Local Subgroup Discovery on Attributed Network Graphs

Seminário 2 – Aprendizado Descritivo

2025.1

Historiador

Bernnardo Serafim

Diná Xavier

Gabriel Fadoul

Kênia Gonçalves

Oluwatoyin Joy

Samuel Kfuri

Contextualização

Home > Advances in Intelligent Data Analysis XXIII > Conference paper

Local Subgroup Discovery on Attributed Network Graphs

Conference paper | First Online: 02 May 2025

pp 195–208 | Cite this conference paper

Carl Vico Heinrich, Tommie Lombarts, Jules Mallens, Luc Tortike, David Wolf, and Wouter Duivesteijn

Technische Universiteit Eindhoven, Eindhoven, the Netherlands

Advances in Intelligent Data Analysis XXIII

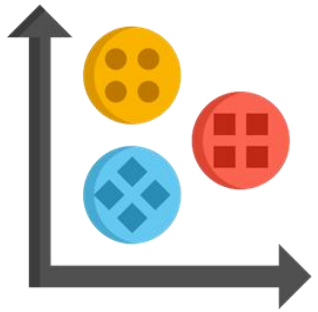
(IDA 2025)

Wouter Duivesteijn é docente da TU/e atuando com mineração de padrões interpretáveis, sobretudo Descoberta de Subgrupos, Modelos Explicáveis e Fairness-aware Learning. Os demais autores são mestrandos.

Renomada instituição holandesa, localizada no importante pólo tecnológico BrainPort, com forte reputação em engenharia da computação, matemática aplicada, ciência de dados e IA.

Symposium on Intelligent Data Analysis (IDA 2025), evento voltado à comunidade científica que estuda mineração de dados, aprendizado de máquina e IA.

Algoritmos relacionados



DESCOBERTA DE SUBGRUPOS (SD)

- Tarefa do aprendizado descritivo supervisionado que se dedica a identificar excepcionalidades estatísticas de subconjuntos em relação a uma propriedade de interesse.
- **LSD (Local Subgroup Discovery):** descoberta de subgrupo local, consiste em restringir o espaço de busca a contextos menores que a população.

GRAFOS

- Grafos são estruturas de dados bastante exploradas para representação de sistemas complexos (nós = entidades, arestas = relações) com interações em diferentes escalas.
- Uma extensão importante são as redes atribuídas, em que nós e arestas possuem rótulos associados.
- **EgoNets:** derivados de um grafo maior, capturam estruturas locais ao redor de um nó central (ego), incluindo seus vizinhos diretos (alters).

Comparativo com métodos existentes

Método	O que incorpora do método?	O que o algoritmo evolui?
SD	Descoberta de subgrupos	Busca local e em grafos
LSD	SD com busca local	Incorpora LSD a grafos (originalmente só tabular)
Grafos Atribuídos	Estrutura de dados rotuladas	Incluir a estrutura na análise, otimizando mineração de dados mais complexos
EgoNets	Análise de rede com foco em um nó específico	Adiciona análise estatística à descrição estrutural focada

Local Subgroup Discovery on Attributed Network Graphs

Definição formal

Local Subgroup Discovery on Attributed Network Graphs consiste em identificar subgrupos de nós em um grafo com atributos, cujo comportamento se destaca de forma estatisticamente significativa em relação à sua **vizinhança local**, combinando informações da **estrutura da rede** com os **atributos dos nós e arestas** e uma **variável-alvo de interesse**.

Contribuições técnicas

- Primeiro algoritmo de LSD para grafos com atributos.
- Mede a divergência local com base em topologia + atributos.
- Seleciona automaticamente (e dinamicamente) o grupo local de referência.

Aplicações práticas

- Redes sociais (identificar bolhas de comportamento).
- Redes biológicas (detectar padrões moleculares locais).
- Dados interligados (web, citações, sistemas complexos).

Formalizador

Caio Jorge Carvalho Lara
Juan Marcos Braga Faria
Luisa Vasconcelos de Castro Toledo
Luiza Sodré Salgado
Mateus Reis Evangelista
Samuel Henrique Miranda Alves

Padrões Minerados

- O objetivo é encontrar grupos "interessantes" dentro do grafo
- Um grupo é qualquer conjunto de nós, com suas respectivas arestas
- Estamos trabalhando em grafos com **classes nos nós** (*attributed networks*)
  - Cada nó também possui uma variável binária T (*target variable*), indicando se aquele nó possui ou não uma determinada classe alvo
  - A distribuição dessa variável binária nos grupos será o indicativo de relevância ("*interestingness*")

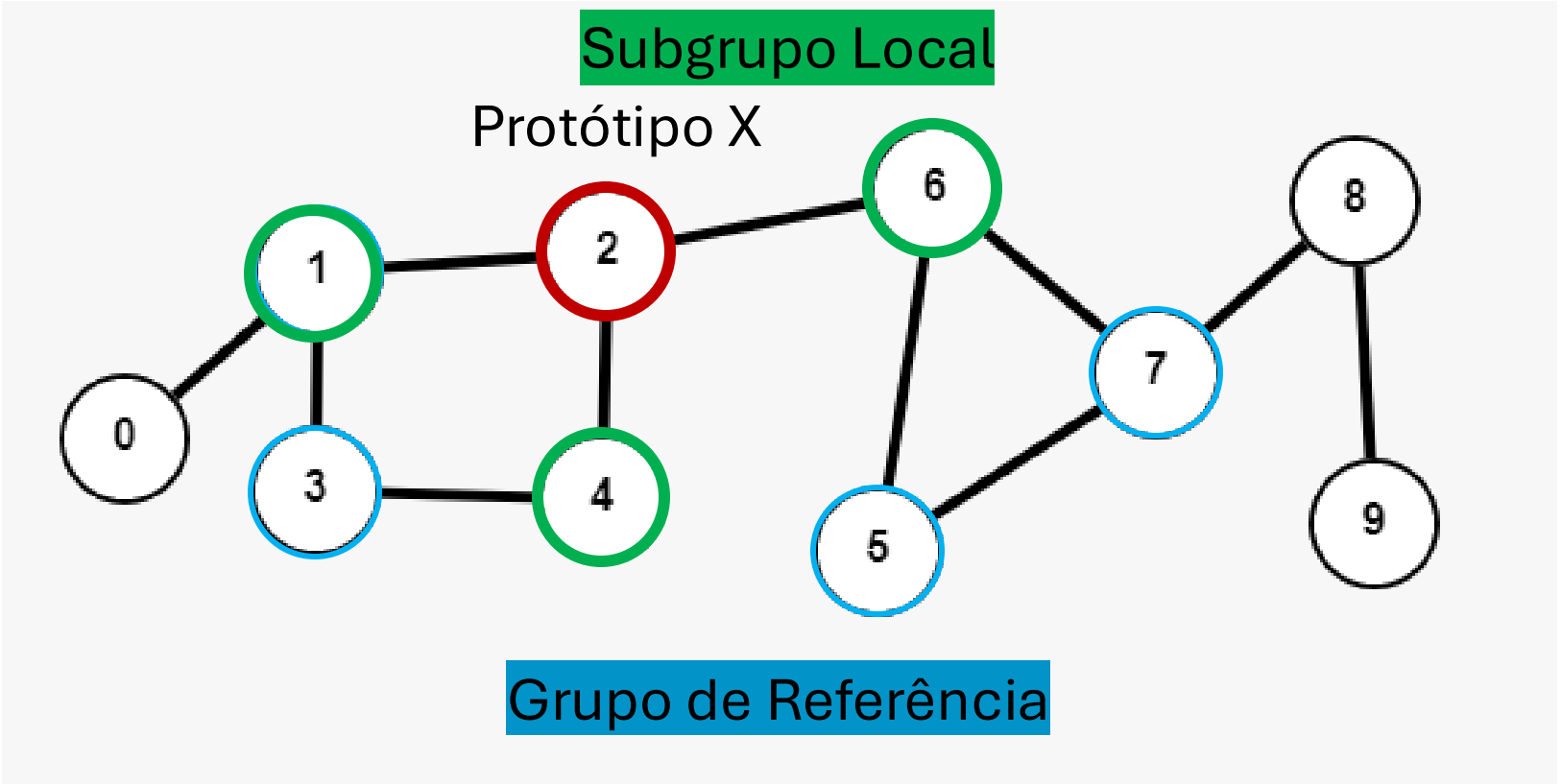
Padrões Minerados

- Particularmente, queremos encontrar subgrupos "interessantes" dentro de um grupo específico no grafo
  - Assim, conseguimos encontrar padrões interessantes que revelam distinções particulares de um local específico no grafo
  - Os subgrupos são chamados de **subgrupos locais**
  - Os grupos analisados são chamados de **grupo de referência**

- Um grupo de referência é "interessante" se ele difere-se do grafo global

- Um subgrupo local é "interessante" se ele difere-se do seu grupo de referência

Padrões Minerados



Grafo Global

Obs: Um subgrupo local é sempre menor que um de referência e, por definição, está sempre contido nele

Conceitos Fundamentais

- Definimos um nó chamado de protótipo, que será o referencial (centro) para a definição dos grupos
  - Os grupos são encontrados pela distância dos nós ao protótipo

- Utilizaremos a métrica de qualidade da **divergência ponderada de Kullback-Leibler** para medir a relevância dos grupos encontrados

$$q(S,R) = \frac{|S|}{|R|} \cdot \sum_{y \in \{0,1\}} P_S(y) \log\left(\frac{P_S(y)}{P_R(y)}\right) \Bigg| P_A(y) = \mathbb{P}\big(t(x) = y \mid x \in A\big)$$

Mede o quanto a distribuição da variável alvo t se difere entre o subgrupo local (S) e o grupo de referência (R), ou seja, o quão distinto o subgrupo local é comparado com o de referência

Cálculo das Distâncias

- Usaremos como medida de distância o menor caminho possível entre 2 nós (Figura 1), e ela será calculada pelo algoritmo de Dijkstra

- Em casos onde a distância de dois nós para o protótipo seja igual, usaremos a distância de Gower para desempatá-los (Figura 2). Essa distância leva em consideração os atributos dos nós.
  - Caso ainda estejam empatados depois disso, eles são escolhidos aleatoriamente

$$d_{uv} = \min_{P_{uv}} \left(\sum_{(u,v) \in P_{uv}} w(u,v) \right)$$

Figura 1

$$d(u,v) = \frac{1}{h} \sum_{k=1}^{h} \delta_k(u,v)$$

Figura 2

Algoritmo 1

Algorithm 1 Generate Distance Ranking

procedure RANKING(x, S)

$D \leftarrow Dijkstra(x, S)$

$R \leftarrow SortOn(R, D)$

for d **in** $unique$(R) **do**

$r, l \leftarrow idxmin(R, d), idxmax(R, d)$

$gowerlist \leftarrow Gower(x, n_i) \ \forall \ n \in R[r : l]$

$SortOn(R[r : l], gowerlist)$

return R

- Executamos **Dijkstra** iniciando em ***x***
- Ordenamos pela distância entre os vértices
- Para cada vértice com mesma distância, pegamos o grupo, calculamos a distância de **Gower** e reordenamos com base nela

Algoritmo 2

Algorithm 2 Generate ρ and σ

```
procedure DISCOVERY(R ← ranking)
    ρ, σ, b ← 0, 0, 0
    for i in len(R) do
        q ← Quality(R[0 : i], S)
        if q ≥ b then
            b ← q
            ρ ← i
    b ← 0
    for i in [0 : ρ] do
        q ← Quality(R[0 : i], R[0 : ρ])
        if q ≥ b then
            b ← q
            σ ← i
    return ρ, σ
```

- Iteramos sobre todos os prefixos possíveis de **R**, isto é, testa **R[0:i]** como candidatos a grupo de referência $\mathbf{R_p}$
- Avaliamos a qualidade do grupo de referência candidato comparado ao conjunto todo **S**
- Se **q** for a melhor qualidade até agora, atualizamos **ρ** com esse valor de **i**.
- Itera sobre todos os subgrupos possíveis dentro de $\mathbf{R_p}$.
- Avalia a qualidade de **R[0:i]** como subgrupo local $\mathbf{S_s}$, comparado ao grupo de referência **R[0:ρ]**.
- Salva o melhor tamanho **σ** que maximiza essa segunda qualidade.

Pós-Processamento

Após rodar este algoritmo para múltiplos protótipos $\mathbf{x}$ (aleatoriamente escolhidos), teremos vários pares **(ρ, σ)**, ou seja, muitos subgrupos candidatos.

Como os protótipos podem ser próximos, muitos subgrupos se sobrepõem bastante.

Então o artigo faz um pós-processamento com base na sobreposição de **Dice-Sørensen**:

$$O(S_1, S_2) = \frac{2}{|S_1| + |S_2|} \sum_{n \in S_1} \mathbb{1}_{\{n \in S_2\}}$$

Definimos um limiar **θ** como limite superior da sobreposição máxima permitida entre subgrupos.

Subgrupos que têm sobreposição acima de um limiar **θ** são descartados.

A filtragem é feita ordenando todos os subgrupos por qualidade decrescente, e escolhendo os melhores que não se sobrepõem muito.

Metodologista

Alexis Duarte Guimarães Mariz
Amanda Mendes Pinho
João Vítor Fernandes Dias
Kael Soares Augusto
Lucas Xavier Veneroso

Datasets Utilizados

- OGBG-MolPCBA
  - 41 127 grafos, 9-dimensional
- Twitch PT
  - 1 912 nós, 64 510 arestas, 128-dimensional
- WebKB Cornell
  - 1703 features por nó (parte de um dataset maior)

OGBG-MolPCBA

Table 2: Protein 349519 subgroups

Prot.	ρ	σ	q	Acc. R_ρ	Acc. S_σ
46	9	3	0.321	0.56	0.00
28	14	5	0.231	0.21	0.60
253	13	5	0.229	0.23	0.60
35	13	5	0.229	0.23	0.60
147	13	5	0.229	0.23	0.60
11	11	7	0.207	0.55	0.29
167	5	3	0.207	0.60	0.33
268	5	3	0.207	0.60	0.33
45	9	4	0.204	0.56	0.25
272	16	5	0.196	0.25	0.60

Twitch PT

Table 3: Twitch PT subgroups

Prot.	ρ	σ	q	Acc. R_ρ	Acc. S_σ
385	9	4	0.278	0.67	0.25
1378	26	13	0.218	0.62	0.31
97	12	4	0.192	0.83	0.50
403	103	14	0.173	0.61	0.14
1032	220	58	0.164	0.49	0.17

WebKB Cornell

Table 4: `WebKB Cornell` subgroups

Prot.	ρ	σ	q	**Acc.** R_ρ	**Acc.** S_σ
78	21	6	0.255	0.19	0.67
135	36	10	0.252	0.22	0.70
30	39	6	0.246	0.21	0.83
60	22	4	0.242	0.18	0.75
16	33	3	0.228	0.24	1.00

Ablation Study

- OGBG-MolPCBA
  - Overlap de nós: 0%
  - Poder Discriminativo: 0.215 vs. 0.0533
  - Cobertura t(v) = 1: 0.00% vs. 31.56%
- Twitch PT
  - Overlap de nós: 0,62%
  - Poder Discriminativo: 0.2349 vs. 0.0798
  - Cobertura t(v) = 1: 63.84% vs. 28.3%
- WebKB Cornell
  - Overlap de nós: 8,99%
  - Poder Discriminativo: 0.2211 vs. 0.1103
  - Cobertura t(v) = 1: 48.6% vs. 81%

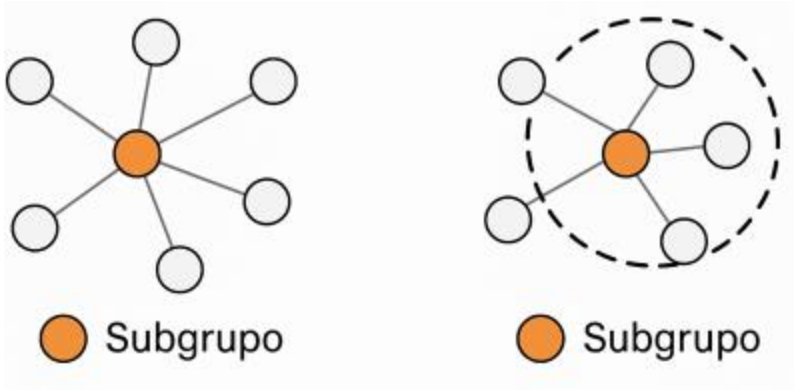
Assessor Social

Henrique Rotsen
Alexandre Cassimiro Silva Araújo
Wesley Marques Daniel Chaves
Caíque Bruno Fortunato
Arthur Yochio R. Codama

Recapitulando...

O artigo apresenta o método Local Subgroup Discovery (LSD).

A principal diferença dessa abordagem é que, **em vez de comparar um subgrupo com toda a rede (o padrão global), ela o compara com um "grupo de pares" local e mais relevante** com o objetivo de **encontrar subgrupos "localmente excepcionais"**



Aplicações do método LSD

- **Biologia Molecular** - OGBG-MolPCBA
  - Identifica subgrupos de moléculas com padrões estruturais únicos
    - Potencial para acelerar descobertas de medicamentos
- **Redes sociais** – Twitch PT
  - Detecta streamers com comportamento incomum (Twitch PT)
    - Pode apoiar moderação, marketing e análise de comunidades

- **Organização da Informação** – WebKB Cornell
  - Classifica páginas web com padrões diferentes (WebKB Cornell)
    - Útil para detectar spam, phishing e estruturar grandes bases

- **Fraudes e anomalias** – Redes bancárias
  - Encontra padrões anômalos em transações
    - Ex.: lavagem de dinheiro em subgrupos bancários

Outros cenários possíveis

- **Cibersegurança**: Detecta acessos e tráfegos incomuns por grupo de dispositivos
- **Cadeia de Suprimentos**: Descobre fornecedores que estão se saindo diferente dos outros parecidos com eles.
  - Atrasando entregas ou apresentando mais problemas do que o normal
- **Segmentação de Clientes**: Captura preferências atípicas dentro de nichos de mercado
- **Grupos Radicais e Desinformação:** Detecta clusters que compartilham conteúdos extremistas localmente
- **Transporte e ecologia:** Identifica grupos de espécies em um ambiente natural que estão interagindo de forma estranha ou sofrendo quedas fora do padrão local (possível problema ambiental)

Problemas de Privacidade no uso do LSD

Invasão de Privacidade

- A combinação entre os dados dos atributos e a estrutura das conexões na rede pode permitir a reidentificação de indivíduos
- Mesmo quando os dados são anonimizados, o contexto local em que o nó está inserido pode revelar quem é a pessoa
  - Exemplo: em redes sociais, o padrão de interação de um usuário e as características associadas ao seu perfil podem ser suficientes para identificá-lo.

Identificação Explícita de Grupos

- Subgrupos “excepcionais” podem ser usados para:
  - Vigilância sem consentimento
  - Campanhas de marketing predatório
  - Negação de acesso a serviços com base em padrões de comportamento

Problemas de Privacidade no uso do LSD

- **O comportamento de um grupo pequeno pode implicar diretamente seus membros**
  - Exemplo: imagine um fórum de mães de primeira viagem. Um pequeno grupo começa a postar com mais frequência sobre ansiedade pós-parto. O sistema pode rotular esse grupo como “de risco”, mesmo que estejam apenas buscando apoio. Isso pode levar a bloqueios, alertas automáticos ou vigilância desnecessária.
- **Ferramentas externas (cookies, redes sociais, logs) podem combinar dados de LSD para tracking avançado**
  - Exemplo: Um site de compras online identifica um subgrupo de clientes com interesse fora do padrão em produtos de saúde mental. Ao cruzar essas informações com cookies de redes sociais, esses clientes passam a receber anúncios específicos em outras plataformas, mesmo sem nunca terem pesquisado o tema diretamente nesses espaços.

Equidade, Discriminação e Igualdade

Discriminação Algorítmica

- Atributos como raça, gênero ou status socioeconômico podem estar embutidos nos dados
- O LSD pode destacar subgrupos “anômalos” que, na prática, são grupos historicamente marginalizados
  - Exemplo: Um subgrupo de candidatos a empregos de uma determinada faixa etária, com um histórico de carreira não-linear, pode ser automaticamente desconsiderado em sistemas automatizados.

Estigmatização e Exclusão Social

- Rótulos como “excepcional”, “fora do padrão” ou “desviante” podem ser mal interpretados
- Isso pode gerar exclusão de oportunidades, tratamentos ou políticas públicas
  - Exemplo: pacientes com doenças raras classificados como “fora do perfil” e não elegíveis para determinado tratamento

Equidade, Discriminação e Igualdade

Desigualdade no acesso ou tratamento

- Subgrupos identificados como “exceções” podem ser tratados de forma desigual por sistemas automáticos
  - Exemplo: em serviços sociais, famílias com perfis atípicos em relação à vizinhança podem ser priorizadas negativamente ou até ignoradas.

Falha ao Reconhecer Desigualdades Estruturais

- O método pode “marcar” comportamentos causados por falta de acesso a recursos, não por desvio real
  - Exemplo: alunos com notas baixas em uma escola vulnerável podem ser vistos como “casos extremos”, quando na verdade refletem problemas sistêmicos

Boas práticas recomendadas...

- Adoção de critérios éticos e transparentes no uso do método
- Garantia de supervisão humana e contextualização dos resultados
- Implementação de salvaguardas contra uso indevido, manipulação ou exclusão social

Hacker

Gabriel Tonioni Duarte
Enilda Alves Coelho
Fábio César Marra Filho
Iasmin Correa Araujo
Jose Vinicius de Lima Massarico
Larissa Duarte Santana
Marcelo Lommez Rodrigues de Jesus

Repositório 👉 https://github.com/TUeEMM/LSD-ATNG

LSD-ATNG Public

main · 1 Branch · 1 Tag · Go to file t · Add file · Code

cvheinrich chore: add requirements.txt — 29ef743 · 4 months ago — 42 Commits

data	chore: finalize code	4 months ago
.gitattributes	Initial commit	8 months ago
.gitignore	Initial commit	8 months ago
LICENSE.txt	chore: add license	4 months ago
OGBG-MolPCBA_dataset.ipynb	chore: add requirements.txt	4 months ago
OGBG-MolPCBA_inspecting.ipynb	chore: add requirements.txt	4 months ago
OGBG-MolPCBA_visualization.ipynb	chore: add requirements.txt	4 months ago
README.md	chore: add requirements.txt	4 months ago
Twitch-PT_dataset.ipynb	chore: add requirements.txt	4 months ago
WebKB-Cornell_dataset.ipynb	chore: add requirements.txt	4 months ago
ablation_metrics.py	docs(ablationStudy): add docstrings to ablation study	4 months ago
methods.py	chore: finalize code	4 months ago
requirements.txt	chore: add requirements.txt	4 months ago

Languages

Jupyter Notebook 99.1% · Python 0.9%

methods.py

- func top_k_func
- func get_attributes
- func ranking
- func q_func
- func discovery
- func process_node
- func find_groups

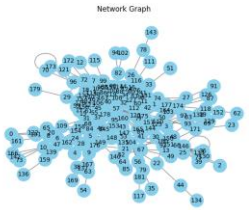
ablation_metrics.py

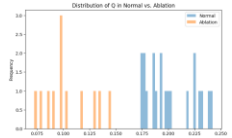
- func compare_subgroups
- func evaluate_pattern_subgroups

requirements.txt

```
numpy==1.26.4
pandas==2.2.3
tqdm==4.67.1
networkx==3.4.2
gower==0.1.2
matplotlib==3.10.0
ogb==1.3.6
torch_geometric==2.6.1
notebook==7.3.2
```

*.ipynb





Dados de Entrada 👉

```
find_groups(G, 20, lu, ablation_mode=False)
```

Grafo (G)

👇

```
import networkx as nx
```

```
edge_index = data.edge_index
num_nodes = data.num_nodes
G = nx.Graph()

G.add_nodes_from(range(num_nodes))

edges = list(zip(edge_index[0].tolist(), edge_index[1].tolist()))
G.add_edges_from(edges)
```

20

👇

O método irá retornar os 20 melhores grupos encontrados.

Lookup Table (lu) - Atributos dos Nós

👇

```
attributes = graph['node_feat']
lu = pd.DataFrame(attributes)
lu['target'] = binary_target == 1
lu.head()
```

	0	1	2	3	4	5	6	7	8	9	...	1694	1695	1696	1697	1698	1699	1700	1701	1702	target
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	False
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	False
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	True
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	False
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	False

5 rows × 1704 columns

Dados de Saída 👉 `find_groups(G, 20, lu, ablation_mode=False)`

Melhores Grupos Encontrados

👇

node	rho	sigma	q	ranks	reference	subgroup
78	22	4	0.242274	[(78, True), (26, False), (143, True), (57, Tr...	[78, 26, 143, 57, 170, 119, 14, 59, 168, 18, 5...	[78, 26, 143, 57]
60	24	4	0.238145	[(60, True), (57, True), (119, False), (59, Tr...	[60, 57, 119, 59, 168, 18, 132, 174, 157, 131,...	[60, 57, 119, 59]
76	36	4	0.231481	[(76, True), (57, True), (60, True), (127, Tru...	[76, 57, 60, 127, 119, 168, 174, 59, 45, 14, 1...	[76, 57, 60, 127]
135	36	11	0.228925	[(135, True), (64, True), (5, False), (57, Tru...	[135, 64, 5, 57, 22, 110, 18, 103, 45, 92, 59,...	[135, 64, 5, 57, 22, 110, 18, 103, 45, 92, 59]
165	36	5	0.225677	[(165, False), (149, True), (67, True), (97, T...	[165, 149, 67, 97, 57, 174, 110, 119, 144, 157...	[165, 149, 67, 97, 57]
30	43	6	0.224416	[(30, False), (25, True), (67, True), (146, Tr...	[30, 25, 67, 146, 57, 66, 110, 13, 144, 174, 1...	[30, 25, 67, 146, 57, 66]
113	33	3	0.219281	[(113, True), (57, True), (60, True), (119, Fa...	[113, 57, 60, 119, 59, 168, 40, 157, 174, 18, ...	[113, 57, 60]
157	19	3	0.202165	[(157, False), (48, True), (57, True), (119, F...	[157, 48, 57, 119, 174, 18, 168, 59, 131, 170,...	[157, 48, 57]
59	20	3	0.200104	[(59, True), (57, True), (119, False), (168, F...	[59, 57, 119, 168, 18, 131, 170, 98, 157, 174,...	[59, 57, 119]
133	40	3	0.198549	[(133, True), (146, True), (66, True), (39, Fa...	[133, 146, 66, 39, 144, 49, 75, 3, 130, 41, 30...	[133, 146, 66]
79	28	5	0.193178	[(79, False), (35, True), (104, True), (56, Fa...	[79, 35, 104, 56, 57, 43, 174, 119, 144, 124, ...	[79, 35, 104, 56, 57]

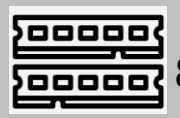
Roteiro de Execução

1. Clonar o repositório ou baixar o código fonte em releases.
2. Criar um grafo a partir de um conjunto de dados.
3. Criar uma tabela com os atributos e o rótulo binário de cada nó.
4. Executar o método *find_groups(...)*.

Recursos Gastos 👉 Ryzen 5 1600x 3.6GHz • 16 GB RAM DDR4 2666MHz • Twitch Dataset

- 1912 nós com 128 atributos e 1 rótulo.
- 64510 arestas.


16 minutos


8 GB

Referências

Referências bibliográficas