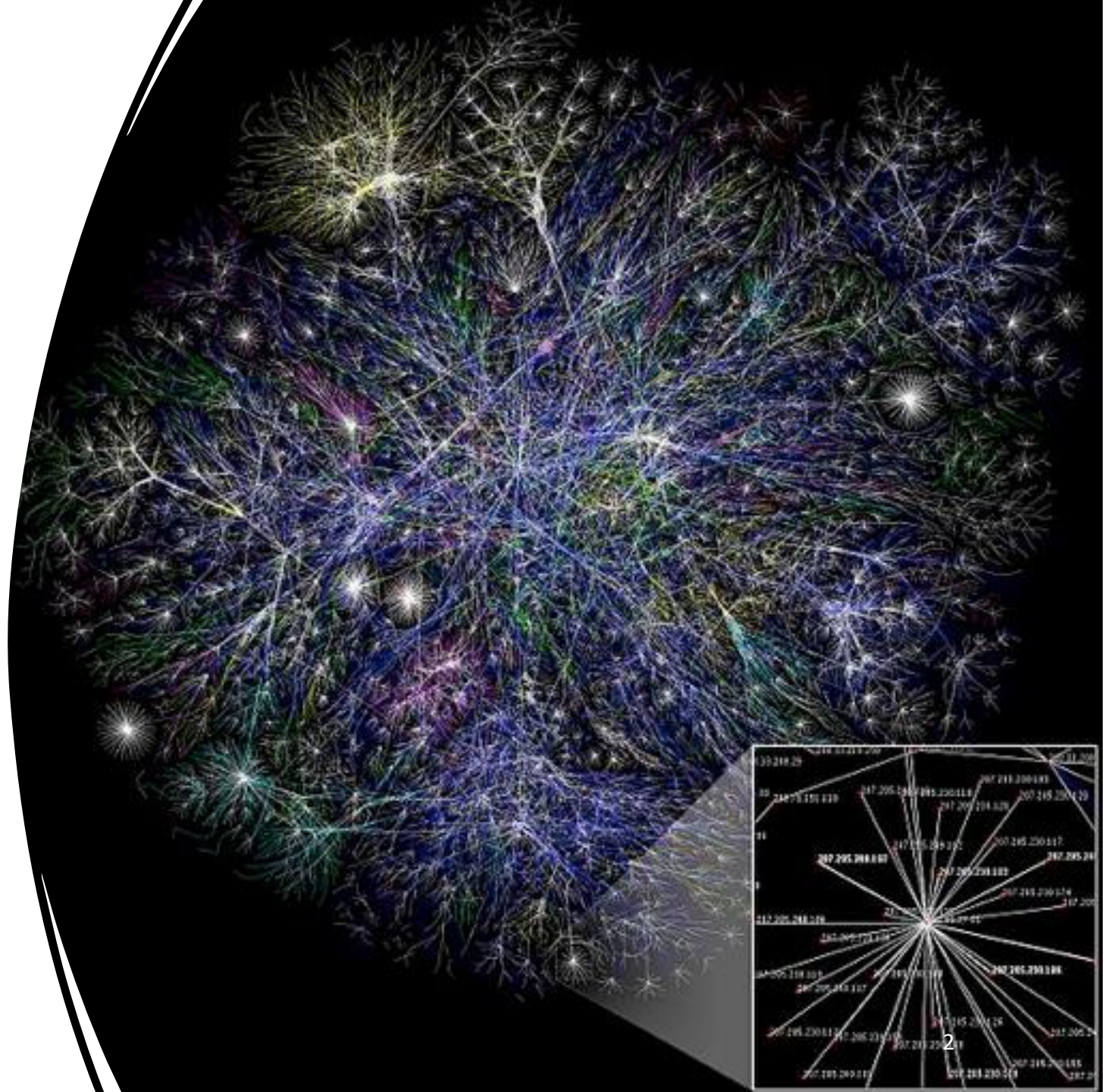


Анализ графов

О чем поговорим

- Основные понятия теории графов
- Библиотека Networkx (CPU)
- Библиотека cuGraph (GPU/Multi-GPU)



Теория сложных сетей

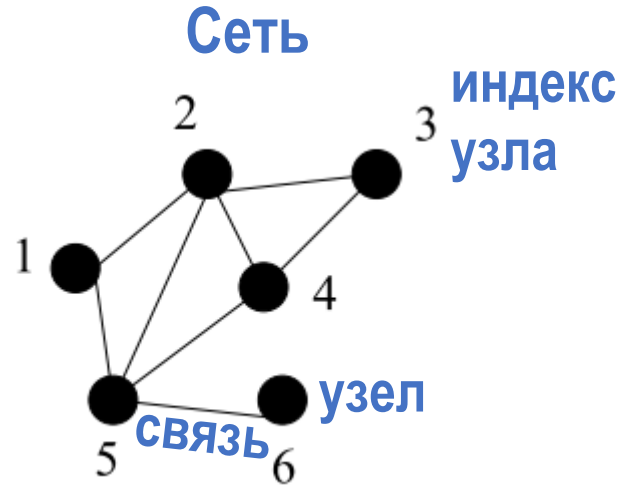
Теория сложных сетей (теория графов) изучает сложные взаимодействующие системы, которые могут быть представлены в виде графа

Примеры:

- Коммуникационные сети (звонки, email...)
- Интернет и веб граф (связь между роутерами, ссылки в интернете)
- Рекомендательные сервисы (сети музыки, фильмов, товаров)
- Социальные сети (друзья, сообщения)
- Семантические сети (связь слов, понятий)
- Технологические, транспортные сети (сети дорог, транспорта, сети электропередач)

Основные понятия теории графов

Сеть(граф) – множество узлов (вершин) и связей (ребер), соединяющий пары узлов.



Матрица смежности сети

$$A = (a_{ij}) = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Список связей сети

(1, 2), (1, 5), (2, 3), (2, 4), (2, 5), (3, 4), (4, 5), (5, 6)

Интегральные характеристики

Количество узлов – $N=6$; количество связей – $L=8$.

Средняя степень узлов сети: $\langle k \rangle = \frac{2L}{N} = 16/6 = 2,67$

Основные понятия теории графов

Ориентированная сеть – сеть, в которой связь имеет направление;

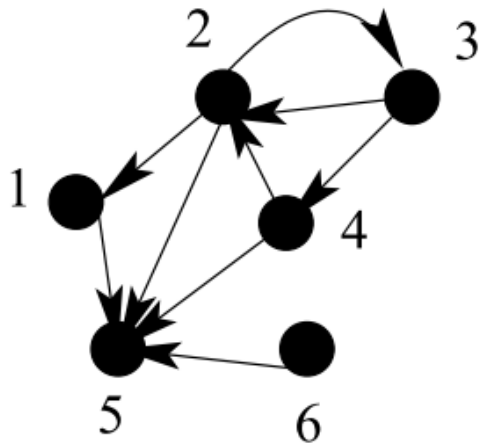
Взвешенная сеть – сеть, в которой ребра имеют вес (например, вес характеризует расстояние);

Степень узла – количество связей у узла (входящая/исходящая степень для ориентированных сетей);

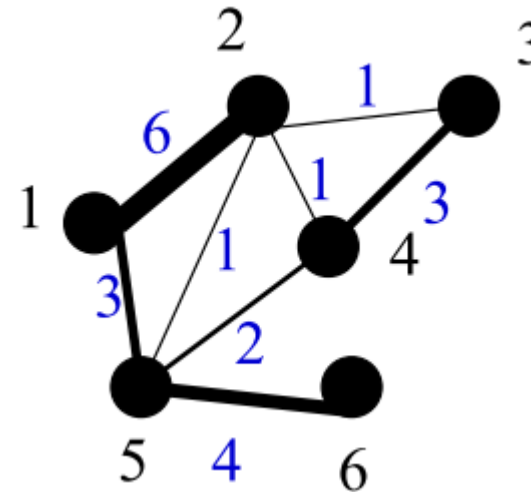
Средняя степень узлов сети - $\langle k \rangle = \frac{2L}{N}$, где N – число узлов, L – количество связей

Расстояние от одного узла до другого – наикратчайший путь от одного узла к другому

Диаметр сети – максимальное расстояние между узлами сети



Ориентированная сеть



Взвешенная сеть

Мера центральности

В графе есть узлы, связи. Но можно ли ответить на вопрос, на сколько конкретный узел (связь) важен для сети?

Важность можно толковать различными способами:

- Узел пропускает большое количество сетевого трафика
- Узел является узким местом сети
- Связь является единственной при соединении двух крупных подграфов
- Узел является наиболее цитируемым
- ...

Основное предположение: важные узлы или связи находятся в центре сети.

Наиболее популярные меры центральности:

- Центральность по степени (degree centrality);
- Центральность по близости (closeness centrality);
- Центральность по посредничеству (betweenness centrality);
- Центральность по собственному вектору (eigenvector centrality).

Меры центральности (наиболее популярные)

Центральность по степени (degree centrality) – центральность узла равна степени узла. Часто степень узла нормируется на максимальное возможное количество связей у узла. (внутренняя и внешняя для направленного графа)

$$C_{DN}(i) = \frac{k_i}{N-1}, C_{DN}(i) \in [0,1]$$

Интерпретация: кто является наиболее активным участником сети.

Центральность по близости (closeness centrality) – суммируются все кратчайшие пути от узла до остальных узлов сети l_{ij} и находится обратная величина.

$$C_C(i) = 1 / \sum_j l_{ij}$$

Обычно она также нормируется на максимально возможное количество связей.

$$C_{CN}(i) = (N-1) / \sum_{j \neq i} l_{ij}, C_{CN}(i) \in [0,1]$$

Интерпретация: на сколько быстро доходит информация в сети от данного узла к остальным.

Меры центральности (наиболее популярные)

Центральность по посредничеству (betweenness centrality) – для всех пар узлов кроме рассматриваемого вычисляется доля кратчайших путей, проходящих через рассматриваемый узел, затем эти доли суммируются (количество кратчайших путей между j и k : σ_{jk} - всего; $\sigma_{jk}(i)$ - проходящих через узел i):

$$C_B(i) = \sum_{j < k; j, k \neq i} \sigma_{jk}(i) / \sigma_{jk}$$

Центральность по посредничеству обычно нормируется по количеству пар узлов в сети (без рассмотрения текущего узла):

$$C_{BN}(i) = C_B(i) / [(N - 1)(N - 2)/2], C_{BN}(i) \in [0, 1]$$

Интерпретация: центральность по посредничеству – это мера контроля, определяющего важность того или иного узла как передаточного звена. Если у какого-либо узла высокий показатель центральности по посредничеству, можно предположить, что он – единственная связь между различными частями сети.

Центральность по собственному вектору (eigenvector centrality) - скажи мне кто твой друг и я скажу кто ты. Узел имеет высокую центральность по собственному вектору, если он соединен с большим количеством узлов или соединен с другими узлами с высокой центральностью.

$$\frac{1}{\lambda} \sum_j a_{ij} x_j = x_i \quad \Rightarrow \quad A^T \vec{x} = \lambda \vec{x}$$

(в матричном виде) (задача поиска собственного вектора)

PageRank

В чем заключается подход?

Когда страница А ссылается на страницу Б, то:

- 1) Автор А считает, что в Б содержится интересная и важная информация;
- 2) Ссылка из А и Б добавляет репутации странице Б

Отсюда еще один аспект – не все ссылки одинаковы. Чем «важнее» страница, тем «дороже» ссылка от нее

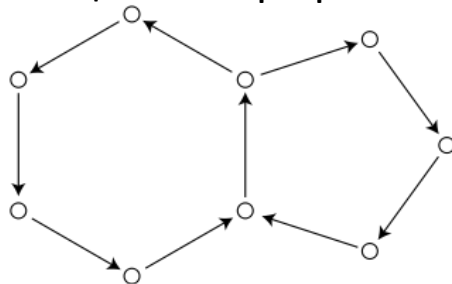
PageRank – постановка задачи

«Авторитет» каждой страницы разделяется между всеми ссылками с этой страницы. Суммарный «авторитет» страниц равен 1:

$$\begin{aligned} \sum_j a_{ij} (x_j / k_j^{out}) &= x_i \\ \sum_j x_j &= 1 \end{aligned} \Rightarrow \begin{aligned} &M - \text{нормированная по строкам } A \text{ (м-а смежности)} \\ &\text{(в матричном виде)} \quad \text{(задача поиска собственного вектора,} \\ &\quad \text{отвечающего собственному значению 1)} \end{aligned}$$
$$M^T \vec{x} = \vec{x}$$

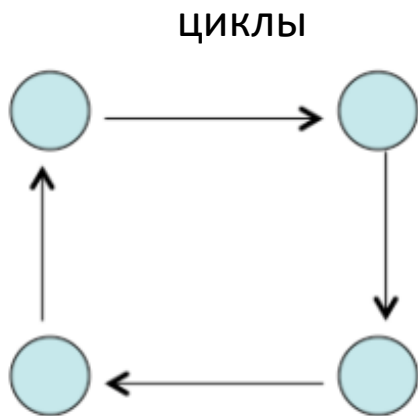
Задача имеет решение, если:

- Ориентированный граф сильно связный (т.е. из любого узла можно построить путь до любого узла)
- Граф апериодический (нет никакого целого числа $k > 1$, которое делит длину каждого цикла в графе... иными словами если наибольший общий делитель длин его циклов равен единице)

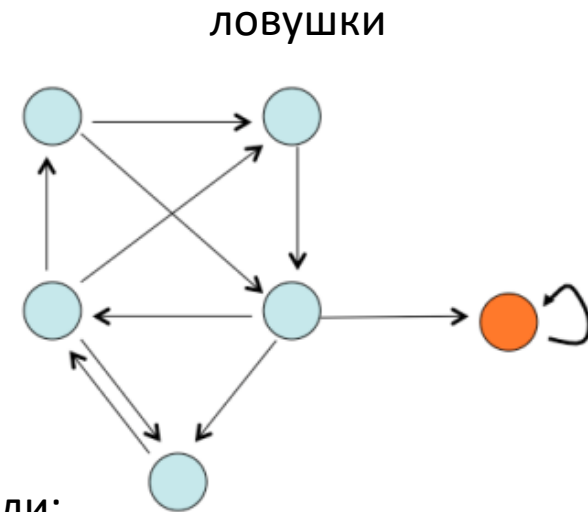


PageRank – ограничения

Для поиска решения задачи $M^T \vec{x} = \vec{x}$ существуют существенные ограничения на сеть.



Граф апериодический (нет никакого целого числа $k > 1$, которое делит длину каждого цикла в графе).



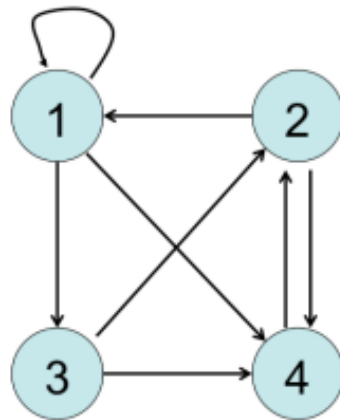
Задача имеет решение, если:

Ориентированный граф сильно связный (т.е. из любого узла можно построить путь до любого узла). Иначе можем застрять...

PageRank – снятие ограничений

Подход: рассчитать, как будет обходить сеть посетитель, случайно выбирающий ссылки.

С каждой страницы (узла) посетитель может равновероятно перейти на любую страницу, на которую ссылается данная страница. Вероятность посещения каждой страницы после очень длительного блуждания и есть ее «авторитет»



$$M^T = \begin{pmatrix} \frac{1}{3} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 1 \\ \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

$$\vec{x}(t) = M^T \vec{x}(t-1)$$

$$t \rightarrow \infty$$

$$\vec{x} = M^T \vec{x}$$

$$\vec{x}(0)^T = (1, 0, 0, 0)$$

$$\vec{x}(1)^T = (1/3, 0, 1/3, 1/3)$$

$$\vec{x}(2)^T = (0.11, 0.50, 0.11, 0.28)$$

$$\vec{x}(10)^T = \vec{x}(11)^T = (0.26, 0.35, 0.09, 0.30)$$

PageRank – снятие ограничений

Подход: добавить возможность случайного равновероятного перехода на любую из страниц (узлов) сети.

С вероятностью $(1 - \alpha)$ случайно перемещаемся на любую из страниц сети (E - единичная матрица, размера $N \times N$). Стохастическая матрица M преобразуется в стохастическую матрицу G , для которой ограничения выполнены аperiodичности и сильной связности выполняются автоматически.

$$G = \alpha M + (1 - \alpha) \frac{1}{N} E \quad \Rightarrow \quad G^T \vec{x} = \vec{x}$$

- **Граф для G сильно связный**
(т.е. из любого узла можно построить путь до любого узла).
- **Граф для G – аperiodический**

(задача поиска собственного вектора, отвечающего собственному значению 1)

Реализуют обычно именно этот вариант и $\alpha = 0.85$

Выявление сообществ (кластеризация)

Необходимо находить такие сообщества (кластеры), чтобы связи внутри них были сильными, а связи между кластерами были слабыми.

Есть множество алгоритмов кластеризации.

- **Алгоритм Гирвана-Ньюмена (Girvan-Newman)**. Идея заключается в подсчете количества проходов кратчайших путей через ребра и удаляем ребро с максимальным посредничеством. После каждого удаления у нас образуются сообщества и алгоритм работает до формирования заданного количества сообществ.
- **Алгоритм Лёвина (Louvain)** (<http://arxiv.org/abs/0803.0476>). Состоит следующих этапов:
 - 1) Каждый узел – уникальное сообщество (кластер);
 - 2) Для каждого узла рассматриваем возможность его перехода в соседнее сообщество при условии увеличения меры модулярности;
 - 3) Каждое новое сообщество – новый узел;
 - 4) Связь между новыми узлами определяется количеством связей исходных узлов;
 - 5) Повторяем все этапы до достижения максимума меры модулярности.

Мера модулярности (modularity) — показатель качества сообщества (отношение количества связей внутри сообществ к связям вовне).

Выявление сообществ (кластеризация)

Алгоритм распространения меток. Каждый узел – сообщество. На каждом шаге узел выбирает себе то сообщество, с которым он имеет связь и сообщество должно быть самым крупным. Если их несколько, то выбор происходит случайным образом.

Spectral clustering. Кластеризация на основе разрезов сообществ таким образом, чтобы минимизировать количество внешних ребер.

И многие другие методы.



NetworkX
Network Analysis in Python

Networkx

- Библиотека написана для анализа графов в Python. Является очень удобной и легкой в освоении, но есть недостаток – медленная для больших графов и работает только на CPU.
- Но очень много всего уже реализовано.

Немного про синтаксис

Создание графа

```
G = nx.Graph()
```

Добавление связей

```
G.add_edges_from([(1, 2), (1, 3)])
```

Добавление нод как по 1, так и по несколько.

```
G.add_node(1)
```

```
G.add_nodes_from([2, 3])  
G.nodes
```

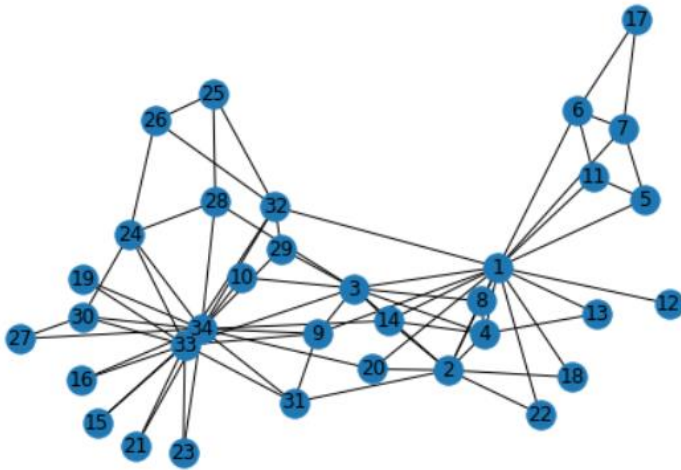
```
NodeView((1, 2, 3))
```

Можно все красиво добавить из pandas

```
G = nx.Graph()  
G = nx.from_pandas_edgelist(edges, source='source', target='destination')
```

Визуализация в networkx

```
nx.draw(G, with_labels=True)
```



Есть один нюанс... При каждой визуализации положение узлов может меняться, но через параметр `pos` можно зафиксировать положение узлов.

```
pos=nx.spring_layout(G)
```

```
nx.draw(G, pos=pos)
```

`node_color` – цвет узлов;

`node_size` – размер узлов; можно разным узлам давать разный размер;

`labels` – подписи узлов;

The RAPIDS logo is a purple rectangle with the word "RAPIDS" in white, bold, sans-serif capital letters. It is centered within a large, light blue circular graphic that has a textured, splattered edge.

RAPIDS

cuGraph

- Библиотека экосистемы rapids. А если мы говорим про rapids, то это сразу говорит нам о GPU и Multi-GPU, но с жертвованием разнообразием реализованного.

Немного про синтаксис

Создание графа

```
G_gpu = cg.Graph()
```

Add_node нет, только через add_nodes_from

```
G_gpu.add_nodes_from([1])  
G_gpu.nodes()
```

В cuGraph нельзя, к сожалению, добавлять новые ноды или связи в граф. Это обусловлено сложностью реализации данного процесса на GPU. Но всегда есть выход – можно вытянуть `cudf.DataFrame` со всеми связями, добавить туда новые записи и инициализировать новый граф.

```
G_gpu.edges()
```

	src	dst
0	1	2
1	1	3



```
appended_values = cudf.DataFrame({  
    'source': [2],  
    'destination': [3]  
})  
edge_list = edge_list.append(appended_values, ignore_index=True)
```



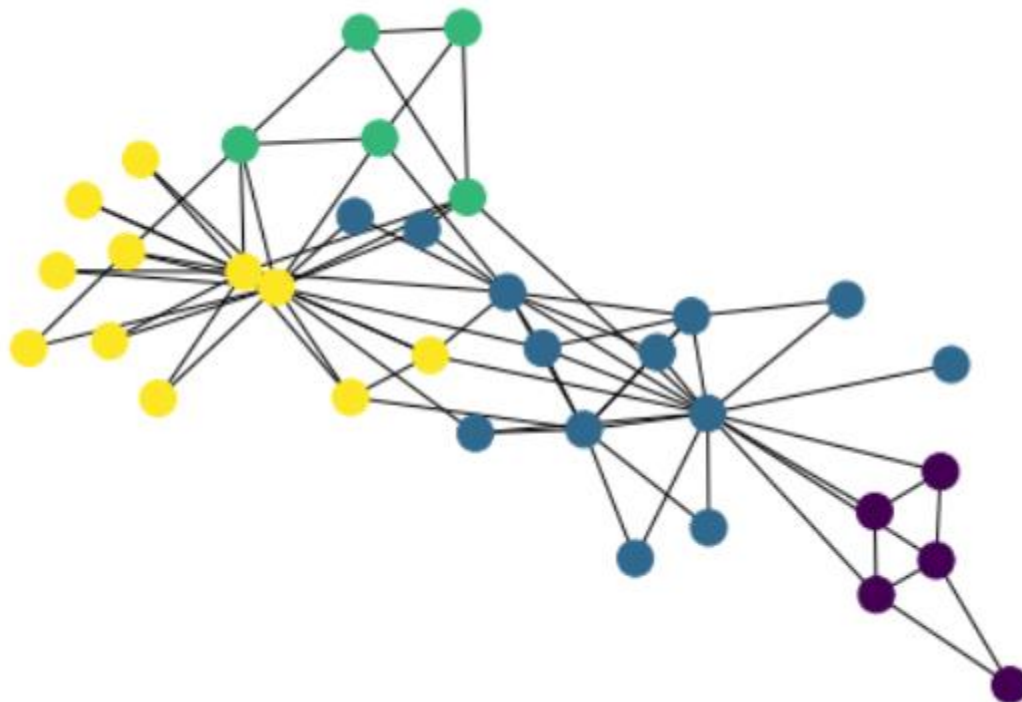
```
G_gpu = cg.Graph()  
G_gpu.from_cudf_edgelist(edge_list, source='source', destination='destination')
```

Кластеризация в cuGraph (поиск сообществ)

Пока в networkx почти ничего нет в части кластеризации графов, в cuGraph постарались решить эту проблему и подготовили уже готовые решения.

- ECG
- K-Truss
- Leiden
- Louvain
- Spectral Clustering
- Subgraph extraction
- Triangle Counting

Алгоритм Louvain, пример



<https://arxiv.org/pdf/1406.2518.pdf>

<https://arxiv.org/pdf/0803.0476.pdf>

Совместимость NetworkX и cuGraph

Можно удобно создать граф в networkx, что-то с ним сделать, а для анализа передать уже в cuGraph.

```
import networkx as nx
import time
import operator

# create a random graph
G = nx.barabasi_albert_graph(N, M)

... do some NetworkX stuff ..

t1 = time.time()
bc = nx.betweenness centrality(G)
t2 = time.time() - t1

print(t2)
```

NetworkX

```
import networkx as nx
import time
import operator
import cudgraph as cnx

# create a random graph
G = nx.barabasi_albert_graph(N, M)

... do some NetworkX stuff ..

t1 = time.time()
bc = cnx.betweenness centrality(G)
t2 = time.time() - t1

print(t2)
```

NetworkX + RAPIDS cuGraph

Что еще есть?

- 1) Различные методы определения сходства графов (подграфов)
- 2) Анализ взаимосвязей, например PageRank
- 3) Различные алгоритмы обхода графа

Немного бенчмарков

Считали betweenness centrality

Node	Edges	Speedup	NetworkX Runtime (sec)	cuGraph Runtime (sec)
100	1,344	0.45	0.05	0.11
200	2,944	1.14	0.20	0.18
400	6,144	2.64	0.84	0.32
800	12,544	5.26	3.84	0.73
1,600	25,344	12.99	17.51	1.35
3,200	50,944	26.5	78.10	2.95
6,400	102,144	48.62	338.73	6.97
12,800	204,544	89.81	1,539.73	17.14
25,600	409,344	180.42	7,804.06	43.25
51,200	818,944	328.05	47,763.09	145.60

Немного бенчмарков

