روش اول**Louvain Method** :

الگوریتم **Louvain**یک روش سلسله‌مراتبی (hierarchical) برای بیشینه‌سازی **modularity**است که توسط Blondel و همکارانش در سال 2008 معرفی شد.

ابتدا هر گره در یک جامعه مجزا قرار می‌گیرد.

گره‌ها به صورت محلی بین جوامع جابه‌جا می‌شوند تا بیشترین افزایش در modularity حاصل شود.

جوامع حاصل به عنوان گره‌های جدید در یک شبکه فشرده شده، در نظر گرفته شده و مراحل تکرار می‌شود.

مزایا:

- مقیاس‌پذیر (مناسب برای شبکه‌های بزرگ)
- اجرای سریع
- **معیار اصلی بهینه‌سازی:**
- **Modularity (Q)** – معیاری برای مقایسه چگالی یال‌ها در داخل جامعه نسبت به بیرون

معیارModularity

برای شبکه‌ای به صورت
$G = (V, E)$
که در آن:

- A_{ij}: عنصر (i, j) از ماتریس مجاورت است که مقدار آن برابر 1 است اگر بین رأس i و j یال وجود داشته باشد، و 0 در غیر این صورت.
- $k_i = \Sigma_j A_{ij}$: درجه‌ی رأس i است (تعداد یال‌های متصل به آن).
- $m = |E|$: تعداد کل یال‌ها در گراف.
- $\delta(c_i, c_j)$: تابع دلتای کرونکر که مقدار آن برابر 1 است اگر رأس‌های i و j در یک جامعه قرار داشته باشند، و در غیر این صورت 0 است.

فرمول محاسبه‌ی مدولاریتی به صورت زیر تعریف می‌شود:

$$Q = (1 / 2m) * \Sigma_{i,j} [A_{ij} - (k_i * k_j) / (2m)] * \delta(c_i, c_j)$$

تفسیر:
Modularityیا مدولاریتی، معیاری است برای ارزیابی کیفیت تقسیم گراف به جوامع. این معیار تفاوت بین:

- تعداد واقعی یال‌های درون هر جامعه
-

- تعداد مورد انتظار یال‌ها بین همان گره‌ها در یک شبکه‌ی تصادفی با توزیع درجات مشابه را اندازه‌گیری می‌کند

اگر مقدار Q بزرگ باشد (مثلاً نزدیک به 1)، نشان‌دهنده‌ی این است که جوامع داخلی دارای اتصال زیادی هستند و ارتباط بین جوامع کم است.

حد تفکیک:(Resolution Limit)
مدولاریتی نمی‌تواند جوامع بسیار کوچک را در گراف‌های بزرگ تشخیص دهد. به این محدودیت «Resolution Limit» گفته می‌شود. یعنی ممکن است چند جامعه‌ی کوچک با ساختار واضح، توسط الگوریتم‌هایی که Modularity را بهینه می‌کنند، در یک جامعه‌ی بزرگ‌تر ادغام شوند، فقط به این دلیل که مقدار Q افزایش یابد.

ایده‌ی کلی

۱ .**مرحله‌ی محلی**:

- هر راس در جامعه‌ی خود است.
- برای هر راس i، همسایه‌هایش را پیمایش کرده و در صورت افزایش Q، آن را به جامعه‌ای دیگر منتقل می‌کند.
- تکرار تا تثبیت هیچ جابه‌جایی که Qرا افزایش دهد وجود نداشته باشد.
  ۲ .**مرحله‌ی فشرده‌سازی**:
- هر جامعه را به یک «ابرراس» تبدیل می‌کنیم.
- ماتریس مجاورت جدید را می‌سازیم (وزن یال‌ها جمع درجات بین جامعه‌ها).
- دوباره مرحله‌ی محلی روی این شبکه‌ی فشرده اعمال می‌شود.
  ۳ .تا زمانی ادامه می‌یابد که هیچ بهبودی در Qرخ ندهد.

نکات تکمیلی

- **ورژن** :**Leiden** اصلاحات برای رفع مسأله اجتماع‌های ضعیف پایداریافته
- **پارامتر** :**Resolution γ** امکان تغییر حساسیت مدولاریتی به اندازه‌ی جوامع
- **پایداری** :کاملاً تصادفی نیست (اثر ترتیب گره‌ها کم است)

روش دوم**Girvan-Newman Algorithm** :

این الگوریتم مبتنی بر حذف تدریجی یال‌هایی با بیشترین **betweenness centrality** (یال‌هایی که روی بیشترین مسیرهای کوتاه قرار دارند) است.

مراحل کلی:

1. محاسبه betweenness برای تمام یال‌ها
2. حذف یالی با بیشترین betweenness
3. بازمحاسبه و تکرار تا شکسته شدن شبکه به چند بخش مجزا
4. در هر مرحله، modularity اندازه‌گیری شده و مرحله با بهترین Q به‌عنوان بهترین تفکیک جامعه انتخاب می‌شود

نقاط قوت و ضعف

- **مزایا**
  - شفاف و قابل فهم
  - انتخاب مرحله‌ی بهینه بر اساس Q
- **معایب**
  - هزینه محاسباتی بسیار بالا
  - بازمحاسبه‌ی مکرر Betweenness
  - عدم مقیاس‌پذیری به شبکه‌های بزرگ‌تر از چند هزار راس

مقایسه عددی روی گراف Karate Club

مقیاس‌پذیری	زمان اجرا (تقریبی)	Q نهایی	تعداد جوامع	روش
شبکه‌های کوچک	0.15 ثانیه	0.402	4	Girvan–Newman
شبکه‌های بزرگ	0.01 ثانیه	0.419	4	Louvain

جمع‌بندی :

1. شبکه‌های کوچک (≥ هزار راس)
   - اگر به تفسیر گام‌به‌گام و شفافیت هر حذف یال نیاز دارید → Girvan–Newman
2. شبکه‌های متوسط تا بزرگ (≤ هزار راس)
   - برای دستیابی به سرعت و مدولاریتی بالاتر → Louvain

3. دقت در جوامع ریز
   - حتماً از پارامتر Resolution γ یا روش‌های چندمعیاره مثل multilayer Louvain استفاده کنید
4. ابزارهای عملی
   - کتابخانه‌ی NetworkX :پیاده‌سازی ساده Girvan–Newman
   - python-louvain یا igraph :نسخه‌ی بهینه Louvain / Leiden

پیاده‌سازی نمونه (پایتون)

```
import networkx as nx
import community as community_louvain

G = nx.karate_club_graph()

# Louvain
partition = community_louvain.best_partition(G)
Q_louvain = community_louvain.modularity(partition, G)

# Girvan–Newman (با اندازه ثابت 4 جامعه)
from networkx.algorithms.community import girvan_newman
comp = girvan_newman(G)
top_level_communities = tuple(sorted(c) for c in next(comp))
Q_gn = nx.community.modularity(G, top_level_communities)

print("Modularity Louvain:", Q_louvain)
print("Modularity GN:", Q_gn)
```

با این کد می‌توانید به سادگی روی هر شبکه‌ی دلخواه خود Q دو روش را مقایسه کنید