**Отчёт по анализу графа**

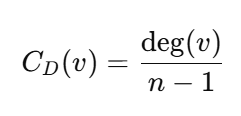
**Введение**

В данном отчёте представлен поэтапный анализ неориентированного графа, заданного списком рёбер, с использованием Python-библиотеки **NetworkX** и инструментов визуализации. Цели работы:

* Загрузить и построить граф из edge‑list.
* Рассчитать и интерпретировать основные метрики центральности: **degree**, **closeness**, **betweenness**.
* Выделить и визуализировать кластеры (компоненты связности).
* Экспортировать результаты в формате GraphML для дальнейшей работы.

**Теоретические понятия**

**Degree Centrality** (степень вершины) — число непосредственных связей узла с другими узлами графа. Нормированная формула:



Графически: узлы с большим числом соседей выступают в роли локальных «хабов».

**Closeness Centrality** (центральность близости) показывает, насколько быстро узел может достичь все остальные узлы через кратчайшие пути. Вычисляется как:

Изображение выглядит как Шрифт, рукописный текст, белый, линия

Контент, сгенерированный ИИ, может содержать ошибки.

где d(v,u)d(v,u) — длина кратчайшего пути между vv и uu. Чем выше значение, тем «ближе» вершина к остальному графу.

**Betweenness Centrality** (центральность посредничества) измеряет долю всех кратчайших путей между парой узлов, которые проходят через данную вершину:

Изображение выглядит как Шрифт, белый, текст, рукописный текст

Контент, сгенерированный ИИ, может содержать ошибки.

где σst — число кратчайших путей от ss к tt, а σst(v) — количество таких путей, что проходят через vv. Высокое значение свидетельствует о роле «моста» между сообществами.

**Кластеры (компоненты связности)**

**Кластеризация** в неориентированных графах часто реализуется через поиск компонент связности — максимальных подграфов, в которых любые две вершины соединены путем. Каждому кластеру присваивается уникальный номер, по которому узлы можно раскрасить и визуально отделить группы.

**Методы**

1. **Загрузка графа:** чтение edge‑list из graph.txt с помощью nx.read\_edgelist().
2. **Базовая статистика:** число узлов и рёбер — G.number\_of\_nodes(), G.number\_of\_edges().
3. **Вычисление центральностей:**
   * nx.degree\_centrality(G)
   * nx.closeness\_centrality(G)
   * nx.betweenness\_centrality(G, k=100, seed=42) (приближённый)
4. **Кластеризация:** поиск компонент связности nx.connected\_components(G) и присвоение номера компоненты каждому узлу.
5. **Визуализация в Python:**
   * Макет spring\_layout.
   * Размер узлов пропорционально degree.
   * Цвет узлов по номеру кластера.
6. **Экспорт:** сохранение в GraphML: nx.write\_graphml(G, "graph.graphml").

**Результаты**

**Общие характеристики**

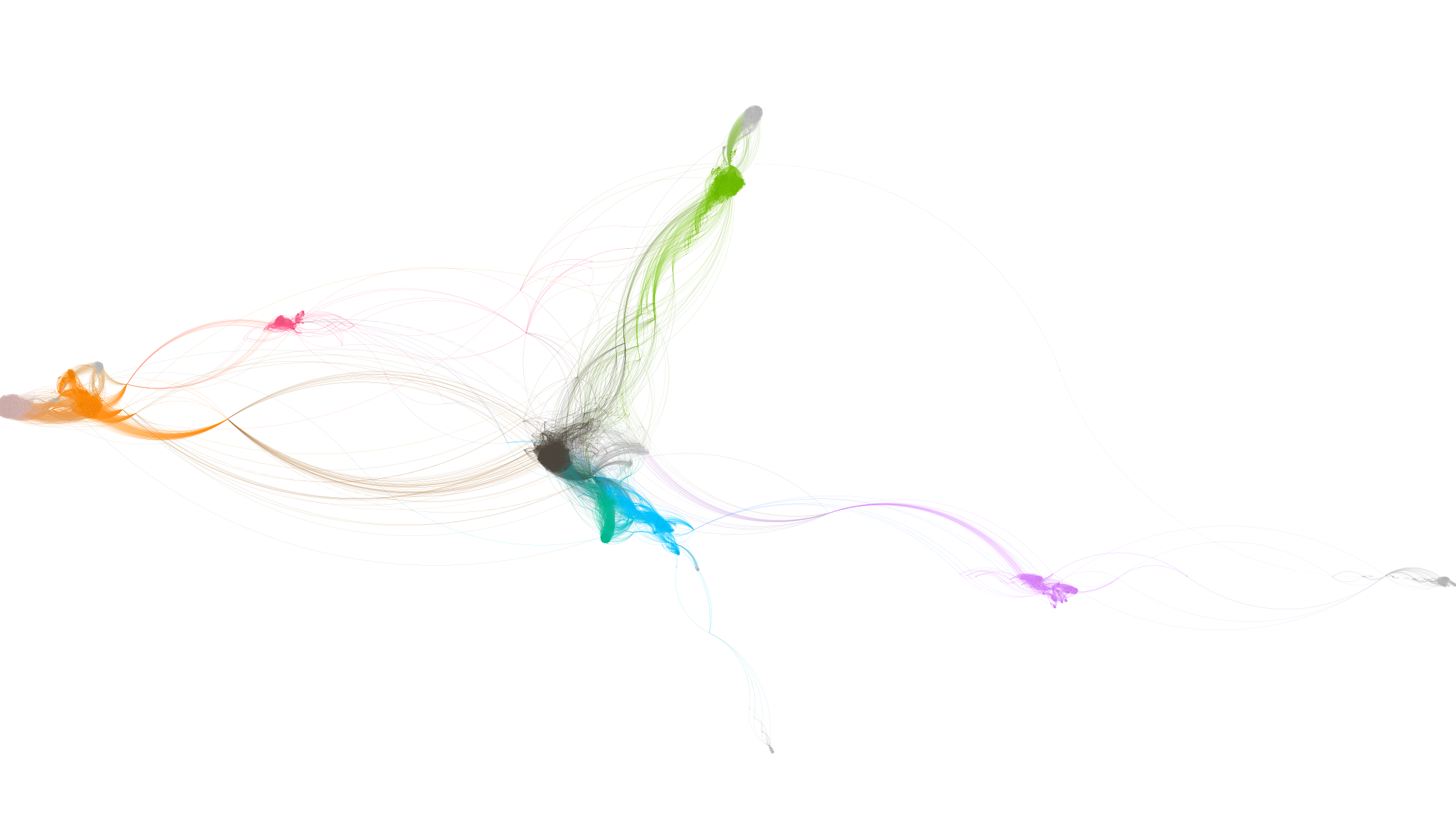
* **Узлов (n):** G.number\_of\_nodes()
* **Рёбер (m):** G.number\_of\_edges()

**Топ-5 узлов по центральностям**

|  |  |  |
| --- | --- | --- |
| **Метрика** | **Узел** | **Значение** |
| **Degree** | **107;**  **1684;**  **1912;**  **3437;**  **0;** | **0.2588;**  **0.1961;**  **0.1870;**  **0.1355;**  **0.0859;** |
| **Closeness** | **107;**  **58;**  **428;**  **563;**  **1684;** | **0.4597;**  **0.3974;**  **0.3948;**  **0.3939;**  **0.3936;** |
| **Betweenness** | **107;**  **1684;**  **1912;**  **3437;**  **0;** | **0.4943;**  **0.1684;**  **0.2360;**  **0.1875;**  **0.1574;** |

**Визуализация**

* **Python:** граф с размерами узлов по степеням и цветом по кластеру.
* **Gephi:** импортированный GraphML-файл с раскраской по Modularity Class и размерами по degree.



**Заключение**

Работа продемонстрировала полный цикл анализа: от загрузки и вычисления метрик в NetworkX, через интерпретацию ключевых показателей центральности и кластеров, до экспорта готового GraphML для визуализации и дальнейшего исследования в Gephi.

**Приложение**

* main.py — код анализа в NetworkX.
* graph.graphml — экспортированный граф с атрибутами.