

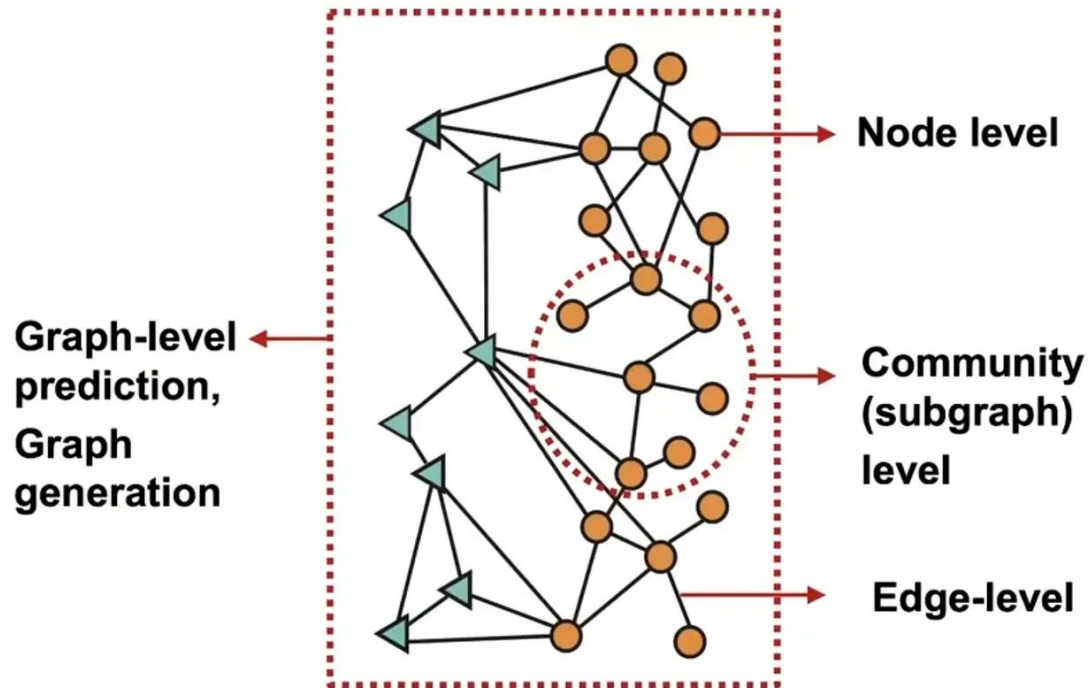


ІІТМО

**Анализ графовых
данных и глубокое
обучение**

Азимов Рустам
Высшая школа цифровой культуры

В предыдущих сериях

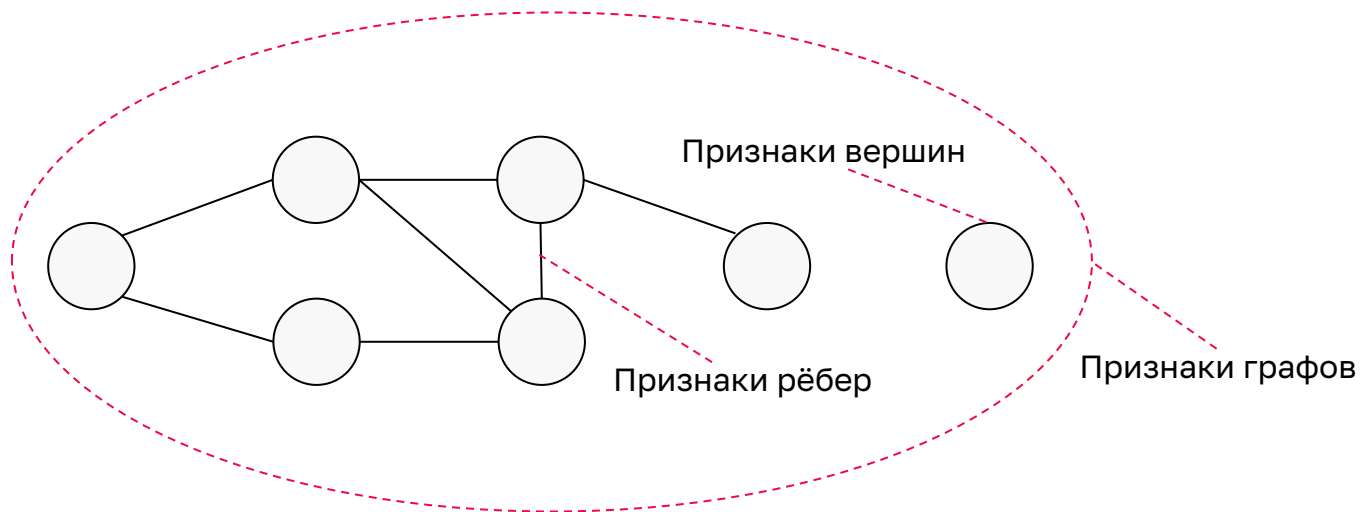


ІТМО

Классический ML для анализа графов

Классический подход

- Разработать правила получения признаков для вершин/рёбер/графов
- Заполнить данные для тренировочного набора и обучить модель



Hand-crafted features

- Выбор признаков для описания вершин/рёбер/графов является ключевым для качественного решения задачи
- В этой лекции рассмотрим варианты таких признаков
- Для простоты работаем только с неориентированными графами



Постановка задачи

- **Задача:** предсказать некоторые значения на новых объектах
- **Признаки:** d -мерные вектора
- **Объекты:** вершины/рёбра/графы
- **Функция** потерь: зависит от задачи



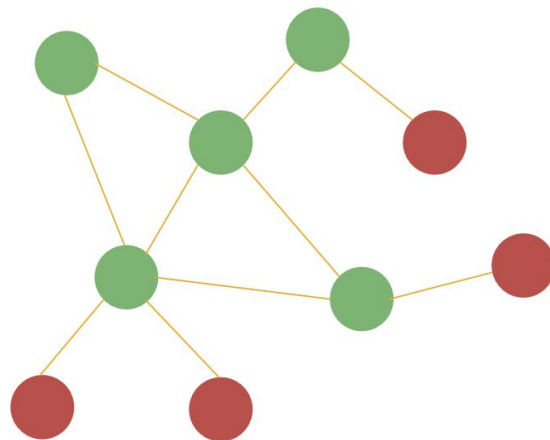
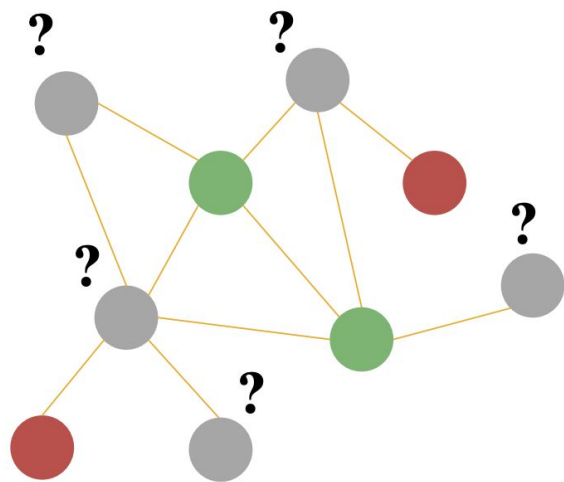


IITMO

Node-level tasks

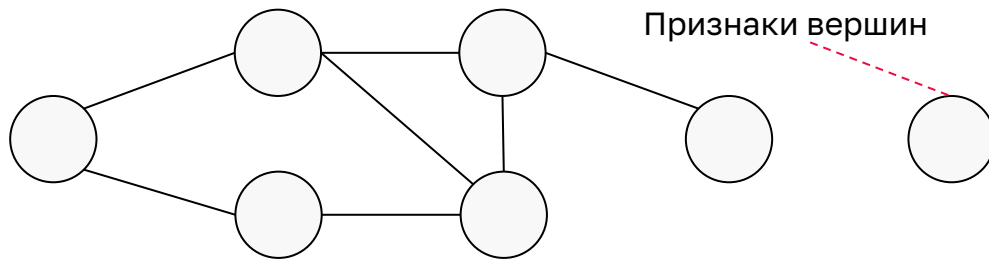
Node classification

VITMO



Признаки для вершин

- Степень вершины
- Показатель центральности
- Коэффициент кластеризации
- Графлеты

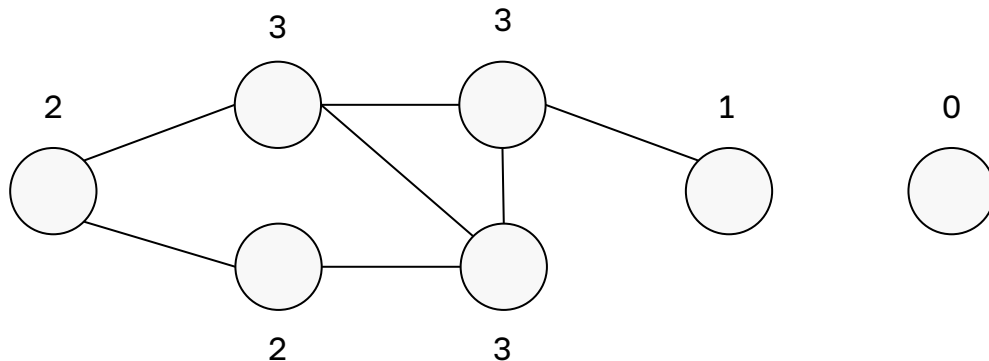


Степень вершины

- Пусть A - матрица смежности графа, тогда **степень** вершины u



$$d_u = \sum_{v \in V} A[u, v]$$



- Степень вершин не учитывает важность соседей
- Это можно сделать, например, с помощью показателя

центральности

- Eigenvector centrality
- Betweenness centrality
- Closeness centrality



Eigenvector centrality

- Важность вершины v зависит от важности соседей $N(v)$



$$c_v = \frac{1}{\lambda} \sum_{u \in N(v)} c_u$$

- В матричной форме $\lambda c = A c$
- Для центральности используется c_{max} , который соответствует наибольшему собственному значению λ_{max}

Betweenness centrality

- Важность вершины зависит от количества кратчайших путей, которые проходят через неё

$$c_v = \sum_{s \neq v \neq t} \frac{\text{number of shortest paths between } s \text{ and } t \text{ that contain } v}{\text{number of shortest paths between } s \text{ and } t}$$

Closeness centrality

- Вершина важна, если от неё можно быстро добраться до других



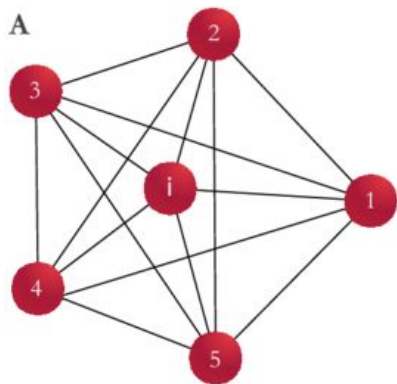
$$c_v = \frac{1}{\sum_{u \neq v} \text{length of the shortest path between } u \text{ and } v}$$

Clustering coefficient

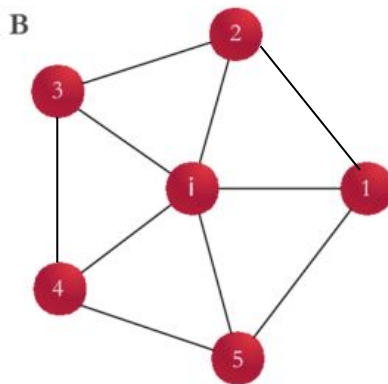
- Измеряет степень связности соседей вершины v



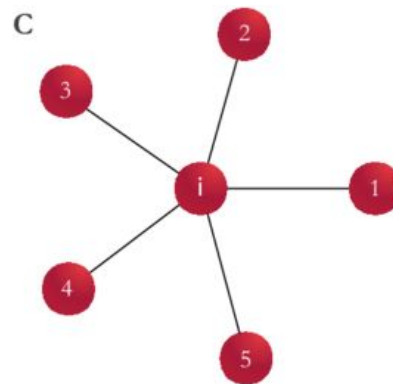
$$e_v = \frac{\text{number of edges among the neighbours of } v}{C_{|N(v)|}^2}$$



$$e_i = 1$$



$$e_i = 0.5$$



$$e_i = 0$$

- **Графлеты** - небольшие подграфы для описания структуры соседей вершины
- **Graphlet Degree Vector (GDV)** - вектор из подсчитанного количества графлетов для конкретной вершины
 - Для графлетов размера от 2 до 5, GDV - вектор размера 73
- GDV может использоваться в качестве hand-crafted признака вершины и описывает локальную топологию
- Сравнение GDV двух вершин более детально чем сравнение степеней вершин или коэффициента кластеризации

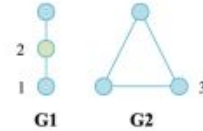


Graphlets

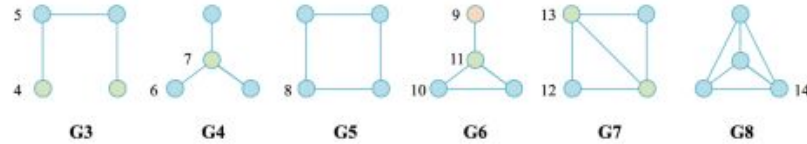
2-node Graphlets



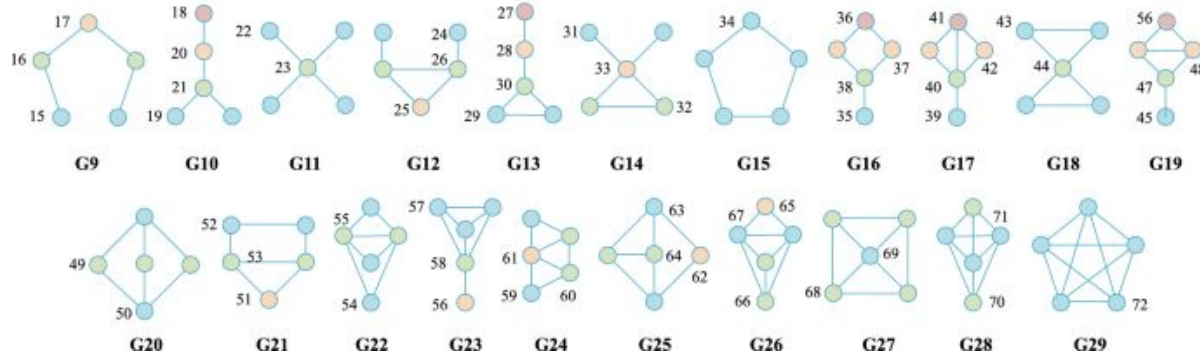
3-node Graphlets



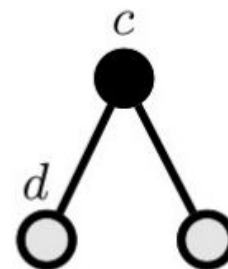
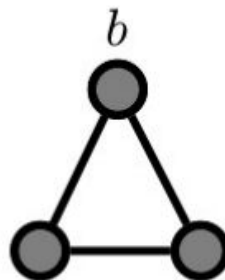
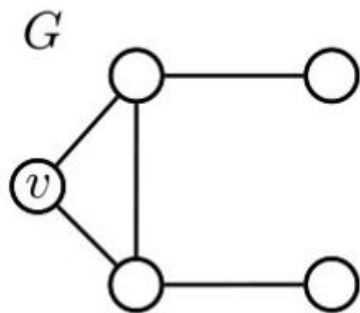
4-node Graphlets



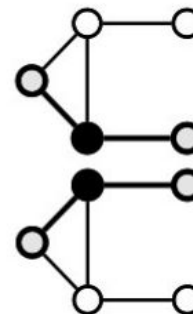
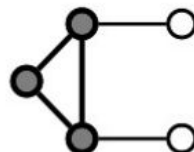
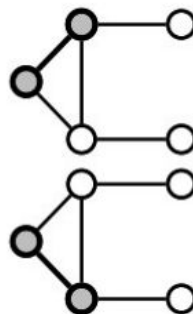
5-node Graphlets



Пример



$orbit$	a	b	c	d
$GDV(v)$	2	1	0	2





IITMO

Edge-level tasks

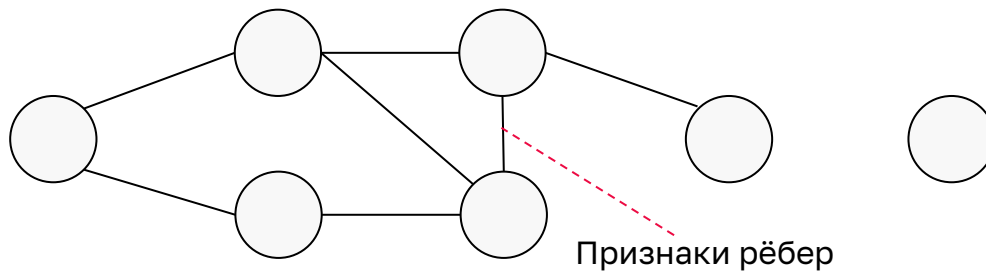
Предсказание связей

- Удалить случайные рёбра в графе и попытаться их предсказать
- Или граф меняется с течением времени, предсказать какие рёбра появятся
 - Для каждой пары вершин считаем score
 - Отранжировать предсказания и сравнить с реально появившимися рёбрами



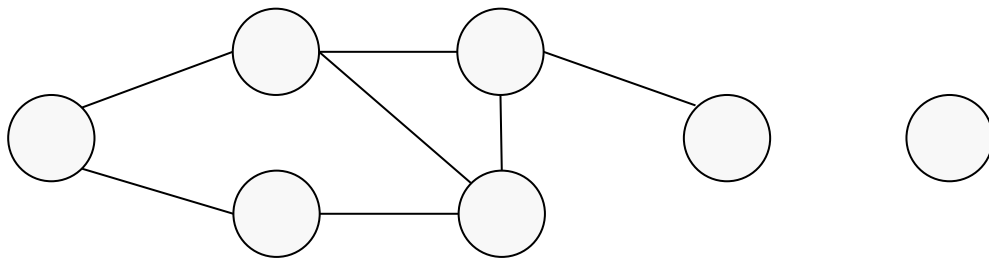
Признаки рёбер

- Distance-based
- Local neighborhood overlap
- Global neighborhood overlap



Distance-based

- Длина кратчайшего пути
- Не различает окружения вершин, например есть ли общие соседи
- Может быть несколько кратчайших путей



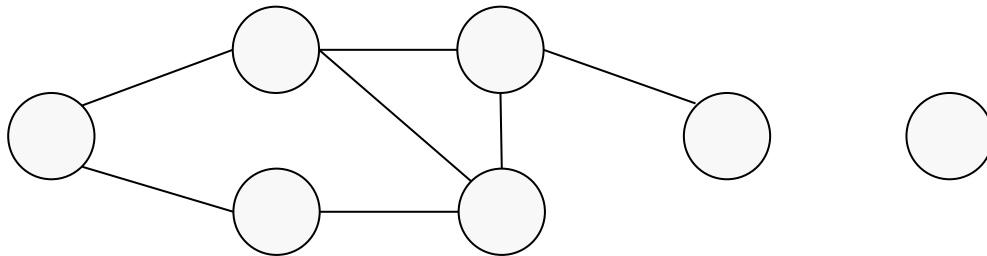
Local neighborhood overlap

- Смотрит на общих соседей у двух вершин v_1 и v_2



- Общие соседи $|N(v_1) \cap N(v_2)|$
- Jaccard's coefficient $\frac{|N(v_1) \cap N(v_2)|}{|N(v_1) \cup N(v_2)|}$
- Adamic-Adar index

$$\sum_{u \in N(v_1) \cap N(v_2)} \frac{1}{\log(k_u)}$$



Local neighborhood overlap

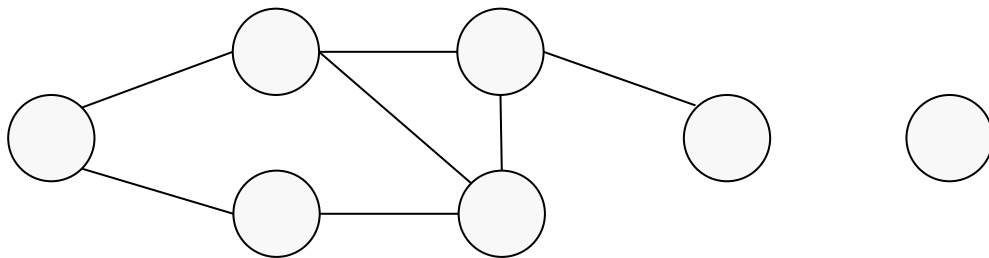
- Если вершины не имеют общих соседей, то все эти показатели равны нулю, хотя в будущем вероятность появления ребра имеется
- Нужно рассмотреть граф целиком



Global neighborhood overlap

- **Katz index** - количество путей между парой вершин
- Можно подсчитать с помощью возведения в степень матрицы смежности над стандартным полукольцом

$$S[v_1, v_2] = \sum_{l=1}^{\infty} \beta^l A^l[v_1, v_2]$$

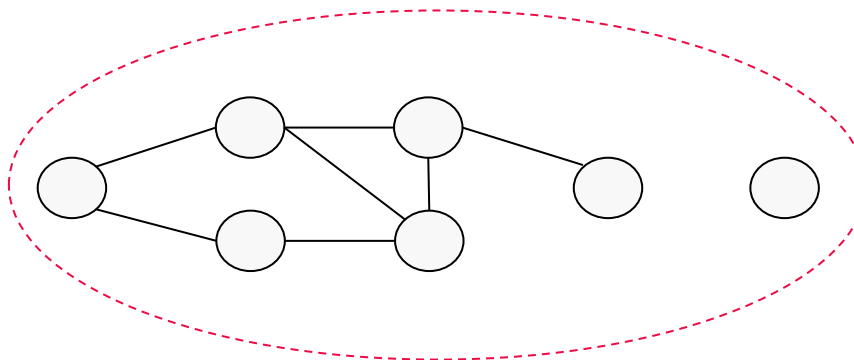




IITMO

Graph-level tasks

- Хотим характеризовать граф целиком
- Из простого - можно агрегировать признаки вершин и рёбер
- **Graph Kernels** $K(G, G')$ оценивают схожесть графов и можно перейти к SVM
 - Graphlet Kernel
 - Weisfeiler-Lehman Kernel



Признаки графов

Graph Kernel

- Основная идея - использовать для описания графа вектор наподобие Bag-of-Words
- “Bag of nodes”
- “Bag of node degrees”



Graphlet Features

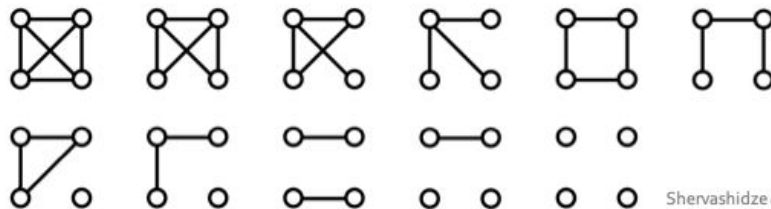
- “Bag of graphlets”: есть отличия от GDV для вершин
 - Не все вершины графлета обязаны быть связанными
 - Нет выделенной вершины (корня)



- For $k = 3$, there are 4 graphlets.

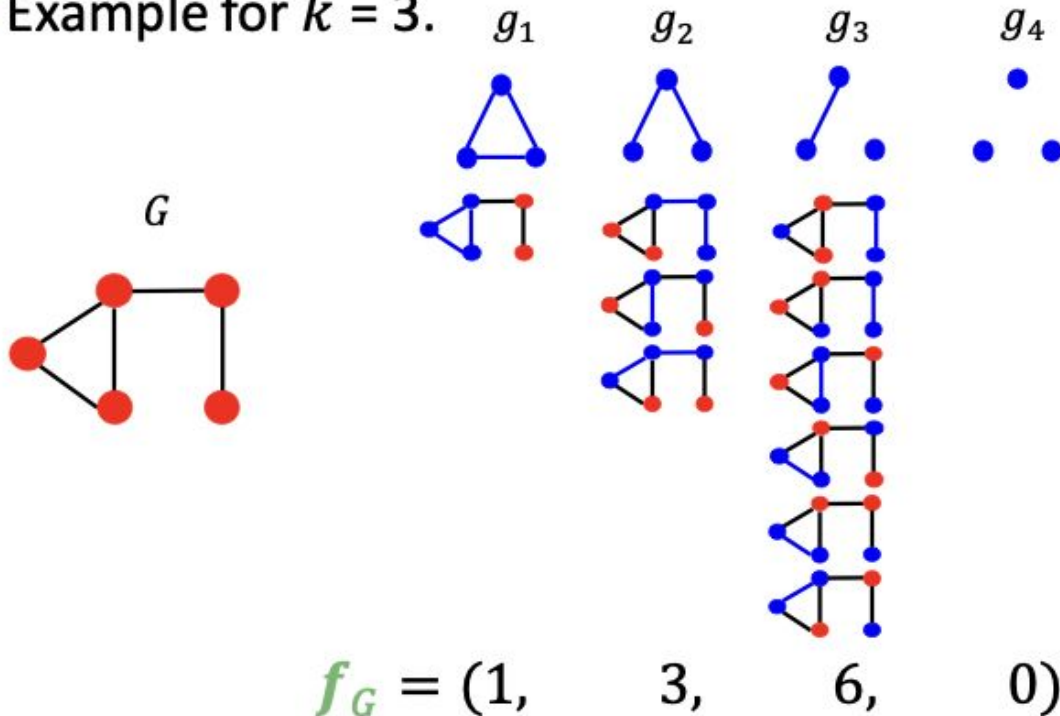


- For $k = 4$, there are 11 graphlets.



Graphlet Features

- Example for $k = 3$.



$$K(G, G') = \langle f_G, f_{G'} \rangle = f_G^T f_{G'}$$



- Графы могут быть существенно разных размеров, так что лучше нормализовать каждый вектор (поделить на сумму значений в нём)
- Дорого для вычисления на больших графах

- Хотим ядровую функцию, которую можно вычислить быстрее
- Обобщаем идею “Bag of node degrees”
- Итеративно раскрашиваем граф и рассматриваем всё более далеких соседей

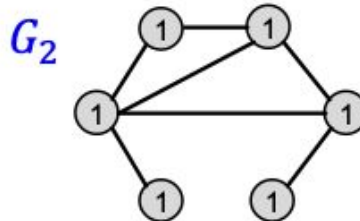
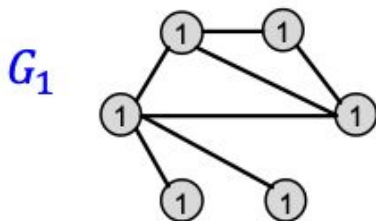
$$c^{(k+1)}(v) = \text{HASH} \left(\left\{ c^{(k)}(v), \{c^{(k)}(u)\}_{u \in N(v)} \right\} \right)$$

- Через K итераций получим информацию о K-hop соседстве в графе

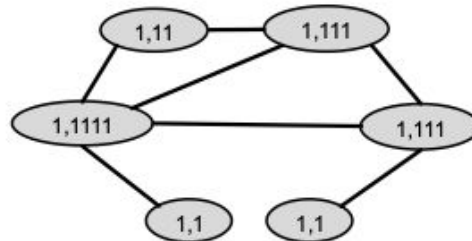
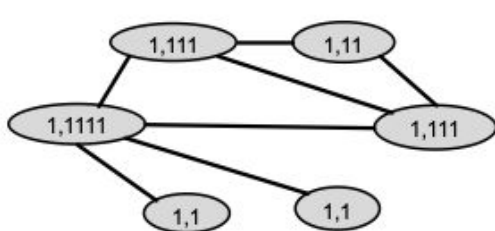
Example of color refinement given two graphs



- Assign initial colors



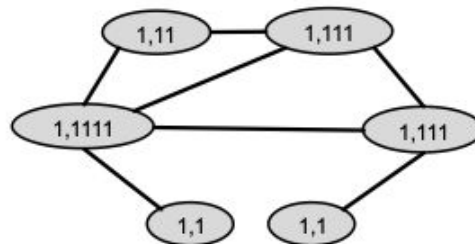
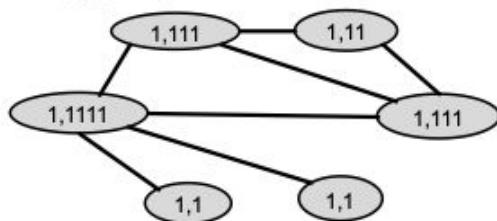
- Aggregate neighboring colors



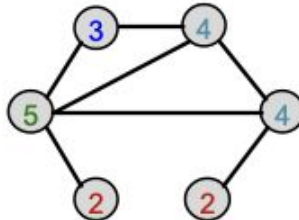
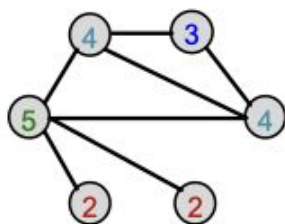
Example of color refinement given two graphs



■ Aggregated colors



■ Hash aggregated colors



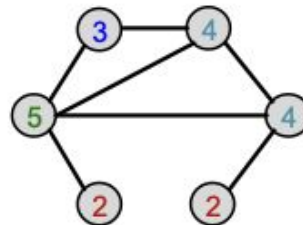
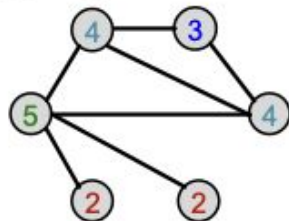
Hash table

1,1	-->	2
1,11	-->	3
1,111	-->	4
1,1111	-->	5

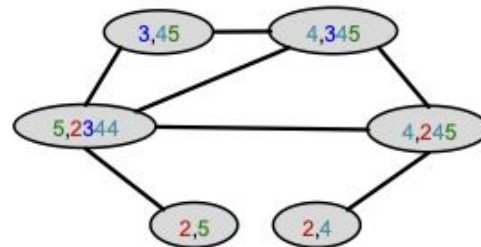
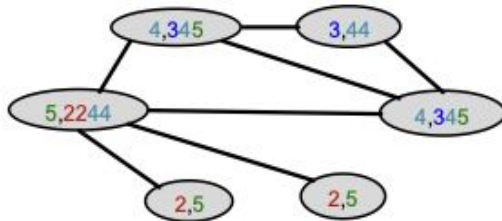
Example of color refinement given two graphs



■ Aggregated colors



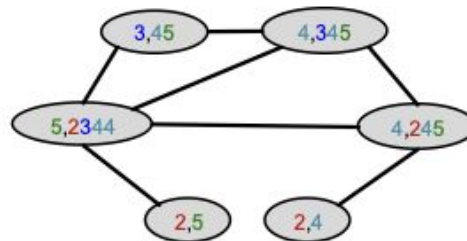
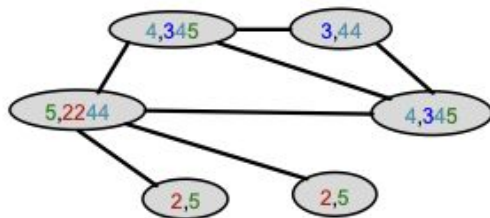
■ Hash aggregated colors



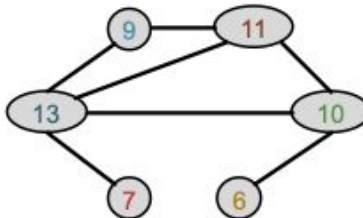
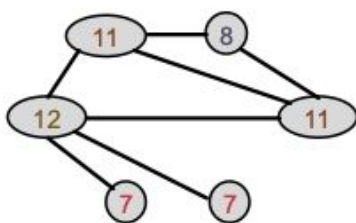
Example of color refinement given two graphs



■ Aggregated colors



■ Hash aggregated colors



Hash table

2,4	-->	6
2,5	-->	7
3,4,4	-->	8
3,4,5	-->	9
4,2,4,5	-->	10
4,3,4,5	-->	11
5,2,2,4,4	-->	12
5,2,3,4,4	-->	13

- После раскраски вычисляется вектор, содержащий информацию сколько вершин каждого цвета
- Ядровая функция опять же скалярное произведение векторов двух графов
- Линейная сложность по количеству рёбер в графах



Заключение

ІТМО

