- gsub(", SAO PAULO, SP", "", yo$paises.affiliation
     )
26
27 yo$Universidade <- "Desconhecido"
28 yo$Instituto <- "Desconhecido"
29
30 yo[c(grep(" UNIVERSITY;" , yo$paises.affiliation)), 2] <- gsub("(.*
     UNIVERSITY);(.*)"                ,"\\1", yo[c(grep(" UNIVERSITY;" , yo$
     paises.affiliation)), 1])
31 yo[c(grep("UNIVERSITY"    , yo$paises.affiliation)), 2] <- gsub("(.*)(
     UNIVERSITY [^;,\\(\\.]*)(.*)"  ,"\\2", yo[c(grep("UNIVERSITY"    , yo$
     paises.affiliation)), 1])
32 yo[c(grep("UNIVERSIDAD " , yo$paises.affiliation)), 2] <- gsub("(.*)(
     UNIVERSIDAD [^;,\\(\\.]*)(.*)" ,"\\2", yo[c(grep("UNIVERSIDAD " , yo$
     paises.affiliation)), 1])
33 yo[c(grep("UNIVERSIDADE ", yo$paises.affiliation)), 2] <- gsub("(.*)(
     UNIVERSIDADE [^;,\\(\\.]*)(.*)","\\2", yo[c(grep("UNIVERSIDADE ", yo$
     paises.affiliation)), 1])
34
35 yo[c(grep("DEPARTAMENTO ", yo$paises.affiliation)), 3] <- gsub("(.*)(
     DEPARTAMENTO [^;,\\(]*)(.*)"   ,"\\2", yo[c(grep("DEPARTAMENTO ", yo$
     paises.affiliation)), 1])
36 yo[c(grep("DEPT"          , yo$paises.affiliation)), 3] <- gsub("(.*)(DEPT
     [^;,\\(]*)(.*)"              ,"\\2", yo[c(grep("DEPT"          , yo$paises
     .affiliation)), 1])
37 yo[c(grep("DEPARTMENT "  , yo$paises.affiliation)), 3] <- gsub("(.*)(
     DEPARTMENT [^;,\\(]*)(.*)"     ,"\\2", yo[c(grep("DEPARTMENT "  , yo$
     paises.affiliation)), 1])
38
39 yo[c(grep("COLLEGE "  , yo$paises.affiliation)), 3] <- gsub("(.*)(COLLEGE
     [^;,\\(\\.]*)(.*)"  ,"\\2", yo[c(grep("COLLEGE "  , yo$paises.
     affiliation)), 1])
40 yo[c(grep("SCHOOL "   , yo$paises.affiliation)), 3] <- gsub("(.*)(SCHOOL
     [^;,\\(\\.]*)(.*)"   ,"\\2", yo[c(grep("SCHOOL "   , yo$paises.
     affiliation)), 1])
41 yo[c(grep("ESCOLA "   , yo$paises.affiliation)), 3] <- gsub("(.*)(ESCOLA
     [^;,\\(\\.]*)(.*)"   ,"\\2", yo[c(grep("ESCOLA "   , yo$paises.
     affiliation)), 1])
42 yo[c(grep("FACULDADE ", yo$paises.affiliation)), 3] <- gsub("(.*)(
     FACULDADE [^;,\\(\\.]*)(.*)","\\2", yo[c(grep("FACULDADE ", yo$paises.
     affiliation)), 1])
43 yo[c(grep("INSTITUTO ", yo$paises.affiliation)), 3] <- gsub("(.*)(
     INSTITUTO [^;,\\(\\.]*)(.*)","\\2", yo[c(grep("INSTITUTO ", yo$paises.
     affiliation)), 1])
44 yo[c(grep("FACUL"       , yo$paises.affiliation)), 3] <- gsub("(.*)(FACUL
     [^;,\\(]*)(.*)"          ,"\\2", yo[c(grep("FACUL"       , yo$paises.
     affiliation)), 1])
45 yo[c(grep("INST"        , yo$paises.affiliation)), 3] <- gsub("(.*)(INST
     [^;,\\(]*)(.*)"          ,"\\2", yo[c(grep("INST"        , yo$paises.
     affiliation)), 1])
46
47 yo[grep("UNIVERSITY OF SAO PAULO[;., ]"  , yo$paises.affiliation), '
     Universidade'] <- "UNIVERSIDADE DE SAO PAULO"
48 yo[grep("UNIVERSITY OF SAO PAULO"         , yo$Universidade)       , '
     Universidade'] <- "UNIVERSIDADE DE SAO PAULO"
```

```
49 yo[grep("FEDERAL UNIVERSITY OF SAO PAULO", yo$paises.affiliation), '
     Universidade'] <- "UNIVERSIDADE FEDERAL DE SAO PAULO"
50
51 n <- nrow(uni)
52 for (i in 1:n){
53   yo[grep(paste0("[^[:alnum:]]", uni[i, 2], "[^[:alnum:]]"), yo$paises.
       affiliation), 'Universidade'] <- uni[i, 1]
54   yo[grep(paste0("^", uni[i, 2], "[^[:alnum:]]"), yo$paises.affiliation),
       'Universidade'] <- uni[i, 1]
55   yo[grep(paste0("[^[:alnum:]]", uni[i, 2], "$"), yo$paises.affiliation),
       'Universidade'] <- uni[i, 1]
56   yo[grep(paste0("^", uni[i, 2], "$"), yo$paises.affiliation), '
       Universidade'] <- uni[i, 1]
57 }
58
59 yo[grep("UNESP[;., ]", yo$paises.affiliation), 'Universidade'] <- "
     UNIVERSIDADE ESTADUAL PAULISTA JULIO DE MESQUITA FILHO"
60 yo[grep("UNESP"       , yo$paises.affiliation), 'Universidade'] <- "
     UNIVERSIDADE ESTADUAL PAULISTA JULIO DE MESQUITA FILHO"
61 yo[grep("MESQUITA"    , yo$paises.affiliation), 'Universidade'] <- "
     UNIVERSIDADE ESTADUAL PAULISTA JULIO DE MESQUITA FILHO"
62
63 yo_backup2 <- yo
64
65 # Copia o affiliation para todos os que forem iguais a "" em universidade
66 yo[!(yo$paises.affiliation == "" | is.na(yo$paises.affiliation)) & yo$
     Universidade == "Desconhecido", 2] <- yo[!(yo$paises.affiliation == ""
     | is.na(yo$paises.affiliation)) & yo$Universidade == "Desconhecido", 1]
67
68 author <- select(author, -affiliation)
69
70 author$university <- yo$Universidade
71 author$institute <- yo$Instituto
72
73 author$university <- gsub("^\\s+|\\s+$", "", author$university)
74 author$institute  <- gsub("^\\s+|\\s+$", "", author$institute)
75
76 # author_name
77
78 author$author_name <- gsub("[\r\t]", "", author$author_name)
79 author$author_name <- gsub("[\r\n]", "", author$author_name)
80 author$author_name <- gsub("  ", " ", author$author_name)
81 author$author_name <- gsub("^\\s+|\\s+$", "", author$author_name)
82
83 author$author_name <- gsub("<[^>]*>"        , "" , author$author_name)
84 author$author_name <- gsub("[^[:alpha:]. ]", "" , author$author_name)
85 author$author_name <- gsub("  "             , " ", author$author_name)
86 author$author_name <- gsub("^\\s+|\\s+$"    , "" , author$author_name)
87
88 # country
89
90 yo_backup <- data.frame(paises$country)
91 yo <- yo_backup
92
93 yo$paises.country     <- gsub("[\r\n]", "", yo$paises.country)
94 yo$paises.country     <- gsub("[\r\t]", "", yo$paises.country)
95
96 Encoding(yo$paises.country) <- "latin1"
97
```

```
yo$paises.country <- to.plain(yo$paises.country)
yo$paises.country <- tolower(yo$paises.country)
yo$paises.country <- gsub("the", "", yo$paises.country)

yo$paises.country <- gsub("[^[:alnum:] ]", "", yo$paises.country)
yo$paises.country <- gsub("brasil", "brazil", yo$paises.country)
yo$paises.country <- gsub("^\\s+|\\s+$", "", yo$paises.country)
# yo[is.na(yo$paises.country), 'paises.country'] <- "Desconhecido"

paises_estranhos <- read.csv(paste0(home, "tab_countryArranged.txt"), sep
    = ",", encoding="UTF-8",stringsAsFactors=FALSE)
paises_corretos  <- read.csv(paste0(home, "tab_countries.txt"), sep = ",",
     encoding="UTF-8",stringsAsFactors=FALSE)
names(paises_estranhos) <- c("Name","Code")
names(paises_corretos) <- c("Name","Code")
paises <- rbind(paises_corretos,paises_estranhos)

paises$Name <- to.plain(paises$Name)
paises$Name <- tolower(paises$Name)
paises$Code <- tolower(paises$Code)
paises$Name <- gsub("the", "", paises$Name)
paises$Name <- gsub("[^[:alnum:] ]", "", paises$Name)
paises$Name <- gsub("brasil", "brazil", paises$Name)
paises$Name <- gsub("^\\s+|\\s+$", "", paises$Name)

yo$country <- NA

n <- nrow(paises)
for (i in 1:n){
  yo[grep(paste0("^", paises[i, 2], "$"), yo$paises.country), 'country']
      <- paises[i, 2]
  yo[grep(paste0("^", paises[i, 1], "$"), yo$paises.country), 'country']
      <- paises[i, 2]
}

yo$country <- toupper(yo$country)
author$country <- yo$country

con = file("vf_authorsInfo.csv", encoding = "UTF-8")
write.csv(author, file = con, row.names=FALSE)
```

C.3 Arquivo metricsInfo.csv

Código usado para a limpeza e padronização do arquivo metricsInfo.csv e a criação de dois arquivos, um contendo as informações das métricas de *download* e o outro com as de visualizações.

```
library(stringdist)

home = '/'
metrics <- read.csv(paste0(home, 'metricsInfo.csv'),
                    sep = ',',
                    encoding='UTF-8',
                    stringsAsFactors = FALSE)

# metric_type

```

```
11 metrics <- metrics[metrics$metric_type == 'ojs::counter', ]
12 metrics <- select(metrics, -metric_type)
13
14 # city
15
16 normaliza_cidade <- function(x) {
17   dfb$city[amatch(x, dfb$city, maxDist = 2)]}
18
19 dfb <- world.cities[world.cities$country.etc=="Brazil",]
20 dfb <- select(dfb, name, lat, long)
21 dfb$name <- toupper(dfb$name)
22 names(dfb) <- c("city", "lat", "long")
23
24 metrics$city <- to.plain(metrics$city)
25 metrics$city <- toupper(metrics$city)
26 metrics$city <- gsub('[^[:alnum:] ]', '', metrics$city)
27 metrics$city <- sapply(metrics$city, normaliza_cidade)
28
29 metrics$country_code <- "Desconhecido"
30 metrics$city_name    <- "Desconhecida"
31
32 metrics[!(metrics$city == "" | is.na(metrics$city)), 'city_name']
33   <- metrics[!(metrics$city == "" | is.na(metrics$city)), 'city']
34 metrics[!(metrics$country_id == "" | is.na(metrics$country_id)),
35         'country_code'] <- metrics[!(metrics$country_id == "" |
36                                        is.na(metrics$country_id)),
37                                      'country_id']
38
39 metrics$country_id <- gsub('A[12]', '', metrics$country_id)
40 metrics <- select(metrics, -city, -country_id)
41
42 # date
43
44 metrics$year  <- metrics$day
45 metrics$month <- metrics$day
46
47 metrics[, 'year'] <-
48   as.numeric(gsub("([0-9]{4})([0-9]{4})","\\1",metrics$year))
49 metrics[, 'month'] <-
50   as.numeric(gsub("([0-9]{4})([0-9]{2})([0-9]{2})","\\2",metrics$month))
51 metrics[, 'day'] <-
52   as.numeric(gsub("([0-9]{4})([0-9]{2})([0-9]{2})","\\3",metrics$day))
53
54 # assoc_type
55
56 d <- metrics[metrics$assoc_type == 260, ]
57 v <- metrics[metrics$assoc_type != 260, ]
58
59 d <- select(d, -assoc_type)
60 v <- select(v, -assoc_type)
61
62 write.csv(d, 'vf_downloadInfo.csv', row.names = FALSE)
63 write.csv(v, 'vf_visualizInfo.csv', row.names = FALSE)
```

Apêndice D

Código para Detecção de Comunidade

Para realizar as análises de detecção de comunidade sobre os dados contido no banco Neo4j foi utilizado R.

Conexão com o Neo4j:

```
1 library(RNeo4j)
2 library(igraph)
3 library(visNetwork)
4
5 neo4j = startGraph("http://localhost:7474/db/data/", username = "neo4j",
      password = "")
```

D.1 Primeiro Experimento

Para gerar o grafo completo:

```
1 nodes_query = "
2 MATCH (:Artigo)-[:FOI_ESCRITO_POR]->(a:Autor)
3 RETURN DISTINCT ID(a) AS id, a.nomeAutor AS label
4 "
5
6 edges_query = "
7 MATCH (a1:Autor)<-[:FOI_ESCRITO_POR]-(:Artigo)-[:FOI_ESCRITO_POR]->(a2:
      Autor)
8 RETURN ID(a1) AS from, ID(a2) AS to, COUNT(*) AS weight
9 "
10
11 nodes = cypher(neo4j, nodes_query)
12 edges = cypher(neo4j, edges_query)
13
14 g = graph_from_data_frame(edges, directed=F, vertices=nodes)
```

D.2 Segundo Experimento

Para gerar o subgrafo do segundo experimento:

Primeiro, selecionamos os identificadores dos 10 autores que publicaram mais artigos:

```
1 MATCH (:Artigo)-[r:FOI_ESCRITO_POR]->(a:Autor)
2 RETURN ID(a), a.nomeAutor AS label, count(r) AS cnt  ORDER BY cnt DESC
      LIMIT 10;
```

```
nodes_query1 = "
MATCH (:Artigo)-[:FOI_ESCRITO_POR]->(a:Autor)
WHERE ID(a) IN [90840, 90726, 98124, 97748, 105573, 98122, 160657, 91622,
    92232, 113381]
RETURN DISTINCT ID(a) AS id, a.nomeAutor AS label ORDER BY id
"

nodes_query2 = "
MATCH (a1:Autor)<-[r:FOI_ESCRITO_POR]-(art1:Artigo)-[:FOI_ESCRITO_POR]->(
    a2:Autor)
WHERE ID(a1) IN [90840, 90726, 98124, 97748, 105573, 98122, 160657, 91622,
     92232, 113381]
RETURN DISTINCT ID(a2) AS id, a2.nomeAutor AS label ORDER BY id
"

edges_query = "
MATCH (a1:Autor)<-[:FOI_ESCRITO_POR]-(:Artigo)-[:FOI_ESCRITO_POR]->(a2:
    Autor)
WHERE ID(a1) IN [90840, 90726, 98124, 97748, 105573, 98122, 160657, 91622,
     92232, 113381]
RETURN ID(a1) AS from, ID(a2) AS to, COUNT(*) AS weight
"

nodes1 = cypher(neo4j, nodes_query1)
nodes2 = cypher(neo4j, nodes_query2)
edges = cypher(neo4j, edges_query)

nodes <- rbind(nodes1, nodes2)
nodes <- data.frame(unique(nodes))

g = graph_from_data_frame(edges, directed=F, vertices=nodes)
```

D.3 Terceiro Experimento

Para o subgrafo do terceiro experimento:
Primeiro, selecionamos o identificador do artigo com mais autores:

```
MATCH (art:Artigo)-[:FOI_ESCRITO_POR]->(a:Autor)
RETURN art, count(a) ORDER BY count(a) DESC LIMIT 1
```

```
nodes_query1 = "
MATCH (a1:Autor)<-[:FOI_ESCRITO_POR]-(art1:Artigo)
WHERE art1.artigoID = 103327
RETURN DISTINCT ID(a1) AS id, a1.nomeAutor AS label ORDER BY id
"

nodes_query2 = "
MATCH (a1:Autor)<-[:FOI_ESCRITO_POR]-(art1:Artigo)
WHERE art1.artigoID = 103327
MATCH (a3:Autor)<-[:FOI_ESCRITO_POR]-(:Artigo)-[:FOI_ESCRITO_POR]->(a2:
    Autor)
WHERE ID(a3) = ID(a1)
RETURN DISTINCT ID(a2) AS id, a2.nomeAutor AS label ORDER BY id
"

edges_query = "
```

```
16 MATCH (a1:Autor)<-[:FOI_ESCRITO_POR]-(art1:Artigo)
17 WHERE art1.artigoID = 103327
18 MATCH (a3:Autor)<-[:FOI_ESCRITO_POR]-(:Artigo)-[:FOI_ESCRITO_POR]->(a2:
      Autor)
19 WHERE ID(a3) = ID(a1)
20 RETURN ID(a3) AS from, ID(a2) AS to, COUNT(*) AS weight
21 "
22
23 nodes1 = cypher(neo4j, nodes_query1)
24 nodes2 = cypher(neo4j, nodes_query2)
25 edges = cypher(neo4j, edges_query)
26
27 nodes <- rbind(nodes1, nodes2)
28 nodes <- data.frame(unique(nodes))
29
30 g = graph_from_data_frame(edges, directed=F, vertices=nodes)
```

D.4 Quarto Experimento

Para o subgrafo do quarto experimento:

```
1 MATCH (inst:Instituto)-[:PERTENCE_A]->(u:Universidade)
2 WHERE inst.nomeInstituto =~ '.*MATEMATICA.*'
3 AND u.nomeUniversidade = 'UNIVERSIDADE DE SAO PAULO'
4 RETURN inst
```

```
1 nodes_query1 = "
2 MATCH (inst:Instituto)<-[:FOI_PUBLICADO_POR]-(r:Revista)<-[:FOI_PUBLICADO_
      EM]-(art:Artigo)-[:FOI_ESCRITO_POR]->(a:Autor)-[E_FILIADO_A]->(inst2:
      Instituto)
3 WHERE inst.nomeInstituto IN [
4    'INSTITUTO DE MATEMATICA. E ESTATISTICA',
5    'INSTITUTO DE MATEMATICA E ESTATFSTICA',
6    'INSTITUTO .DE MATEMATICA E ESTATISTICA',
7    'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO',
8    'DEPARTAMENTO DE MATEMATICA E ESTATSTICA',
9    'DEPT. DE MATEMATICA E ESTATISTICA',
10   'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO IME
        USP.',
11   'INSTITUTO DE MATEMATICA',
12   'DEPTO. DE MATEMATICA E ESTATISTICA',
13   'DEPARTAMENTO DE MATEMATICA E ESTATISTICA',
14   'DEPT DE MATEMATICA E ESTATISTICA',
15   'INSTITUTO DE MATEMATICA E ESTATISTICAS',
16   'DEPARTAMENTO DE MATEMATICA',
17   'INSTITUTO DE MATEMATICA E ESTATISTICA',
18   'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO.'
19   ] OR
20 inst2.nomeInstituto IN [
21   'INSTITUTO DE MATEMATICA. E ESTATISTICA',
22   'INSTITUTO DE MATEMATICA E ESTATFSTICA',
23   'INSTITUTO .DE MATEMATICA E ESTATISTICA',
24   'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO',
25   'DEPARTAMENTO DE MATEMATICA E ESTATSTICA',
26   'DEPT. DE MATEMATICA E ESTATISTICA',
27   'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO IME
        USP.',
```

```
28     'INSTITUTO DE MATEMATICA',
29     'DEPTO. DE MATEMATICA E ESTATISTICA',
30     'DEPARTAMENTO DE MATEMATICA E ESTATISTICA',
31     'DEPT DE MATEMATICA E ESTATISTICA',
32     'INSTITUTO DE MATEMATICA E ESTATISTICAS',
33     'DEPARTAMENTO DE MATEMATICA',
34     'INSTITUTO DE MATEMATICA E ESTATISTICA',
35     'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO.'
36     ]
37 RETURN DISTINCT ID(a) AS id, a.nomeAutor AS label ORDER BY id
38 "
39
40 nodes_query2 = "
41 MATCH (inst:Instituto)<-[:FOI_PUBLICADO_POR]-(r:Revista)<-[:FOI_PUBLICADO_
       EM]-(art:Artigo)-[:FOI_ESCRITO_POR]->(a:Autor)-[E_FILIADO_A]->(inst2:
       Instituto)
42 WHERE inst.nomeInstituto IN [
43 'INSTITUTO DE MATEMATICA. E ESTATISTICA',
44 'INSTITUTO DE MATEMATICA E ESTATFSTICA',
45 'INSTITUTO .DE MATEMATICA E ESTATISTICA',
46 'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO',
47 'DEPARTAMENTO DE MATEMATICA E ESTATSTICA',
48 'DEPT. DE MATEMATICA E ESTATISTICA',
49 'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO IME
       USP.',
50 'INSTITUTO DE MATEMATICA',
51 'DEPTO. DE MATEMATICA E ESTATISTICA',
52 'DEPARTAMENTO DE MATEMATICA E ESTATISTICA',
53 'DEPT DE MATEMATICA E ESTATISTICA',
54 'INSTITUTO DE MATEMATICA E ESTATISTICAS',
55 'DEPARTAMENTO DE MATEMATICA',
56 'INSTITUTO DE MATEMATICA E ESTATISTICA',
57 'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO.'
58 ] OR
59 inst2.nomeInstituto IN [
60 'INSTITUTO DE MATEMATICA. E ESTATISTICA',
61 'INSTITUTO DE MATEMATICA E ESTATFSTICA',
62 'INSTITUTO .DE MATEMATICA E ESTATISTICA',
63 'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO',
64 'DEPARTAMENTO DE MATEMATICA E ESTATSTICA',
65 'DEPT. DE MATEMATICA E ESTATISTICA',
66 'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO IME
       USP.',
67 'INSTITUTO DE MATEMATICA',
68 'DEPTO. DE MATEMATICA E ESTATISTICA',
69 'DEPARTAMENTO DE MATEMATICA E ESTATISTICA',
70 'DEPT DE MATEMATICA E ESTATISTICA',
71 'INSTITUTO DE MATEMATICA E ESTATISTICAS',
72 'DEPARTAMENTO DE MATEMATICA',
73 'INSTITUTO DE MATEMATICA E ESTATISTICA',
74 'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO.'
75 ]
76 MATCH (a)<-[:FOI_ESCRITO_POR]-(:Artigo)-[:FOI_ESCRITO_POR]->(a2:Autor)
77 RETURN DISTINCT ID(a2) AS id, a2.nomeAutor AS label ORDER BY id
78 "
79
80 edges_query = "
81 MATCH (inst:Instituto)<-[:FOI_PUBLICADO_POR]-(r:Revista)<-[:FOI_PUBLICADO_
       EM]-(art:Artigo)-[:FOI_ESCRITO_POR]->(a:Autor)-[E_FILIADO_A]->(inst2:
```

```
    Instituto)
WHERE inst.nomeInstituto IN [
'INSTITUTO DE MATEMATICA. E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATFSTICA',
'INSTITUTO .DE MATEMATICA E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO',
'DEPARTAMENTO DE MATEMATICA E ESTATSTICA',
'DEPT. DE MATEMATICA E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO IME
    USP.',
'INSTITUTO DE MATEMATICA',
'DEPTO. DE MATEMATICA E ESTATISTICA',
'DEPARTAMENTO DE MATEMATICA E ESTATISTICA',
'DEPT DE MATEMATICA E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATISTICAS',
'DEPARTAMENTO DE MATEMATICA',
'INSTITUTO DE MATEMATICA E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO.'
] OR
inst2.nomeInstituto IN [
'INSTITUTO DE MATEMATICA. E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATFSTICA',
'INSTITUTO .DE MATEMATICA E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO',
'DEPARTAMENTO DE MATEMATICA E ESTATSTICA',
'DEPT. DE MATEMATICA E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO IME
    USP.',
'INSTITUTO DE MATEMATICA',
'DEPTO. DE MATEMATICA E ESTATISTICA',
'DEPARTAMENTO DE MATEMATICA E ESTATISTICA',
'DEPT DE MATEMATICA E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATISTICAS',
'DEPARTAMENTO DE MATEMATICA',
'INSTITUTO DE MATEMATICA E ESTATISTICA',
'INSTITUTO DE MATEMATICA E ESTATISTICA DA UNIVERSIDADE DE SAO PAULO.'
]
MATCH (a)<-[:FOI_ESCRITO_POR]-(:Artigo)-[:FOI_ESCRITO_POR]->(a2:Autor)
RETURN ID(a) AS from, ID(a2) AS to, COUNT(*) AS weight
"

nodes1 = cypher(neo4j, nodes_query1)
nodes2 = cypher(neo4j, nodes_query2)
edges = cypher(neo4j, edges_query)

nodes <- rbind(nodes1, nodes2)
nodes <- data.frame(unique(nodes))

g = graph_from_data_frame(edges, directed=F, vertices=nodes)
```

D.5 Girvan-Newman

A função que implementa este método é a cluster_edge_betweenness(graph).

```
clusters = cluster_edge_betweenness(g, weights = E(g)$weight)
nodes$group = membership(clusters)
nodes$value = NULL
max(clusters$modularity)
```

```
5 max(membership(clusters))
6 visNetwork(nodes, edges)
```

D.6 Louvain

A função que implementa este método é a cluster_louvain(graph).

```
1 clusters = cluster_louvain(g, weights = E(g)$weight)
2 nodes$group = clusters$membership
3 nodes$value = NULL
4 max(clusters$modularity)
5 max(membership(clusters))
6 visNetwork(nodes, edges)
```

D.7 Autovalores e Autovetores da Matriz de Modularidade

```
1 clusters = cluster_leading_eigen(g, weights = E(g)$weight) # cluster_
      leading_eigen
2 nodes$group = clusters$membership
3 nodes$value = NULL
4 max(clusters$modularity)
5 max(membership(clusters))
6 visNetwork(nodes, edges)
```

D.8 Passeios Aleatórios

A função que implementa este método é a cluster_walktrap(graph).

```
1 clusters = cluster_walktrap(g, weights = E(g)$weight)
2 nodes$group = clusters$membership
3 nodes$value = NULL
4 max(clusters$modularity)
5 max(membership(clusters))
6 visNetwork(nodes, edges)
```

D.9 Medição do Tempo de Execução

Para mostrar o tempo de execução de cada algoritmo, basta executar, em R:

```
1 system.time(cluster_edge_betweenness(g, weights = E(g)$weight))
2 system.time(cluster_louvain(g, weights = E(g)$weight))
3 system.time(cluster_leading_eigen(g, weights = E(g)$weight))
4 system.time(cluster_walktrap(g, weights = E(g)$weight))
```

Quando o tempo é menor que 0,10 segundos, utilizamos a biblioteca *microbenchmark* para uma maior precisão:

```
1 library(microbenchmark)
2 microbenchmark(cluster_edge_betweenness(g, weights = E(g)$weight), cluster
      _edge_betweenness(g, weights = E(g)$weight))
3 microbenchmark(cluster_louvain(g, weights = E(g)$weight), cluster_louvain(
      g, weights = E(g)$weight))
4 microbenchmark(cluster_leading_eigen(g, weights = E(g)$weight), cluster_
      leading_eigen(g, weights = E(g)$weight))
5 microbenchmark(cluster_walktrap(g, weights = E(g)$weight), cluster_
      walktrap(g, weights = E(g)$weight))
```

D.10 Contagem do Número de Nós em Cada Comunidade

Para contar o número de nós das comunidade detectada por cada um dos métodos, foi utilizado o seguinte código:

```
1 aux <- as.data.frame(table(nodes$group))
2 aux[order(-aux[,2]), ]
```

Apêndice E

Código para Visualização dos Acessos aos Artigos

Para realizar as visualizações das métricas sobre os dados contido no banco Neo4j foi utilizado R.

E.1 Impactos Regionais

A consulta realizada no banco de dados Neo4j para encontrar o artigo com maior número de acessos, ou seja, maior número de visualizações e *download*, foi:

```
1 MATCH (ar:Artigo)-[r]->(ci:Cidade)
2 WITH ar, count(r) AS rel_count
3 RETURN ar.tituloArtigo, ar.artigoID
4 ORDER BY rel_count DESC
5 LIMIT 1;
```

Para obtenção dos nomes das cidades de onde vieram os acessos ao artigo selecionado, foi feita a seguinte consulta no banco de dados orientado a grafos:

```
1 MATCH (ar:Artigo)-[r]->(c:Cidade) -[:ESTA_LOCALIZADA_EM]->(p:Pais)
2 WHERE ar.artigoID = 48 AND p.nomePais = "Brazil"
3 WITH count(ar) as count, c.nomeCidade as city
4 ORDER BY count DESC
5 RETURN city, count;
```

Para isso obter o nome dos países e a contagem dos acessos ao artigo "On the qmeromorphic Weyl algebra" realizados nestes países, foi feita a seguinte consulta:

```
1 MATCH (ar:Artigo)-[r:TEM_VISUALIZACAO_VINDO_DE]->(c:Cidade)
2       -[:ESTA_LOCALIZADA_EM]->(p:Pais)
3 WHERE ar.artigoID = 48
4 WITH count(ar) as rels, p.siglaPais as country, p.nomePais as countryName
5 ORDER BY rels DESC
6 RETURN countryName, country, rels;
```

Para avaliar a influência de um autor, basta contabilizar todos os acessos à todos os artigos publicados por este autor, utilizando o seguinte código:

```
1 MATCH (au:Autor) <-[:FOI_ESCRITO_POR]-          (ar:Artigo)
2                   -[r:TEM_VISUALIZACAO_VINDO_DE]-> (c:Cidade)
3                   -[:ESTA_LOCALIZADA_EM]->         (p:Pais)
4 WHERE au.nomeAutor = "<nome_do_autor>"
5 WITH count(au) as rels, p.nomePais as country
```

```
6 ORDER BY rels DESC
7 RETURN country, rels;
```

E.2 Quantidade dos Acessos

A seguinte consulta foi utilizada para obter as informações dos acessos dos 250 artigos selecionados:

```
1 MATCH (ar:Artigo)-[r]->(ci:Cidade)
2 WITH ar, count(r) AS rels
3 ORDER BY rels DESC LIMIT 250
4 MATCH (ar)-[r]->(:Cidade)
5 RETURN ar.artigoID AS id, r.dia AS dia, r.mes AS mes, r.ano AS ano
```

Referências Bibliográficas

Angles e Gutiérrez(2008) Renzo Angles e Claudio Gutiérrez. Survey of graph database models. *ACM Comput. Surv.*, 40(1). Citado na pág. 6

Blondel *et al.***(2008)** Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte e Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008. Citado na pág. 9, 10

Elmasri e Navathe(2010) Ramez Elmasri e Shamkant B. Navathe. *Fundamentals of Database Systems*. Pearson. Citado na pág. 5, 6

Fortunato(2009) Santo Fortunato. Community detection in graphs. *CoRR*, abs/0906.0612. URL http://arxiv.org/abs/0906.0612. Citado na pág. 2

Newman(2006) Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104. Citado na pág. 9, 11

Newman e Girvan(2004) Mark EJ Newman e Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113. Citado na pág. 7, 8, 9, 10

Penteado *et al.***(2014)** Raqueline RM Penteado, Rebeca Schroeder, Diego Hoss, Jaqueline Nande, Ricardo M Maeda, Walmir O Couto e Carmem S Hara. Um estudo sobre bancos de dados em grafos nativos. *X ERBD-Escola Regional de Banco de Dados*. Citado na pág. 1

Pons e Latapy(2005) Pascal Pons e Matthieu Latapy. Computing communities in large networks using random walks. Em *International Symposium on Computer and Information Sciences*, páginas 284–293. Springer. Citado na pág. 9, 11, 12

Robinson *et al.***(2015)** Ian Robinson, Jim Webber e Emil Eifrem. *Graph Databases: New Opportunities for Connected Data*. O'Reilly Media. Citado na pág. 7

Vukotic *et al.***(2014)** Aleksa Vukotic, Nicki Watt, Tareq Abedrabbo, Dominic Fox e Jonas Partner. *Neo4j in Action*. Manning Publications. Citado na pág. 1, 6, 7