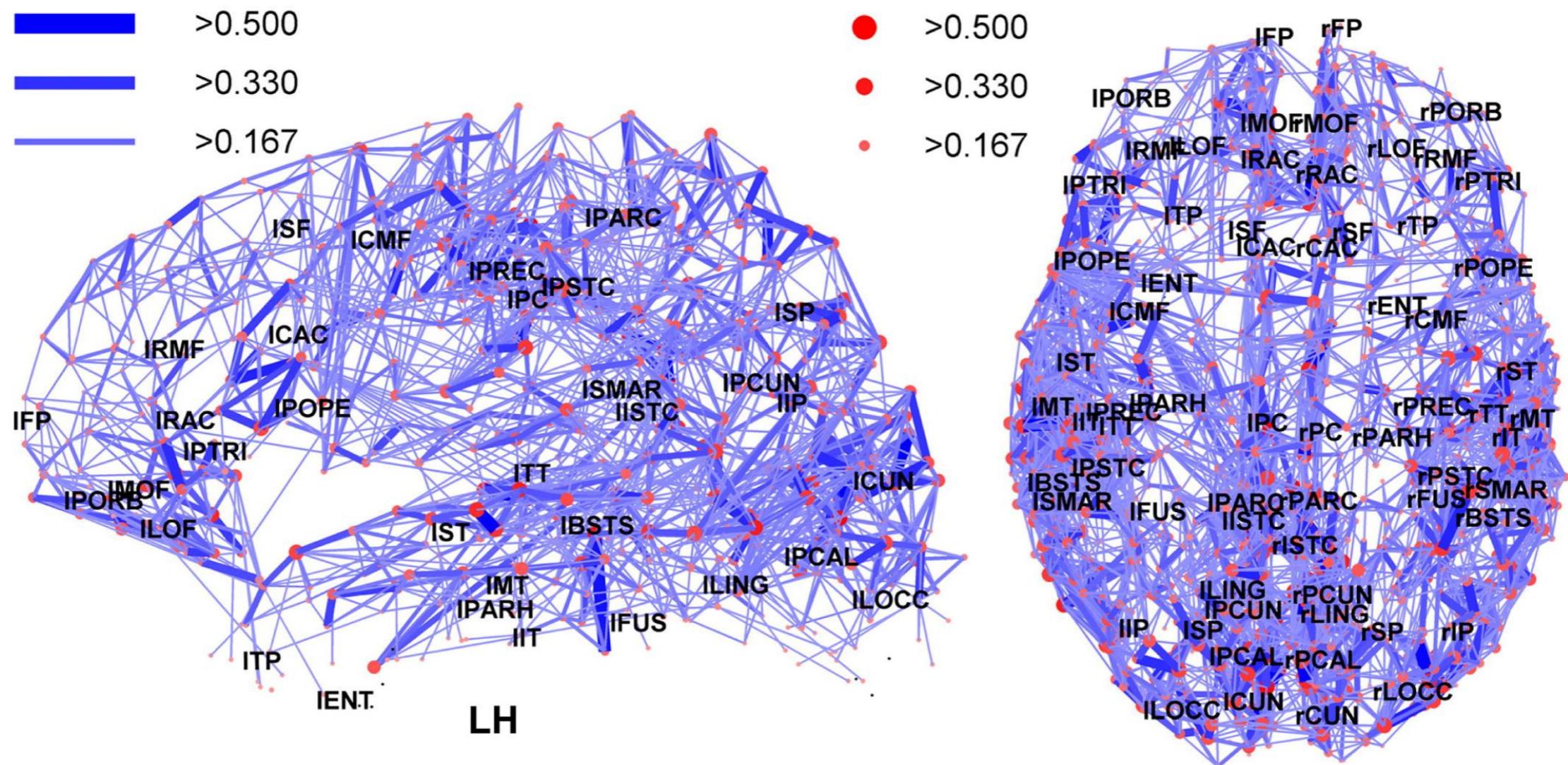


Поиск шаблонов в графах, моделирующих нейронную сеть головного мозга человека

Соколова Полина
371 группа

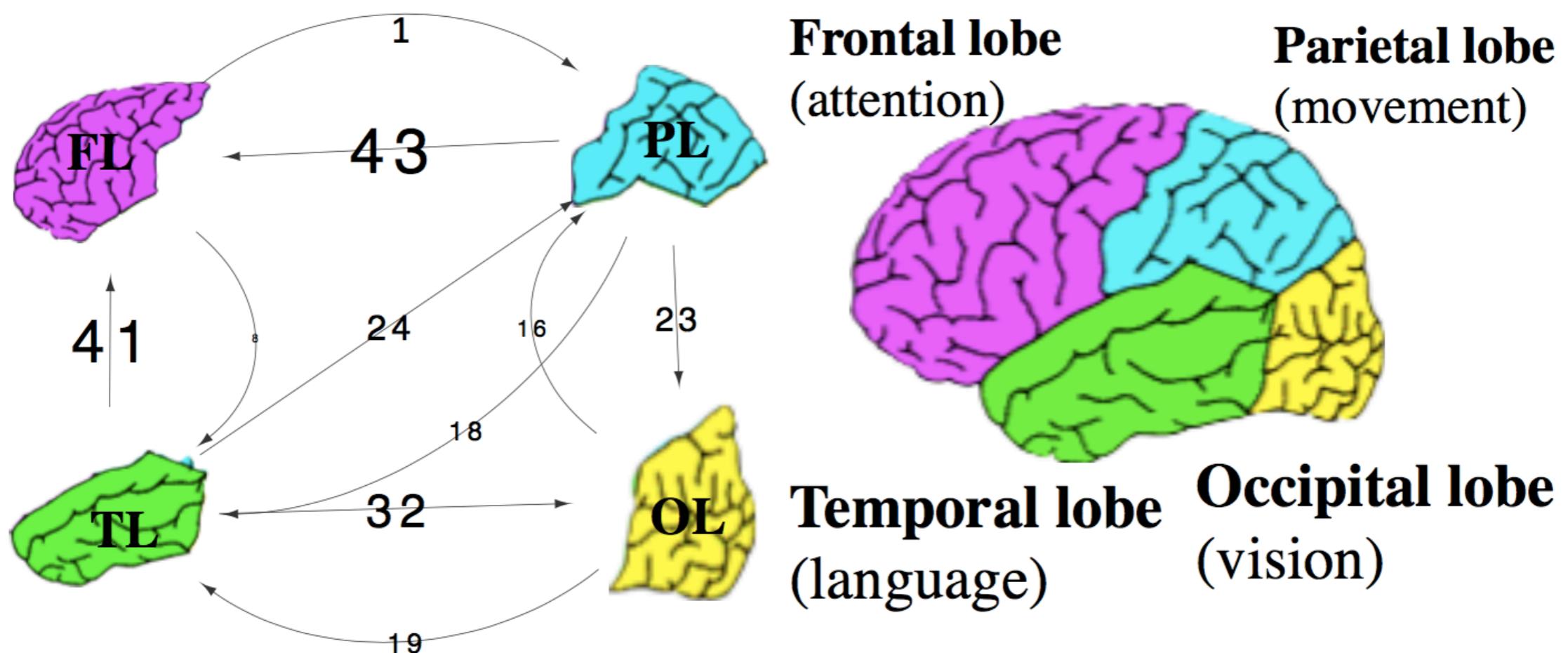
2017

Структурная связь



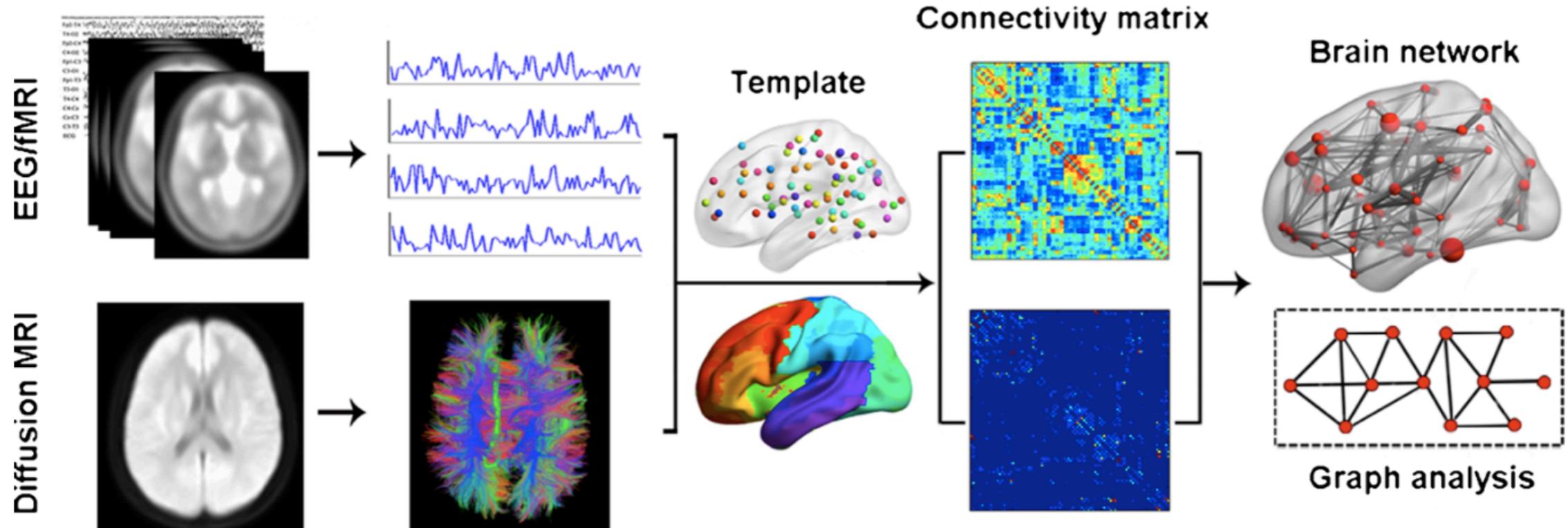
Пространственное представление структурной матрицы связности мозга человека с 66 областями с относительной плотностью связей между областями, указанными толщиной линии.

Функциональная связь



Веса на ребрах показывают количество предполагаемых соединений.

Построение сети



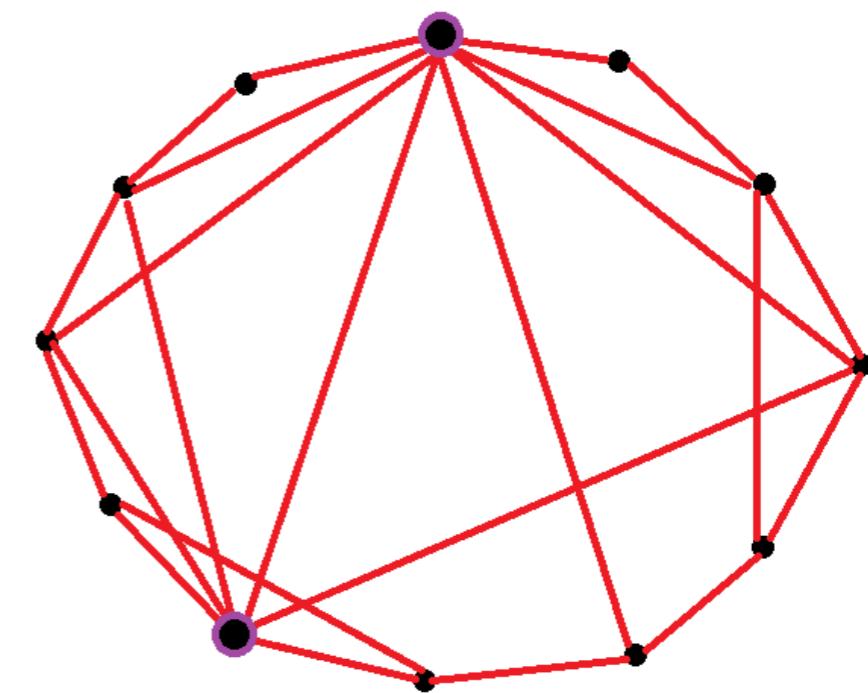
- Временные интервалы (фМРТ), пути распространения волокон (дМРТ)
- Разделение на части
- Матрицы связности
- Визуализация

Граф «Мир тесен»

- Высокий коэффициент кластеризации

$$C = \frac{3 \times \text{число треугольников в сети}}{\text{число «вилок»}}$$

- Изобилие хабов
- Малая длина кратчайшего пути в среднем



Пример графа «Мир тесен», выделены хабы

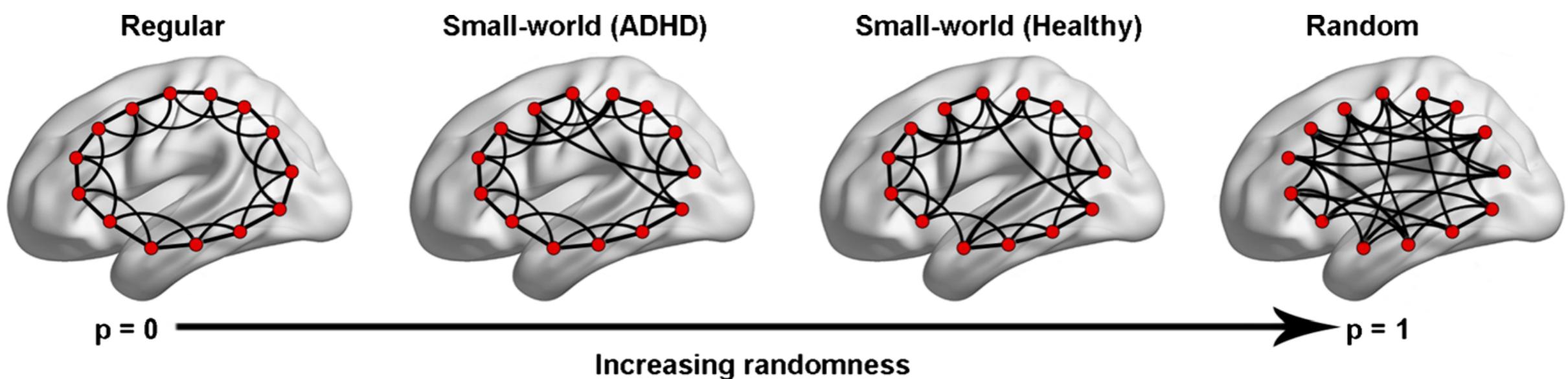
Средняя степень вершины = 1.917

Средняя длина кратчайшего пути = 1.803

Коэффициент кластеризации = 0.522

Синдром дефицита внимания с гиперактивностью

- Более высокий коэффициент кластеризации (локальная эффективность) и меньшая длина пути (глобальная эффективность)
- Задержка в развитии функциональных связей как всей сети мозга, так и подсетей

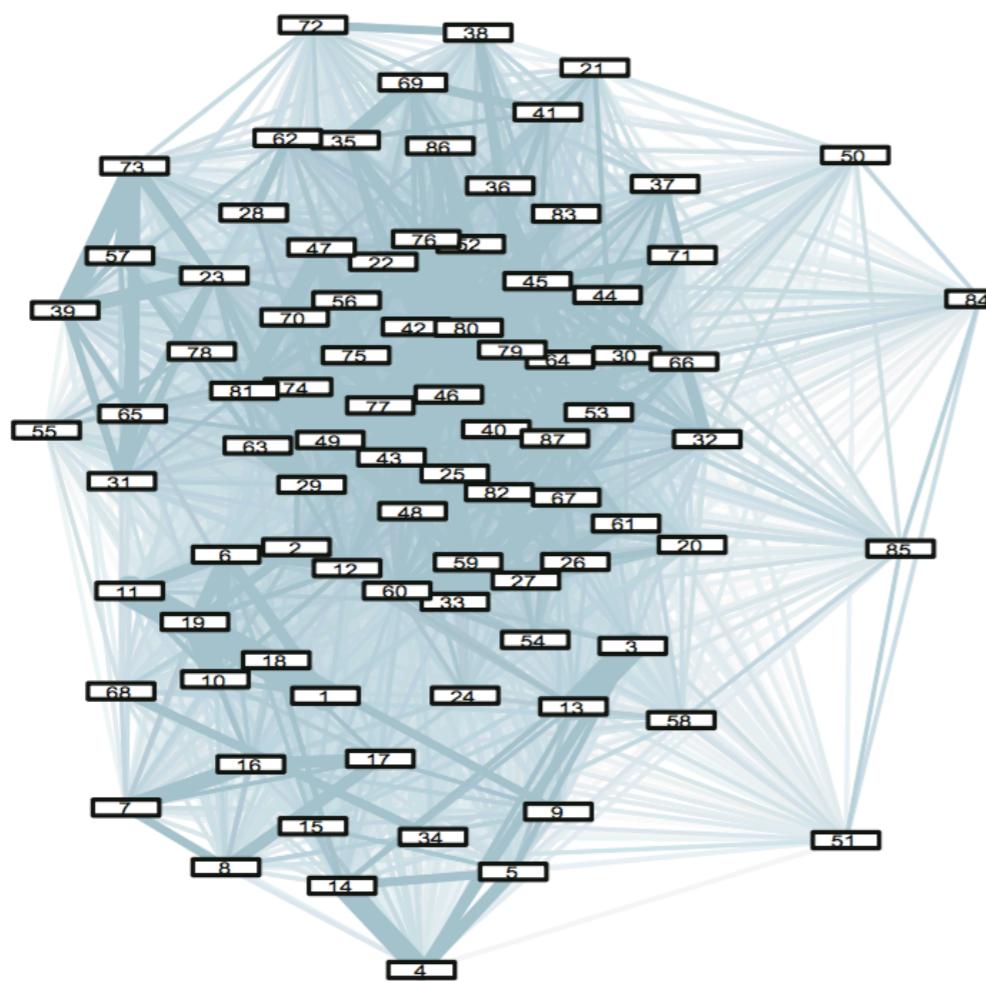


Параметры для сравнения сетей

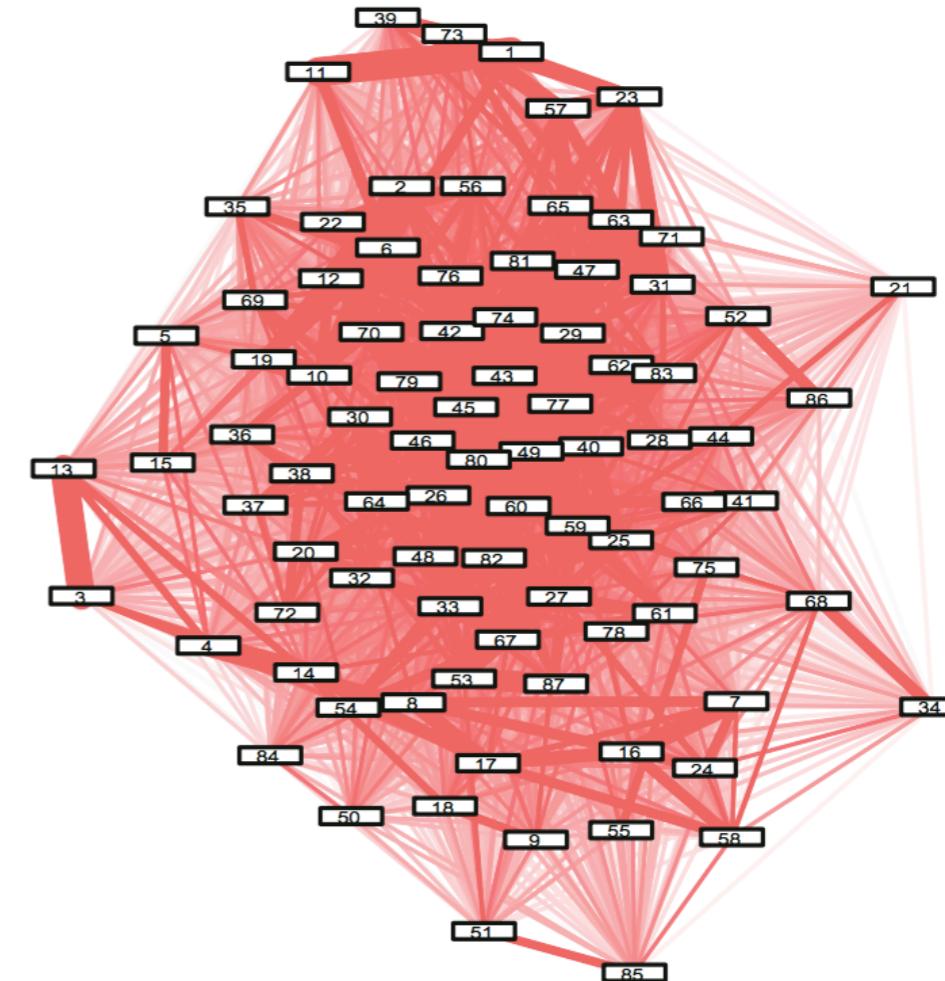
- Показатель эффективности сети (поддержка параллельной обработки информации)
- Свойства графа «Мир тесен» (коэффициент кластеризации, длина пути)
- Мера сегрегации (усредненное значение количества кратчайших путей, проходящих через узел)
- Мера сложности сети (~ 0 - у минимально связанных и полностью связанных, среднее - у сложных структур)
- Энтропия (топологический информационный контент, Индекс Берца и др.)

Общий сетевой анализ

- Нет существенных отличий по ключевым параметрам



Сеть мозга здорового человека



Сеть мозга человека с болезнью Альцгеймера

Кластеризация графов

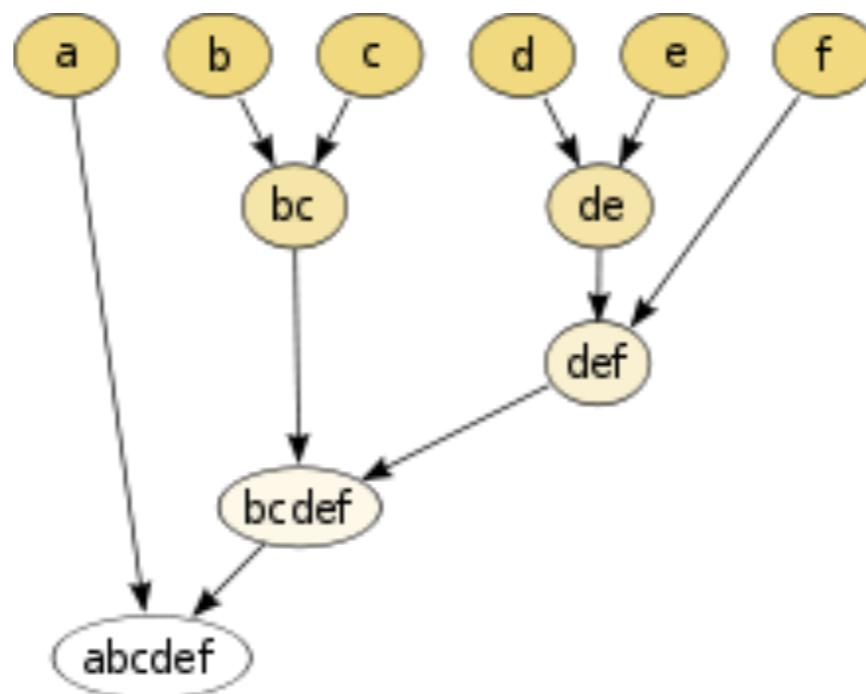
Интервальная оценка ребра - количество путей, содержащих это ребро

Алгоритм:

1. Вычисление интервальных оценок для всех ребер в сети
2. Поиск ребра с максимальной интервальной оценкой и удаление его из сети
3. Пересчет интервальных оценок для оставшихся ребер
4. Повтор шага 2

Алгоритм кластеризации

- На выходе: дендрограмма, представляющая целую вложенную иерархию возможных делений сети на подсети



Пример дендрограммы

Разделение на кластеры

- Мера модульности

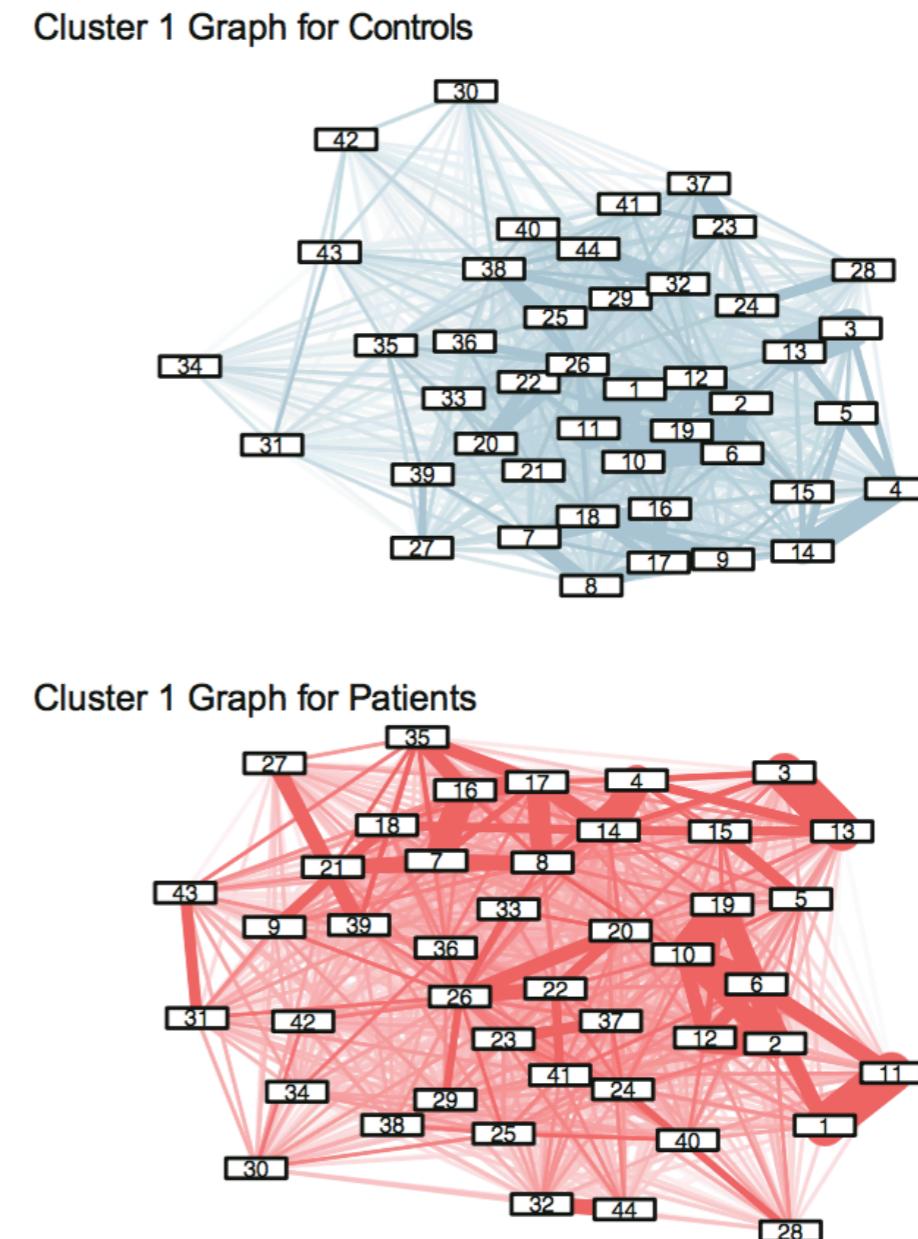
$$Q = \sum_{i=1}^k (e_{ii} - a_i^2)$$

$$e_{ii} = \frac{|\{(u, v) : u, v \in V_i, (u, v) \in E\}|}{|E|}$$

$$a_i = \frac{|\{(u, v) : u \in V_i, (u, v) \in E\}|}{|E|}$$

Сравнение параметров первых кластеров

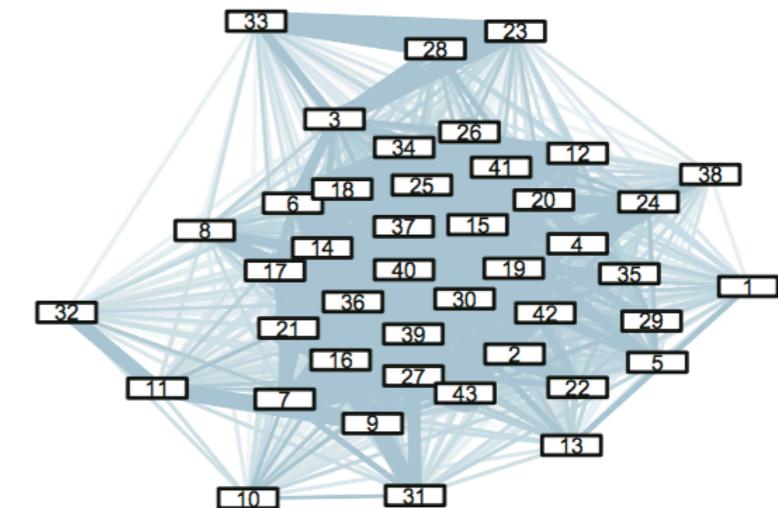
- Уменьшение коэффициента кластеризации
- Уменьшение структурного разнообразия
- По остальным параметрам нет существенных изменений



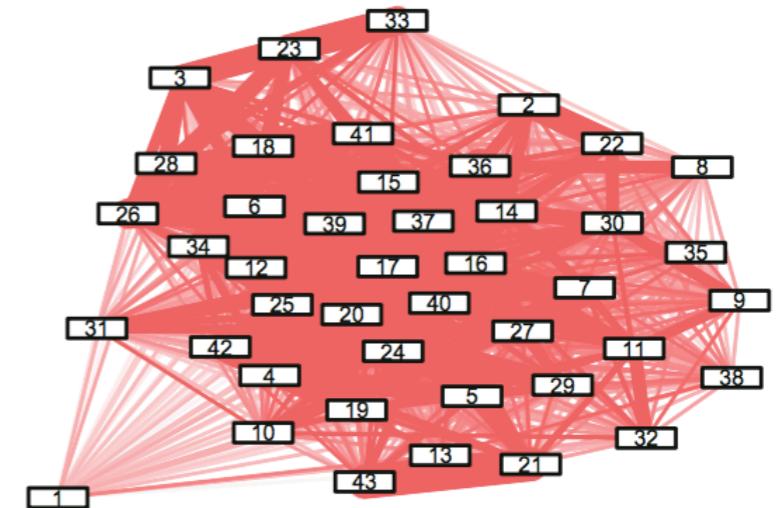
Сравнение параметров вторых кластеров

- Уменьшение структурного разнообразия
- По остальным параметрам нет существенных изменений

Cluster 2 Graph for Controls

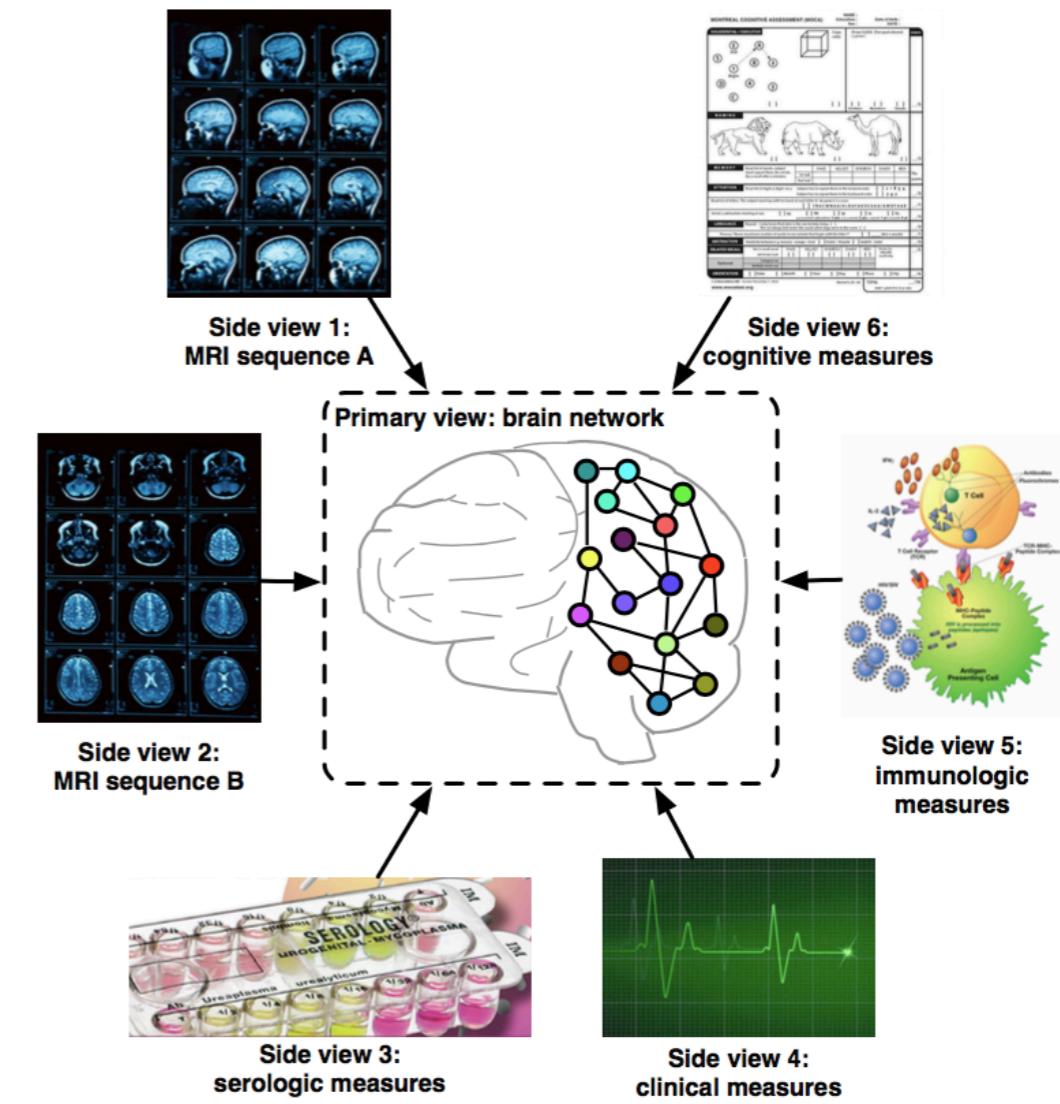


Cluster 2 Graph for Patients



Изучение боковых видов для классификации графов

- Боковые виды:
клинические,
иммунологические,
серологические и
когнитивные меры
- Критерий оценки
полезности шаблонов
подграфов - gSide
- Алгоритм gMSV - поиск
оптимальных признаков
подграфа



gMSV

$\mathcal{D} = \{G_1, \dots, G_n\}$ набор графов

$\mathcal{S} = \{g_1, \dots, g_m\}$ множество шаблонов подграфов из \mathcal{D}

\mathcal{T} множество выбранных шаблонов $\mathcal{T} \subseteq \mathcal{S}$

min_sup минимальный порог частоты

k количество выбранных шаблонов

$\lambda^{(p)}$ вес р-бокового вида (default: 1)

$\kappa^{(p)}$ функция ядра р-бокового вида (default: RBF-ядро)

Input: $\mathcal{D}, min_sup, k, \{\lambda^{(p)}, \kappa^{(p)}\}_{p=1}^v$

Output: \mathcal{T} : Set of optimal subgraph patterns

```

1:  $\mathcal{T} = \emptyset, \theta = Inf$ 
2: while unexplored nodes in the DFS code tree  $\neq \emptyset$  do
3:    $g =$  currently explored node in the DFS code tree
4:   if  $freq(g) \geq min\_sup$  then
5:     if  $|\mathcal{T}| < k$  or  $q(g) < \theta$  then
6:        $\mathcal{T} = \mathcal{T} \cup \{g\}$ 
7:     if  $|\mathcal{T}| > k$  then
8:        $g_{max} = \text{argmax}_{g' \in \mathcal{T}} q(g')$ 
9:        $\mathcal{T} = \mathcal{T} / \{g_{max}\}$ 
10:      end if
11:       $\theta = \max_{g' \in \mathcal{T}} q(g')$ 
12:    end if
13:    if  $\hat{q}(g) < \theta$  then
14:      Depth-first search the subtree rooted from  $g$ 
15:    end if
16:  end if
17: end while
18: return  $\mathcal{T}$ 

```

Поиск шаблонов в графе

- gApprox (различия в метках и структурные различия в ребрах)
- APGM (различия в названиях вершин)
- VEAM (различия в метках)

ФУНКЦИЯ ПОХОЖЕСТИ

$$f_1(g_1, g_2) = \kappa_1 v_{edit} + \kappa_2 e_{edit}$$

$$g_1 = (V_1, E_1, L_1) \quad g_2 = (V_2, E_2, L_2) \quad m \subseteq V_1 \times V_2$$

κ_1 и κ_2 - стоимости редактирования по вершинам и ребрам

$$\kappa_1 + \kappa_2 = 1$$

$$f_2(P, g) = \frac{\kappa_1 v_{edit}}{|V_P| + |V_g|} + \frac{\kappa_2 e_{edit}}{|E_P| + |E_g|}$$

$$P = (V_P, E_P) \quad g = (V_g, E_g)$$

$$v_{edit} = \sum_{v \in V_1 \setminus R_{V_1}} d_v(v, m(v)) + |R_{V_1}| + |R_{V_2}|$$

$$e_{edit} = \sum_{(u, v) \in E_1 \setminus R_{E_1}} d_e((u, v), (m(u), m(v))) + |R_{E_1}| + |R_{E_2}|$$

$$R_{V_1} = \{v_1 \in V_1 \mid \nexists v_2 \in V_2 \text{ such that } m(v_1) = v_2\}$$

$$R_{V_2} = \{v_2 \in V_2 \mid \nexists v_1 \in V_1 \text{ such that } m(v_1) = v_2\}$$

$$R_{E_1} = \{(u, v) \in E_1 \mid \nexists (u', v') \in E_2 \text{ such that } m(u) = u', m(v) = v'\}$$

$$R_{E_2} = \{(u', v') \in E_2 \mid \nexists (u, v) \in E_1 \text{ such that } m(u) = u', m(v) = v'\}$$

$$d_v(v_1, v_2) = \text{стоимость замещения } L_1(v_1) \text{ на } L_2(v_2)$$

$$d_e((u, v), (u', v')) = \text{стоимость замещения } L_1((u, v)) \text{ на } L_2((u', v')).$$

AGraP

Алгоритм позволяет находить вхождение шаблона в графе с помощью неточного соответствия

Input: G : анализируемый граф, σ : порог частоты, delta : порог подобия,

D : словарь эквивалентных марок (опционально)

Output: P : набор частых шаблонов в G .

$P \leftarrow \emptyset;$

for $v \in G$ **do**

Пометить вершину v ;

$Mv \leftarrow$ Список вершин с одинаковой или эквивалентной (судя по словарю) меткой вершины v и стоимость их редактирования;

if $|Mv| \geq \sigma$ **then** Add $\{v\}$ to P

$P_{\text{Traverse}} \leftarrow \text{Traverse}(\{v\}, Mv);$

$P \leftarrow P \cup P_{\text{Traverse}};$

Traverse(P, Mp)

Берет непомеченные вершины и создает из них список для передачи в Expand

Input: P: паттерн-кандидат, Mp: список вхождений P и стоимость их редактирования.

Output: Pexp: набор паттернов, которые можно получить из P.

$V_{exp} \leftarrow$ Непомеченные вершины, соединенные с P;

for $u \in V_{exp}$ **do**

 marked(u) \leftarrow True;

$P_{exp} \leftarrow \text{Expand}(P, Mp, V_{exp})$;

for $u \in V_{exp}$ **do**

 marked(u) \leftarrow False;

Expand(P , M_p , V_{exp})

Расширяет шаблон P до каждой вершины из V_{exp} и исследует новые шаблоны, вызывая $Traverse$

Input: P : паттерн-кандидат, M_p : список вхождений P и стоимость их редактирования, V_{exp} : список непомеченных вершин, связанных с P .

Output: P_p : набор паттернов, которые можно получить из P .

$P_p, P_e, P_t \leftarrow \emptyset;$

for $v \in V_{exp}$ **do**

$P' \leftarrow P \cup \{v\};$

$M_p' \leftarrow \text{ExpandOccurrence}(P', M_p, v);$

Рассчитать функцию похожести для P' . Если значение удовлетворяет σ - добавить P' в P_p ;

$V_{exp} \leftarrow V_{exp} \setminus \{v\};$

if $|M_p'| > \sigma$ **then**

$P_e \leftarrow \text{Expand}(P', M_p', V_{exp});$

$P_t \leftarrow \text{Traverse}(P', M_p');$

$P_p \leftarrow P_p \cup P_e \cup P_t$

ExpandOccurrence(P, Mp, newVertex)

Идентифицирует вхождение шаблона, здесь допускаются структурные различия между графиками

Input: P: паттерн-кандидат, Mp: список вхождений P и стоимость их редактирования, newVertex: вершина, соединенная с P.

Output: Mp' : список вхождений P = P ∪ {newVertex} и стоимость их редактирования.

```
Mp' ← Ø;  
for occurrence O in Mp do
```

Добавить O в Mp' и отметить отсутствие новой вершины символом '-'. Рассчитать и сохранить запись стоимости редактирования между O и P'; // допускаем вхождения с меньшим количеством вершин, чем в шаблоне

Оценить максимальную длину пути (l), которая может существовать между O и новой вершиной, не превышая порог подобия delta; // допускаем вхождения с большим количеством вершин, чем в шаблоне (+ далее)

Получить множество вершин (Neigh), связанных с O путем длины меньше либо равным l;

Определить подмножество Ve вершин в Neigh, которые имеют метку, равную или эквивалентную метке newVertex;

Добавить каждый граф O ∪ {v}, v ∈ Ve, в список Mp и вычислить (и сохранить) стоимость редактирования между этим новым графом и P. // а еще здесь допускаем ребра, отличающиеся от ребер шаблона (когда считаем стоимость редактирования)

Return list Mp'.

AGraP

- Находит больше вхождений, чем gApprox
- Сложность: $O(n^2 n!)$
- Не может обрабатывать большие (> 1000 вершин) высокосвязные графы с низким разнообразием меток
- Может быть применим, например, при анализе нейронных сетей головного мозга