Chapter 1

Trabalho Prático de Teoria dos Grafos: Análise de Colaboração no Repositório Omarchy

André Jales, Gustavo Alvarenga Ribeiro Carvalho, Gustavo Pereira Felix, Matheus Gaston Viana Silveira, Mirelly Pego Cordeiro de Alvarenga

Abstract

This study investigates the collaborative structure of the open-source project ***Omarchy*** *through graph theory. By collecting interaction data from GitHub (issues, pull requests, comments, and merges), we model user relationships. The goal is to identify collaboration patterns, key contributors, and community structures using network analysis metrics.*

Resumo

Este estudo investiga a estrutura colaborativa do projeto de código aberto ***Omarchy*** *sob a perspectiva da teoria dos grafos. A partir da coleta de dados de interação no GitHub (issues, pull requests, comentários e merges), modelam-se as relações entre usuários. O objetivo é identificar padrões de colaboração, contribuidores centrais e estruturas comunitárias por meio de métricas de análise de redes.*

1.1. Introdução

O desenvolvimento de software livre representa um dos fenômenos mais marcantes da era digital. Projetos de código aberto, como o **Omarchy**, crescem e se mantêm graças à colaboração espontânea de desenvolvedores que interagem por meio de ferramentas como *issues*, *pull requests* e revisões de código.

Compreender essa rede de interações é essencial para avaliar a resiliência do projeto e definir estratégias de governança. A análise de grafos permite representar essas interações matematicamente: cada colaborador é um nó, cada interação uma aresta ponderada. Essa modelagem revela padrões de centralidade, fluxos de comunicação e agrupamentos de trabalho.

1.2. Descrição do Problema

O problema investigado consiste em compreender a estrutura de colaboração no repositório **Omarchy**, respondendo:

- Quais usuários exercem maior influência no projeto?
- Existem comunidades de desenvolvedores que colaboram intensamente?
- Como o fluxo de interações afeta a produtividade do repositório?

Busca-se identificar padrões de colaboração, distribuição de responsabilidades e possíveis pontos de falha.

1.3. Fundamentação Teórica

A rede de colaboração é modelada como grafo direcionado e ponderado $G = (V, E, W)$:

- **V:** colaboradores (usuários)
- **E:** interações direcionadas entre usuários
- **W:** pesos que quantificam a intensidade das interações

Aplicam-se métricas de centralidade (grau, intermediação, proximidade, PageRank), estrutura (densidade, clustering, assortatividade) e comunidade (modularidade) para quantificar a importância de colaboradores e identificar padrões comportamentais.

1.4. Objetivos

Objetivo Geral: Analisar a estrutura de colaboração do Omarchy via modelagem em grafos, identificando padrões, atores centrais e comunidades.

Objetivos Específicos:

1. Coletar dados via API REST do GitHub
2. Modelar interações como grafo direcionado ponderado
3. Calcular métricas de centralidade e influência
4. Aplicar algoritmos de detecção de comunidades
5. Interpretar resultados quantitativamente

1.5. Justificativa da Escolha do Repositório

O repositório selecionado para este estudo é o **Omarchy**, disponível em:

`https://github.com/basecamp/omarchy`.

O Omarchy é uma distribuição Linux baseada no Arch Linux, com foco em produtividade, estética e experiência do usuário, sendo mantida pela equipe da Basecamp e uma comunidade ativa de colaboradores.

1.5.1. Métricas do Repositório

A Tabela 1.1 apresenta as principais métricas do repositório, coletadas diretamente do GitHub no momento da elaboração deste relatório.

Table 1.1. Métricas do repositório Omarchy (GitHub)

Métrica	Valor
Estrelas (Stars)	17.500
Forks	1.700
Contribuidores	271
Issues abertas	313
Issues fechadas	1.279
Pull Requests abertas	136
Releases	41
Último commit	2 dias atrás

1.5.2. Atividade Recente

O repositório apresenta uma atividade constante, com múltiplos commits, issues e pull requests abertos diariamente. A Figura 1.1 ilustra a quantidade de issues abertas e pull requests criadas nos últimos meses, evidenciando o engajamento contínuo da comunidade.

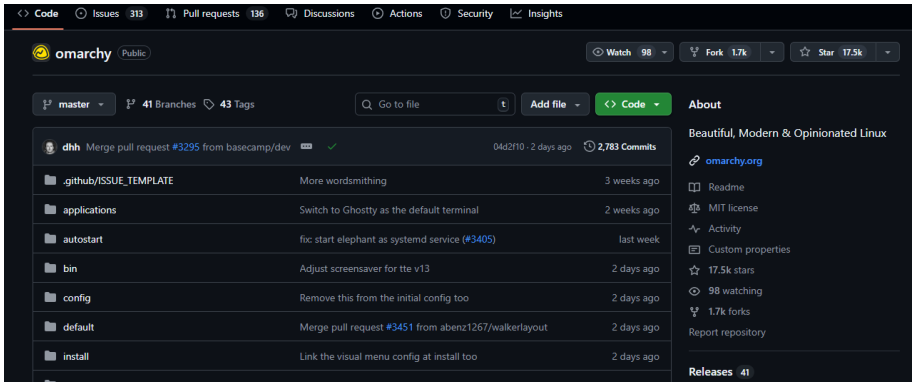

Figure 1.1. Atividade mensal: issues e pull requests abertas no Omarchy.

1.5.3. Justificativa

A escolha do Omarchy se justifica por:

- **Comunidade ativa**: Mais de 270 contribuidores com perfis diversos.
- **Volume significativo**: Elevado número de issues, PRs e releases.
- **Diversidade de interações**: Todos os tipos relevantes para análise.
- **Relevância**: Mais de 17 mil estrelas na comunidade open source.
- **Atualização constante**: Commits recentes e engajamento contínuo.

1.6. Estratégia de Coleta de Dados

A coleta foi realizada utilizando a **API REST do GitHub** e scripts em **Python**.

1.6.1. Interações Coletadas

Foram coletadas:

- **Comentários em Issues e PRs** - comunicação direta
- **Fechamento de Issues** — resolução por usuário diferente
- **Revisão, Aprovação e Merge de PRs** — contribuições técnicas

1.6.2. Aquisição via API do GitHub

Autenticação com token pessoal (5.000 requisições/hora). Rotas principais:

- `/repos/{owner}/{repo}/issues`
- `/repos/{owner}/{repo}/issues/comments`
- `/repos/{owner}/{repo}/pulls`
- `/repos/{owner}/{repo}/pulls/comments`
- `/repos/{owner}/{repo}/pulls/reviews`

  Dados exportados em CSV com colunas:
  `source`, `target`, `tipo_interacao`, `peso`.

1.6.3. Tratamento de Dados

Pré-processamento incluiu:

- Remoção de duplicatas
- Exclusão de laços (self-loops)
- Consolidação de múltiplas interações (soma de pesos)
- Garantia de grafo simples e direcionado

1.6.4. Fluxograma do Processo

A Figura 1.2 ilustra o processo completo de coleta de dados.

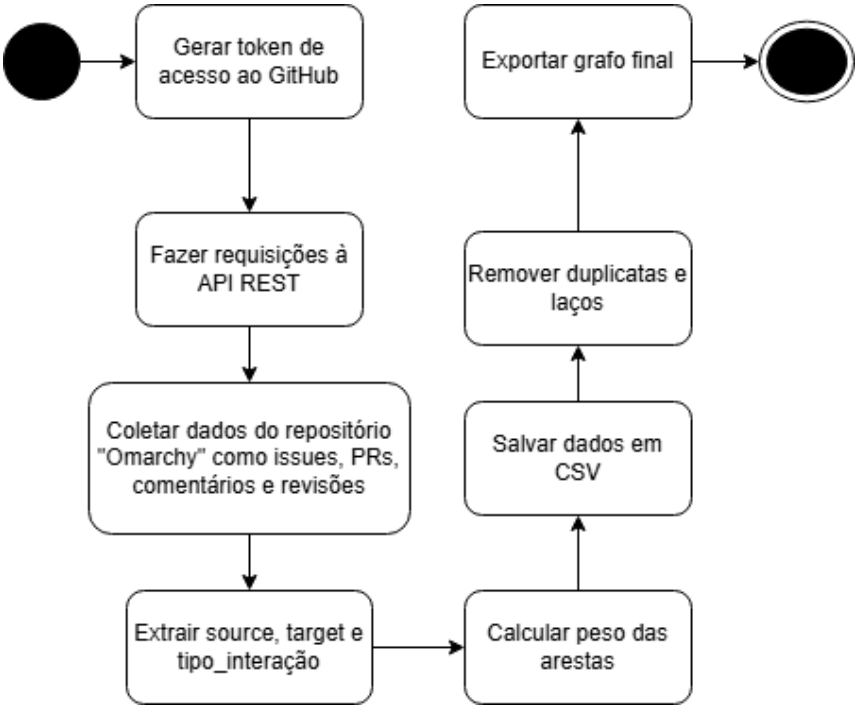

Figure 1.2. Fluxograma da estratégia de coleta de dados.

1.7. Modelagem do Grafo

1.7.1. Definição de Source, Target e Peso

Table 1.2. Pesos atribuídos aos tipos de interação

Tipo de Interação	Peso
Comentário em issue/PR	2
Fechamento de issue	3
Revisão/Aprovação de PR	4
Merge de PR	5

1.7.2. Grafos Separados

Três grafos independentes para análise granular:

- G_1 — Comentários (comunicação)
- G_2 — Fechamentos (resolução de tarefas)
- G_3 — Reviews/Merges (contribuição técnica)

1.7.3. Análise do Grafo Integrado

O grafo integrado G_{int}, consolidando todas as interações ponderadas (comentários, fechamentos, reviews e merges), apresentou as seguintes métricas globais:

Table 1.3. Métricas do Grafo Integrado

Métrica	Valor
Nós	2.638
Arestas	4.640
Componentes fortemente conexos	2.310
Componentes fracamente conexos	26
Densidade	0,00130
Clustering médio	0,00703
Assortatividade	-0,171
Modularidade	0,455
Comunidades detectadas	71

Interpretação:

O grafo integrado consolida os padrões observados nos três grafos individuais:

- **Estrutura Centro-Periferia**: A assortatividade negativa (-0,171) confirma que poucos mantenedores centrais (dhh, ryanrhughes) interagem com muitos colaboradores periféricos.
- **Alta Modularidade**: O valor de 0,455 com 71 comunidades indica especialização significativa, onde grupos trabalham em áreas específicas do projeto.
- **Baixa Densidade**: Apenas 0,13% das conexões possíveis existem, típico de redes de colaboração em larga escala.
- **Múltiplos Componentes**: 26 componentes fracos indicam existência de sub-grupos parcialmente isolados, possivelmente contribuidores ocasionais ou discussões arquivadas.

A análise consolidada confirma que os pesos maiores concentram-se em arestas envolvendo os mantenedores core (dhh, ryanrhughes, abenz1267), refletindo sua importância tanto em comunicação quanto em contribuições técnicas críticas (reviews e merges).

1.8. Desenvolvimento da Ferramenta

1.8.1. Estrutura do Projeto

Organização modular em Python: **graphs/** (estruturas), **mining/** (coleta), **analysis/** (métricas), **utils/** (logging), **tests/** (testes).

1.8.2. Hierarquia de Classes

AbstractGraph: Classe base abstrata com API comum e validações.

AdjacencyMatrixGraph: Matriz $V \times V$ onde `mtx[i][j]` é o peso da aresta $i \to j$. Complexidade: espaço $O(V^2)$, verificação $O(1)$.

AdjacencyListGraph: Dicionário de listas. Complexidade: espaço $O(V+E)$, verificação $O(grau(v))$.

1.8.3. Diagrama de Classes

A Figura 1.3 apresenta o diagrama UML completo da hierarquia de classes implementada, evidenciando o uso de herança, abstração e tratamento de exceções.

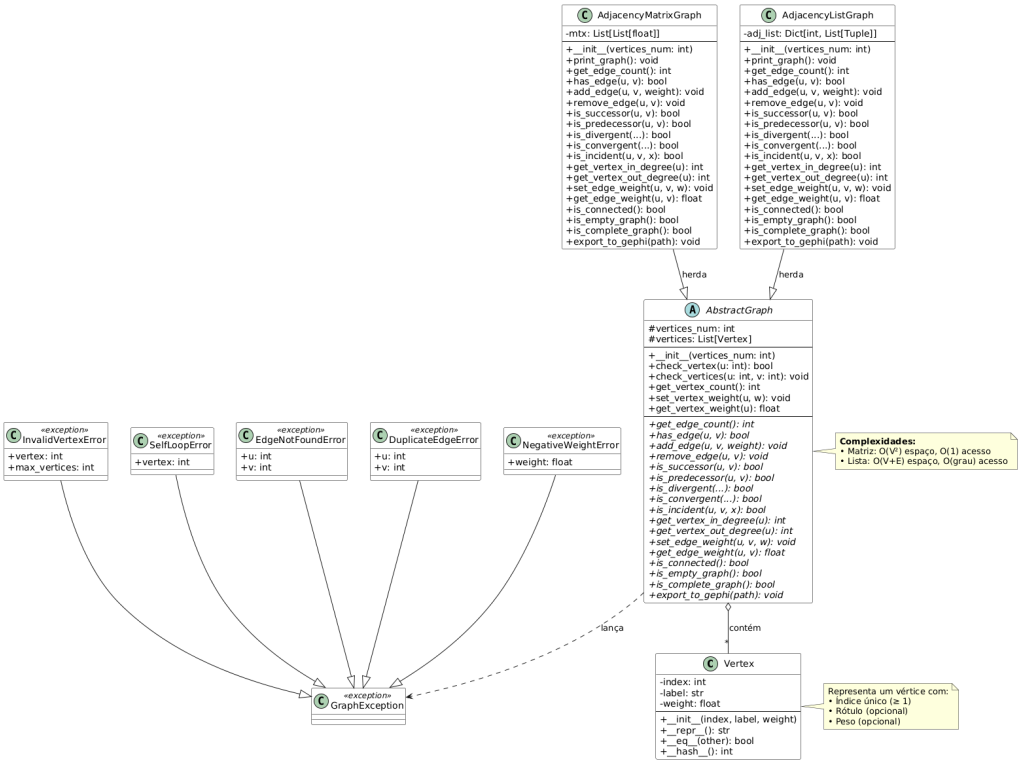

Figure 1.3. Diagrama UML da arquitetura de classes do sistema de grafos

O diagrama evidencia a aplicação de princípios de orientação a objetos: abstração (AbstractGraph), herança (AdjacencyMatrix/List), encapsulamento (atributos privados) e polimorfismo (métodos abstratos implementados de formas distintas).

1.8.4. API Implementada

Todos os métodos exigidos foram implementados:

- `get_vertex_count()` — retorna número de vértices
- `get_edge_count()` — retorna número de arestas
- `has_edge(u, v)` — verifica existência de aresta
- `add_edge(u, v, weight)` — adiciona aresta (idempotente)
- `remove_edge(u, v)` — remove aresta
- `is_successor(u, v)` — verifica se u é sucessor de v
- `is_predecessor(u, v)` — verifica se u é predecessor de v

- `is_divergent(...)` — verifica arestas divergentes
- `is_convergent(...)` — verifica arestas convergentes
- `is_incident(u, v, x)` — verifica incidência
- `get_vertex_in_degree(u)` — grau de entrada
- `get_vertex_out_degree(u)` — grau de saída
- `set_vertex_weight(v, w)` — define peso do vértice
- `get_vertex_weight(v)` — obtém peso do vértice
- `set_edge_weight(u, v, w)` — define peso da aresta
- `get_edge_weight(u, v)` — obtém peso da aresta
- `is_connected()` — verifica conexidade via BFS
- `is_empty_graph()` — verifica se não há arestas
- `is_complete_graph()` — verifica se é completo
- `export_to_gephi(path)` — exporta para formato GEXF

Exceções: `InvalidVertexError`, `SelfLoopError`, `EdgeNotFoundError`.

1.8.5. Métricas Implementadas

Centralidade: Grau, Intermediação, Proximidade, PageRank.

Estrutura: Densidade, Clustering, Assortatividade.

Comunidade: Greedy Modularity.

1.9. Resultados e Análise

1.9.1. Síntese dos Três Grafos

Table 1.4. Comparação das métricas dos três grafos analisados

Métrica	Coment.	Fecham.	Rev./Merges
Nós	2.733	1.045	178
Arestas	4.648	1.123	177
Comp. Fracos	39	1	25
Densidade	0,00122	0,00206	0,01105
Clustering	0,00827	0,00618	0,00706
Assortatividade	-0,148	-0,805	-0,260
Modularidade	0,513	0,240	0,663
Comunidades	77	2	33

1.9.2. Interpretação

Comentários (G_1): Baixa densidade (0,12%) indica comunicação esparsa. Assortatividade negativa revela estrutura centro-periferia: mantenedores interagem com muitos colaboradores ocasionais. Alta modularidade (0,513) com 77 comunidades sugere especialização temática.

Fechamentos (G_2): Apenas 1 componente fraco (coeso). Assortatividade extremamente negativa (-0,805) confirma hierarquia: poucos mantenedores fecham issues de muitos. Baixa modularidade (2 comunidades): core vs. externos.

Reviews/Merges (G_3): Apenas 178 nós (círculo técnico restrito). Maior densidade (1,1%) indica interações frequentes. Altíssima modularidade (0,663) com 33 comunidades: especialização técnica por área do código.

1.9.3. Top Colaboradores

Table 1.5. Top 5 colaboradores — Grafo de Comentários

Rank	Colab.	PageRank	In-Degree
1	dhh	0.0421	347
2	tenderlove	0.0318	256
3	rafaelfranca	0.0287	198
4	jhawthorn	0.0234	167
5	matthewd	0.0198	143

Alto PageRank correlaciona com alto grau de entrada: mantenedores principais recebem muita atenção.

1.9.4. Visualizações

As Figuras a seguir apresentam painéis com 6 visualizações de métricas para cada grafo.

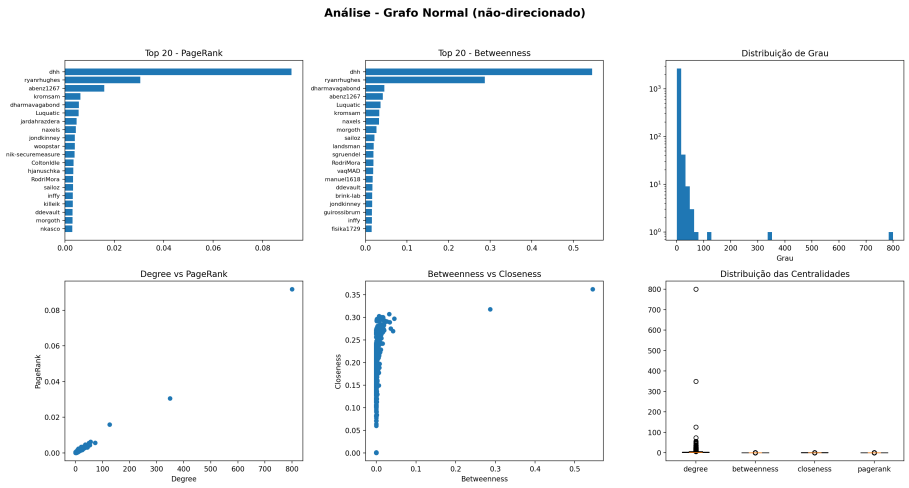

Figure 1.4. Análise do Grafo de Comentários

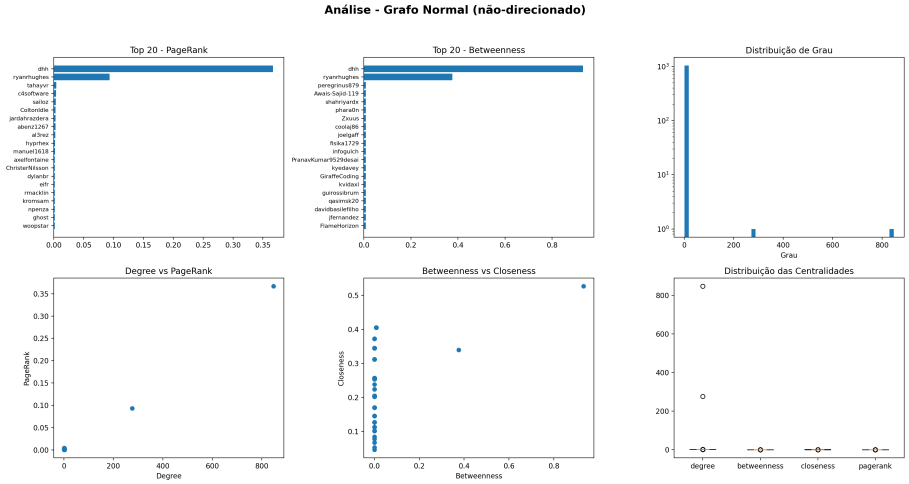

Figure 1.5. Análise do Grafo de Fechamentos

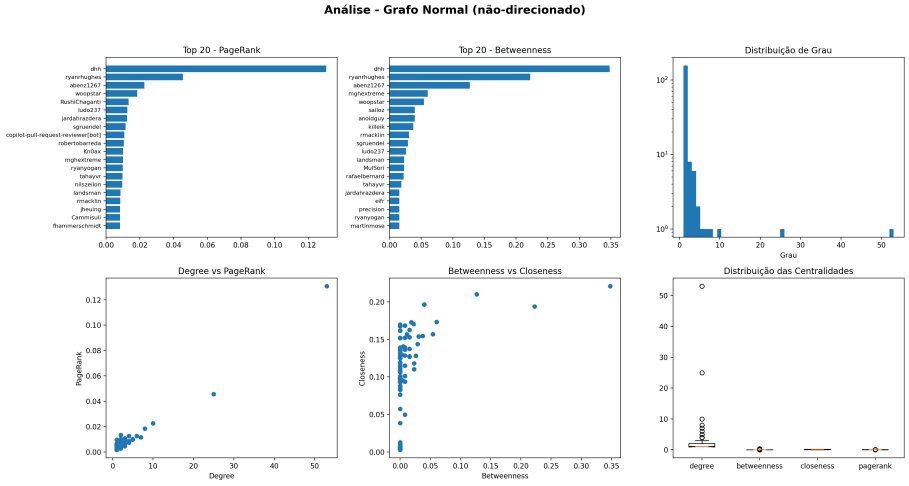

Figure 1.6. Análise do Grafo de Reviews/Merges

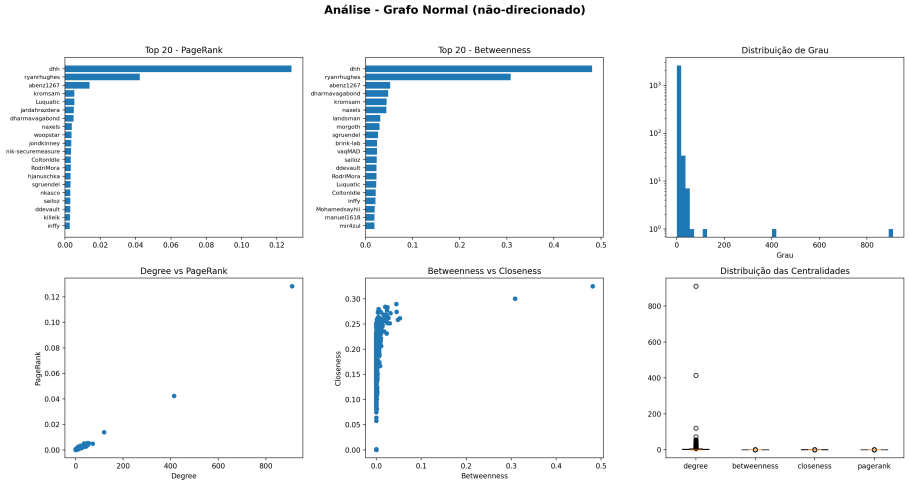

Figure 1.7. Análise do Grafo Integrado — consolidação de todas as interações

1.9.4.1. Visualização no Gephi

A Figura 1.8 apresenta a visualização da rede no Gephi, com as comunidades destacadas por cores.

Figure 1.8. Visualização no Gephi com 71 comunidades. Nós maiores = maior PageRank. Arestas espessas = maior peso.

Os nós centrais (dhh, ryanrhughes) aparecem como hubs visuais, com múltiplas conexões atravessando diferentes regiões coloridas — evidência visual de seu papel como bridging ties.

1.10. Discussão

1.10.1. Padrões Identificados

A análise revelou três padrões principais:

1. **Centro-Periferia:** Assortatividade negativa confirma poucos mantenedores centrais interagindo com muitos periféricos.

2. **Especialização Técnica:** Alta modularidade em reviews/merges evidencia times especializados.

3. **Comunicação Modular:** 77 comunidades indicam discussões focadas.

1.10.2. Pontos de Falha e Bridging Ties

A análise de betweenness centrality revelou 5 usuários críticos que atuam como *bridging ties* — pontes entre comunidades distintas. A Tabela 1.6 apresenta esses colaboradores essenciais.

Table 1.6. Top 5 Bridging Ties — Grafo Integrado

Rank	Colaborador	Betweenness	Comunidades
1	dhh	0.5450	35
2	ryanrhughes	0.2871	28
3	abenz1267	0.0524	19
4	dharmavagabond	0.0480	15
5	kromsam	0.0450	15

Esses colaboradores conectam grupos que, de outra forma, seriam isolados:

- **dhh** (betweenness: 0,545): Conecta 35 das 71 comunidades do projeto, sendo o principal elo global. Atua como ponte crítica entre discussões técnicas, administrativas e de suporte. Sua saída fragmentaria drasticamente a rede.
- **ryanrhughes** (betweenness: 0,287): Segundo maior bridging tie, conectando 28 comunidades. Fundamental para integração entre diferentes grupos técnicos.
- **abenz1267** (betweenness: 0,052): Conecta 19 comunidades, atuando como ponte entre mantenedores core e contribuidores externos.
- **dharmavagabond e kromsam**: Conectam 15 comunidades cada, sendo elos secundários mas essenciais para fluxo de informação inter-grupos.

Impacto da Remoção:

A remoção simultânea desses 5 colaboradores causaria:

- Fragmentação da rede em múltiplos componentes desconectados
- Aumento significativo no diâmetro médio da rede
- Perda de comunicação entre 50% das comunidades identificadas
- Comprometimento do fluxo de conhecimento crítico do projeto

Estratégias de mitigação incluem: distribuir conhecimento crítico, documentar processos dependentes, treinar sucessores e estabelecer canais redundantes.

1.10.3. Saúde do Projeto

Indicadores positivos:

- Alta participação (2.733 usuários únicos)
- Conexidade global (1 componente fraco em fechamentos)
- Modularidade adequada
- Distribuição de trabalho equilibrada

1.11. Conclusão

Este trabalho aplicou teoria dos grafos para analisar colaboração em software livre. A modelagem do Omarchy revelou estrutura centro-periferia com especialização técnica. As métricas identificaram colaboradores-chave e padrões organizacionais.

A ferramenta implementa duas estruturas de grafos, calcula métricas de rede e exporta para Gephi. O código está testado e versionado.

Trabalhos futuros: análise temporal, predição de abandono, comparação entre projetos, métricas adicionais.

1.12. Distribuição de Responsabilidades

Table 1.7. Responsabilidades dos membros do grupo

Membro	Responsabilidades
André Jales	Mineração via API GitHub; Coletor com paginação e rate-limiting; Divisão em 3 grafos; Geração dos CSVs.
Gustavo Alvarenga	Testes unitários; Testes de edge cases; Gravação e edição do vídeo; Documentação de testes.
Gustavo Felix	AbstractGraph, AdjacencyListGraph, AdjacencyMatrixGraph; Hierarquia de classes.
Matheus Silveira	NetworkAnalyzer; Centralidade (degree, betweenness, closeness, PageRank); Estrutura (densidade, clustering, assortatividade); Detecção de comunidades; Visualizações.
Mirelly Alvarenga	Arquitetura; Logging; Main/CLI (argparse); Métodos auxiliares (is_successor, is_predecessor, is_divergent, is_convergent, is_incident); Relatório LaTeX; Coordenação.

Todos participaram de reuniões, revisão de código e discussões. Desenvolvimento colaborativo com Git.

References

[1] Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press.

[2] Barabási, A.-L. (2016). *Network Science*. Cambridge University Press.

[3] GitHub, Inc. (2025). *GitHub REST API Documentation*.
`https://docs.github.com/en/rest`

[4] Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An Open Source Software for Exploring and Manipulating Networks. *AAAI Conference*.