Сетевое программирование

Лекция 12

Визуализация сетей. Методы визуализации в NetworkX. Алгоритмы Layout. Интеграция с библиотекой Matplotlib

Усіпов Н.М., PhD · КазНУ имени аль-Фараби

Цель и план лекции

Цель

Освоение средств визуализации графов в NetworkX и Matplotlib, изучение алгоритмов укладки, а также формирование умения графически отображать атрибуты сети и обоснованно интерпретировать полученные изображения.

1. Введение
2. Средства визуализации в NetworkX
3. Алгоритмы укладки (Layout)
4. Принципы построения информативных изображений
5. Интеграция с Matplotlib
6. Оценка качества укладки
7. Математические основы укладок
8. Принципы восприятия и оформления

1 Введение

- Визуализация является важным инструментом анализа сетей. Правильно построенное изображение позволяет быстро заметить концентраторы, изолированные сегменты, обходные пути и ошибки в исходных данных.
- Вместе с тем изображение графа может вводить в заблуждение. Положение узлов на плоскости в большинстве случаев произвольно: одна и та же сеть при разных укладках выглядит совершенно по-разному.

2 Средства визуализации в NetworkX

NetworkX предоставляет базовые функции рисования, которые основаны на библиотеке Matplotlib:

draw и draw_networkx

Функции для быстрого построения всего графа с общими параметрами оформления.

draw_networkx_nodes

Отображает только узлы и позволяет задавать индивидуальные размер, цвет и форму.

draw_networkx_edges

Отображает рёбра с заданной толщиной, цветом и стилем, а также стрелки для ориентированных графов.

draw_networkx_labels

Добавляет подписи узлов.

Раздельное применение этих функций позволяет отобразить на рисунке значения атрибутов.

2 Средства визуализации в NetworkX (продолжение)

draw_networkx_edge_labels

Выводит подписи рёбер, например значения задержки или полосы.

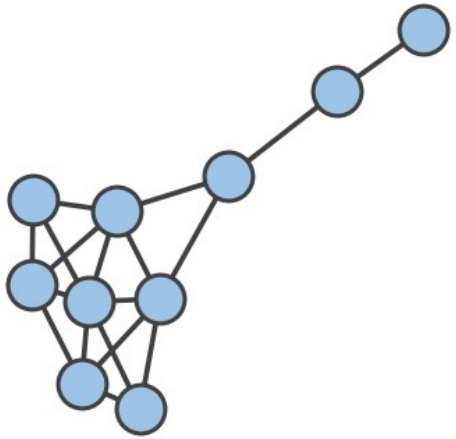
Раздельное применение этих функций позволяет отобразить на рисунке значения атрибутов.

3 Алгоритмы укладки (Layout)

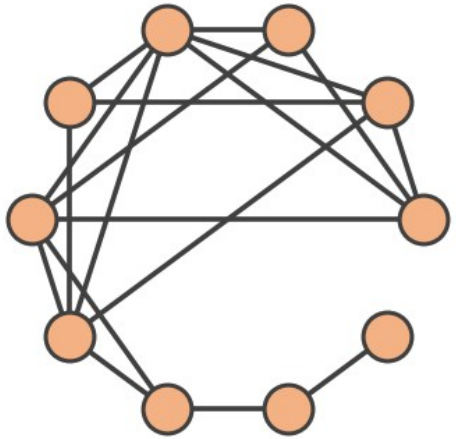
- Алгоритм укладки вычисляет координаты узлов. Выбор алгоритма зависит от структуры сети и цели изображения.

3 Алгоритмы укладки (Layout) (продолжение)

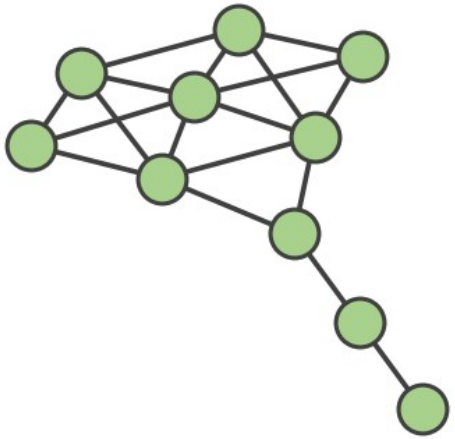
spring_layout



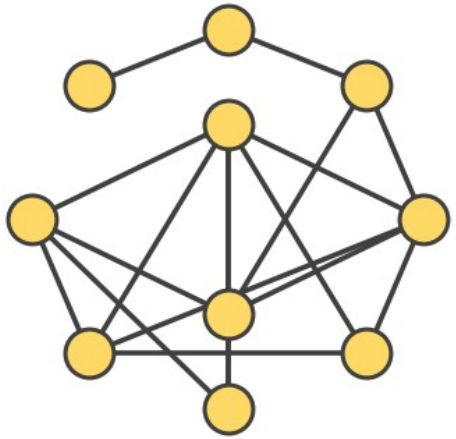
circular_layout



kamada_kawai_layout



shell_layout



Один и тот же граф при различных алгоритмах укладки

3 Алгоритмы укладки (Layout) (продолжение)

Силовая укладка (spring_layout)

Реализует алгоритм Фрухтермана–Рейнгольда (1991). Связанные узлы притягиваются, как при действии пружин, а все узлы взаимно отталкиваются.

Укладка Камады–Каваи (kamada_kawai_layout)

Стремится сделать евклидово расстояние между узлами пропорциональным расстоянию в графе.

Круговая укладка (circular_layout)

Размещает узлы по окружности. Подходит для кольцевых топологий и обзора плотности соединений.

Оболочечная укладка (shell_layout)

Размещает узлы на концентрических окружностях, заданных пользователем. Это наглядно отражает уровни иерархии – ядро, агрегацию и доступ.

3 Алгоритмы укладки (Layout) (продолжение)

Спектральная укладка (spectral_layout)

Использует собственные векторы матрицы Лапласа. Метод быстрый и подчёркивает кластерную структуру.

Укладка по заданным координатам

Если известны географические координаты устройств, их передают непосредственно в словаре `pos`.

4 Принципы построения информативных изображений

Отображение атрибутов

Размер узла может обозначать степень или загрузку, цвет – тип оборудования, а толщина ребра – пропускную способность.

Подписи

Узлы и рёбра необходимо подписывать только в том случае, если это не загромождает изображение.

Цветовая шкала

Для непрерывных значений необходимо добавлять цветовую шкалу с указанием величины и единиц измерения.

Воспроизводимость

Укладку следует вычислять однажды и применять одни и те же координаты для всех рисунков, которые сравниваются между собой.

4 Принципы построения информативных изображений (продолжение)

Масштаб сети

При размере более нескольких сотен узлов изображение теряет наглядность. В этом случае используют фильтрацию, выделяют наибольшую компоненту, отображают только сообщества либо применяют специализированные программы, такие как Gephi и Cytoscape.

5 Интеграция с Matplotlib

- Использование Matplotlib позволяет сохранять рисунки в файлы высокого разрешения, создавать составные рисунки и объединять графы с диаграммами.

5 Визуализация атрибутов сети

Листинг 12.1.

```
import matplotlib.pyplot as plt
import networkx as nx

G = nx.Graph()
G.add_edge("Ядро1", "Ядро2", bw=100)
G.add_edge("Ядро1", "Агр1", bw=40)
G.add_edge("Ядро2", "Агр2", bw=40)
G.add_edge("Агр1", "Агр2", bw=10)
for i in range(1, 5):
    G.add_edge("Агр1" if i < 3 else "Агр2", f"Дост{i}", bw=1)

color = {"Я": "tomato", "А": "gold", "Д": "skyblue"}
node_color = [color[v[0]] for v in G]
node_size = [300 * G.degree(v) for v in G]
width = [1 + d["bw"] / 25 for _, _, d in G.edges(data=True)]
```

5 Визуализация атрибутов сети (продолжение)

Листинг 12.1.

```
pos = nx.spring_layout(G, seed=42)
fig, ax = plt.subplots(figsize=(7, 5))
nx.draw_networkx_edges(G, pos, ax=ax, width=width)
nx.draw_networkx_nodes(G, pos, ax=ax, node_color=node_color,
                       node_size=node_size, edgecolors="black")
nx.draw_networkx_labels(G, pos, ax=ax, font_size=9)
ax.set_title("Иерархическая сеть (толщина – полоса)")
ax.set_axis_off()
fig.savefig("network.png", dpi=300, bbox_inches="tight")
print("Рисунок сохранён, узлов:", len(G))
```

5 Результат: Визуализация атрибутов сети

```
Рисунок сохранён, узлов: 8
```

Важно отметить, что сначала рисуются рёбра, а затем узлы: иначе линии будут перекрывать вершины.

6 Оценка качества укладки

- Качество укладки можно оценивать не только визуально. Одним из показателей служит соответствие между расстоянием на рисунке и расстоянием в графе.

6 Сравнение алгоритмов укладки

Листинг 12.2.

```
import itertools
import numpy as np

G = nx.krackhardt_kite_graph()
d = dict(nx.all_pairs_shortest_path_length(G))
pairs = list(itertools.combinations(G, 2))

def quality(pos):
    x = [np.linalg.norm(pos[u] - pos[v]) for u, v in pairs]
    y = [d[u][v] for u, v in pairs]
    return np.corrcoef(x, y)[0, 1]

layouts = {"spring": nx.spring_layout(G, seed=2),
           "kamada_kawai": nx.kamada_kawai_layout(G),
           "circular": nx.circular_layout(G),
           "spectral": nx.spectral_layout(G)}
for name, pos in layouts.items():
    print(f"{name:13} корреляция = {quality(pos):.3f}")
```

6 Результат: Сравнение алгоритмов укладки

```
spring         корреляция = 0.969
kamada_kawai   корреляция = 0.982
circular       корреляция = 0.330
spectral       корреляция = 0.716
```

Приведённые значения относятся к одному небольшому графу и одной случайной инициализации силовой укладки, поэтому их нельзя распространять на все сети.

7 Математические основы укладок

- Силовая укладка представляет сеть в виде физической системы. Фрухтерман и Рейнгольд определили силы притяжения и отталкивания как функции расстояния d:

$$f_a(d) = d^2 / k, \quad f_r(d) = k^2 / d,$$

- где k – оптимальное расстояние, обычно вычисляемое как k = C√(S/n), где S – площадь области рисования, а n – число вершин.
- В алгоритме Камады–Каваи расчёт основан на минимизации энергии упругой системы:

$$E = \sum_{i<j} \tfrac{1}{2}\, k_{ij}\, (|p_i - p_j| - L\, d_{ij})^2,$$

- где d_{ij} – расстояние в графе, L – желаемая длина ребра, а k_{ij} обратно пропорционален d_{ij}^2.

7 Математические основы укладок (продолжение)

- В спектральной укладке координаты задаются собственными векторами, соответствующими двум наименьшим ненулевым собственным значениям матрицы Лапласа.

8 Принципы восприятия и оформления

- При создании изображения следует руководствоваться правилами, которые учитывают особенности зрительного восприятия.
- Важно уменьшать визуальный шум. При большом количестве рёбер помогают: полупрозрачность линий, исключение несущественных связей, объединение рёбер в пучки, удаление вершин с малой степенью.
- Цветовая схема должна учитывать людей с нарушениями цветового зрения: следует избегать пары красный–зелёный как единственного способа различия, а применять равномерные схемы, такие как `viridis`.
- Для интерактивного изучения топологии используются библиотеки Plotly, PyVis и инструмент Gephi, позволяющие приближать отдельные участки, скрывать слои и получать информацию о выбранном узле.

8 Принципы восприятия и оформления (продолжение)

- Важной задачей является визуализация результатов расчётов. Например, на топологию можно нанести загрузку каналов цветом, отметить узлы, которые разделяют сеть, показать сообщества и выделить маршрут.

9 Разбор примера

1. Найдём равновесное расстояние между двумя соединёнными вершинами силовой модели.
2. Условие равенства сил притяжения и отталкивания $d^2/k = k^2/d$ приводит к $d^3 = k^3$, то есть $d = k$.
3. Таким образом, параметр k действительно задаёт длину ребра в равновесии.
4. Для единичной площади рисунка при $n = 25$ вершинах и $C = 1$ получаем $k = \sqrt{(1/25)} = 0{,}2$.
5. Выберем способ изображения иерархической сети из 3 ядерных, 6 агрегирующих и 24 узлов доступа.
6. Подходит оболочечная укладка с тремя окружностями: ядро внутри, агрегация посередине и доступ снаружи.
7. Цветом целесообразно обозначить уровень, толщиной ребра – полосу.

10 Вопросы для обсуждения на занятии

Выбор укладки

Какое изображение вы выбрали бы для представления руководству и для анализа инженером?

Искажения

Может ли расположение узла в центре рисунка служить показателем его важности?

Сравнение

Как сохранить сопоставимость рисунков сети за разные месяцы?

Контрольные вопросы

1. Почему изображение графа может привести к неверным выводам?
2. Какие функции NetworkX применяются для раздельного отображения элементов графа?
3. Опишите принцип действия силовой укладки. Для чего задаётся параметр seed?
4. В чём заключается идея алгоритма Камады–Каваи?
5. Какую укладку целесообразно использовать для иерархических и кольцевых сетей?
6. Как отобразить на рисунке атрибуты узлов и рёбер?
7. Какие подходы применяются при визуализации очень больших сетей?
8. Какие показатели позволяют количественно оценить качество укладки?

Список литературы

1. Fruchterman T.M.J., Reingold E.M. Graph drawing by force-directed placement // Software: Practice and Experience. – 1991. – Vol. 21, No. 11. – P. 1129–1164.
2. Kamada T., Kawai S. An algorithm for drawing general undirected graphs // Information Processing Letters. – 1989. – Vol. 31, No. 1. – P. 7–15.
3. Hunter J.D. Matplotlib: A 2D graphics environment // Computing in Science & Engineering. – 2007. – Vol. 9, No. 3. – P. 90–95.
4. Bastian M., Heymann S., Jacomy M. Gephi: an open source software for exploring and manipulating networks // Proc. International AAAI Conference on Weblogs and Social Media. – 2009.
5. Hagberg A., Schult D., Swart P. Exploring network structure, dynamics, and function using NetworkX // Proc. 7th Python in Science Conference (SciPy 2008). – Pasadena, 2008. – P. 11–15.
6. NetworkX Documentation [Электронный ресурс]. – URL: https://networkx.org/documentation/stable/